
Zotonic

Release 1.0.0-rc.14

Jan 23, 2023

Contents

1	Organization	3
2	User Guide	5
2.1	User Guide	5
3	Developer Guide	11
3.1	Developer Guide	11
4	Cookbook	267
4.1	Cookbooks	267
5	Best Practices	339
5.1	Best Practices	339
6	Reference	341
6.1	Reference	341
7	Indices and tables	637
7.1	Glossary	637

Note: This documentation is for the 1.x version of Zotonic. For the stable 0.x version, please refer to the [0.x documentation](#).

Welcome to the documentation for [Zotonic](#), the Erlang web framework! This describes 1.0, last updated Jan 23, 2023.

CHAPTER 1

Organization

The documentation is divided into the following parts:

- The *User Guide* (page 5) is an introduction to the Zotonic CMS for end users such as administrators and editors.
- The *Developer Guide* (page 11) is the technical handbook for developers building websites with Zotonic. It guides you through all aspects of the framework.
- The *Cookbooks* (page 267) are a collection of recipes that show you how to solve reoccurring problems.
- *Best Practices* (page 339) as formulated by the Zotonic core team.
- The *Reference* (page 341) is a complete and detailed overview of all Zotonic components.

An introduction to the Zotonic CMS for end users such as administrators and editors.

2.1 User Guide

See also:

if you're a developer who's already familiar with Zotonic, you may want to read the [Developer Guide](#) (page 11) instead.

Welcome to the Zotonic User Guide. This guide is a non-technical introduction to Zotonic.

Table of contents

2.1.1 CMS

Zotonic is a Content Management System (CMS).

Todo

extend CMS introduction

2.1.2 The Zotonic data model

Zotonic's data model can be seen as a pragmatic implementation of the [Semantic Web](#): a mixture between a traditional database and a triple store.

The data model has two main concepts: the [resource](#) and the [edge](#).

Resources, which are often called *pages* in the admin, are the main data unit: they have properties like title, summary, body text; and importantly, they belong to a certain [category](#).

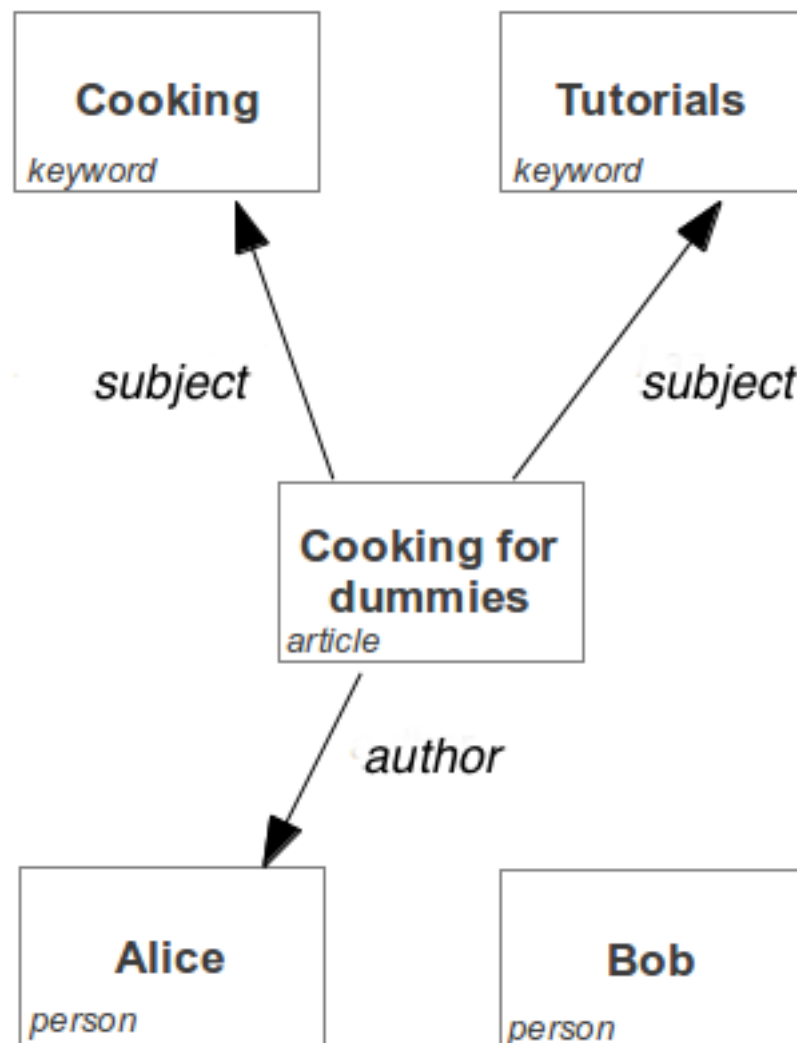
Edges are nothing more than connections between two resources. Each edge is labeled, with a so-called [predicate](#).

By default, Zotonic comes with a set of categories and predicates that makes sense for typical websites.

You can extend the built-in domain model by adding your own categories and predicates.

This manual describes the data model and its related database table in depth.

The data model by example



Take the example data on the right. It shows resources, connected to each other by edges. The rectangular blocks denote the resources, the arrow between them denote the edges.

It shows the *domain model* of a basic blog-like structure: `article` resources which are written by `persons` and articles being tagged with keywords.

The edge between *Alice* and *cooking for dummies* is labelled *author*: indicating that Alice wrote that article.

Note that these edges have a direction. When reasoning about edges, it is the easiest to think of them in terms of grammar: *Subject - verb - object*. In the case of Alice:

- **Cooking for dummies** is authored by **Alice**;
- **Cooking for dummies** has the keyword (subject) **Cooking**;

or more general:

- **subject predicate object**.

In Zotonic, the terms *subject* and *object* (shortened as *s* and *o*) are used in the templates, allowing you to traverse the edges between resources:

```
{{ m.rsc[id].o.author[1].title }}
```

Returns the name of the first author of the article with the given *id*, by traversing to the first *author* object edge of the given *id*. See the *m_rsc* (page 571) model for more details on this.

Pages

See also:

resources (page 26) in the Developer Guide

Resources are the main building block of Zotonic's data model. For simplicity of communication, a resource is often referred to as a page. Every resource usually has its own page on the web site, which is the *HTML representation* of the resource.

A resource can have multiple representations, for example *application/json*.

Different representations are served at different URLs. The *Non-Informational Resource URI* is used to negotiate the requested content-type and redirect to the best matching URI. The non-informational resource URI is usually */id/{id}* where *id* is the numerical id of the resource. The dispatch rule is called *id*.

The dispatch rule for the HTML representation is called *page* or after the name of the category of the resource (for example *article*).

Resource Properties

By default, each page has the following properties.

id	Unique number that identifies the resource and can be used for referring to the resource
title	Main title of the page
summary	Short, plain-text, summary
short_title	Short version of the title that is used in navigation and other places where an abbreviated title is shown
is_published	Only published pages are visible to the general public
publication_start	Date at which the page will be published
publication_end	Date at which the page will be unpublished
name	Alternative for <i>id</i> : used to uniquely identify the resource
page_path	The page's URL on the site. Defaults to <i>/page/{id}/{slug}</i> where the <i>slug</i> is derived from the title
category_id	The category of the resource (see below)

Categories

Each page belongs to exactly one category. The category a page is in determines how it is displayed.

The categories are organized in a hierarchical tree of categories and sub-categories.

For example:

- uncategorized
- text - article - news
- media - image - video - audio - document
- meta - category - predicate - keyword

Edges

See also:

m_edge (page 562)

An *edge* is a labeled connection between two resources.

The *edge* table defines these relations between resources. It does this by adding a directed edge from one *rsc* (resource) record (subject) to another (object). It also adds a reference to the *predicate*: the label of the edge.

In the admin, edges are represented in the “Page connections” sidebar panel, of the edit page of the *subject*: the resource where the edges originate. By convention, edges are said to *belong* to their subject. This is to simplify the access control: if you are allowed to edit the resource, you’re also allowed to edit its *outgoing* edges (“Page connections” in the admin), creating connections to other resources.

Predicates

See also:

m_predicate (page 570)

Edges have a label: like in *The data model by example* (page 6), *author* is a *predicate* of an edge which denotes that a certain *article* was written by a certain *person*

Just like categories, these predicates are themselves also resources: allowing you to specify metadata, give them a meaningful title, et cetera.

Each predicate has a list of valid subject categories and valid object categories (stored in the *predicate_category* table). This is used to filter the list of predicates in the admin edit page, and also to filter the list of found potential objects when making a connection. On their edit page in the admin interface, you can edit the list of valid subject and object categories for a predicate.

Examples of predicates:

- author (from article to person)
- subject (from page to keyword)
- depiction (from any page to an image)
- relation (non defined relation between two pages)
- hasdocument (page has an attached document)

Todo

document categories, predicates and resources

2.1.3 User management

Create new users

Note: This requires *mod_admin_identity* (page 353) to be enabled.

In the admin, under ‘Auth’ > ‘Users’ you can find a list of all users. Use the ‘Make a new user’ button to create a new user with a username/password identity.

2.1.4 Issues and features

If you encounter any issues in using Zotonic, or have ideas for new features, please let us know at the [Zotonic issue tracker](#). You need to have a free GitHub account to do so.

Before submitting a new issue, please search the issues list to see if the issue bug has already been reported.

Every first Monday of the month a new version of Zotonic is *released* (page 101).

Todo

Add more chapters to have a complete run-through of all user-facing aspects of Zotonic.

See <http://zotonic.com/support> for getting help and support.

The technical handbook for developers building websites with Zotonic. It guides you through all aspects of the framework.

3.1 Developer Guide

The technical handbook for developers building websites with Zotonic. It guides you through all aspects of the framework.

3.1.1 Introduction

This is the Zotonic Developer Guide. It takes you through all aspects of Zotonic so you can start building your own sites as quickly as possible.

If you are an end user rather than developer, you may want to read the User Guide instead.

An overview of Zotonic

Zotonic is a complete stack for building advanced websites quickly. It consists of:

1. a content management system (CMS)
2. a web framework
3. a web server.

You can rapidly develop CMS-based sites on top of Zotonic. If you need more flexibility, you can customise all aspects of Zotonic. The included web server means can start working with Zotonic right away, without the need for installing Apache or Nginx separately.

Zotonic is structured into modules. Each module offers a specific piece of functionality; the CMS is itself a module. There are modules for access control, social media integration, TLS, video embeds and much more.

Central to Zotonic is its uniquely flexible data model.

3.1.2 Getting Started

You have two options for running Zotonic: to get started quickly, start our Zotonic container. Or you can install Zotonic on your computer.

Docker

See also:

[Docker](#) (page 14) for more information on using Docker and how to find the password for the Zotonic status site. First [download and install Docker](#) . Then build and start Zotonic with a single command:

```
$ ./start-docker.sh
```

Docker will download and boot the container. The container will start building Zotonic. After which Zotonic can be started with:

```
bash-4.4$ bin/zotonic debug
```

Zotonic will be available on port 8443 on your machine. Open your browser and go to <https://localhost:8443/> to view the Zotonic Status page. If you wish to quit Zotonic, press twice Ctrl+C. If you wish to quite the container, press Ctrl+D at the bash prompt.

To see the configuration for the password (and other configurations) for the zotonic status site, type:

```
bash-4.4$ bin/zotonic config
```

The password is the *password* entry, the username is always `wwwadmin`.

You can now move on to [creating your first site](#) (page 17).

Cloud-Init

A [cloud-init](#) file for [Ubuntu](#) is supplied, this has been tested with Ubuntu 20.04.

This file can be used to install a VPS by providers that support cloud-init. [Hetzner](#), [Microsoft Azure](#) and [TransIP](#) are such providers.

After the cloud-init is done with its installation a new server is up and running on port 80 and 443. It will be using a self-signed certificate, located in `/home/zotonic/.config/zotonic/security/self-signed/` (on macOS the directory is `$HOME/Library/Application Support/zotonic/security/`).

The `wwwadmin` password for the zotonic status site can be found after logging in to your server:

```
root:~# sudo su - zotonic
zotonic:~$ cd zotonic
zotonic:~/zotonic$ bin/zotonic config

Zotonic config for 'zotonic@yourvps':
=====

zotonic:
  environment: production
  zotonic_apps: /home/zotonic/zotonic/apps_user
  security_dir: /home/zotonic/.config/zotonic/security
  password: wXuqsZkC4qp8jlAZHyO3
  timezone: UTC
  ...
```

The server can be stopped and started using the command line:


```
zotonic:~/zotonic$ bin/zotonic stop
Stopping zotonic 'zotonic@yourvps' .... OK
zotonic:~/zotonic$ bin/zotonic start
Waiting for zotonic: . OK
```

Installation

See also:

a more extensive discussion of *all requirements* (page 622)

If you don't like Docker, or you like to do things yourself, you can always install Zotonic on your computer yourself.

Preparation

First prepare your system for running Zotonic. Zotonic needs:

- Erlang/OTP 23 or higher
- PostgreSQL 9.5 or higher
- ImageMagick 6.5 or higher for image resizing
- Git for pulling in external dependencies
- C++ compiler (gcc) for erl_exec and other dependencies
- FFmpeg if you want to use video

Ubuntu / Debian

We recommend you install Erlang from the Erlang solutions website:

<https://www.erlang-solutions.com/downloads/>

The other requirements are easily fetched with apt:

```
sudo apt-get install gcc g++ build-essential git imagemagick postgresql ffmpeg
```

macOS

Install [Homebrew](#), then run:

```
$ brew install erlang git imagemagick postgresql ffmpeg
```

FreeBSD

Erlang and its dependencies can be installed with pkg:

```
# pkg install sudo zip wget bash gmake curl git gcc erlang
```

Also install ImageMagick and PostgreSQL, at the time of writing the commands below can be used, they should be updated with the newest available version:

```
# pkg install ImageMagick7-nox11
# pkg install postgresql10-server
# pkg install ffmpeg
```

Windows

Currently, Zotonic is not officially supported on the Windows platform. However, the main dependencies Erlang, PostgreSQL and ImageMagick do work on Windows, so, if you're adventurous, it should be possible to get it running.

It is advised to use Docker or the Linux subsystem for Windows.

Getting Zotonic

Download the latest Zotonic release ZIP file from the [GitHub releases page](#). For instance:

```
$ wget https://github.com/zotonic/zotonic/archive/|release|.zip
```

Then unzip the file and rename the directory:

```
$ unzip |release|.zip
$ mv zotonic-|release| zotonic
```

Alternatively, clone the latest development version using Git:

```
$ git clone https://github.com/zotonic/zotonic.git
```

You then need to compile the Zotonic sources:

```
$ cd zotonic
$ make
```

Then start Zotonic in debug mode:

```
$ bin/zotonic debug
```

Now point your browser to: <https://localhost:8443/>. You should see a welcome message, 'Powered by Zotonic'. This is the so-called *status website* (page 18). So far, so good! Now it's time to *create your first site* (page 17).

Next steps

- *Create your first site* (page 17).
- Log in to the *status site* (page 18).
- If something goes wrong, read the *troubleshooting reference* (page 96).
- Read more about Zotonic *configuration* (page 624).

3.1.3 Docker

We offer the Docker image [zotonic/zotonic-dev](#) which contains build tools and Erlang.

It can be used for development work on Zotonic itself, where you'll mount your Zotonic directory as a volume in the container.

To use this image, first [download and install Docker](#).

zotonic-dev

You can use the [zotonic/zotonic-dev](#) image for doing development work on Zotonic and Zotonic sites.

Start the container from your local Zotonic clone:

```
$ git clone https://github.com/zotonic/zotonic.git
```

Start the container:

```
$ cd zotonic
$ ./start-docker.sh
```

This uses Docker Compose to start the Zotonic container and a PostgreSQL container.

On first start Zotonic will be compiled and all dependencies are fetched. This can take quite some time if you are not running on Linux (Docker's file i/o is quite inefficient on non-Linux platforms).

A shell prompt will appear after the build is complete. Now Zotonic can be started:

```
bash-4.4$ bin/zotonic debug
```

You can stop Zotonic (Erlang) by typing Ctrl+C twice.

The configuration, including the Zotonic status site password, can be found with:

```
bash-4.4$ bin/zotonic config
```

Zotonic's port 8443 and 8000 are exposed as local ports. You can view the [Zotonic status page](#) (page 18) at `https://localhost:8443/`. You can log in using the username `wwwadmin` and the password from the config.

All configurations, logs and site data used in the container are stored in the `docker-data` directory.

Zotonic is running with a self-signed certificate. The certificate can be found in `docker-data/security/self-signed/`.

You can also run other commands in the container, such as running the tests:

```
$ bin/zotonic runtests
```

Any changes you make in the Zotonic source files will be propagated to the container and *automatically compiled* (page 366).

You can stop the container using Ctrl+D at the Bash shell prompt.

Directories:

- `docker-data/config` The Zotonic and Erlang configuration files
- `docker-data/security` Certificates used by Zotonic and sites
- `docker-data/logs` All log files
- `docker-data/data` Site data and mnesia files
- `_build` All compiled files

3.1.4 Directory structure

Zotonic is a set of regular OTP applications. These can be found in the repository's `apps/` directory:

```
zotonic/
  _build
  _checkouts
  apps/
    zotonic_core/
    zotonic_mod_acl_user_groups/
    zotonic_mod...
    zotonic_site_status
    zotonic_site_testsandbox
```

```
...
apps_user
mysite
...
bin/
doc/
docker/
priv/
```

zotonic/

The main directory contains the startup script “start.sh” and the makefile “Makefile” to build the system.

zotonic/_build/

The Rebar3 build directory.

zotonic/_checkouts

Place here checkouts of dependencies that are in development and should be used instead of the dependencies in `rebar.lock` or as mentioned in the `rebar.config` files of all other apps.

zotonic/bin/

Shell scripts for the Zotonic *Command-line* (page 633) and scripts for maintaining the translation `.po` files.

zotonic/doc/

Documentation sources files.

zotonic/apps/zotonic_core/include/

The main zotonic include files.

zotonic/apps/zotonic_mod.../

All core modules of Zotonic. See the modules documentation for more information.

zotonic/priv/

The priv directory is the place where all non core files statis assets are placed.

zotonic/apps/zotonic_core/src/

The Erlang source for the core Zotonic system.

zotonic/apps/zotonic_core/src/behaviours/

Currently there are two behaviours defined. One for `scomps` and one for `models`.

zotonic/apps/zotonic_core/src/db/

The database drivers. Currently, only PostgreSQL is supported as DBMS. As the modules and models sometimes use database specific queries (especially for full text search) it will not be easy to substitute an alternative database. Read this article why Zotonic only supports PostgreSQL.

zotonic/apps/zotonic_core/src/i18n/

Internationalization support.

zotonic/apps/zotonic_core/src/install/

The database model and basic installation for sites. The files here are used when an empty database is found and needs to be installed.

zotonic/apps/zotonic_core/src/models/

The data model access files. Implements internal APIs for the different data models for use in Erlang modules and templates. Examples of datamodels are `m_rsc`, `m_config` and `m_category`.

zotonic/apps/zotonic_core/src/support/

All base Zotonic source code. Here you will find the source code for site supervisors, module supervisors, image resize server, context routines, and much more.

zotonic/apps/zotonic_core/test/

Contains the EUnit tests for Zotonic.

zotonic/apps/zotonic_site_status

The Zotonic *status site* (page 18).

zotonic/apps/zotonic_site_testsandbox

The Zotonic testsandbox site that tests are run against.

zotonic/apps_user

This directory contains user-modifiable source code which runs in Zotonic, namely user-defined sites, modules and other Erlang/OTP applications.

The `apps_user` directory is the default location of the `ZOTONIC_APPS` environment variable. See *Useful environment variables* (page 94).

3.1.5 Sites

Zotonic has the capability of serving more than one site at a time. You can have multiple sites enabled, each with its own set of templates, database and dispatch rules. Each site has its own hostname.

Create a site

Note: If anything goes wrong, see the *Installation* (page 96).

First, prepare the database. In your terminal, connect to PostgreSQL:

```
$ sudo -u postgres psql (enter your OS password)
```

And create a database user for Zotonic. You may want to change the password:

```
postgres=# CREATE USER zotonic WITH PASSWORD 'zotonic';
```

Now, either give this user create rights to have Zotonic automatically create the database for you (recommended):

```
postgres=# ALTER USER zotonic CREATEDB;
```

Or create the site database manually:

```
postgres=# CREATE DATABASE zotonic WITH OWNER = zotonic ENCODING = 'UTF8';
postgres=# GRANT ALL ON DATABASE zotonic TO zotonic;
```

And quit postgres:

```
postgres=# \q
```

Now that there is a database Zotonic can be started. We do this in debug mode so that all console output is visible:

```
$ bin/zotonic debug
```

In a new terminal window, edit your `/etc/hosts` file, adding an entry for `yoursite.test` (the site hostname) to point at your local host:

```
127.0.0.1    yoursite.test
```

Note: Zotonic has to be running for the `addsite` command to succeed.

Create a new Zotonic site, based on the ‘blog’ skeleton site:

```
$ bin/zotonic addsite -s blog yoursite
```

Finally, point your browser to <https://yoursite.test:8443> to see your new site. The browser will ask to accept a self-signed certificate. Zotonic generates a self-signed certificate for every site. These are stored in `~/ .config/zotonic/security` (on macOS `~/Library/Application Support/zotonic/security`).

You can log into the admin at <https://yoursite.test:8443/admin> using the username `admin` with the password that you can find in your site’s configuration. Use for this the command:

```
$ bin/zotonic siteconfig yoursite
```

The configuration is stored in the file `apps_user/yoursite/priv/zotonic_site.config` in the *zotonic user directory*.

You can stop Zotonic by typing twice `Ctrl-C` at the Erlang command prompt.

If you want to start Zotonic in the background then use:

```
$ bin/zotonic start
```

This can be stopped with:

```
$ bin/zotonic stop
```

Anatomy of a site

A Zotonic site is a folder which lives in the *zotonic user directory* and contains at least:

- a `priv/zotonic_site.config` file: sets the site’s hostname and other parameters
- a `src/sitename.erl` file: initialises the site.
- a `src/sitename.app.src` file: an OTP app source file

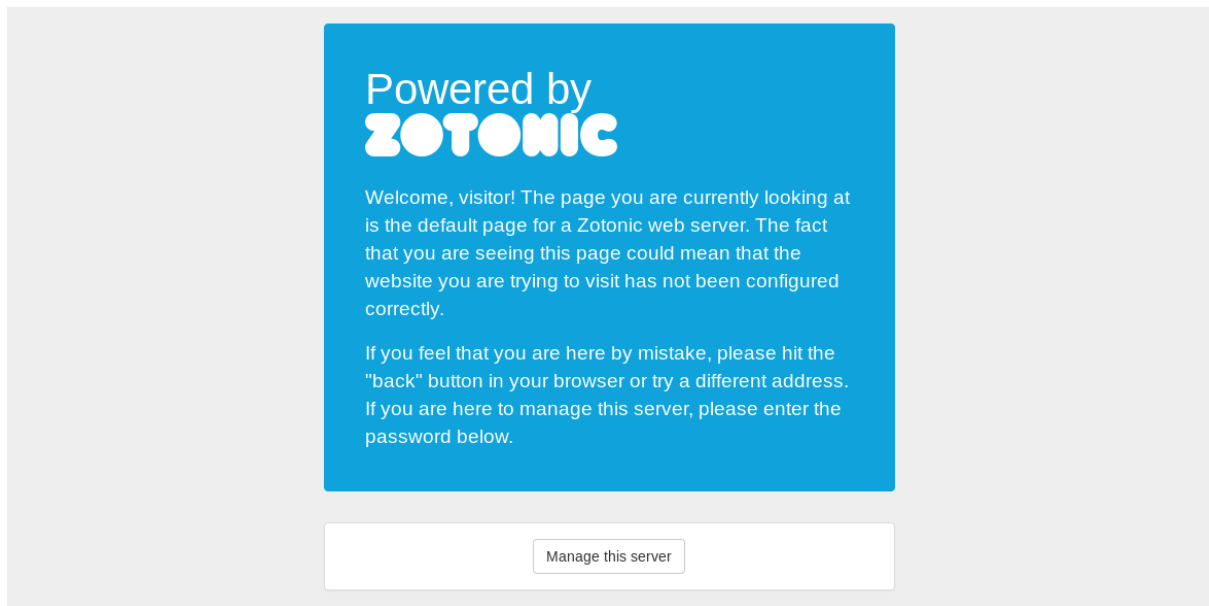
In fact, a site is a special type of *module* (page 59). Like modules, sites usually contain additional resources such as *templates* (page 28), *dispatch rules* (page 22) and *data* (page 64) . Unlike modules, however, sites have their own hostname and database connection.

Next steps

- Consult the reference for all site *configuration parameters* (page 626).
- If something goes wrong, consult the *troubleshooting reference* (page 97).

3.1.6 The Status site

The Zotonic “status” site is the first thing you see once you have installed Zotonic, when you do not have any sites configured yet. This is what it looks like:



This site is also the *fallback* site for Zotonic.

Since Zotonic supports virtual hosting, it uses the `HTTP Host :` parameter to see which site should be served at which URL. If it does not find a `Host :` header, or if the host header does not correspond to any known Zotonic site, it shows the `zotonic_status` site instead.

Logging in

Upon first visit, the site shows a friendly message, which tells visitors that the site they are looking at has probably not been configured correctly yet.

Pressing the `Manage this server` button will bring up the login dialog.

The username for the status site is `wwwadmin`. The password is automatically generated and stored in `~/.config/zotonic/config/[release]/zotonic.config`, where `[release]` is the Zotonic release number, like `1.0`.

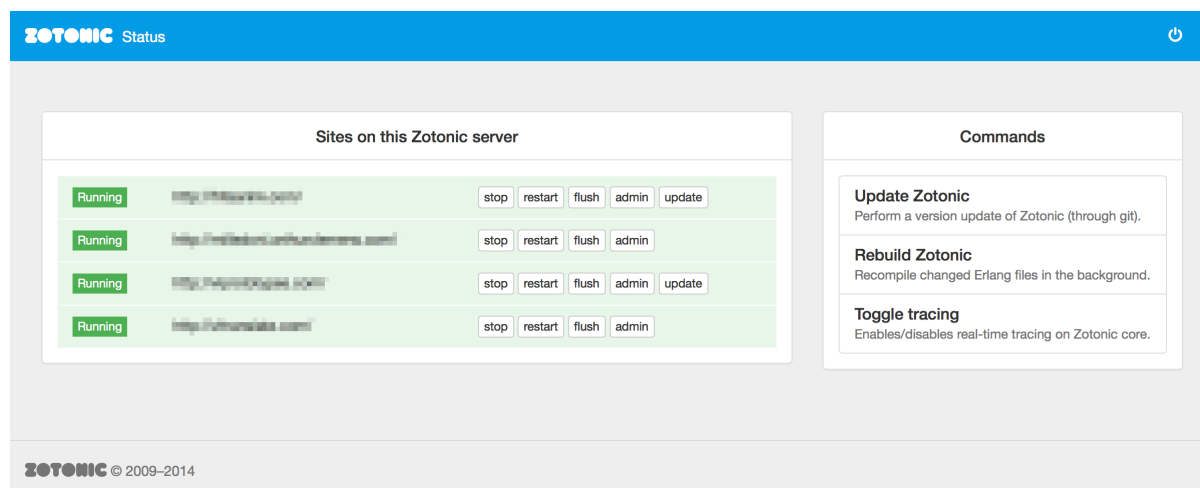
You can see the password by executing:

```
bin/zotonic config
```

You can see the location of the config files with:

```
bin/zotonic configfiles
```

When logged in to the Zotonic status site, you can manage the running sites on the system: starting, stopping and upgrading them.



The “update” buttons only appear when the site (or Zotonic itself) is under Mercurial or Git revision control. These buttons do a “pull” from the repository and then rebuild the system.

Getting the global sites status

The Zotonic status site exposes an API service to check whether all Zotonic sites are running:

```
curl -k 'https://127.0.0.1:8443/api/model/zotonic_status/get/check'
```

The option `-k` was used because the status site is using a self-signed certificate.

If all sites are running, it returns:

```
{"result": "ok", "status": "ok"}
```

If one or more sites are failing then it returns:

```
{"error": "fail", "message": "Not all sites are running.", "status": "error"}
```

"Running" means that a site's status is not "retrying" or "failed"; it ignores sites that are manually stopped or disabled.

This API service can be plugged in to a service like <https://www.pingdom.com/> to monitor the availability of all hosted sites at once.

Restarting sites

Alternatively, from the *Zotonic shell* (page 79):

```
z_sites_manager:restart(yoursitename).
```

3.1.7 Controllers

Controllers are the Erlang modules which decide what happens when a browser requests a page. Zotonic looks at the *dispatch rules* that match the requested URL, and if a dispatch rule matches, the controller that is named in the dispatch rule is used to handle the request.

Anatomy of a controller

Say we have the following dispatch rule, handling `https://localhost/example`:


```
{example_url, [ "example" ], controller_example, []},
```

When hitting `/example`, the `controller_example` controller will be initialized and callback functions on the controller will be called, according to the HTTP protocol flow.

Controllers are pretty self-documenting, thanks to the names of the cowmachine callback functions. For instance, when you define a function `resource_exists/1`, it will be called to decide if the page should return a 404.

The simplest controller to serve HTML:

```
-module(controller_example).

-export([ process/4 ]).

process(_Method, _AcceptedCT, _ProvidedCT, Context) ->
    {<<"<h1>Hello</h1>">>, Context}.
```

Rendering templates

To return the rendered output of a template file in the module's `priv/templates` directory, use `z_template:render_to_iolist/3`:

```
-module(controller_example).

-export([ process/4 ])

process(_Method, _AcceptedCT, _ProvidedCT, Context) ->
    % foo and bam will be available as template variables in mytemplate.tpl.
    Vars = [
        {foo, <<"bar">>},
        {bam, 1234}
    ],
    z_template:render_to_iolist("mytemplate.tpl", Vars, Context).
```

If you need more examples, *mod_base* (page 360) contains many controllers, implementing basic HTTP interaction but also redirects, websockets, et cetera. The *Controllers* (page 435) page lists all available controllers in the Zotonic core.

The [Cowmachine documentation](#) is helpful for understanding controllers.

Differences between Cowmachine and Basho's Webmachine

Zotonic's fork of Webmachine has been named `cowmachine` and lives in its separate repository at <https://github.com/zotonic/cowmachine>).

The main differences with Basho's Webmachine are:

- Uses Cowboy instead of MochiWeb
- All callbacks have a single handler
- Pluggable dispatch handler
- Support for the HTTP Upgrade: header
- Optional caching of controller callbacks results
- Dispatch handler can redirect requests
- Use of process dictionary has been removed
- `webmachine_request` is now a normal (not parametrized) module
- Extra logging

- ping and init callbacks are removed

Together, this is a significant simplification and speed boost.

3.1.8 Dispatch rules

Dispatch rules route incoming requests to *controllers* (page 20).

A dispatch rule contains a pattern that is matched against the URL of an incoming request. When a URL is requested by the web browser, the dispatcher looks at the URL and matches it against all dispatch rules. Based on the match, it calls a to handle the request.

Dispatch rules are also used for the reverse action of generating request URLs in your templates and Erlang code. As long as you use dispatch rules in your application, you don't have to hard-code request URLs.

See also:

mod_custom_redirect (page 364), *mod_base* (page 360)

Defining dispatch rules

Dispatch rules are defined in a dispatch file in the `priv/dispatch` directory of a module or site. This dispatch file contains a list of dispatch rules. For example:

Listing 3.1: sites/yoursite/priv/dispatch

```
%% Example dispatch rules
[
  {home,      [],                      controller_page, [ {template, "home.
↳tpl"}, {id, page_home} ]},
  {features,  ["features"],           controller_page, [ {template,
↳"features.tpl"}, {id, page_features} ]},
  {collection,["collection", id, slug], controller_page, [ {template,
↳"collection.tpl"} ]},
  {category,  ["category", id, slug],   controller_page, [ {template,
↳"category.tpl"} ]},
  {documentation, ["documentation", id, slug], controller_page, [ {template,
↳"documentation.tpl"} ]}
].
```

Each dispatch rule is a tuple that contains four elements:

1. The dispatch rule name (e.g. `collection`).
2. The match pattern for the request URL's path (e.g., `collection/{id}/{slug}`).
3. The controller that will handle the request (*controller_page* (page 447) above).
4. An optional list of controller arguments (in our example above, the template name to be rendered).

Let's look at each of these elements in turn.

1. Dispatch rule name

Each dispatch rule must have a name. This name is used for *generating URLs* (page 24).

Dispatch rules do not have to be unique: if multiple rules with the same name exist, the dispatcher will look at the first rule that matches both the name and the optional extra arguments that were given.

2. Match pattern

In your match patterns, use:

- strings to define fixed parts that must match the URL (e.g. `collection/`)
- atoms to define dispatch rule arguments (e.g. `id` and `slug`) that will bind to the URL part at that position
- the special atom `'*'` to bind to the remaining part of the URL. This must be the last element of the path.

So a request to `http://yoursite/collection/123/nice-slug-you-got-there` matches the `collection` dispatch rule above, with the following bindings:

argument	value
<code>id</code>	<code>123</code>
<code>slug</code>	<code>nice-slug-you-got-there</code>

You can access the values with the `q` template variable:

```
{{ q.id }}
{{ q.slug }}
```

To view all variables, use the *debug tag* (page 536).

Order of rules

Dispatch rules are processed from top to bottom, so make sure the more specific rules are above more general ones.

Match pattern constraints with regular expressions

Some developers need very particular control of dispatch in order for their applications to function as they want them to.

Say you want to only accept numerical arguments as an `id` in:

```
{foo, ["foo", id], controller_foo, []}
```

The you can use a dispatch rule with a regular expression test:

```
{foo, ["foo", {id, "[0-9]+$"}], controller_foo, []}
```

or, you can specify some extra options:

```
{foo, ["foo", {id, "1?2?", [notEmpty]}], controller_foo, []}
```

In this case, the `id` must contain a 1 or a 2, amongst any other characters.

3. Controller

The third element of the dispatch rule is the controller that will handle the incoming request. This can be one of Zotonic's *built-in controllers* (page 435) or a *custom controller* (page 20).

4. Controller arguments

The last element is an optional property list that will be passed as arguments to the controller. Refer to the *documentation for each controller* (page 435) for available arguments.

Generating URLs

In templates

To generate URLs in templates, use the *url tag* (page 532) and pass the dispatch rule name:

```
{% url home %}
```

And with dispatch rule arguments:

```
{% url collection id=123 slug="nice-slug-you-got-there" %}
```

which gives: /collection/123/nice-slug-you-got-there.

Any arguments that are not defined in the dispatch rule are appended as query string parameters, so:

```
{% url features var=1 x="hello" %}
```

will result in the URL /features?var=1&x=hello.

In Erlang

To generate URLs in your Erlang code, use `z_dispatcher`:

```
z_dispatcher:url_for(features, Context).  
  
%% and with dispatch rule arguments:  
z_dispatcher:url_for(collection, [{id, 123}, {slug, "nice-slug-you-got-there"}],  
    ↪Context).
```

Dispatcher details

Organizing dispatch files

A module or site can have multiple dispatch files, and they can have any filename as long as you don't use the extension `.erl`.

The module indexer will load all dispatch files. They can be reloaded with the “rescan” button in the admin modules page. Illegal dispatch files are skipped, showing an error message in the Zotonic shell.

When your dispatch rules don't work, check first if there are any typos, then check if your dispatch rules are not overruled by a module that loads earlier. Modules are loaded on priority first, then on module name.

URL rewriting

Before URLs are matched, they can be rewritten to match something else. This is done using the *dispatch_rewrite notification* (page 589). This allows one to set extra context variables or change the (internal) URL so different dispatch rules get triggered.

An application of URL rewriting allows you to set the Zotonic language based on the domain that is being requested on your site. To set up domain-based language detection using the following code snippet:

```
observe_dispatch_rewrite(#dispatch_rewrite{host=Host}, {Parts, Args}, _Context) ->  
    Language = case Host of  
        <<"example.nl">> -> nl;  
        <<"example.de">> -> de;  
        _ -> en    %% default language
```

```
end,
{Parts, [{z_language, Language} | Args]}.
```

This leaves the request URI intact (the `Parts` variable), but injects the `z_language` variable into the request context, overriding the language selection.

For this setup to work, this requires you to have the `{redirect, false}` option in your site, and the appropriate `hostalias` directives for each host. See *Anatomy of a site* (page 18) for more details.

Dispatch rule options

There is one dispatch rule option that is valid for all dispatch rules: `allow_frame`

Normally pages are not allowed to be displayed inside a frame on another domain than the domain of served frame.

This is done by setting the HTTP header: `X-Frame-Options: sameorigin`

If this option is given then the `X-Frame-Options` header is omitted and the dispatch rule is allowed to be displayed inside a frame on any website.

Unmatched hosts/domains

First, the dispatcher finds the site that matches the HTTP host in the request. If no site can be found then the dispatcher will first check all enabled sites with a *dispatch_host notification* (page 588) to see if any site has a known redirect.

If this fails then the dispatcher will select a default site (usually the *status site* (page 18)) to handle the request.

If no site is running at all then a bare bones '404 Not Found' page will be shown.

See *mod_custom_redirect* (page 364) for redirecting unknown domains.

Unmatched paths

If the dispatcher has found a site to handle the request, it looks for a dispatch rule that matches the request path. If no such rule can be found, a *dispatch notification* (page 588) is triggered.

The module *mod_base* (page 360) will check the request path against the `page_path` property of all resources. After that the module *mod_custom_redirect* (page 364) will check the configured redirect locations.

Dispatch rule BNF

A dispatch rule is built up as follows:

```
{RuleName, UrlPattern, ControllerModule, ControllerArgs}
RuleName = atom()
PathSpec = [PathSegmentSpec]
PathSegmentSpec = StaticMatch | Wildcard | Variable
StaticMatch = string()
Wildcard = '*'
PathVariable = atom() | {atom(), RegExp} | {atom{}, RegExp, ReOptions}
RegExp = string()
ReOptions = [term()]
ResourceModule = atom()
ResourceArgs = [{Key, Value}]
```

All *PathVariables* in the matching rule are made available to the resource through `z_context`. The *ResourceArgs* proplist is passed to `ControllerModule: init/1`.

PathVariables are part of the request-scope configuration of *ControllerModule*. Things like the ID, name or category of a page being requested can be gathered effectively here. Judicious use of *PathVariables* can substantially reduce the number of dispatch rules while making them easier to read.

ControllerArgs is the rule-scope configuration of *ControllerModule*. It makes it possible to reuse a well-designed resource module in many dispatch rules with different needs. *ControllerArgs* is effective for establishing implementation details like the template to be used, whether or not to do caching and where to load static resources from.

3.1.9 Resources

Resources are Zotonic's main data unit. You may want to familiarise yourself with the Zotonic *data model* (page 5) in the User Guide.

Resource properties

Resources are very flexible data units: they can have any property that the developer needs them to have. However, by default, Zotonic's admin is designed to edit a common set of properties. The *m_rsc model* (page 571) is used to edit resources and display them in templates.

Categories

See also:

m_category (page 558) model reference

Every resource belongs to a single category.

There is no real distinction between rsc records that are a person, a news item, a video or something else. The only difference is the *category* of the rsc record, which can easily be changed. Even categories and predicates themselves are represented as rsc records and can, subsequently, have their own page on the web site.

Categories are organized in a hierarchical fashion, and used to organize the resources into meaningful groups. Zotonic has a standard set of categories (see *The Zotonic data model* (page 5)), but it is very usual to define your own in your own site, resulting in a custom *domain model*.

In the database, categories are stored in an extra metadata table, *category*, which defines the hierarchy of categories using the *Nested Set model*. The tree is strictly hierarchical: Every category has at most a single parent category, and every resource belongs to exactly one category. That a resource can't belong to more than a single category is done to maintain the datamodel's simplicity and speed of the searches in the system.

Since in Zotonic, *everything is a resource*, categories *themselves* are also resources, namely, resources of the category *category*. This allows the category to be titled and described, just like other resources. The category table only describes the nested hierarchy of the categories. All other properties of a category are defined by its rsc record.

Medium

See also:

m_media (page 568)

Medium management is described in full in *Media* (page 32). Media metadata is stored in a separate table, called *medium*, since one media is a medium. When a resource contains a medium, this table holds a record describing it. Amongst others, it stores its mime type, width, height and file size.

Besides the `medium` table, a `medium_deleted` table exists. When a medium is deleted then any files referenced by that medium will be added to this table. Zotonic periodically checks this table to delete files that are no longer referenced by any media.

Blocks

Blocks are a specific feature in a resource. The `blocks` property of a resource is a list of blocks which can be dynamically added and removed from the resource in the admin edit page. Each module can define their own blocks, which consist of an edit template and a view template.

The survey module uses the blocks feature to allow you to dynamically create a list of questions which a user has to answer.

Todo

Fix blocks documentation

Manipulating resources

How resources are stored

Each *resource* on a site is stored in the `rsc` database table. The resource's properties are stored in two ways.

- The *core properties* are persisted in separate columns. They include id, name, category, modification date, path, publication period. These properties are used for searching, filtering and sorting resources. As they can have unique or foreign key constraints, they help in preserving data sanity.
- All *other properties* are serialized together into one binary blob column named `props`. This includes any custom properties that you set on the resource. These serialized properties cannot be used for finding or sorting data, but only for later retrieval.

Storing properties in a serialized form is a flexible approach. You can save any property on a resource without having to make changes to your database schema.

Changing resources

See also:

m_rsc model reference (page 571)

Imagine you wish to store whether resources are liked by users. No need to change the database schema, define the property or whatsoever. Just update the resource and set a custom `is_liked` property (using *m_rsc* (page 571)):

```
m_rsc:update(123, #{ <<"is_liked">> => true }, Context).
```

`is_liked=true` will now be stored in the database for resource 123, so you can retrieve it like you would any other property:

```
?DEBUG(m_rsc:p(123, is_liked, Context)).
%% prints: true
```

Or, in a template:

```
{{ id.is_liked }}
```

which is equivalent to:

```
{{ m.rsc[id].is_liked }}
```

To remove the property, just store it as `undefined`:

```
m_rsc:update(123, #{ <<"is_liked">> => undefined }, Context).
```

This flexible approach is fine for custom properties that you only want to retrieve and display. However, if you need to *find* all liked resources, you need to define `is_liked` as a pivot column (see below).

Pivots

Pivot columns

If you want to *filter* or *sort* on any custom defined property, you need to store that property in a separate database column using a *custom pivot* (page 321). If you want to *find* resources based on text values in custom properties, you can change the texts that are pivoted with *pivot templates* (page 324).

The pivot queue

When the version number or modification date of a resource is updated then its id is added to the *pivot queue*. Zotonic has a pivot process running in the background which looks at this queue and for each queued resource, extract all texts and some other information from the record, filling the pivot columns of the rsc record. The pivot columns are used for searching, they contain amongst others the full text index.

The `rsc_pivot_queue` table is used to hold the queue of resource ids that are waiting to be pivoted.

The `pivot_task_queue` holds a second queue for more generic task processing: it holds references to functions which need to be called in the background.

Identities

See also:

m_identity (page 566).

An rsc record can become a user by adding the user's credentials to this table. A single user can have multiple kinds of credentials, think of his/her username, openid uri etc. A user isn't necessarily a person.

Deleted resources

See also:

m_rsc_gone (page 577).

Whenever a resource is deleted, an entry is added to the `rsc_gone` table. The page and id controllers will serve a *410 Gone* when a deleted resource is requested.

3.1.10 Templates

See also:

Media (page 32) on how to include images and other media in your templates.

See also:

Wires (page 56) on using JavaScript to add interaction to your templates.

Templates are text files marked up using the Zotonic template language. Zotonic interprets that mark-up to dynamically generate HTML pages. Zotonic's template syntax is very similar to the Django Template Language (DTL).

Variables

Variables are surrounded by `{{` and `}}` (double braces):

```
Hello, I'm {{ first_name }} {{ last_name }}.
```

When rendering this template, you need to pass the variables to it. If you pass James for `first_name` and Bond for `last_name`, the template renders to:

```
Hello, I'm James Bond.
```

Instead of strings, variables can also be objects that contain attributes. To access the attributes, use dot notation:

```
{{ article.title }} was created by {{ article.author.last_name }}
```

The variables that you add to your templates get their values from three places:

- they can be *passed from controllers* (page 21) when rendering the template;
- they can come from *models* (page 30);
- or they can be *global variables* (page 554).

To print the variables in your template for debugging, you can use the *debug* (page 536) tag:

```
{% debug %}
```

Filters

Optionally, you can pipe template variables through filters. Filters transform the value before rendering it. To apply a filter, you append `|` plus the filter's name to the variable:

```
{{ first_name|lower }} {{ last_name|upper }}
```

Renders:

```
james BOND
```

There's a *reference of built-in filters* (page 454). If you need to, you can also *create your own filters* (page 314).

Tags

With tags, you can add logic and flexibility to your templates. Tags are surrounded by `{%` and `%}`, like this:

```
{% tag %}
```

Some tags have arguments:

```
{% tagname id width=200 height=200 crop %}
```

And some tags are *block tags*, which require both an opening and a closing tag. The name of the closing tag is always `end` plus the name of the opening tag:

```
{% if article.is_published %}
  There you go:: {{ article.title }}
{% else %}
  Sorry, the article hasn't been published yet!
{% endif %}
```

See also the reference *all tags* (page 511). You can also *create your own tags* (page 322).

Models

A template model provides data to templates through the syntax: `m.modelname.property`. For example:

```
{# Get the site's title #}
{{ m.site.title }}

{# Fetch the title of the page with name page_home #}
{{ m.rsc.page_home.title }}

{# Fetch the title of the page whose id is the integer 1 #}
{{ m.rsc[1].title }}

{# Perform a search on all persons #}
{% for p in m.search[query cat='person']] %}{{ p.title }}{% endfor %}

{# Fetch the title of the page whose id is the template variable id #}
{{ m.rsc[id].title }}
```

You'll find that you need `m.rsc[id]` the most, so there's a *recommended shortcut* (page 339) for that:

```
{{ id.title }}
```

See the reference for a list of *all models* (page 555). You can also add *your own models* (page 315).

Template names

All templates are stored in the `priv/templates/` directory of *sites* (page 17) and *modules* (page 59). They have the extension `.tpl`. Templates are referred to by their filename, including their subdirectory name within `priv/templates/` (if any). So if you have these two templates:

- `modules/mod_example/priv/templates/foobar.tpl`
- `modules/mod_example/priv/templates/email/email_base.tpl`

you refer to them as:

- `foobar.tpl`
- `email/email_base.tpl`

The module name itself (`mod_example`) is never part of the template name, because all templates are grouped together. This allows you to override Zotonic's templates.

Overriding templates

If you want to override a template, you create a template with the same name in your site (or module). So what if the `email/email_base.tpl` template from `mod_example` mentioned above is not to your liking? Just create a `email/email_base.tpl` file in your own site: `sites/yoursite/priv/templates/email/email_base.tpl`.

So if multiple templates can have the same name, how does Zotonic know *which* template to render: the one from `mod_example` or the one from `yoursite`? This depends on the *module priority* (page 63). Usually sites have a higher priority than modules, so `yoursite` wins and can serve its template.

If you want to *add* your template instead of overriding, you can use the *all include* (page 512) and *all catinclude* (page 511) tags.

Page templates

Most of your templates will be *page* (page 7) templates. When showing a page, Zotonic's *page controller* (page 447) looks up the appropriate template in order of specificity and renders the first template it finds (assuming the name of the page `page_name`):

1. `page.name.page_name.tpl` (*unique name* (page 571))
2. `page.category.tpl` (*category* (page 26))
3. `page.tpl` (fallback)

So if you have a page in the category 'article' (which is a sub-category or 'text') and that page has a unique name 'my_text_page', Zotonic will look for the following templates:

1. `page.name.my_text_page.tpl` (unique name)
2. `page.article.tpl` (category)
3. `page.text.tpl` (category)
4. `page.tpl` (fallback)

The same lookup mechanism is used for the *catinclude* (page 515) tag.

Template structure

Now you know how Zotonic decides which template to render for a page, let's go into how you can render templates yourself. Usually, you spread template logic for a page over multiple template files. This allows you to re-use these files in other templates.

Including templates

You can include other templates using the *include* (page 526) tag:

```
This is the main template. To include another template:

{% include "other-template.tpl" %}
```

Zotonic will replace the include tag with the output of `other-template.tpl`.

Variants of the include tag are *catinclude* (page 515), *all include* (page 512) and *all catinclude* (page 511): following the *lookup mechanism* (page 30) described above, Zotonic will find the best template based on the page and module priority.

Inheritance

To improve template re-use, it is common to inherit from a base template. A simple base template might look like this:

Listing 3.2: templates/base.tpl

```
<!DOCTYPE html>
<html>
  <head>
    <title>Zotonic{% block title %}{% endblock %}</title>
  </head>

  <body>
    {% block content %}This is default content{% endblock %}
  </body>
</html>
```

You can then extend multiple templates from this single base template using the *extends* (page 516) tag. The base template contains *block* (page 513) tags that can be overridden in child templates:

Listing 3.3: templates/page.tpl

```
{% extends "base.tpl" %}

{% block title %}This is the page title{% endblock %}

{% block content %}
    This will override the content block from base.tpl
{% endblock %}
```

Using the *inherit* (page 527) and *overrides* (page 530) tags, you adapt the inheritance to your needs.

3.1.11 Media

Resources can have media resources attached to them. Resources and their media (images, video and audio) are connected through ‘depiction’ *edges*. Additionally, resources can be *embedded* in some HTML text, such as the resource’s body text.

Adding media

Media is added to resources with the ‘Add media item’ button in the admin. You can embed resources by using the admin’s rich text editor.

Programmatically, use the *m_media* (page 568) model to insert the media resource:

```
{ok, MediaId} = m_media:insert_url("http://some-site/images/img.jpg", Context).
```

Or, with some properties:

```
Props = [
    {title, <<"This will be the title of the media resource">>},
    {category, audio},
    {is_published, true}
],
{ok, MediaId} = m_media:insert_file("/path/to/audio.wav", Props, Context).
```

Then link the owning resource and the media resource:

```
m_edge:insert(Id, depiction, MediaId, Context).
```

In templates

To render some resource’s (id) first depiction, use the *image* (page 520) tag:

```
{% image id %}
```

You can also add inline *parameters* (page 521):

```
{% image id width=200 height=200 crop %}
```

To render embedded media, use the *show_media* (page 469) filter:

```
{{ id.body|show_media }}
```

Media classes

Instead of inline image tag parameters, you can use media classes to define image transformations. The advantage is that this image definition can then be reused amongst templates.

Create a `templates/mediaclass.config` file in your site directory:

```
[
  {"thumb", [
    {width, 200},
    {height, 200},
    crop
  ]}
].
```

This defines a media class called 'thumb', which can be used to display a 120x120 cropped square image. You then only need to refer to this media class in your image tag:

```
{% image id mediaclass="thumb" %}
```

The image URL will have a checksum embedded in it so that when the contents of the media class is changed, all images which use that media class will be regenerated to reflect the new media class.

Responsive images

To provide images in multiple [responsive sizes](#), use the `srcset` attribute:

```
%% templates/mediaclass.config
[
  {"masthead", [
    {width, 1600},
    {height, 900},
    {crop, center},
    upscale,
    {quality, 85},
    {srcset, [
      {"640w", [{quality, 50}]},
      {"1200w", []},
      {"2x", []}
    ]},
    {sizes, "100vw"}
  ]}
].
```

An `{% image id mediaclass="masthead" %}` tag in your template will output:

```
<img src='image-default.jpg'
  sizes='100vw'
  srcset='image-640.jpg 640w, image-1200.jpg 1200w, image-2400.jpg 2x'
  class="mediaclass-masthead">
```

Each `srcset` value is either a [width descriptor](#) or a pixel density descriptor.

- A width descriptor of `640w` will resize the image to a width of 640 pixels.
- A pixel density descriptor of `2x` will resize the image to 2 times the original media class width value, so $2 * 1600 = 3200$.

By default, each `srcset` image will inherit all properties from the parent media class. So, in the example above, the `640w` image will have a reduced quality value of 50 while the `1200w` image will inherit its parent's value of 85.

So you can override the automatically determined width of 3200 for the `2x` descriptor by adding:

```
{ "2x", [{width, 2000}] }
```

Raw ImageMagick options

Besides the normal image processing options, as described in *image* (page 520), it is possible to add literal ImageMagick convert commands to the mediaclass definition.

For example:

```
{magick, "-level 90%,100% +level-colors \\#FE7D18,\\#331575"}
```

(Note that you have to double any backslashes that were needed for the `convert` command line.)

This command is given *as-is* to the ImageMagick `convert` command, therefore it is best to first try it with the command-line `convert` command to find the correct options and command line escapes needed.

There are three variations: `pre_magick`, `magick`, and `post_magick`. The only difference is that the `pre_magick` is added before any other filter argument, `magick` somewhere between, and `post_magick` after the last filter.

In this way it is possible to pre- or post-process an image before or after resizing.

See <http://www.imagemagick.org/Usage/> for examples of using ImageMagick from the command line.

3.1.12 Icons

Including Zotonic icons CSS

Add the CSS file to your template:

```
{% lib
    "css/z.icons.css"
%}
```

This ensures that whenever a Zotonic icon class is used, the image will show up.

Writing Zotonic icons CSS classes

In your template HTML, use this syntax:

```
z-icon z-icon-<name>
```

For instance:

```
<span class="z-icon z-icon-off"></span>
```

Always include `z-icon`, except for the convenience class `zotonic-logo`.

If you want to provide a text label for accessibility (but visually hidden), put the text inside `` or ``:

```
<span class="z-icon z-icon-off"><em>Log off</em></span>
```

Available classes:

```
.z-icon-cross
.z-icon-cross-circle
.z-icon-drag
.z-icon-edit
.z-icon-facebook
```

```
.z-icon-google-plus
.z-icon-help
.z-icon-help-circle
.z-icon-info-circle
.z-icon-instagram
.z-icon-linkedin
.z-icon-logo-z
.z-icon-logo
.z-icon-minus
.z-icon-minus-circle
.z-icon-off
.z-icon-ok
.z-icon-ok-circle
.z-icon-plus
.z-icon-plus-circle
.z-icon-twitter
.z-icon-user
.zotonic-logo
```

Buttons

Icons-as-buttons: 2 convenience classes for commonly used buttons are:

```
.z-btn-remove
.z-btn-help
```

These buttons are styled as round buttons with size 16px. Button text inside `` or `` is hidden.

To have consistent close buttons, Bootstrap close buttons `a.close` and `button.close` are overridden by the `z-btn-remove` style.

Usage:

```
<a class="z-btn-remove"><em>Remove</em></a>
```

Social login buttons

Social login buttons are created with `z-btn-social` and show up with a brand icon at the side. Usage:

```
<a href="#" class="btn z-btn-social"><span class="z-icon z-icon-facebook"></span>
↵Login with Facebook</a>
```

NOTE: the full syntax that opens a new login window:

```
<a href="{% url logon_service service='facebook' is_connect=is_connect %}"
  class="btn z-btn-social do_popupwindow"
  data-popupwindow="height:300"
  style="color: white; background-color: #44609d">
  <span class="z-icon z-icon-facebook"></span>
  {% if is_connect %}
    { _ Connect with Facebook _ }
  {% else %}
    { _ Login with Facebook _ }
  {% endif %}
</a>
```

“popupwindow” must be included in the template:

```
{% lib
    "js/modules/z.popupwindow.js"
%}
```

Writing LESS

If you are writing frontend styles in LESS, Zotonic icons can be extended using mixins (found in `extend.less`).

NOTE: The less files have a dependency with `zotonic_mod_artwork/priv/lib/font-awesome-4`, so you need to include the path to its LESS folder when using the `lessc` command. For example:

```
lessc --include-path="../../../../zotonic_mod_artwork/priv/lib/font-awesome-4/less"
my_input.less my_output.css
```

Or for easier access, create a symlink to the *font-awesome-4* LESS folder and add the symlink to the include-path.

Extending Zotonic icons

To extend a class with a Zotonic icon class, write:

```
.extend_icon(z, @name)
```

For example:

```
.my-btn-help {
    .extend_icon(z, icon-help-circle);
}
```

This will generate the following CSS:

```
(lots-of-classes),
.my-btn-help:before {
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    display: inline-block;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
(lots-of-classes),
.my-btn-help:before {
    font-family: "Zotonic";
}
.my-btn-help:before {
    content: "\e600";
}
```

The `:before` pseudo-class can be extended to further style the icon. For instance to add a plus icon to a link:

```
.my-plus-link {
    position: relative;
    padding-left: 16px;
    .extend_icon(z, icon-plus);
    &:before {
        position: absolute;
        top: 4px;
    }
}
```



```

    left: 0;
    width: 16px;
    font-size: 13px;
  }
}

```

Extending Material Design icons

1. Enable module `mod_artwork`.
2. In LESS, add parameter 'md' to the *extend* mixin and pass the character code:

```

.btn-bookmark {
  .extend_icon(md, "\f019");
}

```

The icon (variable) characters can be found in the [icons cheatsheet](#).

Extending FontAwesome 4 icons

1. Enable module `mod_artwork`.
2. In LESS, add parameter 'fa' to the *extend* mixin:

```

.btn-bookmark {
  .extend_icon(fa, fa-var-bookmark);
}

```

The icon (variable) names can be found in `zotonic_mod_artwork/priv/lib/font-awesome-4/less/variables.less`.

3.1.13 Forms and validation

You [should validate all input data entered in forms](#). In Zotonic you create forms by writing plain HTML. You can attach one or more validators to each input element. Validators define acceptable values for the input data.

See also:

the [validate](#) (page 550) tag

See also:

the [postback](#) (page 42) validator

Let's say you have a required input field 'title'. To make sure some text is entered in it, attach the [presence](#) (page 43) validator:

```

<input type="text" id="title" name="title" />
{% validate id="title" type={presence} %}

```

The validated form field is available to Erlang code using the `z_context:get_q_validated/2` function:

```

Title = z_context:get_q_validated(<<"title">>, Context).

```

Client- and server-side

Zotonic's validators work both client- and server-side:

- JavaScript prevents the form from being submitted until the input data conforms to the validators.

- All validation is done on the server as well, which protects against users bypassing the validation checks in their browser.

Validators

Zotonic comes with some commonly needed validators:

acceptance

- Module: *mod_base* (page 360)

See also:

Forms and validation (page 37)

Check if an input value evaluates to true.

Can be used in combination with a check box that must be checked on submit.

For example:

```
<input type="checkbox" id="accept" name="accept" value="1" />
{% validate id="accept" type={acceptance} %}
```

Arguments

Argument	Description	Example
fail- ure_message	Message to be shown when the input is true. Defaults to “Must be accepted.”	failure_message="Please agree to our TOS."

confirmation

- Module: *mod_base* (page 360)

See also:

Forms and validation (page 37)

Check if two inputs are the same.

Useful when a form requires double entry of a password or e-mail address to prevent typos.

Accepts an additional parameter *name* with the name of the other input field.

For example:

```
<input type="password" id="password" name="password" value="" />
<input type="password" id="password2" name="password2" value="" />
{% validate id="password" type={confirmation match="password2"} %}
```

Arguments

Argument	Description	Example
match	The id of the input field that should have the same value.	match="field1"
fail- ure_message	Message to be shown when the two fields are unequal. Defaults to “Does not match.”	failure_message="Please retry."

custom

- Module: *mod_base* (page 360)

See also:

Forms and validation (page 37)

Support for custom client-side (JavaScript-based) validators.

This call simply adds a `z_add_validator()` JavaScript call which adds a custom validator based on the arguments given in the validator.

Example:

```
<input type="text" name="foobar" id="foobar" value="" />
{% validate id="foobar" type={custom against="window.validate_foobar"} %}
```

And then `validate_foobar` is defined as follows:

```
function validate_foobar(value, args, isSubmit, submitTrigger)
{
    // ... some logic to check the value
    return true or false;
}
```

The `args` are available if the validation is added using the LiveValidation JavaScript API.

`isSubmit` is `true` if the validation is triggered by a form submit, if it was triggered by change or focus events then it is `false`.

`submitTrigger` is the DOM tree node triggering the possible form submit.

Note that this validation does not do any server side validation. Because there is no server side validation, the value of the input element is not available via `z_context:get_q_validated/2` but only via `z_context:get_q/2`.

date

- Module: *mod_base* (page 360)

Validate input date against a given date format.

A quick guide to validating a date in Zotonic:

```
<input type="text" id="my_date" name="my_date" value="" />
{% validate id="my_date" type={date separator="-" format="b"} %}
```

This code validates when the user enters a date in the following format:

```
yyyy-mm-dd
```

Arguments

Argument	Description	Example
separator	Character used to separate date parts, such as <code>/ - \</code> . Defaults to <code>"-"</code> .	<code>separator="-"</code>
format	Date format, big endian (starting with year), little endian (starting with day) or middle endian (starting with month). Defaults to <code>"l"</code> (little).	<code>format="m"</code>

email

- Module: *mod_base* (page 360)

See also:

Forms and validation (page 37)

Check if the content of the input field is an e-mail address.

For example:

```
<input type="text" id="email" name="email" value="" />
{% validate id="email" type={email} %}
```

Arguments

Argument	Description	Example
failure_message	Message to show when the entered value is not an e-mail address. Defaults to “Incorrect E-mail”	failure_message="Please enter your e-mail address."

email_unique

- Module: *mod_admin_identity* (page 353)

See also:

m_identity (page 566), *username_unique* (page 44), *Forms and validation* (page 37)

Check if an entered e-mail address is unique, by looking in the *m_identity* (page 566) table for the *email* key:

```
<input type="text" id="email" name="email" value="" />
{% validate id="email" type={email} type={email_unique} %}
```

Optionally, an *rsd_id* parameter can be given to the validator to skip that particular id when doing the uniqueness check. This is useful when you are displaying a form in which the user is editing his own email address.

format

- Module: *mod_base* (page 360)

See also:

Forms and validation (page 37)

Regular expression test.

Checks if an input element's value matches a regular expression. There is an optional *negate* argument to validate only when the regular expression does not match.

For example, to test a dutch postal code:

```
<input type="text" id="postcode" name="postcode" value="" />
{% validate id="postcode" type={format pattern="^[0-9][0-9][0-9][0-9] +[A-Za-z][A-↵Z][a-z]$" %}
```

Another example, no digits allowed:

```
<input type="text" id="nodigit" name="nodigit" value="" />
{% validate id="nodigit" type={format pattern="[0-9]" negate} %}
```

Arguments

Argument	Description	Example
pattern	The regular expression to match against.	pattern="[0-9][a-z]+"
negate	Specify negate when you want to accept values that do not match the pattern.	negate
failure_message	Message to show when the input value does not match the pattern (or does match the pattern when the negate argument is given). Defaults to "Not valid."	failure_message="Invalid postcode"

length

- Module: *mod_base* (page 360)

See also:

Forms and validation (page 37)

Check the length of a text input.

Test if the length of the input's value is more than a minimum and/or less than a maximum length.

Arguments are *minimum* and *maximum*. You can give either or both arguments.

For example, a summary that is neither too long nor too short:

```
<textarea id="summary" name="summary"></textarea>
{% validate id="summary" type={length minimum=20 maximum=200} %}
```

Arguments

Argument	Description	Example
is	Use when the value must be a specific length.	is=4
minimum	The minimum length of the value.	minimum=4
maximum	The maximum length of the value.	maximum=10
wrong_length_message	Message for when the length is unequal to the value of the "is" argument. Defaults to "Must be . characters long."	
too_short_message	Message for when there are not enough characters entered. Defaults to "Must not be less than . characters long."	
too_long_message	Message for when there are too many characters entered. Defaults to "Must not be more than . characters long."	

name_unique

- Module: *mod_base* (page 360)

See also:

Forms and validation (page 37), *username_unique* (page 44)

A *validator* (page 37) to check whether a resource's name is unique:

```
<input type="text" id="name" name="name" value="" />
{% validate id="name" type={name_unique} %}
```

Optionally, pass an *id* parameter to exclude that particular id when testing for uniqueness. This is useful when you want to exclude the name of the resource currently being edited:

```
<input type="text" id="name" name="name" value="" />
{% validate id="name" type={name_unique id=id} %}
```

You can also pass a `failure_message`:

```
<input type="text" id="name" name="name" value="" />
{% validate id="name" type={name_unique id=id failure_message=_ "Eek! Already used!"} %}
```

numericality

- Module: *mod_base* (page 360)

See also:

Forms and validation (page 37)

Numerical input and range check.

Checks if the input is a number and within a certain range or equal to a fixed value. At the moment only integer inputs are allowed.

Arguments are *is*, *minimum* and *maximum*.

For example, when the input must be 42:

```
<input type="text" id="number" name="number" value="" />
{% validate id="number" type={numericality is=42} %}
```

And for a number within a certain range:

```
<input type="text" id="percent" name="percent" value="" />
{% validate id="percent" type={numericality minimum=0 maximum=100} %}
```

Arguments

Argument	Description	Example
<code>is</code>	Tests for equality.	<code>is=42</code>
<code>minimum</code>	Minimum value.	<code>minimum=1</code>
<code>maximum</code>	Maximum value.	<code>maximum=100</code>
<code>is_float</code>	Boolean flag which tells if the input can be a floating point number. Defaults to false.	<code>is_float</code> <code>is_float=true</code>
<code>not_a_number_message</code>	Message to show when the entered value is not a number. Defaults to “Must be a number.”	<code>not_a_number_message="* "</code>
<code>not_an_integer_message</code>	Message to show when the entered number is not an integer. Defaults to “Must be an integer.”	
<code>wrong_number_message</code>	Message to show when the entered number is unequal to the <code>.is</code> argument. Defaults to “Must be ..”	
<code>too_low_message</code>	Message for when the entered number is less than the minimum allowed. Defaults to “Must not be less than ..”	
<code>too_high_message</code>	Message for when the entered number is greater than the maximum allowed. Defaults to “Must not be more than ..”	

postback

- Module: *mod_base* (page 360)

See also:

Forms and validation (page 37)

Performs a custom server side validation of an input value. This allows you to add your own validation logic to HTML form fields.

Start by adding a validator of the type `postback` to your form field in your template:

```
<input type="text" id="username" name="username" value="" />
{% validate id="username" type={postback event="validate_username"} %}
```

The `event` argument declares the name of the event that will be *notified* (page 69). *Handle this event* (page 71) in your site or module:

Listing 3.4: sites/yoursite/yoursite.erl

```
-export([
    observe_validate_username/2
]).

%% The event name passed in your template as event="validate_username",
%% prefixed with observe_
observe_validate_username({validate_username, {postback, Id, Value, _Args}}, _
    Context) ->
    case is_valid(Value) of
        true ->
            {{ok, Value}, Context};
        false ->
            %% The validation message will be shown in the form
            {{error, Id, "Sorry, that's not valid. Try again!"}, Context}
    end.

%% Some function that returns true or false depending on the validity of the
%% input value
is_valid(Value) ->
    %% ...
```

presence

- Module: *mod_base* (page 360)

See also:

Forms and validation (page 37)

Check if an input has been filled in or checked.

For example when a title must be entered:

```
<input type="text" id="title" name="title" value="" />
{% validate id="title" type={presence} %}
```

Arguments

Argument	Description	Example
failure_message	Message to be shown when field is empty. Defaults to “*”	failure_message="Please enter."

username_unique

- Module: *mod_admin_identity* (page 353)

See also:

m_identity (page 566), *email_unique* (page 40), *Forms and validation* (page 37)

Check if an entered username is unique, by looking in the *m_identity* (page 566) table for the given username:

```
<input type="text" id="username" name="username" value="" />
{% validate id="username" type={username_unique} %}
```

Optionally, an `id` parameter can be given to the validator to skip that particular id when doing the uniqueness check. This is useful when you are displaying a form in which the user is editing his own user name.

If you need to implement custom validation logic, use the *postback* (page 42) validator. For JavaScript-only custom validation, use the *custom* (page 39) validator.

3.1.14 Search

Using the query search API you can retrieve lists of resources in various ways. In your templates, you do so through the *search model* (page 578):

```
{% for id in m.search[{query (options go here...) }] %}
```

For instance, to select all news items, ordered by their modification date, newest first:

```
{% for id in m.search[{query cat='news' sort='-rsc.modified'}] %}
  {{ id }}
{% endfor %}
```

Trying it out

Note: `mod_atom_feed` automatically sorts on last-modified date, `api/model/search/get` doesn't.

Of course you can create your own `for`-loop in a template, but there are easier ways to check out the inner workings of the query model: through your browser.

The query-model is exposed to the browser in (currently) 2 URLs: the Atom feed module for creating a customized update feed, and the API for receiving lists of ids in JSON.

Get all resources of the “documentation” category on zotonic.com:

<https://zotonic.com/api/model/search/get?qcat=documentation>

Get a feed of most recent documentation containing the word “filter”:

<https://zotonic.com/feed/search?cat=documentation&text=filter>

Query arguments

cat

Filter resources on a specific category:

```
cat='news'
```

Specifying multiple ‘cat’ arguments will do an OR on the categories. So to select both news and person resources:


```
cat='news' cat='person'
```

cat_exact

Filter resources to include the given category, but exclude any subcategory:

```
cat_exact='news'
```

cat_exclude

Filter resources to exclude the given category:

```
cat_exclude='meta'
```

id

Filter resources to only include the ones with the given ids:

```
id=123
```

id_exclude

Filter resources to exclude the ones with the given ids:

```
id_exclude=123
```

filter

Filtering on columns:

```
filter=['pivot.title', 'Hello']
filter=['pivot.mypivot.foo', 'bar']
filter=['facet.somefacet', 'baz']
```

In its most simple form, this does an ‘equals’ compare filter. The `filter` keywords expects a list. If the list is two elements long, we expect the first column to be the filter column name from the database table, and the second column name to be the filter value:

```
filter=['facet.numeric_value', `>`, 10]
```

If the filter is a three-column list, the second column is the operator. This must be an atom (surround it in backquotes!) and must be one of the following: `eq`, `ne`, `gt`, `gte`, `lt`, `lte`; or one of `=`, `<>`, `>`, `>=`, `<`, `<=`:

```
filter=['facet.numeric_value', `>`, 10]
```

It is possible to define an OR query for multiple terms:

```
filter=[ ['facet.numeric_value', `>`, 10], ['facet.numeric_value', `<=`, 0] ]
```

hassubject

Select all resources that have an *incoming edge* from the given page, which is specified by the argument (the page id 123 in the example, or the unique page name `tag_gift`). Optionally, you can pass the name of a predicate as the second argument, to specify that the connection should have this predicate.

So, to select all resources that have an incoming edge from a subject with id 123:

```
hassubject=123
```

Alternatively, use the subject's unique name:

```
hassubject='tag_gift'
```

Specifying this multiple times does an AND of the conditions:

```
hassubject=123  
hassubject=[123, 'author']
```

hasobject

Like `hassubject`, but selects all pages that have an **outgoing edge** to the given page, which is specified by the argument. Optionally, you can pass the name of a predicate as the second argument, to specify that the connection should have this predicate:

```
hasobject=123  
hasobject='tag_gift'  
hasobject=[123, 'hasdocument']
```

hasanyobject

Like `hasobject`, but allows to define an OR operation on the edge. You can define multiple combinations of predicates and objects; any resource having such an outgoing edge will be matched. The argument is a list. Each element in the list is either an id or an id/predicate combination.

To select all resources that have an outgoing edge to an object with id 1, 2 or 3:

```
hasanyobject=[1, 2, 3]
```

For each list element, you can add the connection's predicate. So, to select all resources that have an outgoing 'author' edge to an object with id 123:

```
hasanyobject=[[123, 'author']]
```

And to do the same but also include resources that have an 'editor' edge to an object with id 456:

```
hasanyobject=[[123, 'author'], [456, 'editor']]
```

Substitute '*' for the object id to match *any* object. So, to select all resources that have any author or editor edge:

```
hasanyobject[['*', 'author'], ['*', 'editor']]
```

You can also mix the two types of elements. To select all resources that have an author or a connection (with any predicate) to resource 2 or 3:

```
hasanyobject[['*', 'author'], 2, 3]
```

hasmedium

Only returns resources that have a `medium` record attached or only those that do not have a `medium` record attached.

For example:

```
hasmedium
```

Return only resources with a medium record.

Or:

```
hasmedium=false
```

Return only resources without a medium record.

The joined medium record is *medium*, that enables sorting on, for example, the medium record's creation date with `sort=medium.created`.

match_objects

Find the resources that have similar object edges as the given resource. This is done using a full text query. The resource with most overlapping objects ids will be returned first:

```
match_objects=1234
```

An `id_exclude=...` is automatically added for the resource in the argument.

match_object_ids

Find the resources that have similar object edges to the given resources. This is done using a full text query. The resource with most overlapping objects ids will be returned first:

```
match_object_ids=[108, 238, 1234]
```

is_authoritative

Boolean, filters whether a resource is considered authoritative (originating from this site) or not:

```
is_authoritative
```

is_featured

A boolean option that specifies if a page should be featured or not:

```
is_featured
```

is_published

Select published, unpublished or omit the publish check. Legal values are true, false or all:

```
is_published='all'
```

is_findable

A boolean option that specifies if a page should be findable or not:

```
is_findable
```

This checks the resource's `is_unfindable` flag. To be findable in searches the flag must be set to `false`, which is the default.

is_unfindable

A boolean option that specifies if a page should not be findable:

```
is_unfindable
```

This checks the resource's `is_unfindable` flag.

upcoming

Specifying 'upcoming' means that you only want to select things that have a start date which lies in the future. Like the name says, useful to select upcoming events:

```
upcoming
```

ongoing

Specifying 'ongoing' means that you only want to select things that are happening now: that have a start date which lies in the past, and an end date which lies in the future:

```
ongoing
```

finished

Specifying 'finished' means that you only want to select things that have a start date which lies in the past:

```
finished
```

unfinished

Specifying 'unfinished' means that you only want to select things that have an end date which lies in the future:

```
unfinished
```

unfinished_or_nodate

Specifying 'unfinished_or_nodate' means that you only want to select things that have an end date which lies in the future or no start date:

```
unfinished_or_nodate
```

sort / asort / zsort

Sort the result on a field. The name of the field is a string which directly refers to the SQL join that is being used. If you specify a dash (-) in front of the field, the order is descending. Leaving this out or specifying a + means ascending.

The sort terms are added in the order: asort, sort, and zsort.

This is useful for e.g. text search. Text search will add a `sort` term on relevance. This relevance sort term is appended *after* any existing sort term. By using `zsort` you can force sub-sorting in case of the same relevance or no text for the query. Example:

```
{query cat='news' text=q.qsort zsort="-rsc.created"}
```

If `q.qsort` is empty, this will return the newest *news* items. If `q.qsort` is not empty then it will search for the text and return the best matches where equally matching news items will have the newest on top. Use `asort` instead of `zsort` to show the newest matching news, regardless on how well they match the search term:

```
{query cat='news' text=q.qsort asort="-rsc.created"}
```

Some sort fields:

- `rsc.modified` - date of last modification
- `rsc.publication_start` - publication date
- `rsc.pivot_date_start` - the start date specified in the admin
- `rsc.pivot_date_end` - the end date specified in the admin
- `rsc.pivot_title` - the title of the page. For multilingual sites, the behavior of sorting on title is undefined.
- `seq` - sequence number of the first edge (ignored if no edge is joined)
- `edge.created` - creation date of the first edge (ignored if no edge is joined)

For all the sort fields, you will have to consult Zotonic's data model. Example sorting on modification date, newest first:

```
sort='-rsc.modified'
```

See also:

[Pivot Templates](#) (page 324)

pivot.name

See also:

[Custom pivots](#) (page 321)

Filter on the named pivot of the `rsc` table. The name is prefixed with `pivot_`.

Available pivot fields are:

- `pivot.category_nr`,
- `pivot.first_name`
- `pivot.surname`
- `pivot.gender`
- `pivot.date_start`
- `pivot.date_end`

- `pivot.date_start_month_day`
- `pivot.date_end_month_day`
- `pivot.street`
- `pivot.city`
- `pivot.state`
- `pivot.postcode`
- `pivot.country`
- `pivot.geocode`
- `pivot.title`
- `pivot.location_lat`
- `pivot.location_lng`

These fields can also be used in `filter` and the `sort` terms.

pivot.mypivot.name

Filter on the named pivot of the custom pivot `mypivot`.

Here `mypivot` is a custom pivot table defined with `z_pivot_rsc:define_custom_pivot/3`

Check the explanation and examples in [Custom pivots](#) (page 321) for more information.

These fields can also be used in `filter` and the `sort` terms.

facet.name

Add a join with the `facets` table and filter on the named facet.

The `facets` table is filled from the `pivot/facet.tpl` template, each block is a facet that can be used for filters or for the `facets` query.

The name must be the name without types of a block. That is, if a block is called `foo_int` then the name is `foo` and the query term is `facet.foo`.

The value can be an operator:

```
>123
>=123
<123
<=123
<>123
```

For example, the last one translates to the SQL clause `facet.name <> 123`.

The facet fields can also be used in `filter` and the `sort` terms.

hasobjectpredicate

Filter on all things which have any outgoing edge with the given predicate:

```
hasobjectpredicate='hasdocument'
```

hassubjectpredicate

Filter on all things which have any incoming edge with the given predicate:

```
hassubjectpredicate='author'
```

text

Perform a fulltext search on the primary “rsc” table. The result will automatically be ordered on the relevancy (rank) of the result:

```
text="test "
```

Use prefix id: to find specific resources by id or name:

```
text="id:1000"
text="id:1000,1001,1002"
text="id:category,1"
```

query_id

See also:

[Query resources](#) (page 53)

Load the query arguments from the saved query resource:

```
query_id=331
```

qargs

Take all the arguments from the current request and use these. The arguments have to start with a q, for example:

```
http://example.com/search?q=test&qcat=text
```

With the query:

```
m.search.paged[{query qargs page=q.page pagelen=20}]
```

Will find all pages containing the string *test* in the category *text*.

As *qs* is the usual text search argument in forms it is mapped to *text*. All other arguments have the *q* removed and should map to known query-model arguments.

publication_month

Filter on month of publication date

```
publication_month=9
```

publication_year

Filter on year of publication date

```
publication_year=2012
```

date_start_after

Select items with a start date greater than given value

```
date_start_after="2010-01-15"
```

It also possible to use relative times:

- now
- +0 sunday (last sunday or the current sunday)
- +0 monday (last monday or the current monday)
- +1 minute
- +1 hour
- +1 day
- +1 week
- +1 month
- +1 year

Negative offsets are allowed as well. There //must// be a + or – sign.

date_start_before

Select items with a start date smaller than given value:

```
date_start_before="2010-01-15"
```

date_start_year

Select items with an “event start date” in the given year:

```
date_start_year=2012
```

date_end_after

Select items with a end date greater than given value:

```
date_end_after="2010-01-15"
```

date_end_before

Select items with a end date smaller than given value:

```
date_end_before="2010-01-15"
```

date_end_year

Select items with an “event end date” in the given year:

```
date_end_year=2012
```


content_group

Select items which are member of the given content group (or one of its children):

```
content_group=public
```

name

Find resources with a matching unique name. A wildcard can be defined, for example:

```
name=page_*
```

Searching on an empty name or just `*` will return all resources with a defined name. The given name will be trimmed and converted to lowercase before searching.

language

Find resources with a certain language. The language must be a valid ISO 639-1 language code. Search terms with invalid language codes are ignored.

Find all resources with a German translation:

```
language=de
```

Filter behaviour

All of the filters work as AND filter. The only exception to this is the `cat=` filter: if you specify multiple categories, those categories are “OR”ed together, to allow to search in multiple distinct categories with a single search query.

Query resources

See also:

- [mod_search](#) (page 388) reference: Zotonic’s built-in search module.
- [mod_elasticsearch](#) on using Elasticsearch with Zotonic.
- [mod_search_solr](#) on using Solr for search.

Query resources are, as the name implies, [Resources](#) (page 26) of the special category *query*. In the admin this category is called “search query”. it is basically a stored (and thus content manageable) search query. You create an editable search query in an admin page that then is invoked from a template.

When creating such a resource in the page, you will see on the admin edit page an extra text field in which you can add search terms. Each search term goes on its own line, and the possible search terms are equal to the ones described on this page (the *Query-model arguments*).

3.1.15 Translation

Many sites need to support content and templates in multiple languages. Luckily, Zotonic is completely multilingual, out of the box. [mod_translation](#) (page 400) does all the work for you.

Zotonic will try to serve the page in the language of the user (based on the browser’s [HTTP Accept header](#)) or the selected language from the URL. If this fails, Zotonic falls back to the default language as configured on the Translation admin page.

In Zotonic, two kinds of information are being translated.

- Static translations: The fixed texts in your templates, like button labels, confirmation texts, etc.
- Translated content, as edited in the admin. The text fields of each *resource* can be translated in multiple languages.

Enabling multiple languages

Before you can use multiple languages, you first need to enable *mod_translation* (page 400).

Next, you need to tell Zotonic which languages you are going to use. You can do this in admin: Structure > Translation.

On this page you can add and remove languages, enable/disable languages and more.

The selected language can optionally get displayed in the URL of the page - useful if language versions of the page should be accessible as permanent URLs, for instance for search engine indexing.

The language code in the URL is either a 2-letter code (e.g. `en`), a 2-2 language-territory combination (e.g. `en-gb`), a 2-4 language-script combination (e.g. `zh-hant`), or a language-territory-territory combination (e.g. `zh-hant-hk`).

Translation sources

Texts in templates are made localizable with different forms of underscore syntax.

Translate tag:

```
{_ Translate me _}
```

Interpolation trans tag:

```
{% trans "Hello {foo} World" foo=_("happy" %)}
```

As part of a tag parameter:

```
{% button text=_("Click me" %) }
```

Texts in `erl` files use the `?__()` syntax:

```
?__("Translate me", Context)
```

and with binary strings:

```
?__(<<"Translate me">>, Context)
```

using string concatenation:

```
[?__("Edit:", Context), " " ++ Title]
```

and with binary strings:

```
?__(<<"Edit:"/binary, " ", Title/binary>>, Context)
```

Generation of translations

The fixed texts in a Zotonic website are translated using the [GNU gettext](#) `.po` file format and its related tools. If you're not using our *Docker* (page 14) images, you may have to install gettext first:

```
$ sudo apt-get install gettext
```

In Zotonic, translations files are placed in two locations:

- for the core modules, the translation files are consolidated in `priv/translations/`;
- third-party modules and sites, including your own, have their translation files in a `priv/translations/` subdirectory in the module itself.

In the translations directory you can find the `.po` files containing the translations. They are marked with the their language code. (Optionally you can name your file like: `nl.foobar.po` as Zotonic will only look at the part till the first `.` for the language code):

```
mod_foo
  priv/translations/
    template/mod_foo.pot
    tr.po
    zh-hant.po
```

This shows that module `mod_foo` has been translated into Dutch (*nl*), Turkish (*tr*) and Chinese traditional script (*zh-hant*).

The `.po` translation files are based on translation templates (`.pot` files). The templates are located in `translations/templates`:

- `priv/translations/template/zotonic.pot` for the core modules;
- `mod_foo/translations/template/mod_foo.pot` for custom modules.

This `.pot` file is regenerated when you click on the ‘Generate .pot files’ button on the [Translation page](#) (page 400) in the admin. Alternatively, from your Zotonic shell:

```
mod_translation:generate(Context) .
```

Zotonic will parse all your templates and Erlang modules for translatable strings. These strings are then added to the `.pot` files.

Creating a new translation for a module

First, add a language in the admin with the language code for that language. See the [Translation page](#) (or the code in `src/i18n/languages.erl`) for possible languages.

Say, we’re adding Polish, `pl`. Now copy the `.pot` template file to the language code `.po` file:

```
$ cd modules/mod_foo
$ cp translations/template/mod_foo.pot translations/pl.po
```

Now, open `pl.po` in your favorite editor and start translating the strings. A good po file editor can be found at: <http://www.poedit.net/>

Updating translation strings

When templates change, often the translatable strings change: more strings are added, strings are changed, or removed. When this happens, the translation files need to be kept in sync.

In admin: Structure > Translation > Translation Status you can find the overview of the amount of strings that are translated for each language.

After pressing the *Generate .pot files* button in the translation admin the `.pot` files are updated, but the existing `.po` files are not.

GNU gettext comes with a tool, `msgmerge`, which looks at the changed strings in a `.pot` file and changes the translated strings in a language’s `.po` file accordingly:

```
$ cd modules/mod_foo/translations
$ msgmerge -U -N nl.po template/mod_foo.pot
```

This will merge the new strings into the existing `nl.po` file.

To update all `.po` files in the directory:

```
$ cd modules/mod_foo/translations
$ find . -name "*.po" -print0 | xargs -0 -I file msgmerge -U -N file template/*.pot
```

After doing this, you'll need to edit each `po` file again to check if there are new strings which need translation, edit existing strings, etc.

Helpful commands

To remove duplicates (and make a backup first), use:

```
$ cat nl.po > nl~.po && msguniq nl.po -o nl.po
```

To do this for all files, without backup:

```
$ find . -name "*.po" -print0 | xargs -0 -I file msguniq file -o file
```

Translated content

When you have enabled languages on the Translation page of the admin you will see a Translations item on the right of the edit page.

Each language has a checkbox next to it: When you click a checkbox, the language will become visible as a tab on your content items.

Resources in Zotonic are translated on a per-page basis. This allows you to start translating your site by translating the most important pages first.

Text searches, translations and stemming

For text searches a full text index is maintained. This full text index is stemmed according to the site's configured default language.

All translations are added to the same full text index. This combined text is stemmed using a single stemmer. The selected stemmer depends on the configured default language (config key `i18n.language`). The stemmer can be overruled by setting the config key `i18n.language_stemmer` to the two letter language code of the language matching with the stemmer. You have to make sure that the stemmer is configured in PostgreSQL otherwise the pivot process will crash with a SQL error.

3.1.16 Wires

- Module: [mod_wires](#) (page 403)

Wires are the older way to code actions and client/server interaction. It is now advised to use MQTT topics with [mod_mqtt](#) (page 381) instead.

Actions

The action defines what should happen when the wire is triggered. Actions can be client-side (such as JavaScript animations) or server-side postbacks.

Trigger actions from JavaScript

See also:

[Auto-generated identifiers](#) (page 58), [Create a custom action](#) (page 314)

To trigger an action from an HTML element, you attach a wire to the element:

```
<a href="#" id="link">Click me!</a>
{% wire type="click" id="link" action={fade_out target="link"} %}
```

The wire's `id` value must match the `id` value of the HTML element. This wires up a link with a *fade_out* (page 409) action, so that when the link is clicked, it fades away.

Actions can be called from the template, but can also be called when some server-side event occurs.

Server postbacks

See also:

[postback reference](#) (page 419)

Postbacks are server-side actions. For instance, to submit a form asynchronously through Ajax, use a postback:

```
{% wire type="submit" id="myform" postback="form_submitted" delegate="mysite" %}
<form id="myform" method="post" action="postback">
  <input name="username" />
  <button>Submit form</button>
</form>
```

This will submit the form over Ajax; the result is that a function will be called in the specified delegate module `mysite.erl`, called `event/2`:

```
event(#submit{}, Context) ->
  io:format("The value of 'username' is: ~s~n", z_context:get("username",
↳Context),
  Context.
```

Trigger browser actions from the server

See also:

listing of all *actions* (page 403).

Named actions

If you want to trigger actions from your JavaScript code, give the action a name:

```
{% wire name="my_action" action={growl text="Hello World"} %}
```

You can then refer to it in your JavaScript code:

```
z_event("my_action");
```

And pass arguments to the action:

```
z_event("my_action", { foo: bar });
```

The argument `foo` will become a query argument, that you can access in your Erlang module with `z_context:get_q(foo, Context)`.

Auto-generated identifiers

If you include a template many times (i.e. from a for loop), then having fixed element identifiers are no good. Zotonic provides a mechanism to generate an identifier which has a unique value within the template.

To prefix the id with a unique value (per invocation of the template) prefix the id with a #-sign:

```
<div id="{ #foo }">
```

This special notation will replace `#foo` with an auto-generated identifier, which will expand to something like this:

```
<div id="ubifgt-foo">
```

Unique ids can also be generated inside a for loop:

```
{% for id in mylist %}
  <li id="{ #foo.id }">{{ id.title }}</li>
{% endfor %}
```

This will generate HTML like this:

```
<li id="gdjqa-foo-1234">Some great news</li>
```

When using a *wire* (page 551) tag, that same unique id can be referenced:

```
{% for id in mylist %}
  <li><a id="{ #list.id }" href="#">{{ m.rsc[id].title }}</a></li>
  {% wire id=#list.id action=some_action %}
{% endfor %}
```

3.1.17 Access control

Access control is about defining who is allowed to access certain resources. It takes two steps:

See also:

mod_acl_user_groups (page 341)

1. The user is *authenticated*, that is, the user's identity is determined based on some form of identification in the request, such as a session cookie.
2. After the user has been identified, we can decide whether that user is *authorized* to access a certain URL or resource.

Authentication

For each HTTP request that Zotonic receives, it looks for some form of credentials that can identify the user. This can be a username/password combination when the user logs in for the first time, and a *session cookie* for subsequent requests. When Zotonic finds some credentials, it checks them for validity. If the credentials are valid, the user is said to be authenticated and authorization can start.

The first request, that does not yet have a session cookie or whose session cookie has expired, needs to contain some credentials in order to be authenticated. The *authentication controller* (page 440) takes care of processing login requests and checks for the presence of a 'remember me' cookie for automatic login. It then responds with a fresh session cookie that the client will send along with subsequent requests.

Authenticating subsequent requests, that have a session cookie, does not take place until a session is ensured (or continued) for the request *context*. This is commonly done by the *controller* handling the request by a call to `z_context:ensure_all/1`.

Customizing authentication

Zotonic relies on a number of notifications to perform authentication. Observe any of the authentication notifications to customize Zotonic's authentication behaviour. See the reference for a list of all [authentication notifications](#) (page 585).

Authorization

Once the request has been authenticated, authorization is initialized by sending an `#acl_logon{}` notification. Should the session get logged out, losing its authentication, the authorization is cleared by sending a `#acl_logoff{}` notification.

Once authorization has been initialized for a request [context](#), operations against objects can be checked by the `z_acl` module from Erlang code, and by the `m_acl` (page 555) model from [templates](#) (page 28).

Protecting access to controllers

See also:

[ACL options](#) (page 448)

The first point of contact for authorization is in the [controller](#)'s `is_authorized/2` function. Both [controller_page](#) (page 447) and [controller_template](#) (page 452) check for [ACL options](#) (page 448) in the [dispatch rule](#) that matched the current request.

Protecting access to resources and modules

Zotonic ships with [mod_acl_user_groups](#) (page 341), a powerful user group-based authorization module. With this module you can define access control rules that determine which user groups are allowed to access which groups of content.

Customizing authorization

No matter what authorization module you use, you can always override Zotonic's behaviour by observing the authorization or ACL notifications. This is especially useful if your application has some authorization logic that is not easily expressed in ACL rules. See the reference for a full list of [ACL notifications](#) (page 582).

3.1.18 Modules

See also:

The reference for a list of [all modules](#) (page 341).

See also:

You can [roll your own module](#) (page 325).

Modules are the building blocks of Zotonic. They add functionality to your Zotonic website such as:

- the admin web interface
- embedding videos in your content
- search engine optimization (SEO)
- social media integration.

Structurally, a module is a directory containing the module's Erlang code, [templates](#) (page 28), [controllers](#) (page 20), [dispatch rules](#) (page 22) and more.

Activating modules

Before you can use a module, you need to activate it. You can do so in two ways.

1. For testing, you can enable the module in the *admin interface* (page 355), under System > Modules.
2. If you decide to use the module in your site, it's best to declare so in your *site configuration* (page 629). This ensures that the module is activated not only for you but also for other developers and on other servers that the website may run on (e.g. a production server). Add the module name to the `sites/your-site/config` file, under the `modules` key:

```
[
  % ...
  {modules, [
    % ...,
    mod_example
  ]}
  %...
].
```

Then *restart the site* (page 18) for the changes to be picked up.

Module configuration

Some modules can be configured to influence their behaviour. The module's documentation will tell you about its configuration parameters.

Directory structure

A module groups related functions together into a single directory. It contains an Erlang module (from here on called the 'module file') and subdirectories for templates, actions, tags, dispatch rules and more.

The generic structure is:

```
zotonic_mod_example/
  priv/dispatch/
  priv/templates/
  priv/lib/
  priv/lib-src/
  src/mod_example.erl
  src/zotonic_mod_exampe.app.src
  src/models/...
  src/filters/...
  rebar.config
```

The module file

At the very minimum, a Zotonic module must have a module file. The name of the module file is an Erlang file that must be the same as the name of the module's directory. Zotonic scans this file for metadata about the module and uses it to start the module:

```
-module(mod_example).
-author("Nomen Nescio <nomen@example.com>").
-mod_title("Your module title").
-mod_description("Description what this module does.").
-mod_prio(500).
```

In this case, the module code only consists of some metadata properties, there is no real code in there. This is fine for a lot of modules: since Zotonic already provides so many functions, there is often little need to write custom code.

The `mod_title` and `mod_description` properties describe your module in natural language: these properties will be visible on the admin modules page. The `mod_prio` property defines the *priority* (page 63) of the module.

In cases where you need to execute code when the module starts, you can export an optional `init/1` function. The parameter is a context record initialized for the site the module will be running in. This is useful when you need to initialize the database or other data structures for which you don't need a running process. When you also need to execute code when a module stops you can export an optional `terminate/2` function. This function will be called when the module terminates. The first parameter is a Reason parameter which indicates why the module stopped. The second a context record similar to the one in the `init/1` function.

When you do need a running process, read about those in the next topic, *gen_server based modules* (page 67).

Module subdirectories

Besides the module code file, a module usually has one or more subdirectories. These are specially named; different parts of Zotonic scan through different folders.

This section describes what each of the module folders hold.

priv/dispatch/

See also:

Dispatch rules (page 22)

This directory contains files with *dispatch rules* (page 22). You can name your files however you want, just don't give them the extension `.erl`, because then the Makefile will try to compile them.

priv/lib/

See also:

the *lib* (page 528) template tag.

The `lib` (short for *library*) directory contains static images, CSS and javascript files. These files will be served with via the *lib* (page 528). The usual layout of the lib directory is:

```
priv/lib/css/
priv/lib/images/
priv/lib/js/
priv/lib/misc/
```

priv/lib-src/

This directory contains the source files for the `lib` directory. Examples are Less, Scss, and other files.

If a file in the lib-src directory changes then the system will search for a `Makefile` in the directory of the changed file or one of its parent directories. If a `Makefile` is found then it is executed.

Scss files starting with a `_` (like `_home.scss`) are known to be include files. If no `Makefile` is found then a Scss file without underscore is searched and used to generate the corresponding css.

If no `Makefile` is found then any input file is converted to a similar sub-directory in the `lib` directory. For example `lib-src/foo/bar.scss` is used to generated `lib/foo/bar.css`

There are standard file handlers for the following extensions/formats:

- `.scss` (Makefile, `sassc`, or `sass`)
- `.less` (Makefile or `lessc`)

- `.coffee` (Makefile or coffee)

priv/templates/

See also:

[Templates](#) (page 28)

This directory contains all [templates](#) (page 28). Templates do not have any prefix in their name, as they are not (directly) compiled as Erlang modules.

The following naming conventions for templates are used:

- All templates have the extension “.tpl”
- Templates used as a complete page can have any name: “my_special_page.tpl”
- Templates used as the base of other templates, using the [extends](#) (page 516) tag, have the word “base” in them: “base.tpl”; “email_base.tpl”.
- Templates only used by including them in other templates start their name with an underscore: “_example.tpl”
- The template for the home page of a site is called “home.tpl”
- Templates for displaying resources are called “page.tpl”

src/actions/

See also:

[Actions](#) (page 56)

This directory holds the [actions](#) (page 56) defined by the module. Every action name must be prefixed with the word “action” and the module name (without the `mod_`). For example the filename for the action `dialog_open` in the module `mod_base` will be `action_base_dialog_open.erl`

src/scomps/

See also:

[Tags](#) (page 29)

Any custom tags that you define yourself go into the `src/scomps/` directory.

Scomps are prefixed in the same way as actions, except that the word “scomp” is used. For example the `scomp_button` in the module `mod_base` has as file name `scomp_base_button.erl`.

src/controllers/

See also:

[Controllers](#) (page 20)

This directory contains Erlang modules which define controllers which are called from the dispatch system to handle incoming HTTP requests.

Controllers must have unique names, as they are compiled and loaded in the Erlang system. The convention is to prefix every controller with `controller_` and the name of the module, for example `controller_admin_edit.erl`.

src/models/

See also:

[Models](#) (page 30)

This directory contains Erlang modules, each of which is a [model](#) (page 30).

The module name of a model always starts with `m_`, for example `m_comment`. This model is then to be used in the templates as `m.comment`. Be careful to give your models a unique name to prevent name clashes with other models and Erlang modules.

src/filters/

See also:

[Filters](#) (page 29)

This directory holds Erlang modules, each of which defines a [template filter](#) (page 29).

Each filter must have a unique name, reflecting the filter's name. For example, the filter “tail” resides in the Erlang module `filter_tail.erl` and exports the function `tail/1`. Filters are added in the filters directory. The template compiler will insert references to the correct modules into the compiled templates. A missing filter will result in a crash of the compiled template.

src/validators/

See also:

[Forms and validation](#) (page 37)

This directory holds Erlang modules, each of which defines a [validator](#) (page 37).

Validators are prefixed in the same way as actions and scomp, except that the word “validator” is used. For example the validator “email” in the module “mod_base” has the file name: “`validator_base_email.erl`”

Changing / recompiling files

Changes to the Erlang files in a module are visible after issuing the `zotonic update` CLI command, or `z:m()` from the Zotonic shell. Any new lib or template files, or changes in the dispatch rules are visible after the module indexer has rescanned all modules. You can do this with the “rescan modules” button on the modules page in the admin. Changes to templates are directly visible.

Priority

The *module priority* is a number defined in the module's code and is usually a number between 1 and 1000; the default is 500. A lower number gives a higher priority. Modules with higher priority are checked first for [templates](#) (page 30), actions, custom tags etc.

The priority is defined by `mod_prio` in the [module file](#) (page 60). For a site, the priority is usually set to 1, to make sure that its templates etc override the ones from the Zotonic modules.

When two modules have the same priority then the modules are sorted by their name. That means that, given the same priority number, `mod_aloha` has higher priority than `mod_hello`.

Dependencies

Modules can have dependencies on other modules. These are expressed via the module's metadata, as follows:

```
-mod_depends([mod_admin]).
```

This states that the current module is dependent on `mod_admin` to be activated.

Sometimes, explicitly depending on a module name is not a good idea: there might be more modules that perform the same functions but are competing in implementation. In that case, such modules can export a `mod_provides` meta tag, so that dependent modules can depend on what one of these modules provides.

Example: `mod_a` and `mod_b` both provide some functionality, `foo`:

```
-module(mod_a).  
-mod_provides([foo]).
```

and:

```
-module(mod_b).  
-mod_provides([foo]).
```

Now, another module, `mod_bar`, needs the “foo” functionality:

```
-module(mod_bar).  
-mod_depends([foo]).
```

Now, the module manager will require either (or both!) of the `mod_a` and `mod_b` modules to be activated, before `mod_bar` can be activated.

A module automatically provides its own module name, as well as its name minus the `mod_`. So, `mod_bar` has implicitly the following provides constructs:

```
-module(mod_bar).  
-mod_provides([mod_bar, bar]).
```

These two provides are there even when a module adds its own provides clauses.

Module startup order

Note that when a site starts, its modules are started in order of module dependency, in such a way that a module's dependencies are always started before the module itself starts.

Module versioning

Note: The function `manage_schema/2` is called inside a transaction, so that any installation errors are rolled back. `manage_data/2` is called outside a transaction, and after all resources, predicates etc. are installed, but before the current module version number is updated.

Modules can export a `-module_schema()` attribute which contains an integer number, denoting the current module's version. On module initialization, `Module:manage_schema/2` is called which handles installation and upgrade of data.

The `manage_schema/2` function returns either `ok`, a `#datamodel{} record` or a list of `#datamodel{} records`:

```
-spec manage_schema( install | {upgrade, integer()}, z:context() ) ->  
    ok | #datamodel{} | [ #datamodel{} ].
```

In a `#datamodel{}` record you can define:

- categories
- predicates
- resources
- edges
- *ACL rules* (page 345).

After the `manage_schema/2` function is called, the optional `manage_data/2` function is called. The function `manage_data/2` is called if and only if the `manage_schema/2` is called. If you only want a `manage_data/2` function, then add a dummy `manage_schema/2` function that returns *ok* and does nothing else.

For example:

Listing 3.5: `mod_twitter.erl`

```
-module(mod_twitter).
-mod_title("Twitter module").
-mod_schema(3).  %% we are currently at revision 3

-export([
    manage_schema/2,
    manage_data/2
]).

%% ...

manage_schema(install, Context) ->
    %% Return a #datamodel{} record with items that will be created when
    %% the module starts.

    #datamodel{
        categories = [
            {
                tweet,  %% unique category name
                text,   %% parent category (can be undefined)

                %% category properties
                [
                    {title, <<"Tweet">>}
                ]
            },
        ],
        predicates = [
            {
                tweets,  %% unique predicate name

                %% predicate properties
                [
                    {title, {trans, [
                        {en, <<"Tweets">>},
                        {nl, <<"Twittert">>}
                    ]}},
                ],
                %% predicate from/to categories:
                [
                    {person, tweet}
                ]
            },
        ],
    }
```

```
resources = [
  {
    person_tweeter,  %% resource's unique name
    person,          %% category

    %% resource properties
    [
      {title, <<"Test Tweeter">>},
      {name_first, <<"Sir">>},
      {name_surname, <<"Tweetalot">>}
    ]
  },
  {
    silly_tweet,
    tweet,
    [
      {body, <<"What's your favourite colour?"/utf8>>}
    ]
  }
],
edges = [
  %% subject      predicate      object
  {person_tweeter, tweets,      silly_tweet}
]
};

manage_schema({upgrade, 2}, Context) ->
  %% code to upgrade from 1 to 2
  ok;

manage_schema({upgrade, 3}, Context) ->
  %% code to upgrade from 2 to 3: update the person_tweeter resource
  #datamodel{
    resources = [
      {
        person_tweeter,
        person,
        [
          {name_surname, <<"Tweetalot the Second">>}
        ]
      }
    ]
  }.

manage_data(_Version, Context) ->
  %% Whatever data needs to be installed after the datamodel
  %% has been installed.
  ok.
```

Note that the install function should always be kept up-to-date according to the latest schema version. When you install a module for the first time, no upgrade functions are called, but only the `install` clause. The upgrade functions exist for migrating old data, not for newly installing a module.

Using categories defined by other modules

When your site needs to add resources which are defined by other module's `manage_schema` functions, you need to make sure that those modules manage functions are called first. This can be realised by adding a dependency to those modules, as explained in *Module startup order* (page 64).

For instance, when you want to create a custom menu for your site:

```
manage_schema(install, _Context) ->
    #datamodel{
        resources=[
            {help_menu, menu, [
                {title, "Help"},
                {menu, [...]}
            ]}
        ]
    }.

```

You also need to make sure that you add a *dependency* (page 64) to `mod_menu`, which creates the menu category for you:

```
-mod_depends ([mod_menu]) .
```

gen_server based modules

See also:

`gen_server` in the Erlang documentation.

When you need a running process, i.e., a module that does something in the background, then it is possible to implement your module as a `gen_server` (or supervisor). A `gen_server` is a standard way to implement a reliable Erlang worker process.

In that case you will need to add the behaviour and `gen_server` functions. You also need to change the `init/1` function to accept a property list, which contains the site definition and a `{context, Context}` property.

This server module will be started for every site in a Zotonic system where the module is enabled, so it can't be a named server.

If you want to observe Zotonic's *notifications* (page 69) and handle them through your module's `gen_server`, export `pid_observe_...` functions (instead of the regular `observe_...` ones). These function will then be passed the `gen_server`'s PID:

```
export([
    pid_observe_custom_pivot/3
]).

pid_observe_custom_pivot(Pid, #custom_pivot{} = Msg, _Context) ->
    gen_server:cast(Pid, Msg).

handle_cast(#custom_pivot{id = Id}, State)
    %% Do things here...
    {noreply, State}.

```

A minimal example

```
%% Zotonic modules always start with 'mod_'
-module(mod_example).

%% The author - also shown in the admin ui
-author("Nomen Nescio <nomen@example.com>").

%% A module can be a 'gen_server', a 'supervisor', or just a module
%% without behaviour.
-behaviour(gen_server).

%% The title of your module
-mod_title("Your module title").

```

```
%% A short description, shown in the admin ui
-mod_description("Description what this module does.").

%% Priority, lower is higher prio, 500 is default.
-mod_prio(500).

%% The modules or services this module depends on.
%% This module is only started after the mentioned modules
%% or services are started.
%% List of atoms.
-mod_depends([]).

%% The modules or services this module provides.
%% A module always provides itself ('mod_example' in this case)
%% List of atoms.
-mod_provides([]).

-export([init/1, handle_call/3, handle_cast/2, handle_info/2, terminate/2, code_
↳change/3]).
-export([start_link/1]).

-include_lib("zotonic_core/include/zotonic.hrl").

-record(state, {
    context :: z:context()
}).

%% Module API

%% The Args is a proplists with the site config and a context.
%% The {context, z:context()} is added to it so there is
%% an instantiated site context. The site context is
%% authenticated as the admin.
start_link(Args) when is_list(Args) ->
    gen_server:start_link(?MODULE, Args, []).

%% gen_server callbacks

init(Args) ->
    {context, Context} = proplists:lookup(context, Args),
    % Instantiate a new, empty, and anonymous site context.
    {ok, #state{ context = z_context:new(Context) }}.

handle_call(Message, _From, State) ->
    {stop, {unknown_call, Message}, State}.

handle_cast(Message, State) ->
    {stop, {unknown_cast, Message}, State}.

handle_info(_Info, State) ->
    {noreply, State}.

terminate(_Reason, _State) ->
    ok.

code_change(_OldVsn, State, _Extra) ->
    {ok, State}.
```

As you can see, this code is almost identical to the standard Erlang `gen_server` boilerplate, with the exception of the metadata on top.

You also see that the `start_link/1` function is already implemented. Note that in this function the `gen_server` is started without registering the server under a name: this is done because the module can be started multiple times; once for each site that needs it.

The `init/1` function contains some more boilerplate for getting the `context{}` argument from the arguments, and storing this context into the server's state. This way, you'll always have access to the current context of the site in the rest of the `gen_server`'s functions.

3.1.19 Notifications

See also:

[list of all notifications](#) (page 582)

At different moments in the lifecycle of the web request, Zotonic sends notifications. By *observing* these notifications you can *override* (page 325) Zotonic's behaviour. You can also add your own notifications.

Zotonic's notifier system makes it possible to create modular components with a pluggable interface. The notifier system is used by internal core Zotonic components like the authentication mechanism, the logging system and more.

The notification system can not only act as a traditional event subscription system but also as an advanced priority based function dispatch system. It uses the same priority system which is used to select templates. This makes it possible to override pre-defined default behaviour of core Zotonic modules.

A notification message is a tagged tuple. The first element of the tuple is the type of notification message. An observer can use this to subscribe to specific messages it is interested in.

Example:

```
{acl_logon, 1234}
```

Or:

```
{module_activate, mod_stream, <0.32.0>}
```

You can find a [list of all notifications](#) (page 582) in the reference section.

Sending notifications

As mentioned earlier, the notification system can not only be used to just send events to observers. Observers can also return values back. They can do this in various ways described in the methods below.

Notification types

notify

Send a message to all observers. This is used if you want to notify other observers about a specific event. In Zotonic this is used a lot. For instance, it is used to notify modules of about user logons, or notify when modules are activated and deactivated.

notify1

Notify the first observer. This is useful for if you want to be sure just one observer can do something with the message.

first

Call all observers, and use the first non undefined answer. This is used to get information from one of the observers. By using the notification system it makes sure that modules are decoupled.

map

Call all observers and get a list of all answers.

foldl

Do a fold over all observers, high prio observers first.

foldr

Do a fold over all observers, low prio observers first.

notify_sync

Synchronous notification, wait till all observers have processed the notification before continuing. The *notify* (page 69) will notify all observers asynchronously and returns before all notifications are processed.

The synchronous notification also shares the current database transaction (if any) and other context data with the observers. This allows to process the notification within a transaction.

Subscribing to notifications

Registering as an observer works as follows:

```
z_notifier:observe(NotificationType, Handler, Priority, Context)
```

If the module is either a Zotonic module or a site module, the `Priority` parameter can be omitted. The observer will then get the same priority as the module.

NotificationType The type of notification you want to observe, an atom.

Handler Can be a `pid()`, or a `{Module, Fun}` tuple. When the handler is a `pid()` and the notification is sent with `notify` or `notify1` the `gen_server` process receives a `handle_cast`. When an answer is expected back `handle_call` is used. This is the case for `first`, `map`, `foldl` and `foldr`.

Priority The priority of the observer. This influences the order in which they are called.

Context The Zotonic context.

Example:

```
z_notifier:observe(acl_logon, {mysitewww, handle_logon}, Context)
```

Subscription shorthands

Modules and sites can use shortcuts for registering as an observer. When the Zotonic module exports a function with the prefix `observe_` or `pid_observe_` Zotonic's module manager will register the observer for you.

For example exporting `observe_acl_logon/2` will register that function as an observer. It will be triggered when the `acl_logon` notification is fired. Functions prefixed with `pid_observe_` are for *gen_server based modules* (page 67): they get passed the `gen_server`'s PID as the first argument.

Handling notifications

When a notifications is sent the `z_notifier` module looks in its tables to see if there are any observers interested in receiving it. There are three types of notifications.

Cast notifications This is the simplest notification. The notifier does not expect an answer back the result of the handler is ignored. This kind of notification is triggered by calling `z_notifier:notify/2` or `z_notifier:notify1/2`. They are useful for letting other modules know about a certain even or condition. This makes it possible for other modules to act on it.

For example, *mod_development* (page 364) uses call notifications to trigger builds and reloads. By doing this other modules can notify `mod_development` to trigger builds. But when `mod_development` is disabled nothing will happen.

Call notification For this kind of notification, `z_notifier` expects an answer back. This answer is returned back to the notifier. This kind of notifications is used to decouple modules. For instance a module can ask another module for a special URL to go to after logging in without knowing which module will do this. Call notifications are triggered by: `z_notifier:first/2` and `z_notifier:map/2`.

For example, *mod_signup* (page 393) uses a call notification to find out what page to redirect to after a successfull signup. This allows one to customize the signup process.

Fold notifications

Fold notifications are called, with `z_notifier:foldl/3` or `z_notifier:foldr/3`. It works similarly to the `lists:foldr` and `lists:foldl` functions of Erlang's `lists` module.

The fold function calls each observer in sequence, either starting at highest (`foldl`) or at lowest (`foldr`) priority, passing values and an initial accumulator value.

Each observer can adapt values in the accumulator, and needs to return it, for passing on to the next observer. The final value of the accumulator is returned as result. This is useful if you want multiple modules to be able to adapt and use values in the accumulator.

For example, *mod_admin* (page 346) uses a fold notification (called `admin_menu`) to build up the admin navigation menu, where each observer is called to add menu entries to the menu.

3.1.20 Browser/server interaction

There are multiple ways to set up interaction between server-side Zotonic code and client-side JavaScript.

If you want to initiate the interaction in the client (browser)

1. publish/subscribe.
2. API methods
3. Wires
4. Notifications
5. Transport (deprecated)

Publish/subscribe (MQTT)

The preferred transport is using publish and subscribe with MQTT topics. Message that are published on the bridge topics are relayed between the server and the browser (and vice versa).

See *mod_mqtt* (page 381) for more information.

An example of MQTT PubSub usage is the custom tag *live* (page 542).

API Methods

All model interfaces `m_get`, `m_post`, and `m_delete` are directly exposed as JSON APIs.

For example, the `m_get` call for the Erlang file `m_mymodel.erl` is available via a HTTP GET on the path: `/api/model/mymodel/get/...`

Similar for the POST and DELETE HTTP methods.

Wires

This is part of *mod_wires* (page 403) and is the older method of transporting data between the server and the browser.

It is fully functional and the *admin* interface is using these methods.

If you creating a new site then it is advised to start using the MQTT topics and isolate the places where you are using wires, so that these can be replaced in due time.

Wired events

First, define a named action:

```
{% wire name="test" action={growl text="Hello world"} %}
```

Then, call that action from your JavaScript code:

```
z_event("test");
```

Trigger server side notifications from JavaScript

Trigger an Erlang *notification* (page 69) in Zotonic with the `z_notify` JavaScript function:

```
z_notify("mymessage");
```

or:

```
z_notify("mymessage", {foo: bar, who: "world"});
```

This will trigger a call to:

```
z_notifier:first(#postback_notify{message = <<"mymessage">>}, Context)
```

Which you can handle in any Zotonic module by defining:

```
-export([observe_postback_notify/2]).
observe_postback_notify(#postback_notify{message = <<"mymessage">>}, Context) ->
    Who = z_context:get_q(who, Context),
    z_render:growl(["Hello ", z_html:escape(Who)], Context);
observe_postback_notify(_, _Context) ->
    undefined.
```

All arguments are available via the `z_context:get_q/2` function (and friends).

Transport

Note: It is strongly advised to use MQTT topics instead of the `z_transport` mechanism. See [mod_mqtt](#) (page 381) for more information.

Zotonic has a message bus to transport data between server and browser. It transports structured data in different formats and supports retransmission in case of lost messages.

Zotonic uses two mechanisms to transport data from the browser to the server:

- WebSocket with bidirectional transports using [controller_mqtt_transport](#) (page 446)
- AJAX calls to publish via [controller_mqtt_transport](#) (page 446) to a topic. This is used to post forms with files to the server.

The WebSocket connection it used to transport data from the server to the browser.

From browser to server

To send a message from the browser to the server:

```
z_transport("mod_example", "ubf", {hello: "world"});
```

And then on the server, use Erlang to process the message:

```
-module(mod_example).
-export([event/2]).

-include_lib("zotonic_core/include/zotonic.hrl").

event(#z_msg_v1{data=Data}, Context) ->
    io:format("~p", [Data]),
    Context;
```

This will print on the console:

```
[{<<"hello">>, <<"world">>}]
```

Quality of service

The message will be sent with a quality of service of 0. That means the browser will try to send the message, but will not check if it arrived. Alternatively, you can send with a qos of 1, in that case the browser will wait for an ack, and if that doesn't arrive in 30 seconds, then a duplicate message will be queued for transport:

```
z_transport("mod_example", "ubf", {hello: "world"}, {qos: 1});
```

It is possible to define a callback function that will be called if an ack is received:

```
z_transport("mod_example", "ubf", {hello:"world"}, {
    qos: 1,
    ack: function(ackMsg, callOptions) {
        alert(ackMsg);
    }
});
```

From server to browser

Sending JavaScript (or other data) from the server to the browser is straightforward:

```
z_transport:page(javascript, <<"alert('Hello World');">>, Context);
```

This transports the JavaScript to the page associated with `Context`. This JavaScript will then be evaluated in the browser.

The default quality of service is 0 (see above); to let the page queue retry delivering the message it is possible to specify another quality of service:

```
z_transport:page(javascript, <<"alert('Hello World');">>, [{qos, 1}], Context);
```

It is also possible to send a message to all open pages of a session, or to all sessions of a user:

```
z_transport:session(javascript, <<"alert('Hello World');">>, [{qos, 1}], Context);
z_transport:user(javascript, <<"alert('Hello World');">>, [{qos, 1}], Context);
```

Or transport to a specific page, session or user, but then you will need to specify the message and the message-queue:

```
Msg = z_transport:msg(session, javascript, <<"alert('Hello World');">>, [{qos, 1}
↪]).
z_transport:transport_user(Msg, UserId, Context).
```

The message queue is either `session` or `page`. It defines which queue will be responsible for resending the message and where the ack message is received. If `user` is specified as queue then it will be replaced by `session`.

3.1.21 E-mail handling

Any Zotonic system is capable of sending and receiving e-mail messages over SMTP.

Zotonic implements a mailing system for sending text and HTML messages to one or more recipients.

Out of the box, e-mail sending should “just work”.

Site-specific configuration

Module	Key	Value
site	email_from	Set this to the from-address you want to e-mail to appear from, e.g. something like <code>noreply@yoursite.com</code> .
site	email_override	If set, all e-mail messages that get sent from Zotonic will arrive at this address. Useful if you are testing but don't want to confuse other people with your test e-mails.
site	smtphost	The hostname where you want messages to appear from. Mostly used for bounce message handling and the EHLO handshake. Defaults to the site's hostname, but can be overridden
site	admin_email	E-mail address of the admin user, the address where admin log/debug messages get sent to when using <code>z_email:send_admin/3</code> .
site	bounce_email	E-mail address where bounces are sent to. Normally a special bounce address is generated. See also the discussion about <code>smtp_bounce_email_override</code> below.
site	email_images_inlined	Images in emails are inlined if they are smaller than 1MB. Setting this config disables the inlining of images, the html image tags will be unchanged.

Zotonic-wide configuration

The file `~/config/zotonic/1/zotonic.config` (on macOS `~/Library/Application Support/zotonic/config/1/zotonic.config`) can be configured to hold any of the configuration options below. They are in effect for every site running in the Zotonic instance.

Zotonic-wide configuration for sending email

Key	Description
<code>smtp_relay</code>	Whether or not to use a SMTP relay host. Boolean value, defaults to false.
<code>smtp_host</code>	The hostname for the SMTP relay host, only needed if <code>smtp_relay</code> is enabled.
<code>smtp_ssl</code>	Whether or not to use SSL on the relay host, only needed if <code>smtp_relay</code> is enabled.
<code>smtp_username</code>	The username for the relay host, only needed if <code>smtp_relay</code> is enabled.
<code>smtp_password</code>	The password for the relay host, only needed if <code>smtp_relay</code> is enabled.
<code>smtp_no_mx_lookup</code>	Set to true to not do a MX lookup before sending mail. (default: false)
<code>smtp_verp_as_from</code>	Use the “from” address as VERP for bounce handling (default: false)
<code>smtp_bcc</code>	Optionally send a BCC of every sent to this address
<code>email_override</code>	A global e-mail override. The override logic first checks the site override, and then the global override address. Useful for testing and development.
<code>smtp_bounce_domain</code>	Which domain to use for bounce VERP messages. Defaults to the smtp domain of the site sending the email.
<code>smtp_bounce_email</code>	The email address for bounce handling. Only use if all else fails (see the paragraphs after the next one).
<code>smtp_is_blackhole</code>	Drop all outgoing email, the mail is still logged. Good for testing large mailings.

Zotonic-wide settings for receiving email

To receive email the SMTP server has to listen on the correct IP address and port. Spam filtering is done by checking DNSBL (DNS Block List) servers and optionally using Spamassassin.

Key	Description
<code>smtp_listen_domain</code>	The domain announced in the HELO
<code>smtp_listen_ip</code>	IP address to listen on for incoming SMTP connections. Defaults to "127.0.0.1" Set to <code>none</code> to disable the smtp server.
<code>smtp_listen_port</code>	IP address to listen on for incoming SMTP connections. Defaults to 2525. Set to <code>none</code> to disable the smtp server.
<code>smtp_dnsbl</code>	List for the DNS block lists used for checking incoming email connections. Defaults to ["zen.spamhaus.org", "dnsbl.sorbs.net"]
<code>smtp_dnswl</code>	List for the DNS white lists used for checking incoming email connections. Defaults to ["list.dnswl.org", "swl.spamhaus.org"]
<code>smtp_spamd_ip</code>	Optional IP address for a spamassassin host
<code>smtp_spamd_port</code>	Optional port number for a spamassassin host

The sender's domain

Recipients of e-mail want a valid sender's address on the envelope. This section describes how to set your e-mail bounce/sender domain and fixing domain errors when sending e-mail.

You have to think of e-mail as normal snail mail. There is a message and an envelope.

The `email_from` is the address that is written on the *message*. It could be anything, and is generally not used when delivering your mail. Just like with snail mail, the postman only looks on the *envelope*, not on the message.

The address on the envelope is the most important address. It is where your e-mail is returned to when the message can't be delivered (so called bounces). Often it is also checked for validity when the e-mail is delivered at a SMTP server.

You need a valid domain for this envelope sender address. The part before the @ is generated by Zotonic and is used for identifying the original message and recipient when the message bounces.

If the generated part is not acceptable then you can force an envelope address by setting the `smtp_bounce_email_override` option. Setting the bounce/envelop address manually disables Zotonic's build-in handling of bounces that happen *after* the e-mail was accepted for delivery by the remote SMTP host.

The bounce e-mail address can also be set on a per-site basis using the `site.bounce_email_override` configuration. See the site specific settings table above.

How does Zotonic know the domain?

It checks in order:

- site's config: `bounce_email_override` (you can also set this with the admin config as `site.bounce_email_override`)
- global `zotonic.config`: `smtp_bounce_domain` setting
- site's config: `smtphost`
- site's config: `hostname`

Any `bounce_email_override` configuration must be a complete email address. For example: `bounces@example.org`

If no `bounce_email_override` is used then the part before the @ is generated by Zotonic itself, for administration and detection of bounces. A typical sender address on the envelope looks like: `noreply+mlcm6godbz2cchtgdvom@example.org`

Sending E-mail

Once configured, you can use the following Erlang commands to send e-mail from Zotonic code:

Command	Explanation
<code>z_email:send/3</code>	Sends a quick e-mail to the site administrator. Handy to notice the site admin that something is wrong, a job has finished, etc... The e-mail that is used is the <code>admin_email</code> address that is specified in the site's config file.
<code>z_email:send/4</code>	Sends a text message with a subject to a specified recipient.
<code>z_email:send/4</code>	Renders a template and sends it as a HTML message to a specified recipient.
<code>z_email:send/2</code>	Sends an email defined by a <code>#email{}</code> record.

Send e-mail from a template

First, create a *template* (page 28) for the e-mail. Just like normal templates, it can contain *variables* (page 29). The `<title>` tag becomes the e-mail's subject:

Listing 3.6: email.tpl

```
<html>
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <title>An e-mail from {{ sender_name }} at {{ m.site.title }}</title>
  </head>

  <body>
    Hello {{ id.name_first }},
```



```

    Hope you're doing fine!

    Cheers,

    {{ sender_name }}
  </body>
</html>

```

To include inline images, just add an `` tag:

Listing 3.7: email.tpl

```

<body>
  
</body>

```

Use the *image tag* (page 520) to include a *media depiction* (page 32):

Listing 3.8: email.tpl

```

<body>
  {% image id.depiction %}
</body>

```

After you've created the template, you can send the e-mail. You can provide the template variables (we had `id` and `sender_name` in the template) as `vars`:

```

-include_lib("zotonic_core/include/zotonic.hrl").

some_function(SomeId, Sender, Context) ->
  %% Create an e-mail record
  Email = #email{
    to = "someone@somewhere.com",
    vars = [
      {id, SomeId},
      {sender_name, Sender}
    ],
    html_tpl = "email.tpl"
  },

  %% And send the e-mail
  z_email:send(Email, Context).

```

Send e-mail to the admin

The `z_email:send_admin/3` command actually looks in three different places for determining the admin e-mail address: the config key `zotonic.admin_email`, then the `site.admin_email` key, and finally the *email* property of the admin user (user with id 1).

If no admin email address is found then the address `wwwadmin@example.com` is used, where `example.com` will be your site's hostname.

Receiving E-mail

In its default configuration, Zotonic starts an SMTP server on port 2525 for receiving e-mail messages. You can write your own code to decide what happens if somebody sends e-mail to the system, by implementing the `email_received` notification (see below).

The SMTP server is also used to receive bounce messages from other servers, when sending of a message has failed. *mod_mailinglist* (page 379) uses this functionality to automatically deactivate invalid e-mail addresses.

Configuring incoming E-mail

To send messages to Zotonic, the domain part of the e-mail address should have an A or MX record which points to the server where Zotonic is able to receive on port 25. This means that you have to add a firewall rule to redirect port 25 to 2525.

If you were to set up e-mail receiving for a site called `example.com`, you could test if this is working by using the *netcat* program, like this:

```
nc example.com 25
```

Then, you should be greeted by Zotonic in the following way:

```
220 example.com ESMTP Zotonic 1.0.0
```

Press ctrl-c to exit.

Handling incoming E-mail

When receiving an e-mail message, Zotonic looks at the domain part of the e-mail address to determine which *Zotonic site* is configured to handle this message. It looks at the `host` and `hostalias` fields in the site's config file to match the recipient domain.

If no site matches the e-mails domain, the message is dropped, and a warning logged.

For handling incoming messages in your site, you need a hook in your site module to do something with the received messages, implementing the `email_receive` notification.

The code in your module looks like this:

```
-include_lib("kernel/include/logger.hrl").

observe_email_received(E, _C) ->
    ?LOG_WARNING("Email from: ~p: ~p", [E#email_received.from,
                                         E#email_received.email#email.subject]),
    ok.
```

Export this function and then restart your site. Now, send an e-mail message to any address `@example.com`, and notice that it arrives in Zotonic:

```
(zotonic001@host.local)9> 20:57:54.174 [warning] Email from: <<
↪"arjan@miraclethings.nl">>: <<"Hello!">>
```

Feedback

If you need feedback on messages that have been sent, enable *mod_logging* (page 378) which provides an overview of sent/received and bounced messages.

Troubleshooting

Check in the admin the log and smtp log. If a message bounces back to the Zotonic SMTP server, you will see errors there. A typical error looks like this:

```
SMTP: bounce: 504 5.5.2 <noreply+mlcm6godbz2cchtgdvom@oops>: Sender address_
↪rejected: need fully-qualified address To: piet@example.com (1234) From:
↪<noreply+mlcm6godbz2cchtgdvom@oops>
```

3.1.22 Command-line shell

See also:

Command-line (page 633)

The Zotonic shell gives you access to a running Zotonic instance with its code and data.

To connect to the background Zotonic server instance within an EShell (Erlang shell):

```
zotonic shell
```

Alternatively, use the *debug* command to launch the Zotonic server interactively and get an EShell on the running instance:

```
zotonic debug
```

The `start.sh` command in the root folder is a shortcut for this command.

Tab completion

The Zotonic shell has tab completion.

z_<tab> Lists Zotonic library modules.

m_<tab> Lists Zotonic models.

mod_<tab> Lists Zotonic modules.

Add a colon to get all available functions from a module, for instance: `m_rsc:<tab>`.

Often used commands

The `z` module provides shortcuts for the most used commands. From within the Zotonic shell:

z:m() . (Re)makes all erlang source modules and resets the caches.

z:flush(sitename) . or **z:flush()** . Resets all caches, reloads the dispatch rules and rescans all modules.

z:ld(modulename) . or **z:ld()** . Reloads an Erlang module, or reloads all changed Erlang modules

z:restart(sitename) . or **z:restart()** . Restarts the site, or restarts Zotonic

z:c(sitename) . Gets the current site context. This call is often used inside another function call - see examples below.

Activating/deactivating modules

z_module_manager:deactivate(mod_modulename, z:c(sitename)) . Deactivates a module.

z_module_manager:activate(mod_modulename, z:c(sitename)) . Activates a module.

Resetting a user password

Sometimes it happens that you want to reset an user's password from the Erlang shell. You can do this from the Erlang shell without using the admin or the reset-password mail/dialog.

m_identity:set_username_pw(1234, "username", "password", z:c(sitename)) .

Where 1234 is the rsc id of your user (1 for the admin), this must be an integer.

Accessing data

Sometimes it is desirable to access data stored in Zotonic from the EShell. This is useful for experimenting with code without having to go through an involved process or through HTTP requests to test it.

These calls use the site context as parameter. Get it using:

```
Context = z:c(sitename).
```

The `m_rsc` model module provides functions for retrieving and interacting with pages and other resources stored in Zotonic's datastore for the site.

RscProps = m_rsc:get(Id, Context). Returns the entire resource as a proplists structure. Id is the resource Id number.

```
Title = m_rsc:p(Id, title, Context).
```

You can also use the Erlang proplists module:

```
Title = proplists:get_value(title, RscProps).
```

Searching

To perform a search on the running site, use the `z_search` module.

rr(z_search). Loads all records defined in `z_search` (including those defined in include files such as the `#search_result` record).

Results = z_search:search(<<"latest">>, #{ <<"cat">> => text }, 1, 20, Context).
Returns the search result record, for the search on pages from the category *text*. You can specify the category as a string, binary, atom or integer.

To retrieve the list of pages, we access the `result` property of the record data:

```
Pages = Results#search_result.result.
```

Use `m_rsc:get(Id, Context)` to retrieve Page information of each search result (see above).

Debugging

We have added the `recon` application to the zotonic deps. This allows one to easily debug many aspects of a running zotonic node. It contains tools for tracing calls, check memory and bandwidth usage and a lot more. For more information see: [Stuff Goes Bad: Erlang in Anger](#).

3.1.23 Logging

See also:

[Logging to Logstash](#) (page 328) cookbook

See also:

To log messages to the database, you can use [mod_logging](#) (page 378).

Zotonic uses [Logger](#) for logging. Logger metadata is automatically set by Zotonic in the controller functions.

To log a message from your code, simply call `Logger` directly:

```
-include_lib("kernel/include/logger.hrl").  
  
?LOG_ERROR("Something went very wrong with ~p", [SomeParam]).
```

which will write the following in the `error.log` file:

```
2022-01-28 17:17:45 ERROR <0.9.0> [some_module:some_function/0:42] text=
↳ "Something went very wrong with whatever"
```

Configuration

In *The erlang.config file* (page 625) file, change the `kernel` section to configure log handlers and formatters. See the [Logger documentation](#) for more information.

For instance, to configure Logger for logging to Logstash, uncomment the `logstash` handler in the logger configuration of your `erlang.config` file and check the `logstash` configuration:

Listing 3.9: erlang.config

```
{kernel, [
    % Minimum log level for all loggers below.
    {logger_level, info},

    {logger, [

        %% To use logstash:
        %% - Enable the logstash_h handler
        %% - Configure logstash (see below the kernel config)
        %%
        % {handler, logstash, logstash_h,
        %     #{
        %         level => info
        %     }
        % },

        % Log of all information to the terminal/console.
        % The 'logger_level' above defines what is shown on the console.
        {handler, default, z_logger_h,
            #{
                level => debug,
                formatter => {z_logger_formatter,
                    #{
                        prettify => true,
                        colored => true,
                        time_designator => $\s,
                        time_offset => "",
                        time_unit => second,
                        strip_tz => true,
                        level_capitalize => true
                    }
                }
            }
        },

        % Error log on disk.
        % LOG_DIR is replaced with the default Zotonic ZOTONIC_LOG_DIR directory.
        {handler, error_log, z_logger_h,
            #{
                level => error,
                config => #{
                    type => file,
                    file => "LOG_DIR/error.log",
                    max_no_files => 10,
                    max_no_bytes => 52428800 % 10 x 5mb
                },
                formatter => {z_logger_formatter,
                    #{
```

```

        prettify => true,
        colored => false,
        time_designator => $\s,
        time_offset => "",
        time_unit => second,
        strip_tz => true,
        level_capitalize => true
    }
}

},

% Console log on disk.
% LOG_DIR is replaced with the default Zotonic ZOTONIC_LOG_DIR directory.
{handler, console_log, z_logger_h,
    #{
        level => debug,
        config => #{
            type => file,
            file => "LOG_DIR/console.log",
            max_no_files => 10,
            max_no_bytes => 52428800 % 10 x 5mb
        },
        formatter => {z_logger_formatter,
            #{
                prettify => true,
                colored => false,
                time_designator => $\s,
                time_offset => "",
                time_unit => second,
                strip_tz => true,
                level_capitalize => true
            }
        }
    }
}

}],

}],

%% Logstash configuration.
%% If a logger handler with 'logstasher_h' is defined then zotonic_core will start_
↳ the
%% logstasher application.
{logstasher, [
    {transport, console}, % tcp | udp | console
    {host, "localhost"}, % inet:hostname()
    {port, 5000} % inet:port_number()
}],
}
```

3.1.24 Testing sites

It is possible to create end-to-end integration tests for Zotonic websites. Tests like these are called *sitetests*. They run within the Zotonic shell, starting the website under test using a special database schema so that the main database is not affected by the tests.

To run the sitetests for a site called `example`, run the following command:

```
z_sitetest:run(example).
```

This will stop the `example` site, create a new database schema (called `z_sitetest`), start the site (which installs all the site's data, e.g. those defined by the [Module versioning](#) (page 64), into the new schema), and

then scans all compiled Erlang modules for modules named `example_*_sitetest.erl`. These found test modules are then run using Erlang's standard [EUnit test framework](#).

Note: The filename pattern of the tests is `(sitename)_(testname)_sitetest.erl`, where `sitename` is the name of the site under test (lowercase alphanumerics + underscores), and `testname` is a name for the test suite (lowercase alphanumerics only). `testname` *cannot* contain any underscores.

Besides running the tests from the Erlang shell, they can also be run from the terminal [Command-line](#) (page 633):

```
$ zotonic sitetest example
```

This is convenient for integration into CI systems.

Example sitetest module

Put the following inside the example site under the filename `tests/example_administrator_sitetest.erl`:

```
-module(example_administrator_sitetest).
-compile(export_all).

-include_lib("eunit/include/eunit.hrl").

sudo_context() ->
    z_acl:sudo(z:c(example)).

administrator_name_test() ->
    ?assertEqual(<<"Site Administrator">>, m_rsc:p(1, title, sudo_context())),
    ok.
```

Test Driven Development

Running the sitetests is integrated into the filewatcher hooks. As soon as you edit a `*_sitetest.erl` file, all tests in that specific sitetest file will be executed.

While developing a site it is often handy to continuously run the tests while you are developing the site.

To establish this, you can call `z_sitetest:watch(site)` to start watching all Erlang files inside the site under development. When the filetest is watching a site, all filetests are triggered as soon as any Erlang module inside your site is being recompiled.

To stop the automatic test running again, call `z_sitetest:unwatch(site)`.

Example testing output

Running the test command `z_sitetest:run(example) .`, will produce output similar to the following:

```
(zotonic001@host)33> z_sitetest:run(example).
15:45:18.162 [info] Site stopped: example (<0.17763.0>)
15:45:18.682 [warning] [example] Database connection failure: noschema
15:45:18.688 [warning] [example] Creating schema "z_sitetest" in database "example"
15:45:18.693 [info] [example] Retrying install check after db creation.
15:45:18.702 [warning] [example] Installing database with db options: [{dbschema,
↳ "z_sitetest"}, {dbdatabase, "example"}, {dbhost, "localhost"}, {dbport, 5432}, {dbuser,
↳ "zotonic"}]
15:45:19.226 [info] example: Install start.
15:45:19.226 [info] Inserting categories
15:45:19.257 [info] Inserting base resources (admin, etc.)
```

```

15:45:19.264 [info] Inserting username for the admin
15:45:19.267 [info] Inserting predicates
15:45:19.290 [info] example: Install done.
15:45:19.299 [info] Site started: example (<0.22082.0>)
15:45:19.575 [info] [example] info @ z_datamodel:169 Creating new category 'menu'
15:45:19.604 [info] [example] info @ z_datamodel:169 Creating new menu 'main_menu'
15:45:19.723 [info] [example] info @ z_datamodel:169 Creating new category
↳ 'content_group'
15:45:19.755 [info] [example] info @ z_datamodel:169 Creating new content_group
↳ 'system_content_group'
15:45:19.775 [info] [example] info @ z_datamodel:169 Creating new content_group
↳ 'default_content_group'
15:45:19.893 [info] [example] info @ z_datamodel:169 Creating new category 'acl_
↳ user_group'
15:45:19.959 [info] [example] info @ z_datamodel:169 Creating new category 'acl_
↳ collaboration_group'
15:45:20.001 [info] [example] info @ z_datamodel:169 Creating new predicate
↳ 'hasusergroup'
15:45:20.033 [info] [example] info @ z_datamodel:169 Creating new predicate
↳ 'hascollabmember'
15:45:20.044 [info] [example] info @ z_datamodel:169 Creating new predicate
↳ 'hascollabmanager'
15:45:20.059 [info] [example] info @ z_datamodel:169 Creating new acl_user_group
↳ 'acl_user_group_anonymous'
15:45:20.071 [info] [example] info @ z_datamodel:169 Creating new acl_user_group
↳ 'acl_user_group_members'
15:45:20.079 [info] [example] info @ z_datamodel:169 Creating new acl_user_group
↳ 'acl_user_group_editors'
15:45:20.086 [info] [example] info @ z_datamodel:169 Creating new acl_user_group
↳ 'acl_user_group_managers'
15:45:20.393 [info] [example] info @ z_datamodel:169 Creating new category 'admin_
↳ content_query'
===== EUnit =====
example_testone_sitetest: administrator_name_test (module 'example_administrator_
↳ sitetest')...[0.022 s] ok
=====
Test passed.
ok

```

3.1.25 Deployment

So you have built your Zotonic site, and now you want to show it to the world. This page tells you how to configure your zotonic environment so that it is ready for real-world website visitors.

As per the Installation Instructions, up until now you probably have always started Zotonic using the `zotonic debug` command. This is fine for debugging purposes, because it gives you an Erlang shell in which you can view the output of the server, the `?DEBUG` messages that are triggered, and try out Erlang expressions.

However for a production system, you don't need this shell, you want Zotonic running in the background, and started at startup.

This manual describes the various ways how Zotonic can be run and how it works in combination with widely used HTTP frontends like Varnish and nginx.

Table of contents

Automatic startup on system boot

Once you have Zotonic running, you want to make sure that it automatically starts up when the server reboots, so that the website keeps running after a power failure, for example.

Creating an init script

The *Zotonic shell command* (page 633) can start Zotonic in the background and stop it again.

An init script will just need to call the zotonic command with either `start` or `stop`. On debian systems it might look like this:

```
#!/bin/sh -e

### BEGIN INIT INFO
# Provides:          zotonic
# Required-Start:    $local_fs $remote_fs $network $time postgresql
# Required-Stop:     $local_fs $remote_fs $network $time postgresql
# Should-Start:
# Should-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Zotonic
### END INIT INFO

# Set any environment variables here, e.g.:
#export ZOTONIC_IP="127.0.0.1"

/usr/bin/sudo -u zotonic -i /home/zotonic/zotonic/bin/zotonic $@
```

For other environment variables which can be set, see *Useful environment variables* (page 94).

Server configuration

See also:

- See *Exometer metrics* (page 330) on how to monitor your Zotonic system.
- For general Erlang troubleshooting, Fred Hebert's free ebook *Stuff goes bad: Erlang in anger* is a very good resource.

This chapter describes how to configure your Linux server for running Zotonic.

File descriptors

By default, Linux limits the number of file descriptors (i.e. the maximum number of open files) at 1024. Running Zotonic at scale will require many more than that, particularly because Zotonic uses *WebSockets* (page 446) extensively; remember that every open port is an open file.

The limit applies on several levels:

1. the process, in our case the Erlang VM (BEAM) that runs Zotonic
2. the Zotonic user
3. system-wide.

To check the current process usage, find BEAM's PID:

```
# pidof beam.smp
10006 (for instance)
```

Then count its open files:

```
# lsof -a -p 10006 | wc -l
```

And compare it to the process limit:

```
# cat /proc/10006/limits | grep 'Max open files'
```

You can raise the process limit by adding `ulimit -n` to your Zotonic init script:

```
ulimit -n 50000
```

Or change the limit in your system definition. For instance, when using `systemd`:

```
[Service]
LimitNOFILE=50000
```

Finally, make sure to check [your system-wide limits](#), too.

Network connections

If you have stateful connection tracking enabled, high-traffic Zotonic sites may [overflow your conntrack table](#).

Compare the current number of connections with the limit:

```
# sysctl net.netfilter.nf_conntrack_max
# sysctl net.netfilter.nf_conntrack_count
```

When you increase the maximum number of connections in the connection tracking table it is important to increase the size of the table which stores the connections. The rule of the thumb for this is: `nf_conntrack_buckets = nf_conntrack_max / 8`.

If you need to raise the limit, edit `/etc/sysctl.conf`:

Listing 3.10: `/etc/sysctl.conf`

```
net.netfilter.nf_conntrack_max = some_number
net.netfilter.nf_conntrack_buckets = some_other_number
```

Moreover, to reduce the number of open connections, you can decrease their time-out values so they get closed sooner. By default, inactive connections can stay in the conntrack table for 5(!) days. To change this:

Listing 3.11: `/etc/sysctl.conf`

```
net.netfilter.nf_conntrack_tcp_timeout_established = 600
```

If you have a proxy in front of Zotonic (e.g. HAProxy or Varnish), you need to change the limits on the proxy, too.

Running on Port 80 and Port 443

Using standard ports helps visitors discover your page and removes the awkward port number from URLs.

The HTTP and HTTPS protocols are normally served on TCP ports 80 and 443. It is beneficial to run a these services on those standard ports: it aids discovery and lends a air of polish while a non-standard port number suggests something is incomplete.

*nix systems only allow the superuser (root) to bind to ports below 1024. HTTP & HTTPS use ports 80 & 443 respectively. So setting up Zotonic to serve from those ports without running as superuser presents a problem, as *nix considers all ports below 1024 to be “privileged”, requiring special access.

Other webservers (like nginx) typically do not have this problem, as they usually run their main unix process as root, but forking off child processes as non-privileged workers. Zotonic cannot be made to work like that because it is just one unix process, and running Zotonic entirely as the root user is considered harmful.

This manual outlines three different methods to let Zotonic listen on port 80. All of them are for *nix based systems only.

Pre-launch notes

Before Zotonic serves its sites on a privileged ports, the hostname portions of your Zotonic sites need to be changed to reflect this.

For production release of your new Zotonic site you need to:

- Make sure that your server has a public IP address, and that the server accepts connections on port 80.
- For each of your Zotonic sites, configure their DNS (e.g. *www.mysite.com*) to point to your server's IP address.
- Ensure `{hostname, "mysite"}` is set to `{hostname, "www.mysite.com"}` in `apps_user/mysite/priv/zotonic_site.config`. This last change enables the virtual hosting: it makes sure that Zotonic knows which site is being requested when somebody visits *www.mysite.com*.

Note: Your actual site location might be different, see the [zotonic user directory](#).

Running behind another web server / proxy

You run another web server to proxy requests from port 80 to 8000 and from 443 to 8443. [Varnish](#) (page 89) and [nginx](#) (page 92) are both very capable web servers for doing this.

However, Zotonic itself is also very capable of directly serving web content without needing an extra caching layer in between. The other methods listed below explain how Zotonic can obtain direct access to the privileged ports.

Using authbind

Note: Instructions below assume site is named *mysite* and Zotonic is installed in `/home/zotonic/zotonic`. Replace as appropriate.

Install authbind:

```
zotonic:~$ sudo apt-get install authbind
```

Configure authbind to allow zotonic user to access port 80 and 443:

```
zotonic:~$ sudo install -m 500 -o zotonic /dev/null /etc/authbind/byport/80
zotonic:~$ sudo install -m 500 -o zotonic /dev/null /etc/authbind/byport/443
```

When you want to run zotonic's incoming mail server on port 25, you can make a similar authbind configuration for this port.

Set up the environment.

Make sure zotonic will listen on port 80 and 443. Either by setting this in the environment variables described below, or configuring it in zotonic's `zotonic.config` file.

Add the following entries to `/home/zotonic/.profile`, then save file & exit:

```
export ZOTONIC_LISTEN_PORT=80
export ZOTONIC_SSL_LISTEN_PORT=443
```

Source the file to update the environment:

```
zotonic:~$ . ~/.profile
```

Or put the following entries in zotonic's `zotonic.config` file:

```
{listen_port, 80},
{ssl_listen_port, 443}
```

Stop zotonic if already running:

```
zotonic:~$ ~/zotonic/bin/zotonic stop
```

Start zotonic:

```
zotonic:~$ authbind --deep ~/zotonic/bin/zotonic start
```

Browse to <https://yoursite/> and verify that everything is working like it should.

Using setcap

Warning: this is a much broader approach as it grants privileged bind to all Erlang VM processes (the beam and beam.smp executables). Unless you are the sole user of such a machine this is not a great idea.

From a shell, install the setcap program:

```
sudo apt-get install libcap2-bin
```

Now configure setcap to allow Erlang BEAM processes user to bind to ports lower than 1024:

```
sudo setcap 'cap_net_bind_service=+ep' /usr/lib/erlang/erts-5.9.2/bin/beam
sudo setcap 'cap_net_bind_service=+ep' /usr/lib/erlang/erts-5.9.2/bin/beam.smp
```

Note that the exact paths to the beam and beam.smp can be different, depending on the Erlang version.

During package upgrades Erlang may be upgraded and your site will seem to be broken. Just make sure to check the ERTS version and rerun these setcaps commands for the new version.

For more granular control, you could create an Erlang release that only the Zotonic User can access. Once the release is created setcap could be applied to the beam and beam.smp within that release only.

Using iptables

If authbind and setcap will not work for you, using the system firewall to redirect the ports can be an option.

Firewall prerouting can be enabled as follows to forward communication on port 80 to port 8000 and port 443 to port 8443:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to 8000
iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to 8443
```

You also need two more rules so that the site can reach itself. In the following firewall rules, replace your.ip.address with your external IP address:

```
iptables -t nat -A OUTPUT -p tcp -d your.ip.address --dport 80 -j REDIRECT --to_
↪8000
iptables -t nat -A OUTPUT -p tcp -d your.ip.address --dport 443 -j REDIRECT --to_
↪8443
```

The downside of using the firewall is that Zotonic still also listens on port 8000. This might be a cause for confusion.

For instructions on how to save these firewall rules and reinstate them after a system reboot, consult the [Ubuntu firewall administration manual](#).

HTTPS support

Zotonic has built-in support for HTTPS and TLS (previously SSL) certificate handling.

All sites share a single TLS port using Server Name Indication for virtual hosting and certificate selection.

The following guides are useful:

Running on Port 80 and Port 443 (page 86) Let connect to the default https (443) and http (80) ports.

Port configurations (page 630) Configure all Zotonic listen ports, depending on your server configuration and proxy usage.

Proxying Zotonic with nginx (page 92) If you want to use nginx as a terminating proxy for SSL connections.

mod_ssl_letsencrypt (page 397) Automatically fetch free SSL certificates from Let's Encrypt.

mod_ssl_ca (page 395) Use SSL certificates obtained from a Certificate Authority.

Using Varnish as frontend for Zotonic

Using the [Varnish HTTP frontend](#), you can speed up your Zotonic even more as this web server caches static files intelligently.

Your Varnish `config.vcl` needs to define a *backend* for Zotonic:

```
backend zotonic {
    .host = "127.0.0.1";
    .port = "8000";
    .first_byte_timeout = 300s;
    .connect_timeout = 300s;
    .between_bytes_timeout = 300s;
}
```

Then, in `vcl_recv`, specify the Zotonic backend as the default backend:

```
sub vcl_recv {
    set req.http.X-Forwarded-Host = req.http.host;
    set req.backend = zotonic;

    ...
}
```

Full varnish example configuration file

Please see the [Varnish documentation](#) for more information.

```
#
# Varnish configuration for a Zotonic site
#

backend zotonic_com {
    .host = "127.0.0.1";
    .port = "8000";
    .first_byte_timeout = 300s;
    .connect_timeout = 300s;
    .between_bytes_timeout = 300s;
}

backend nginx {
    .host = "127.0.0.1";
    .port = "8080";
    .first_byte_timeout = 300s;
```

```
.connect_timeout = 300s;
.between_bytes_timeout = 300s;
}

sub vcl_recv {
    set req.http.X-Forwarded-Host = req.http.host;

    #####
    ##### VIRTUAL HOSTS #####

    if (req.http.host ~ "(www.|)(zotonic|example).(com|net)$") {
        set req.backend = zotonic_com;
    }
    # Nginx hosted sites
    #
    else {
        set req.backend = nginx;
    }

    #####

    # Add a unique header containing the client address
    unset req.http.X-Forwarded-For;
    set req.http.X-Forwarded-For = client.ip;

    # We only deal with GET and HEAD by default
    if (req.request != "GET" && req.request != "HEAD") {
        return (pass);
    }

    # Cache the home page for a short period (ttl = 1 second, see vcl_fetch)
    if (req.url ~ "^/$") {
        unset req.http.Cookie;
        unset req.http.Authenticate;
        set req.grace = 10s;
        return (lookup);
    }

    # Cache served css and media files
    if (req.url ~ "^/(lib|image|media|favicon.ico)/") {
        unset req.http.Cookie;
        unset req.http.Authenticate;
        set req.grace = 30m;
        return (lookup);
    }

    return (pass);
}

sub vcl_pipe {
    # Note that only the first request to the backend will have
    # X-Forwarded-For set. If you use X-Forwarded-For and want to
    # have it set for all requests, make sure to have:
    set req.http.connection = "close";
    return (pipe);
}

sub vcl_pass {
    if (req.url ~ "^/comet") {
        #breq.connect_timeout = 70;
    }
}
```

```

    #breq.first_byte_timeout = 70;
}
return (pass);
}

sub vcl_fetch {
    if (req.url ~ "^/(lib|image|media|favicon.ico)/") {
        unset beresp.http.Set-Cookie;
        set beresp.grace = 30m;
        set beresp.ttl = 10m;
        return (deliver);
    }
    return (pass);
}

sub vcl_error {
    if (obj.status == 503 && req.restarts < 4) {
        restart;
    }
}

```

Auto-starting Varnish on Mac OSX

To automatically start Varnish on Mac OSX, add the following plist file to launchd.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Disabled</key>
    <false/>
    <key>KeepAlive</key>
    <false/>
    <key>Debug</key>
    <false/>
    <key>Label</key>
    <string>varnishd</string>
    <key>OnDemand</key>
    <false/>
    <key>GroupName</key>
    <string>wheel</string>
    <key>UserName</key>
    <string>root</string>
    <key>ProgramArguments</key>
    <array>
        <string>/usr/local/sbin/varnishd</string>
        <string>-F</string>
        <string>-a</string>
        <string>:80</string>
        <string>-T</string>
        <string>localhost:6082</string>
        <string>-f</string>
        <string>/usr/local/etc/varnish/varnish.zotonic.vcl</string>
        <string>-s</string>
        <string>malloc</string>
        <string>-u</string>
        <string>nobody</string>
    </array>
</dict>
</plist>

```

```
</array>
<key>RunAtLoad</key>
<false/>
</dict>
</plist>
```

Proxying Zotonic with nginx

It is possible to put Zotonic behind the *nginx* <<http://nginx.org/>> web server, for example if you have other, non-Zotonic virtual hosts running on your system.

When proxying, don't forget to check the config files of the sites you are planning to server (the `apps_user/your-site/priv/zotonic_site.config` files). The `hostname` value should not contain any port number, if you run from port 80/443: `{hostname, "test.zotonic.com"}`.

Zotonic configuration

Example of `zotonic.config` (find the location using `bin/zotonic configfiles`) ip/port settings when terminating SSL in nginx proxy and using plain HTTP towards the Zotonic backend:

```
%%% IP address on which Zotonic will listen for HTTP requests.
{listen_ip, any},

%%% Port on which Zotonic will listen for HTTP requests.
{listen_port, 8000},

%%% Port on which Zotonic will listen for HTTPS requests.
%%% Set to the atom 'none' to disable SSL
{ssl_listen_port, none},

%%% Outside port on which Zotonic will listen for HTTP requests.
{port, 80},

%%% Outside port zotonic uses to receive incoming HTTPS requests.
{ssl_port, 443},
```

Nginx configuration

Below is an example configuration file to proxy nginx to zotonic. Be sure to replace all occurrences of `test.zotonic.com` with your own hostname:

```
server {
    listen 80;
    listen [::]:80 default_server ipv6only=on; ## listen for ipv6

    listen 443 ssl http2;
    listen [::]:443 ssl http2 ipv6only=on;

    server_name test.zotonic.com;

    access_log /var/log/nginx/test.zotonic.com.access.log;
    error_log /var/log/nginx/test.zotonic.com.error.log;

    keepalive_timeout 65;
    gzip off;

    ssl_protocols TLSv1.2 TLSv1.3;
```



```

ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH:ECDHE-RSA-
↪AES128-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA128:DHE-
↪RSA-AES128-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES128-GCM-SHA128:ECDHE-
↪RSA-AES128-SHA384:ECDHE-RSA-AES128-SHA128:ECDHE-RSA-AES128-SHA:ECDHE-RSA-AES128-
↪SHA:DHE-RSA-AES128-SHA128:DHE-RSA-AES128-SHA128:DHE-RSA-AES128-SHA:DHE-RSA-
↪AES128-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA384:AES128-
↪GCM-SHA128:AES128-SHA128:AES128-SHA128:AES128-SHA:AES128-SHA:DES-CBC3-SHA:HIGH:!
↪aNULL:!eNULL:!EXPORT:!DES:!MD5:!PSK:!RC4";
ssl_prefer_server_ciphers on;

# Disable preloading HSTS for now. Enable when you know that your
# server certs works. You can use the header line that includes
# the "preload" directive if you understand the implications.
# add_header Strict-Transport-Security "max-age=15768000; includeSubdomains";
↪ # six months
# add_header Strict-Transport-Security "max-age=15768000; includeSubdomains;
↪preload";

ssl_session_cache shared:SSL:10m;
ssl_session_timeout 5m;

# create with: openssl dhparam -out /etc/nginx/dhparam.pem 2048
# ssl_dhparam /etc/nginx/dhparam.pem;

ssl_certificate /path/to/ssl.crt;
ssl_certificate_key /path/to/ssl.key;

location / {
    proxy_pass http://127.0.0.1:8000/;
    proxy_redirect off;

    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_pass_request_headers on;

    client_max_body_size 0;
    client_body_buffer_size 128k;

    proxy_connect_timeout 90;
    proxy_send_timeout 90;
    proxy_read_timeout 90;

    proxy_buffer_size 4k;
    proxy_buffers 4 32k;
    proxy_busy_buffers_size 64k;
    proxy_temp_file_write_size 64k;
}

location /mqtt-transport {
    proxy_pass http://127.0.0.1:8000/mqtt-transport;

    proxy_http_version 1.1;

    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";

    proxy_pass_request_headers on;

```

```
}

location /close-connection {
    keepalive_timeout 0;
    empty_gif;
}

}
```

As shown in the example above:

- Add `X-Forwarded-Proto` to proxied header so that Zotonic knows that HTTPS is used before the proxy even though HTTP is used between the proxy and backend.
- Add `X-Real-IP` and `X-Forwarded-For` headers.

Zotonic always redirects to HTTPS so the proxy needs to be configured for both HTTP and HTTPS.

Zotonic makes use of a websocket connection for MQTT messages at the `/mqtt-transport` endpoint, so you need to pass the `Upgrade` and `Connection` headers.

The `/mqtt-transport` endpoint is also used to POST uploaded files using a HTML multi-part form post.

See the [nginx documentation](#) for more information on its configuration procedure.

Useful environment variables

The following environment variables influence how Zotonic starts up.

ZOTONIC_IP The IPv4 address to bind the web server to. By default, it binds to any IP address. If Zotonic runs behind a proxy like nginx or Varnish, it is wise to put `127.0.0.1` or `localhost` here. Use `any` to bind to all IPv4 addresses, `none` to disable the IPv4 web server.

ZOTONIC_IP6 The IPv6 address to bind the web server to. By default it binds to the `ZOTONIC_IP` address. If Zotonic runs behind a proxy like nginx or Varnish, it is wise to put `::1` here. Use `any` to bind to all IPv6 addresses, `none` to disable the IPv6 web server.

ZOTONIC_PORT Outside port that clients send HTTP requests to. Defaults to `ZOTONIC_LISTEN_PORT`. See [Port configurations](#) (page 630).

ZOTONIC_SSL_PORT Outside port that clients send HTTPS requests to. Defaults to `ZOTONIC_SSL_LISTEN_PORT`. Use `none` to disable the ssl web server. See [Port configurations](#) (page 630).

ZOTONIC_LISTEN_PORT Port on which Zotonic will listen for HTTP requests. Defaults to port 8000. See [Port configurations](#) (page 630).

ZOTONIC_SSL_LISTEN_PORT Port on which Zotonic will listen for HTTPS requests. See [Port configurations](#) (page 630).

ZOTONIC_SMTP_LISTEN_IP The IPv4 address to bind the SMTP server to. Binds to any IP address by default. Use `none` to disable the SMTP server. Use `any` to bind to all IP addresses.

ZOTONIC_SMTP_LISTEN_PORT The port number to bind the SMTP server to. Defaults to port 2525. Use `none` to disable the SMTP server.

ZOTONIC_SMTP_LISTEN_DOMAIN The domain to bind the SMTP server to, if any.

ZOTONIC_CONFIG_DIR The directory with the configuration files. Possible locations are:

- The init argument `zotonic_config_dir`
- The environment variable `ZOTONIC_CONFIG_DIR`
- The directory `$HOME/.zotonic`
- The directory `/etc/zotonic` (only on Unix)
- The OS specific directory for application config files

The OS specific directories are:

- On Unix: `~/.config/zotonic/config/`
- On macOS: `~/Library/Application Support/zotonic/config/`

In those directories the system searches for a `zotonic*` file in the following subdirectories (assuming the version of Zotonic is 1.2.3 and the node is called `zotonic001@foobar`):

- `zotonic001@foobar/`
- `zotonic001/`
- `1.2.3`
- `1.2`
- `1`
- `.`

The default is the OS specific directory, with as subdirectory the major version number of Zotonic (in this case 1). For Linux this would be `~/.config/zotonic/config/1/`

ZOTONIC_SECURITY_DIR The directory to store the certificates and other security related data. Possible locations are:

- The environment variable `ZOTONIC_SECURITY_DIR`
- The `~/.zotonic/security` directory
- The `/etc/zotonic/security` directory (only on Linux)
- The OS specific directory for application data files

The OS specific directories are:

- On Unix: `~/.config/zotonic/security/`
- On macOS: `~/Library/Application Support/zotonic/security/`

The default is the OS specific directory.

ZOTONIC_DATA_DIR The directory to store the data files. Possible locations are:

- The environment variable `ZOTONIC_DATA_DIR`
- The data directory in the Zotonic directory
- The OS specific directory for application data files

The OS specific directories are:

- On Unix: `~/.local/share/zotonic`
- On macOS: `~/Library/Application Support/zotonic/`

The default is the OS specific directory.

ZOTONIC_LOG_DIR The directory to store the log files. Possible locations are:

- The environment variable `ZOTONIC_LOG_DIR`
- The `logs` directory in the Zotonic directory
- The OS specific directory for application log files

The OS specific directories are:

- On Unix: `~/.cache/zotonic/log/`
- On macOS: `~/Library/Logs/zotonic/`

The default is the OS specific directory.

ZOTONIC_APPS The directory used for sites, modules and additional OTP applications. This defaults to `apps_user` in the Zotonic umbrella application.

If a separate checkouts directory is used, then this environment variable must be:

- Defined when building Zotonic with `make compile` or `./rebar3 compile`
- Defined when starting Zotonic

ZOTONIC_PIDFILE Path to zotonic PID file. If set, Zotonic will create it at start and remove it before exit. Note however that Erlang VM is not stopped at this moment and can last for several seconds. See `ZOTONIC_WAIT_VM` to get rid of this.

SNAME The *short name* of the Zotonic Erlang node. This defaults to `zotonic`. If a short name is defined then the Erlang node is started with `-sname`. The name can be like `zotonic@foobar`, but the part after the `@` may not have a dot (`.`), as then it is a long name.

LNAME The *long name* of the Zotonic Erlang node. This defaults to `zotonic`. If a long name is defined then the Erlang node is started with `-name`. The name can have the domain defined, for example: `zotonic@foo.bar.com`. The part after the `@` **must** be a fully qualified domain name. Zotonic will use the OS's domain name if no domain is defined in the `LNAME`.

TMP Where Zotonic puts temporary files. Examples are temporary files for image resizing or URL downloading.

ERL_MAX_PROCESSES Maximum of processes in Erlang, defaults to 500000.

ERL_MAX_ATOMS Maximum number of atoms in Erlang, defaults to 1048576.

ERL_MAX_PORTS Maximum number of ports for the Erlang VM, defaults to the output of `ulimit -n`. If `ulimit` returns `unlimited` then 100000 is used.

ERL_KERNEL_POLL How the kernel polls for i/o. Defaults to `true`. Valid values are `true` and `false`.

The following environment variables influence how Zotonic stops.

ZOTONIC_WAIT_VM If set to 1, ask launcher script to wait for total stop of Zotonic Erlang VM before exit. This can be used to ensure all resources are freed before trying a new start.

3.1.26 Troubleshooting

Installation

Zotonic won't start and shows errors when running zotonic debug

Check your site's database configuration.

Check PostgreSQL Authentication Config (`pg_hba.conf` in `/etc/postgresql`).

If you get connection failures when starting Zotonic you should double-check `pg_hba.conf` and make sure to `/etc/init.d/postgresql reload` to make sure it gets loaded.

The `pg_hba.conf` should contain something like:

# IPv4 local connections:			
host	all	all	127.0.0.1/32 md5
# IPv6 local connections:			
host	all	all	:::1/128 md5

Unable to read boot script

In some cases the Zotonic nodename does not resolve well and you see an error like:

```
[error] Unable to read boot script (start_sasl.script): {error,enoent}
```

or:

```
erl_call: can't ei_gethostbyname(MacBook-Pro-Apple)
Zotonic is not running. You need to start Zotonic first to use this command.
```

Solution: Add the computer name to `/etc/hosts`, for instance:

```
127.0.0.1 MacBook-Pro-Apple
```

Erlang crashes

Here we list some common causes of Erlang crashes.

```
{{badmatch, {error, emfile}} or {reason, emfile}}
```

You need to raise the *File descriptors* (page 85) limit.

Sites

Site doesn't start

Check your database connection configuration in the `zotonic_site.config` file which is located in the *zotonic user directory*. This can be found in `yoursite/priv/zotonic_site.config`. The `priv` directory should also be soft linked in the `_build` directory: `_build/default/lib/yoursite/priv/zotonic_site.config`

Browsers can't connect to `http://yoursite.test:8000`

Check the `/etc/hosts` file and make sure you have an entry like the following:

```
127.0.0.1    yoursite.test
```

Browsers give self-signed certificate warning

If you connect to `https://yoursite.test:8443` the browser warns that the site is not trusted. This is correct, as the certificate is self-signed.

You can connect if you accept the certificate. For Safari and Firefox just follow the instructions shown to you by the browser.

For Chrome you have to do something silly:

- Click anywhere in the white on the window
- Type: `thisisunsafe`

You won't see any characters echo to the screen. After typing the magic strings the page should reload and the site certificate is accepted.

3.1.27 Contributing to Zotonic

We encourage contributions to Zotonic from the community! This chapter describes how you can help improve Zotonic.

1. Fork the `zotonic` repository on Github (at <https://github.com/zotonic/zotonic>).
2. Clone your fork or add the remote if you already have a clone of the repository:

```
git clone git@github.com:yourusername/zotonic.git
```

or:

```
git remote add mine git@github.com:yourusername/zotonic.git
```

3. Create a topic branch for your change:

```
git checkout -b some-topic-branch
```

4. Make your change and commit. Use a clear and descriptive commit message, spanning multiple lines if detailed explanation is needed.

5. Push to your fork of the repository and then send a pull-request through Github:

```
git push mine some-topic-branch
```

6. A Zotonic committer will review your patch and merge it into the main repository or send you feedback. The pull request page on github is the main place to discuss the code and related topics.

Coding standards

The Zotonic code follows [Inaka's Erlang Coding Guidelines](#). This is enforced using the [Elvis](#) code checking tool. You can check the code locally with:

```
$ elvis rock
```

When contributing, simply try to follow the coding style as you find it in the existing code.

Emacs

Provided with the Zotonic distribution is a Zotonic template mode, `zotonic-tpl-mode`, which supports the Zotonic flavor of Django. It is located in the `priv/emacs/zotonic-tpl-mode.el` file, and may be installed in emacs by adding something like this to your `.emacs` file:

```
(add-to-list 'load-path ".../path/to/zotonic/priv/emacs")
(require 'zotonic-tpl-mode)
;; optional, for associating .tpl files with zotonic-tpl-mode
(add-to-list 'auto-mode-alist '("\\.tpl$" . zotonic-tpl-mode))
```

Writing commit messages

The Zotonic commit convention are slightly based on [rebar's README](#).

Structure your commit message like this:

```
prefix: One line summary (less than 50 characters)

Longer description, multiline text that is supposed to wrap at 72
characters.

Fixes #403
```

- **Prefix:** Every commit message must start with one of the designated commit prefixes:
- `mod_foobar`: Changes that are related to a single module should be prefixed with the module name.
- `doc`: For changes to the documentation, everything below `doc/`
- `scripts`: for changes to the `zotonic` command and its helper scripts.

- `build`: for the build system and related changes.
- `tests`: for unit tests and the testsandbox.
- `skel`: for the skeleton sites.
- `zotonic_status`: for the default site.
- `translation`: for new/updated translations.
- `core`: For changes in the `apps/zotonic_core` directory; i.e., anything not covered by another tag.
- The **summary** should be less than 50 characters, and tell what was changed. Use the imperative present tense (fix, add, change). For example: *Add 'foobar' filter, Fix bug in media upload service.*
- The **description** should explain the intention and implementation of your approach, in the present tense.
- Optionally, when your commit **fixes** a bug on github, add *Fixes #1545* on a separate line below the description.

Notice the empty line preceding the longer description and the “Fixes” tag.

Git best practices

- Please maintain commit atomicity by breaking up logical changes into separate commits; e.g., do not commit unrelated fixes into a single commit.
- Make whitespace changes separately.
- When updating from the Zotonic source, please use `git pull --rebase` to prevent unnecessary merge commits.
- Generally, try to [Mind your Git Manners](#).

The CONTRIBUTORS file

When this is your first contribution to Zotonic, you are welcome to add your name and e-mail address to the CONTRIBUTORS file in the root of the project. Please keep the file alphabetically ordered.

Running the tests

Zotonic comes with a basic test suite which can be run the following way:

```
zotonic runtests
```

This starts the Zotonic system and executes all EUnit tests. It will disable all zotonic sites except for the special site `testsandbox`, which will be enabled.

The `testsandbox` site does not have a database configuration and is configured to run on `localhost:8040`.

Contributing documentation

Build the documentation

First, install [Sphinx](#). To build the documentation, Erlang must be installed.

```
$ cd doc/

# Install dependencies
$ pip install -r requirements.txt

# Generate meta-*.rst files:
```

```
$ make stubs

# Then generate HTML files:
$ make html
```

Then view the HTML files in `doc/_build/html/index.html`.

Heading styles

Use the following [convention](#) for headings:

```
First-level heading
=====

Second-level heading
-----

Third-level heading
^^^^^^^^^^^^^^^^^^^^

Fourth-level heading
""""""""""""""""""
```

When writing documentation of modules, actions, etc.; anything under `ref/`; the first level heading is already there for you, generated in the `meta-*.rst` file. So you should only use `-----` and `^^^^^^^^` for the headings in the `ref/` files.

When using Emacs, this little snippet helps with adding underlines to headings:

```
(defun underline-with-char (char)
  (interactive (list (read-from-minibuffer "Char: ")))
  (when (= 0 (length char))
    (error "Need a character"))
  (setq char (aref char 0))           ; Ignore everything but the first char.
  (save-excursion
    (goto-char (point-at-eol))
    (insert "\n"
            (make-string (- (point-at-eol)
                           (point-at-bol))
                          char))))
```

From a mailing list [post](#).

References

Be generous with using references (`:ref: `pagelabel``) in your writing. The more terms are linked to their respective documentation pages, the better. Only make the first occurrence of a term a reference to its page, though; consequent occurrences can be made ``italic``.

Add a `.. seealso::` section at the top to highlight any other pages which are closely related to the current one, for example:

```
.. code-block:: none
```

See also:

[Contributing to Zotonic](#) (page 97)

Table styles

For the easy editing of tables, we use Emacs' [table-mode](#), which at first has a bit of a learning curve but actually works pretty well when creating the ascii-art tables that the RST format requires you to use.

In general, we use this style of table:

Header	Other header	
This is the table	Some more contents	
cell contents		

Writing consistent Cookbook items

A Zotonic Cookbook item is a single-concept solution to a well-defined problem, living in the [Cookbooks](#) (page 267) section of the documentation.

Useful items range from the simplest content management tasks to technically sophisticated module development and site administration solutions. This means that items are welcomed from noobies and wizards alike.

Whenever you struggle to find a solution to a specific problem, fail to find a Cookbook item that addresses it, and work through the solution with a final “Aha!,” you have the raw material for an excellent Cookbook submission.

A well-written item has four sections:

WHY: What problem does this Cookbook item solve? What benefits does it deliver?

Four major reasons for submitting Cookbook items are:

1. The best way to learn is to teach
2. Your Cookbook items documents your efforts; helps you remember what you did next time you encounter a similar problem
3. Each item makes it that much easier for noobies and other community members to advance their Zotonic skills.

ASSUMPTIONS: What does this item assume about operating system, Linux distribution, programming skills, knowledge of Zotonic architecture and conventions etc.

HOW: Step-by-step instructions for implementing your solution.

Don't take user competency for granted. When you specify a command, note what user name you're working under and what directory you are working in. Respect the noobies by including steps that may be obvious to you but not so obvious to folks with less experience.

Think of your instructions as a check-list. A noobie should be able to achieve success by reading, implementing and checking off each instruction. Keep your instructions simple, complete, and clear.

Recruit a noobie to try out your solution. Fix the stumbling blocks s/he encounters. If you can't find a noobie, put yourself in noobie mind. Remember, you too once were one.

Zotonic releases

See also:

[GitHub](#) for the latest release.

Release dates

Zotonic follows a time-based release model. Every first Monday of the month – at the call of the [Dutch test siren](#) – a new Zotonic version is released. Version numbers are incremented according to the [Semantic versioning](#) specification.

Release schedule

Preparation for each release lasts one month:

1. **Development phase:** new features are added and existing ones improved. Commits take place on the current `.x` development branch (for instance, `0.x`).
2. **Stabilisation phase:** five working days before a release, we create a release branch from the development branch, incrementing the minor version number (for instance, `release-0.16.0`). During the stabilisation phase, no new features are added. Instead, the last bug fixes for the release are committed.
3. On the first Monday of each month, the release branch is **tagged** (for instance, `0.16.0`), merged back into the development branch and then discarded.

Hotfix releases

Some bug fixes, such as security fixes, need to be made available immediately. In case a change cannot wait for the next monthly release, we release it as a hotfix, incrementing the patch number (for instance, `0.16.1`).

3.1.28 Release Notes

Every time a Zotonic release is made, we create a document which lists the most important changes. This page contains links to the release notes for each release so far, dating back to Zotonic's initial release in November 2009.

Release 0.1.0

Released on 2009-11-13.

- Initial release.
- Packaged the `zotonic.com` site as the prime “example” site in the default install.

Release 0.2.0

Released on 2009-12-11.

New modules

mod_broadcast Send system messages to all users which are currently logged in in the Zotonic admin.

mod_calendar Shows event resources in a week-overview, and generates ICalendar feeds.

mod_mailinglist Module which allows you to define groups of recipients and send mailings to them. Can also send via the unix sendmail program.

mod_twitter Receives feeds for Zotonic persons, using the Twitter streaming API.

- New core features:

“catincluder” and “all catincluder” tags These include templates based on the category of a resource. Used in the admin to create custom fields based on category. <http://zotonic.com/documentation/760/catincluder>

Query search model Generate lists of resources on the fly. Used in `mod_atom_feed` to generate atom feeds, and has an API endpoint, `/api/search`. <http://zotonic.com/documentation/761/the-query-search-model>

More template filters: `in_future`, `in_past`, `rand`, `twitter`, `escape_ical`

Bugfixes

- Dynamic postgresql pool size, based on system load (issue #4)
- Issue in postgres pooling on stresstesting (#15)
- Uploaded files now get a proper mime type and extension (#5)
- And other issues: #2, #3, #9, #11, #14, #19, #20

Release 0.3.0

Released on 2010-01-25.

New modules

mod_comment Enables a simple commenting system on your site using `mod_comment`.

New core features

A new default site The default site of a vanilla Zotonic install is now modelled after a simple blog-style website, complete with an archive section, keywords, navigation to previous and next posts, atom feeds and comments.

Speed improvements The Webmachine code was restructured to be more lean-and-mean, yielding up to 20% more performance on page requests.

WebSockets support When WebSockets is available in the browser, then it is used as a replacement for the Comet long poll. Currently only Google Chrome supports this feature but it is expected to arrive in other browsers soon.

Admin updates Support for editing a location (Google map picker), a new collection type “query” was added for creating “saved searches”.

EUnit support A start has been made to put the core functionality of Zotonic in unit tests using the EUnit testing framework. As of yet, only a small fraction of the code has been covered, but we’ll keep working on increasing the code coverage of the tests.

Bugfixes

- Resizing animated GIFs (#28)
- Determining EXIF orientation for images (#27)
- The OAuth API key management interface is now available from the admin. (#35)
- Hiding “meta” pages from the admin overview (#12)
- And dozens of small fixes which did not go through the issue tracker.

Release 0.4.0

Released on 2010-04-19.

New modules

mod_pubsub Enables resource sharing over XMPP's PubSub; share content between sites and get realtime updates when content changes. See: <http://scherpenisse.net/id/644>

mod_search_solr Added a module which plugs into Zotonic's search system to support Solr (<http://lucene.apache.org/solr/>). Using Solr enables quick fulltext searching and facetting.

New features

Default site improvements The default site of a vanilla Zotonic install has been improved with nicer graphics, cleaner typography, a "contact form" example and styles for the Twitter module.

"More results" scomp A twitter/facebook style ajaxified "read more" pager, which is a button which will fetch more results for the current search question inline on the same page.

Windows support Initial support for building and running Zotonic on the Windows platform.

Database schema support Multiple sites running inside one Postgres database is now possible thanks to Postgres' support for multiple table namespaces (schema's)

Template expressions It is now possible to use full boolean and arithmetic expressions in the ErlyDTL templates.

Other features:

- Webserver IPv6 support
- Yandex.Video support in mod_video_embed module (#52)
- PID-file for zotonic (#74)
- Support for HTML5 audio/video tags in TinyMCE editor (#75)
- Newer TinyMCE 3.3.2 release from upstream (#69)
- Newer Mochiweb r153 release from upstream

Bugfixes

- page_path controller should not redirect to the default_page_url (#6)
- Get the name of the current dispatch rule (#21)
- zotonic fails after postgresql restart (#49)
- Unreliable pivot? (#50)
- Module manager should feedback when module cannot be started. (#51)
- Do not depend on the 'default' site (#59)
- i18n of scomp_pager (#62)
- Buttons and "Logoff" link problems in Chrome (#63)
- Comment form breaks on new default site (#64)
- Getting an unknown_rsc error on startup (#66)
- Zotonic fails to (re)start if an existing admin panel is open with browser supporting WebSockets (#70)
- can't save location without e-mail (#71)
- Improve the default styles to include list bullets/numbers (#72)
- Twitter module cannot be enabled (#76)

Release 0.5.0

Released on 2010-10-03.

New features

Simpler module system Modules are simpler, do not have to be a fullblown `gen_server`. Registering `z_notifier` for modules is made more simpler by using Erlang's introspection on modules.

i18n support through gettext Gettext `.po` and `.pot` file support for translations. Templates can be translated per module. Pot files are automatically generated from the templates.

Pluggable Access Control system The new ACL structure works through pluggable ACL modules. Two ACL modules are included as examples. `mod_acl_adminonly`, where all users are admins, and `mod_acl_simple_roles`, which implements a simple role based ACL system.

Authentication can now be customized and extended. `mod_authentication` is the basic module used for authentication. This module can be extended. The `mod_facebook` is an (incomplete) example of such an extender. `mod_authentication` implements the username/password authentication, including logon and logoff. It also supports 'password forgotten' e-mails.

User signup Non admin users can sign up using the `mod_signup`. This module works in harmony with the authentication module and authentication extenders.

New OTP supervisor hierarchy. The PostgreSQL connection pool is now part of the individual sites. Sites are more isolated and can be individually started, restarted or stopped. It is possible to add and remove sites without restarting Zotonic. Modules are now isolated and the running status of a module is displayed in the admin's module overview.

A status overview site, `zotonic_status`. `zotonic_status` shows the running status of all sites. When logged in, the user can start/stop/restart sites using his browser. It is also possible to do 'hg pull' updates of sites that contain a mercurial repo.

New ErlyDTL filters `group_by_title_firstchar`, `is_visible`, `pprint`, `urlize`, `without_embedded_media`.

Media preview enhancements `{% image %}` now supports the the following new arguments:

'extent' - create a larger image then the original not by scaling up but by adding a border to the image.

'removebg' - removes the image's background. It accepts an optional fuzziness parameter (range 0..100).

'upscale' - Forces a small image to scale up to the requested dimensions.

Extended support for Websocket connections. The two newest protocols, as used by Chrome and Safari, are supported.

mod_development improvements It now supports turning on or off the concatenation of `{% lib %}` includes as one file or separate files, and can give a live trace of translated templates, showing clearly the template inheritance and selections.

mod_menu improvements It implements the menu now as a template, easing your own menu implementation.

mod_emailer improvements It can now inline images into the e-mails

New: mod_slideshow It can make a slideshow of any collection, you can add your own slide templates.

New: mod_contact Simple contact form which gets sent over e-mail

New: mod_facebook Facebook logon

New: mod_imageclipper A simple javascript image-clipper bookmarklet for grabbing images from other web-sites.

New: mod_logging A realtime log of debug messages and errors in the system.

Other features:

- New ErlyDTL tags: `{% inherit %}`, `{% overrule %}`

- New ErlyDTL support for multiple argument {% with %}: {% with a,b as c,d %}
- New ErlyDTL support for filters with multiple parameters.
- New ErlyDTL test set, including regression tests.
- System wide configuration system (z_config) using a configuration file at 'priv/config'

Bugfixes

- Allow HTML5 audio and video tags (#75)
- Typo in m_config, line 127. undefind -> undefined (#83)
- setting initial admin password does not work (#88)
- After upgrading the code to latest changeset admin authentication causes exception (#91)
- Menu module does not follow ACL rules (#92)
- Crash in start.sh using Erlang R14A on Mac OS X 10.6 (#93)
- Extra Atom Link (#95)
- Makefiles use rm GNUism (#96)
- z_email:split_name_email/1 does not what it says it should do (#97)
- dots in page paths are transformed into dashes (#98)
- attaching media to pages does not work correctly (#99)
- After a module crashes, the new dynamic observe_* methods are not re-initialized (#100)
- setting page path and unique name is broken (#101)
- IF statements on empty rsc_list structures (#104)
- When image is too small, providing only a width should not make the image very large (#105)
- And many various other fixes which users noted on the mailinglist and were fixed quickly.

Release 0.6.0

Released on 2011-02-12.

New features

SSL support Zotonic has gotten support for serving web pages over secure HTTPS connections. When configured, it listens by default on port 8443. See <http://zotonic.com/https-support> for details.

z_logger A new subsystem for the low-level logging and tracing of requests. This module should be used to log lower level events during development time. Higher-level log messages (e.g. events by Zotonic modules) are still handled by 'mod_logging'.

multilingual content Every resource can have be translated in as many languages as you like. The admin has gotten an interface to provide the editing of the multiple language versions. Available languages are fully dynamically configurable.

z_depcache Partial rewrite of depcache system, is now faster and using more the process dictionary of the calling process to cache often used values.

New and changed modules

mod_signal New module providing a handy signal and slot mechanism for use in templates.

mod_tkvstore New module providing a simple typed key/value store for modules and Erlang code.

mod_translation Check if the user has a preferred language (in the user's persistent data). If not then check the accept-language header (if any) against the available languages.

mod_mailinglist Tweaks in the templates, updated dutch translations; do not send mail when deleting recipient from admin; Added 'recipient_id' to some e-mails so that the e-mails are sent in the correct language.

mod_authentication Fix user name display in password reminder e-mail.

mod_emailer Fix for e-mail override, escape the '@' in the original e-mail address. Added flushing of poll messages

mod_seo Added option to set a no-index for a complete site. New Google Analytics tracker code. With thanks to Richard Fergie.

mod_contact Configurable from address for contact email

mod_admin_identity Fix for finding users, select only identity records with type 'username_pw'

mod_calendar Better handling for undefined date_end values.

mod_search Improper months ordering in archive_year_month query. (#134)

mod_menu Possibility to create an arbitrary number of different menu's. Also a new filter (menu_trail) which gets the menu trail for the main menu.

Changes to template filters and tags

'first' filter added optional length parameter

min/max and minmax 3 new filters were added to clamp a value in an (integer) range.

filesizeformat New filter, similar to the Django filesizeformat filter.

lib tag Extended the lib tag with a 'use_absolute_url' option.

confirm/alert actions These actions were changed and now use HTML dialogs instead of javascript popups.

reversed New filter to reverse a list

menu tag Added 'menu_id' parameter to specify which menu to render

date_diff New filter to calculate the difference between two dates

tinymce_add, tinymce_remove New actions to dynamically initialize of de-initialize rich textareas

trigger_event New action to trigger a named wire.

wire Added a new 'visible' wire type, which triggers when the wired element comes into view (by scrolling or using 'show').

lazy New scomp which shows a 'loader' image and performs onetime actions when loader comes into view.

General bug fixes

- Fix for 'double-dot' in e-mails when using postfix. Also encode the \$. characters using quoted-printable.
- Fix for format_price filter. Show thousands when no cents.
- Make video embed code editable.
- Merged various webmachine fixes, updating it to 1.7.3:
- support {stream, TotalSize, StreamFun} body result for range-capable streams

- Add infinity timeout to gen_server calls
- Allow multiple IP/port bindings
- split chunk header on semicolon just in case a client is using chunk extensions
- properly extract peername from all rfc1918 addrs
- change H7 to match on any if-match, not just *
- webmachine: WM-1.7.3(compat) ignores client's Content-Type on HTTP PUT requests (#130)
- webmachine: prevent using chunked transfer encoding with HTTP/1.0.
- increase the startup timeouts for the gen_servers to prevent startup race condition
- Update mochiweb to latest version from mochi/mochiweb github repository (1.5.0)
- Pulled latest epgsql driver to support Postgres notifications.
- Added additional mime types (Office 2007, .rar)
- z_session: Only mark the persistent store as dirty when a persistent value changes.
- pgsq: Fix for a problem where a postgres connection was not returned to the pool in case of a sql error.
- z_media_preview: some files without a preview where not showing an icon.
- fixed an DoS vulnerability in Mochiweb/SSL
- Added flushing for most periodic internal messages (e.g. tick, poll)
- windows: fix build.cmd; remove some unix-specificness from imagemagick shell commands
- mochiweb: Cookie expire date format string now follows rfc2109
- ACL checks on static file serving
- Comet: support for cross-domain comet connections

Release 0.6.1

Released on 2011-07-06.

Konstantin Nikiforov

- jquery actions insert_before, after, replace

Atilla Erdődi

- Activity Streams
- added notification for comments

Arjan Scherpenisse

- Mailinglist - fix database updater
- Fixed zotonic-stop command by using -name instead of -sname
- Services API tweaks
- Made group_title_firstchar filter work with translatable content.
- Fix up base/info service – add translation lookup to administrator title.
- Small tweak to atom entries, better follow atom spec
- Remove link to atom_feed from mod_base's base template to remove dependency on mod_atom_feed.
- Fix lib tag for including multiple files directly from the toplevel 'lib' folder.
- Release depcache locks immediately after exception in z:memo().
- Use translation lookup in internal link action.

- Make dispatch rule reloading more stable.
- Make `z_utils:url_decode/1` work and add unit tests.
- `z_media_tag`: Ensure that the generated URL is a binary, even if it's empty.
- Fix for editing content which used to be multilingual but is no longer multilingual.
- backport changeset 05a5254b6c92 - Fix for embedded media not showing up
- Fix translation statistics for modules in places other than `modules/`
- Fix month/day name capitalization: dutch does not do capitalization.
- Fix TinyMCE double initialization for the `rsc-body`.
- Fix unnecessary sharpen in lossless (PNG) images.
- Backported fix for issue 158 to 0.6 (twitter images)
- Fix solr searching because `z_convert:to_isotime` now returns a binary.
- Allow for recursive file lookups in `z_module_indexer:scan_subdir/4`
- Fix menu fetching on 0.6; update docs
- Fixed essential typo in mailinglist module.
- Only use gzip encoding for mime types that actually benefit from it.
- Compile the Zotonic behaviours before the other source files.
- Fix mailinglist upload button for changed `#upload{ }` record.
- Fixed a bug in `mod_acl_simple_roles` which caused a NULL violation when setting `visible_for`.
- Make `zotonic-1.0.js` compatible with jQuery 1.5
- Remove string reversal from `mod_logging` which was previously needed.
- `fieldreplace`: Removed toggle class from click handler, because change handler is also triggered.

Marc Worrell

- Show the 'voorvoegsel' field for nl, de languages.
- Add `identify/2` for `#upload` records.
- Small style fix for non-translation editing of the body.
- Added support for subscribing persons in the database to a mailinglist.
- Added flipchart mime type.
- Fixes for looking up flipchart, upper case extensions and mime icons.
- Fix for `getValue` on an empty select element.
- Make pivoting dates more resilient against illegal dates.
- Fetch the widget options from `data-...` attributes instead from in the element's class. The class is still parsed for backwards compatibility.
- Move widget options to `data-...` attributes.
- Remove newlines before parsing the widget options.
- always call the `widgetManager` when inserting content.
- Added 'visible' option, triggers the `moreresults` action on visible of the element instead of on click.
- Added `z_comet` to the list of zotonic query args.
- Added 'if' filter.
- Added missing mailinglist footer template.

- Changes in the `acl_add_sql_check` notification, now uses a record, including the search sql.
- Added visibility toggle to `mod_comment`.
- Added support for generic msoffice mime type returned by 'file'
- Fix for resetting the textarea after committing the comment.
- Clear textarea and input fields after a comment has been posted.
- Added randomize filter, shuffles a list.
- Added 'matching' type of question and results view for select in 'narrative'
- Wired the 'postback_notify' event directly into the postback/websockets resources. This makes javascript triggered postbacks more flexible and simpler.
- Allow `data_xxxx` attributes, which map to html5 'data-xxx' attributes.
- Added 'do_slideshow' widget. Needs a data-slideshow with the id of the elements to be shown in a slideshow dialog.
- Added cursor property for `do_slideshow` elements.
- Fix for validator, empty string should pass the format validation.
- Fix for black background when previewing transparent images as jpeg.
- Added form reset action. Fix for submit/reset of 'closest' form. Fix for IE problem with inputoverlay.
- Let `do_inputoverlay` also work for textareas
- Added 'propagate' option to event action (wire scomp). This prevents canceling the event after being handled.
- Added created column to overview.
- Better embedding of images. Now fetches the image using the rights of the sender. Also initialises the random number generator for the inline image cids.
- Removed content disposition of 'inline' images.
- Add X-Attachment-Id to embedded images.
- Added plain text alternative to sent e-mails. Based on markdown text of the html.
- Added 'acl' option, similar to the `acl` option of the `resource_template`.
- Better feedback when a validation fails on form submit.
- Added 'is_public' and 'ongoing' query terms.
- Added `z_template:render/2` to render a `#render{ }` record.
- Added poll, including poll templates. A poll should have a start/end date, used for displaying.
- More subtle error message when form does not validate.
- Fix for a problem where the substr of translated binaries was taken.
- Fix for binary format string.
- Fix for binary results from translations.
- Made `do_listfilter` more generic.
- Make `erlydtl_dateformat` a binary. Correct handling of binaries or lists for day/month representations. Re-enable test set.
- Allow institutions to have an username/password.
- Show date range for surveys. Always show poll results.
- Let the 'remeber me' be valid for 10 years or so instead of two weeks.
- Fix for `max_age` of persist cookie

- Updated modernizr, fixes hanging bug in Chrome.
- Added survey result download in CSV (tab delimited) format.
- Allow to specify 'propagate' with the wire scomp
- Added download link to survey results. Also changed tab delimited results file to comma delimited file.
- Hide results when user is not allowed to see the results.
- Fix for editing multilingual content with or without mod_translation.
- Added possibility to have 'undefined' array values.
- Log an error when we run into a compile error.
- Simple expression evaluator, makes use of the erlydtl parser and runtimes.
- Added conditional jumps on page breaks. Added missing license texts.
- Added fix for 'file' utility identification error on mpeg files.
- Added i18n module for further localization. Currently implements a list of languages by iso code with translations.
- Added a 'whereis' api to z_module_manager to find the pid of a running module.
- Fix for mod_i18n description.
- Fixes for pivoting country names. Expands iso code to country names in languages as defined in the page. Only use the english postgresql indexing to prevent problems when searching in other languages.
- Map upper case iso to lower case for pivoting.
- Workaround for validations of elements that are removed. Needs better solution.
- Fixes for dynamic changes in LiveValidation checked forms. Keep the administration of LiveValidation objects in the data() of the form elements. So that you can dynamically add/remove elements without the LiveValidationForm object getting out of sync.
- Prevent retry loop when providing an illegal page id
- Move the init of non-gen_server modules to the dummy module process instead of the module manager's process.
- Added user_from_page/1 and user_from_session/1 to derive the user id from a page or session pid.
- Fix for a problem where the 'appear' was called on the enclosing container for insert_top/bottom instead of the inserted html fragment.
- Part of previous commit to fetch user_id from page pid.
- Added do_popupwindow widget to open a link in a popup window.
- Replace textual smiley codes with images.
- Supervisor for named processes, based on Erlang's supervisor.erl
- Added firefox smiley.
- Fix for a problem where widgetManager is only called for the first element in the jquery context.
- Fix for popupwindow namespace.
- Fixes for window name on IE, and fixes for correcting size of popup after opening window.
- Delayed size correction, Chrome returns 0 when the window is just opened.
- Added support for anchors and tables to markdown translators.
- Added no_html option to html2markdown filter.
- Make it possible to pass additional variables to the rendered result template.
- Use the no_html option when translating html to plain text.

- Fix for `z_render:set_value/3`
- Allow property names as string values.
- Added cc to email record.
- Added support for Cc
- let `os_escape` of undefined return `[]`
- Fix translation table references when pivoting resources.
- Added 'full' option for popup windows.
- Added live updating `do_timesince`. Added `z_translate` function, uses a lookup object to find translations. Fixed local/utc time problem in 'U' date format. Fixed copyright name in smiley.
- Fix for site/hostname mixup.
- Fix for opening a window in 'full screen' size.
- Echo `q.username` so that we can have direct link to the password reminder when we know the username.
- Added version to activity record.
- Added missing `survey_results` resource.
- Added notifications on edge insert/delete. Incomplete but good enough.
- Made session timeout configurable, use `site.session_expire_1` and `site.session_expire_n`
- Automatically cleanup temporary files when the request process is stopped.
- Allow more than one `do_` widget per element.
- Fix for resizing images embedded in e-mails.
- Added Google Analytics `_trackEvent` `do_gaq_track`
- Added `m.comment.get` for a `comment_id`
- Added reload of pages on logon or logoff - first part of smarter session management and open pages.
- Let the reload action take into account the continuation page of a logon screen.
- Merging 0.7 email additions to help with site transitions from 0.6 to 0.7
- Added `to_flatlist/1`
- Added 'attachments' field to email record, for easing upgrading 0.6 sites to 0.7.
- Added newer jquery and jquery ui to fix drag problems in IE9
- Remove `<style>` tags when converting html to markdown.

Release 0.7.0

Released on 2011-07-28.

New core features

SMTP Native SMTP support for sending and receiving e-mails in any Zotonic site. We integrated Andrew Thompson's `gen_smtp` library which allows us to manage outgoing and incoming mails. `mod_logging` provides a new email log-view for inspecting what mails go in and out.

Commandline A "zotonic" shell command. The "zotonic.sh" shell command has been replaced by a more generic and more powerful shell command with support for pluggable subcommands.

Module repository Zotonic now supports installing system-wide modules which are not part of the core repository. We have created a place where externally contributed modules can be linked at <http://modules.zotonic.com/>. Modules registered on that site can be easily installed through the “addsite” subcommand.

Skeleton sites The default website has been replaced by the notion of “skeleton” sites. The “zotonic addsite” command lets you create a new Zotonic website based on one of the (currently two) website templates.

New modules

mod_email_relay Relay received e-mails to an user’s email address. Serving as an example for the SMTP functionality, this module looks up a username by the local part of a received e-mail and forwards the mail to the mail address the user configured.

mod_email_receive Handle received e-mails, notifies email observers depending on a stored mapping of recipient addresses.

mod_import_csv Fairly generic module for importing CSV files, updating or creating new content on the fly.

mod_import_wordpress Basic import module for WordPress WXR file format, allowing you to migrate a WordPress blog into Zotonic.

Discontinued modules

To make Zotonic more lightweight and remove some of the build dependencies, some infrequently used modules have been removed from the core and moved to their own repository, at <http://code.google.com/p/zotonic-modules/>. These modules are `mod_search_solr`, `mod_pubsub`, `mod_slideshow`, `mod_broadcast`, `mod_imageclipper`, `mod_admin_event` and `mod_calendar`. They can still be easily installed with the help of the “zotonic modules install” command. The `mod_emailer` module (and its `esmtplib` library) has been removed in favor of the native SMTP sending/receiving capabilities.

Each module now also can have its own dependencies by including a “deps” subfolder in the module. This is used for example in the `mod_pubsub` external module which has the `exmpp` library as a dependency.

Other minor features

- `to_json` filter for representing template values as JSON objects
- `split` filter for splitting a string
- `slice` filter for manipulating lists
- Added `{% raw %}...{% endraw %}` support for representing literal code blocks.
- `erlydtl`: Added possibility to define atoms using backquoted strings.
- admin templates are reorganized, allowing to write admin customizations with less code
- translations of the admin updated and more translations added

Bugfixes

Too many bugfixes to list. However, the base system is becoming more stable and this release aims to be a good step towards the 1.0.

Release 0.7.1

Released on 2011-08-03 13:17 by arjan.

Arjan

- Documentation fixes and fixed wrapping of homepage links in the `www.zotonic.com` site
- Fixed the menu editor: Added `resource_menu_admin_menu` again which was removed by accident.
- Reworked the way `mod_logging` notifies the log pages; it now uses `mod_signal`.
- Fixed `quality=xx` parameter to `{% image %}`.
- Added `mod_signal` to the default list of installed modules.

Release 0.7.2

Released on 2011-12-11 19:51 by arjan.

Alain O'Dea (1):

- Ignore compile output and runtime generated files

Arjan Scherpenisse (`mod_mailinglist`):

- Added new bounce handling dialog.
- Attach documents to the mailing for each 'hasdocument' edge instead of 'document'.
- Added administration of worker processes to email server.
- Updated translations, removed references to `mod_emailer`.
- Fixed using the configured address in mailinglist rsc as the "from" address.
- Re-added the option of test-sending a mailinglist page to a single address.
- Removed `already_sent` check from `mod_mailinglist`, which is not needed since the new interface.
- Fix include reference to mailing footer template.

Arjan Scherpenisse (`mod_survey`):

- `mod_survey`: made questions configurably required or not.
- Added dutch translation for `mod_survey`; small template tweaks. Fixes #205
- `mod_survey`: quick hack to put email validation on a field if you name it 'email'.

Arjan Scherpenisse (misc):

- Added support for varying overview lists in the admin on category.
- `mod_facebook`: make 'scope' parameter configurable.
- On win32, mime type returned as `application/octet` for all files.
- Added spanish translation and install the spanish language by default and enable it.
- Added a file with the translators per language.
- Scan all erlang files in the modules for translatable strings.
- Added missing file
- Added 2 new translation template files.
- Add check in `zotonic-addsite` to see if the given site name is valid.
- `tiny_mce`: Fixed the disappearing of inline images. Fixes issue #203.
- Tooltip Y position is now dependent on its height. Fixes issue #207
- Disregard stderr output from `identify` command. Fixes issue #206
- Make windows users happy when redirecting stderr.
- Added dutch month/day names to `mod_110n`.
- `z_datamodel`: do not try to resolve 'undefined' in `valid_for` check

- Generalized `group_title_firstchar` filter into `group_firstchar`.
- Added `date_start_year=` and `date_end_year=` search filters to filter on year of date start/end.
- Fix infinite recursion in `sub_month/3` filter.
- `mod_import_csv`: Added import button to admin status page.
- `m_rsc_update` emptied the `pivot_date_*` fields when `date_` fields where not part of the update.
- Added `application/x-font-woff` for webfonts.
- Allow modules to override admin TinyMCE options which were originally set in `admin-common.js`
- Addressed the issues in the backup module. Fixes #220
- `mod_backup`: make sure we have an archive dir before archiving.
- Added option `email_bounce_override` to override bounce email address.
- Allow id to be either number or unique name for `resource_admin_edit`.
- Export `z_session_manager:get_session_id/1`.
- `z_html`: do not escape/strip HTML when a property name ends in `_html`.
- `z_html:escape_props/1` - Make selecting escape function more safe.
- Fixed picture rotation detection by tweaking the parser of the output of “`exif -m -t Rotation`”.
- Fixed `z_utils:tempfile()` to respect OS environment variables.
- `z_convert:to_json/1` now also accepts floating point numbers.
- Fix SQL error in `z_installer`.
- Add distinctive icon for internal link in TinyMCE. Fixes #189
- Removed `m_identity:{get,set}_props` which were unused and not working.
- Show language selector on admin media page. Fixes #253
- `mod_twitter`: remove invalid `{verbose, trace}` option.
- `lower/upper` filters now try to convert their argument to a list if they’re not.

Release 0.7.3

Released on 2011-12-14 14:43 by arjan.

Arjan Scherpenisse (1):

- Fixed `mod_mailinglist` compilation error on 0.7.x branch.

Release 0.7.4

Released on 2012-01-10 14:44 by arjan.

Arjan Scherpenisse (1):

- `mod_twitter` - use https for streaming API.

Marc Worrell (1):

- Fix for pivoting with a new resource.

Release 0.7.5

Released on 2012-03-11 09:04 by arjan.

Andreas Stenius (1):

- Fix dialog display issue when not using `<html xmlns=...>`

Arjan Scherpenisse (7):

- admin: Show error message when user tries to add the same page connection twice.
- LiveValidation: use the same e-mail regexp as in the backend.
- mod_import_wordpress: More feedback, convert text to proper paragraphs, support for 1.1 WXR schema.
- mod_twitter: Fixed converting unicode -> utf-8 in body text of received tweets.
- mod_oauth: Fixed access/request token uris to work with both GET and POST

Release 0.8.0

Welcome Zotonic 0.8.0, released on April 11, 2012. These are the changes for Zotonic release 0.8.0. The most important changes are summarized first, below that is a full “git shortlog” of all changes since release 0.7.

New core features

Module manager Module manager startup stability fixes, module dependencies and starting/stopping modules in the correct order.

Status site The site status got a redesign to be more in line with the current zotonic.com site. It now shows a friendly welcome message and requires a login to view / manage the running zotonic sites.

PostgreSQL We stabilized the pgsqll connection pool in the presence of database connection failures and improved query timeout handling.

The “host” option in a site’s config file is now optional. When not present it will be derived from the site’s directory name.

New / updated modules

mod_oembed Provides an easier way to embed external content into your site, using the OEmbed standard.

mod_translation added support for RTL languages like Hebrew and Arabic in the admin interface. Content pages that are translated in multiple languages now have a separate URL for each language version. Translations of the admin interface were added for Irish, Spanish, Estonian and Polish.

mod_mailinglist Improved the mailinglist interface. It is now much easier to track to which list a page has been sent to, to preview the mailing and to view and manage bounced emails.

mod_development On Linux, development has been made easier by integrating inotify. Supports on-the-fly compilation of Erlang files, flushing caches, and compiling/minifying LESS/SCSS/Coffeescript.

Other minor features

New filter: `index_of`, which gives the index of an item in a list.

`filter_random:random/3` - create random sublist with length 1.

`range` filter: easily generate lists with integers

Development process

The git master branch switched to using git submodules for the most important external dependencies.

Documentation got updated, most source files now have @doc tags which are generated and available online at from <http://zotonic.com/documentation>

Git shortlog

Alain O'Dea (2):

- Ignore compile output and runtime generated files
- Enhance PostgreSQL security

Andreas Stenius (27):

- z_pivot_rsc: Should also update pivot_category_nr when it is null.
- z_media_identify: Log identify errors.
- z_email_server: inc_timestamp/2 should increment by minutes.
- z_config: report config file parse errors, and terminate.
- site config: improved error reporting and host checks. (for issue #5)
- z_email_server: refactored the ?DELETE_AFTER define to be a customizable option.
- mod_backup: lookup db settings from pgsql_pool.
- m_edge: add set_sequence/4 to control edges.
- admin action "dialog_new_rsc": trigger custom actions for newly created resources.
- mod_base/filters/replace_args: new filter (#193).
- format filter: support filtering binaries. (#251)
- Windows: Update build.cmd for renamed webmachine -> webzmachine.
- mod_signup: urls are returned as binaries.
- Added Michael Connors for his Irish translations.
- Avoid // in path of found Makefiles.
- Pull in lager using git:// rather than https.
- Keep track of row number inside {# ... #-}comments.
- Fix dialog display issue when not using <html xmlns=...>
- fix for prev_year from feb 29.

Arjan Scherpenisse (245):

- Add 'no session' dispatch rule option to prevent starting a session on a page.
- Add check in zotonic-addsite to see if the given site name is valid.
- Add distinctive icon for internal link in TinyMCE. Fixes #189
- Add some cache blocks in case we get HN'd/slashdotted :-)
- Add z_context:{get,set}_cookie
- Added +x permission to zotonic-status script.
- Added .cw (Curaçao) to mod_110n.
- Added .empty file for extensions dir

- Added 2 new translation template files.
- Added François Cardinaux to contributors, fix docstring.
- Added PUT and DELETE as accepted methods for API calls.
- Added a file with the translators per language.
- Added administration of worker processes to email server.
- Added application/x-font-woff for webfonts.
- Added blog section to www.zotonic.com
- Added commands to enable, disable sites
- Added `date_start_year=` and `date_end_year=` search filters to filter on year of date start/end.
- Added dutch month/day names to `mod_l10n`.
- Added dutch translation for `mod_survey`; small template tweaks. Fixes #205
- Added empty `_language_attrs.tpl` to admin module for when `mod_translation` is disabled.
- Added `gen_smtp` as a submodule.
- Added `l10n_date:monthname_short/2` for short month names.
- Added `lager` as logging framework
- Added missing file
- Added `mod_signal` to the default list of installed modules.
- Added option `email_bounce_override` to override bounce email address.
- Added `resource_menu_admin_menu` again which was removed by accident.
- Added sass support to `mod_development`
- Added spanish translation and install the spanish language by default and enable it.
- Added support for scanning module `.erl` files for `?__` syntax.
- Added support for updating Zotonic and sites over Git in `zotonic_status` site.
- Added support for validation error message on radio elements.
- Added support for varying overview lists in the admin on category.
- Added the ability to use the `resource_api` handler for any URL.
- Added `z_utils:percent_encode/1` function.
- Addressed the issues in the backup module. Fixes #220
- Admin link dialog: possibility to add a preconfigured list of defaults.
- Admin: remove tooltip from media attachment to fix dragging images to the right.
- Again fix the embedding of images in TinyMCE. Fixes #286
- Allow id to be either number or unique name for `resource_admin_edit`.
- Allow modules to override admin TinyMCE options which were originally set in `admin-common.js`
- Automatic make of changed `.erl` files works
- Bugfixes in `m_edge:replace/4`.
- Completely remove `cufon` from `zotonic_status` site, remove stats page
- Deal with spaces in provider name for embed template lookup.
- Disregard `stderr` output from `identify` command. Fixes issue #206
- Do not use sass caching

- Enable/disable now starts/stop the site on the node.
- Export `z_pivot_rsc:insert_queue/2`, for the delayed pivoting of a single rsc.
- Export `z_session_manager:get_session_id/1`.
- Facebook: Add possibility to redirect to a custom signup failure page.
- First work on module upgrader.
- Fix calls to `z_sites_dispatcher:update_dispatchinfo/0`
- Fix compilation error in `z_toposort`
- Fix custom server header for Zotonic with the new webzmachine.
- Fix infinite recursion in `sub_month/3` filter.
- Fix stylesheet issues in new hierarchical editor.
- Fix warnings in `m_rsc_update`
- Fixed crash in `inotify` server of `mod_development`.
- d picture rotation detection by tweaking the parser of the output of “`exif -m -t Rotation`”.
- Fixed `quality=xx` parameter to `{% image %}`.
- Fixed some more admin translations
- Fixed typo in TinyMCE. See #286
- Fixed `z_utils:tempfile()` to respect OS environment variables.
- Fixes in twitter/facebook for changed `z_dispatcher:url_for` return value :-/
- Forgot to use `catinclude` after media item add in admin.
- Gave a fresh new look to `zotonic_status`, similar to Zotonic.com.
- Generalized `group_title_firstchar` filter into `group_firstchar`.
- Get the persistent id in template using `{{ m.persistent.persistent_id }}`
- Greatly improved the mailinglist feedback.
- Implemented new schema mechanism in all Zotonic modules.
- Let `m_rsc:get_raw/2` return empty list instead of undefined when result is not found
- Let `z_lib_include` handle an empty request. Fixes #283
- Let `zotonicwww` site also use `manage_schema/2`.
- LiveValidation: use the same e-mail regexp as in the backend.
- Logoff controller now respects ‘p’ argument.
- Make `bin/zotonic` compatible with python 3.x
- Make the “change category” option better accessible
- Make windows users happy when redirecting stderr.
- Make `z_form_submit_validated_do` more stable using `$.each()`
- Makefile - use “find” to locate every Makefile we need, including those behind symlinks.
- Makefile now `init`s/updates git submodules if any.
- Media: classify `application/*` media files as “document”.
- Move webmachine -> webzmachine in its own repository.
- Moved translation-tabs initialization into `mod_translation`.
- New filter: `index_of`, which gives the index of an item in a list.

- OAuth: fix request/acces token with POST
- OEmbed: even better error reporting, and show preview image when creating item.
- OEmbed: make the gen_server site-dependent; do not crash when getting invalid http request.
- OEmbed: when adding an oembed video, set the title if it's not set yet.
- On win32, mime type returned as application/octet for all files.
- Only show text direction controls in TinyMCE when mod_translation is enabled.
- Pass all filters into filter2arg function. Fix background removal for JPG images.
- Prettified the zotonic status commandline script
- Re-added the option of test-sending a mailinglist page to a single address.
- Refactored m_media:replace_file_mime_ok to not use a nested transaction in the insert case.
- Refactored the collecting of dispatch rules.
- Removed already_sent check from mod_mailinglist, which is not needed since the new interface.
- Removed gen_smtp in preparation of it being a submodule
- Removed m_identity:{get,set}_props which were unused and not working.
- Replaced TinyMCE with latest version. Fixed zmedia plugin.
- way mod_logging notifies the log pages; it now uses mod_signal for inter-page communication.
- Rsc pivot: fix case where sometimes pivot title would say 'undefined' and refused to update.
- Show error message when user tries to add the same edge twice.
- Show language selector on admin media page. Fixes #253
- Simplify manage_schema/2 module call allowing to return a #datamodel{ }.
- er *after* sites manager so we can directly collect dispatch rules in dispatcher's init/1.
- Support ISO timestamps with time zone (currently ignored)
- Support for "extensions"; system-wide extra user-defined gen_servers.
- Support for default value in session get / get_persistent
- Tooltip Y position is now dependent on its height. Fixes issue #207
- and Facebook modules now also use #logon_ready_page observe pattern after successful logon.
- Updated the zotonic_install script
- Updated varnish config example to a more recent Varnish version
- Use catinclude in show_media filter for more versatility
- Use newer rebar script for iconv.
- When postgres exists normally, dont print info report.
- action_admin_dialog_edit_basics: custom action= argument(s)
- admin: use catinclude for _edit_media template, so it can be overridden.
- filter_index_of: Removed debug statements
- lower/upper filters now try to convert their argument to a list if it's not.
- m_rsc_update emptied the pivot_date_* fields when date_ fields where not part of the update.
- mod_admin: Made the title of uploaded file optional.
- mod_admin: Press "enter" now saves the edit page.
- mod_backup: make sure we have an archive dir before archiving.

- mod_development - Removed unneeded ensure_server message and commented out trap_exit
- mod_development - flush cache on dispatch rule change.
- mod_development - remove debug statement, fix sass syntax
- mod_development.erl: When detecting new template, flush all cache to make sure it is found.
- mod_development: Added LESS css on-the-fly compilation.
- mod_development: when discovering new .tpl in site, flush its cache.
- mod_facebook: make 'scope' parameter configurable.
- mod_import_csv: Added import button to admin status page.
- mod_import_csv: Added more flexible date import and support for publication start/end.
- mod_import_wordpress tweaks
- mod_110n: added ru.po, ru.country.po
- mod_logging - Fix log message formatting error.
- mod_mailinglist - added bounce handling dialog.
- inglist - attach documents to the mailing for each 'hasdocument' edge instead of 'document'.
- mod_mailinglist - fix include reference to mailing footer template.
- inglist: When sending to single address or to bounces, do not send to subscriber_of edges.
- mod_oauth: do not assume GET
- mod_oauth: fix API authorization check when using OAuth.
- mod_oauth: fix for R15B, changed http_uri:parse/1 return format.
- mod_oauth: fix for accessing public services when authorized
- mod_oauth: more refactoring; API services defined in site modules now also work.
- mod_search: Add creator_id and modifier_id to search query options.
- mod_search: improve the previous/next search function by allowing other dates to be paged on.
- mod_survey - show a counter column in front of every survey result in the editor.
- mod_survey - show questions in the right order
- mod_survey: Added a survey results edit page to the admin.
- mod_survey: Limit entry of "name" field to 32 chars.
- ey: Propagate qargs into the survey templates, make possible to add default values to survey
- mod_survey: made questions configurably required or not.
- mod_survey: normalize survey question names with z_string:to_name/1 instead of with to_slug/1
- mod_survey: quick hack to put email validation on a field if you name it 'email'.
- mod_twitter - use https for streaming API.
- mod_twitter – support for login using Twitter, similar to mod_facebook.
- mod_twitter: Fixed converting unicode -> utf-8 in body text of received tweets.
- mod_twitter: fix redirecting to ready_page by storing it in the session.
- mod_twitter: remove invalid {verbose, trace} option.
- mos_survey: fix chart export when answer name changed for yesno questions.
- oauth: Added allowed methods for access/request token uris.
- resource_api: do not start session when not needed.

- `search_query`: fix Erlang warning about exported variable.
- `tiny_mce`: Fixed the disappearing of inline images. Fixes issue #203.
- `z_convert` added `ip_to_long/1` and `long_to_ip/1`.
- `z_convert`: fix timezone parsing for formats like `2011-10-06T14:44:00+0200`
- `z_convert:to_json/1` now also accepts floating point numbers.
- `z_datamodel`: do not try to resolve 'undefined' in `valid_for` check
- `z_db:ensure_table` – added `primary_key` attribute for custom primary keys
- `z_filewatcher_inotify` - Change `timer:send_after` to `erlang:send_after`
- `z_html`: do not escape/strip HTML when a property name ends in `_html`.
- `z_html:escape_props/1` - Make selecting escape function more safe.
- `z_module_manager`: schema upgrades are allowed to return a `#datamodel{}` now as well.
- `z_session:restart/1` can now take `#context{}` as argument.
- `zotonic-start`: `cd` to `$ZOTONIC` before doing `make`. Fixes #218

Atila Erdodi (4):

- support for per property acl settings (note: you need to implement your own acl module. no example provided yet.)
- page model
- removed unnecessary info messages

François Cardinaux (1):

- New filter to escape JSON strings added to `mod_base`.

Konstantin Nikiforov (7):

- `z_session`, `m_persistent`: move SQL into model, cleanup ugly code
- added `.gitignore`
- `z_search`: added recursive concat for complex `#search_sql{}`
- `m_persistent`: fixed `push()`
- fixed SQL RETURNING behaviour on empty result.
- `filter_random:random/3` - create random sublist with length 1.
- `mod_l10n`: added `ru.po`, `ru.country.po`

Maas-Maarten Zeeman (39):

- Sometimes somebody (e.g. google) uses a smallcaps dtd
- Added a range filter to easily generate lists with integers. Syntax: `18|range:70 -> [18, 19, ..., 70]` or `2012|range:1900:'-1' -> [2012, 2011, ..., 1900]`
- Accidentally removed, generated new template
- Add facebook graph queries
- Added admin page for facebook module
- Added configuration option to increase the maximum number of concurrent connections
- Added facebook model for fql and graph queries
- Added `if` argument for optional caching. Issue #296
- Added spaceless block to strip whitespace between tags. `{% spaceless %}...{% endspaceless %}`
- Also refactored the sort event, again backward compatible

- Also refactored postback_notify. The refactor is backward compatible for postbacks, but not for notifications. The notification now also contains the trigger and target ids of the elements. This can lead to crashes in code which did not use records for notifications.
- Apply filter to a block. `{% filter upper %}`This will `{% endfilter %}`
- Backward compatible refactor for drag and drop events
- Backward compatible refactor. Moved submit and postback to records in order to make things more clear and remove the `_TriggerId`, `_TargetId` (or was it vice versa?) code. Todo are drag, drop, postback_notify and sort events.
- Change to get an object picture via the facebook graph api
- Changed filenames of translation templates
- Changed md5 hash for hmac and use base64url encoding so pickles are url friendly
- Configurable max memory for depcache
- Copied macros not needed anymore
- Couple of wrong renames
- Fix for a nasty loop caused by heart when things fail. Issue #212
- Fixed parsing of quoted attributes. They can contain newlines
- Fixes a race condition in which slots is called before the module is started
- Fixes an error when closing the dialog
- Fixes the edit button of `acl_simple_roles`. Fixes issue #208
- Format validator converts javascripts re's to pcre re's. Allows unicode re's #242
- Generated fresh pot files
- Made the `acl_simple_role` admin templates translatable
- Refactor, Introduced `with_connection` to handle direct pgsqL queries easier
- Refactored facebook code. Now it can do graph posts too
- Refactored `z_service` so it works with modules and methods with underscores
- Removed experimental module
- Renamed translation template files from `en.pot` to `modulename.pot`
- Store `z_notifier` observers in ets instead of a dict
- Support for webm video
- iolist support for `to_lower` and `to_upper`
- `quote_plus` is now exported by `mochiweb_util`, removed copied code

Marc Worrell (136):

- Add 'action' to the `#rsc_update` fold notification, so we can distinguish an insert from an update.
- Add alternative urls to the head for translations of the current page.
- Add <http://> before links starting with `www`.
- Added 'with' support to value expressions. Example: `{{ a with a=3 }}`
- Added dependencies to modules. Fixes issue 230.
- Added download link to media on `page.tpl`
- `_existing_module/1` which checks if a module name can be found (and loaded) as a module. This without creating a new atom when the module does not exist.
- Added `flattr` button

- Added fold `set_user_language`, sets the context language to the `pref_language` of an user.
- Added `freebsd` to `iconv` rebar config. Thanks to Thomas Legg.
- Added `id_exclude` search term. With thanks to Michael Connors.
- Added `is_even` filter. Thanks to Michael Connors
- Added new menu/hierarchy sorter. In use for menu and category editors.
- Added remark about optional host configuration and module dependencies.
- Added `sha1` as filter and `javascript`.
- Added some module dependencies. Changes default module dependencies to include module name with the `provides` and default `[base]` with the `depends`. Refers to issue #230
- Added support for posting `z_notify` javascript notifications directly to a delegate module. This calls the `event/2` handler in the module (instead of the `z_notifier` observer).
- Added `tinyMCE` plugin for inline text direction editing: `zbd0`
- Added url rewrite on dispatch and generation. Now in use for automatically adding `z_language` argument to all paths.
- Added `{continue, NewMessage}` return format to `z_notifier:first/2`.
- Allow binaries for the header values. Fixes issue #257
- Allow included template to be a variable, forcing a runtime include. Fixes issue #256
- Allow non-atoms as language when setting the context language, ignore lists that aren't languages.
- Changed `http` into `httpc` for R15 compatibility.
- Changed the `rsc_update_done` notification to a `#rsc_update_done` notification record.
- Changing startup sequence.
- Check on return value of module activation transaction. Fixes issue #255.
- Fix for URLs with only a language code. Suppress language codes in URL when no languages enabled. Refers to issue #258.
- Fix for a problem where the admin overview crashed when no category was given.
- Fix for detaching `m:f` with `pid` when `pid` is not alive anymore.
- Fix for `do_feedback` on a single input element.
- Fix for loading beam files.
- Fix for loading modules on the fly.
- Fix for range requests
- Fix for saving surname prefix when signing up.
- Fix for the case where the `exif` orientation is an empty string.
- Fix for url encoding values ≥ 16 . With thanks to Charles Won.
- Fix for when the to-be-submitted form disappeared during a feedback wait.
- Fixes #215. Hide the label when there is a value in the overlayed input.
- Fixes #225. Filters image tags with references for `/admin/media/preview`.
- Fixes for `is_required` handling of survey. Make `created` nullable for a smoother upgrade.
- Graceful Not Found in missing lib images.
- Make module startup handle nested upgrades.
- Make sure that text can be prepended/appended into an existing html element. Fixes #214

- Making modules more robust. More to follow.
- Missing argument for string:tokens/2
- Module manager: Only start modules when their dependencies are running.
- Monitor webmachine, in case webmachine goes down without telling z_logger.
- Moved old datamodel/0 to the manage_schema/2.
- No acl check for is_published and visible_for.
- Only allow existing and enabled languages as a language prefix. This fixes issue #258.
- REST API: Added support for jsonp callbacks
- Set min height of tree editor, so a drop on an empty menu is possible.
- Set the language of rendered e-mails to the preferred language of the recipient_id (if any)
- Stabilizing the database connection pool and sites in the presence of database connection failures. Refers to issue #269
- Support for rtl languages. (Arabic and Hebrew)
- Use bind instead of live - as we run into event bubble problems.
- admin: Fix for positioning dialogs that are higher than the window height.
- admin: Fix for unlinking images, result of action was not shown.
- dispatcher: Fix for creating URLs when a parameter has a regexp pattern.
- email: Added default values for relay options. Added option smtp_bounce_email_override to override the VERP for all sites.
- erlydtl: Added {% ifelse ... %} support. fixes #303
- i18n: Added Farsi (Persian) as right-to-left language.
- m_media: Fixed problem where uploading a file with a pre-defined category resulted in a duplicate category_id SQL error.
- m_rsc: Added 'is_import' and 'no_touch' options, this makes it possible to import data from other sites and keep the original created/modified dates.
- m_rsc: More efficient 'exists' check for rsc records.
- mod_import_wordpress: Renamed title of module to be more inline with other import modules.
- mod_menu: Remove invisible menu items when requesting menu_flat or menu_trail. Issue #291
- tinymce: Updated to 3.4.7
- to_list/1 shouldn't flatten a list, to_flatlist/1 should.
- websockets: Added support for hybi17 websocket protocol "13". Thanks to code from Cowboy by Loic Huguin
- z_datetime: Fix for next year on feb 29.
- z_db: Added automatic retry for deadlocked transactions.
- z_db: Added optional timeout argument for (most) database functions. Defaults to ?PGSQL_TIMEOUT. Fixes #301
- z_depcache: Make the process dict flush safe for proc_lib process dict vars.
- z_notifier: Allow programmes to send scripts to connected pages. Allow signals to be a simple atom, for programming simplicity.
- z_notifier: Documented z_notifier notifications. Translated tuple into records.

Michael Connors (5):

- Add Month, Day and Country translations in French and Irish
- Added Irish translation
- Added date validator to Zotonic.

Paul Guyot (1):

- z_db: Fix bug where ensure_table generated bad SQL when adding a column with a default value.

Piotr Meyer (10):

- Added Polish translation

Taavi Talvik (4):

- Added Estonian translation

Release 0.8.1

Released on 2012-08-11 16:36 by arjan.

Ahmed Al-Saadi (1):

- Removed websocket response header to conform to the latest proposed websocket protocol standard (see <http://tools.ietf.org/html/rfc6455#section-4.2.2>, December 2011)

Arjan Scherpenisse (20):

- Fixed the “make docs” command in the Makefile
- Fix 2 typos.
- mod_mailinglist - depend on mod_logging.
- mod_mailinglist - address recipients by name in welcome/confirm/goodbye messages.
- mod_mailinglist: new filter to add recipient details to links in mails.
- mod_backup: Prefix backup files with the site’s name.
- Dynamically calculate the ebin dirs when compiling .erl, to ensure deps/ ebin dir existence.
- Fix wording of SEO webmaster tools configuration. Fixes #310.
- Merge remote-tracking branch ‘origin/release-0.8.x’ into release-0.8.x
- Added optional argument to show_media filter which specifies the used template for media items.
- z_email_server: Force quoted-printable encoding on HTML and Text parts in emails.
- Show e-mail addresses that have been entered in the survey in an easy copy-pastable dialog.
- mod_survey: Added easily-printable output.
- mod_survey: Sort the printable list on name_surname by default. Sort argument from the query string is also supported.
- mod_survey: Use lowercase when sorting printable list
- Backport fixes to Twitter filter to 0.8.
- Fix blog skel indentation. Fixes #329

Maas-Maarten Zeeman (4):

- The session id was not stored when a new session was created. Hopefully this fixes #327.
- Fix for mochiweb:to_html bug with singleton tags. Fixes issue #328
- Make language dependent URL rewriting configurable. Fixes #315
- Fix for logoff option. Fixes #382

Marc Worrell (8):

- Fixed a problem when editing a newly added menu. This was already fixed in the new-admin-design branch.
- Fixed typo in attr name ('check' instead of 'checked')
- Fix cache key of page. Issue #313
- Added travis-ci config from master.
- mod_survey: Added some css to display survey questions.
- mod_survey: Better likert display
- mod_survey: Better matching display
- mod_survey: Better longanswer display

Release 0.8.2

Released on 2012-10-30 11:48 by arjan.

Arjan Scherpenisse (3):

- Full text search: use the default postgres tsearch catalogs.
- Add new webzmachine commit for Firefox Websockets fix.
- mod_import_wordpress: Support schema 1.2; fix XMerl unicode behaviour.

Artur Wilniewicz (1):

- fix multipart form data parsing: add new data at the end of previous data

Marc Worrell (8):

- Fix vulnerability in the random number and id generation. Fixes #369
- Fix mixup of tr/pl iso codes. With thanks to Kunthar.
- Allow searching on texts like 'id:1234' to find a specific id.
- Allow query_id to be a page with 'haspart' objects. Don't throw exceptions on mismatch of categories (this allows a query to be inserted before the referred categories are inserted)
- Only make queries with the 'is_query_live' flag a live query. Also some extra updates to m_rsc_update for later REST API work. Fixes #344
- Also merge the sort terms from 'query_id' includes.
- Add 'publication_before/after' search terms. Fixes issue #361
- Fix query for haspart of a collection.

Release 0.9.0

Welcome Zotonic 0.9.0, released on December 17, 2012. These notes list the most important changes for this new feature release.

New core features

The ability was added to aid and promote “mobile first” development using automatic user agent classification. Amongst others, the template selection mechanism is now based on the detected user agent class.

All base HTML has moved to use the Twitter Bootstrap CSS framework instead of Atatonic. This includes the admin interface (which got a new design) and the base templates for the skeleton sites. All Atatonic CSS files have been removed from Zotonic. Cufon is also no longer included.

While editing resources it is now possible for editors to add different “blocks” of content to the page. Blocks can be custom defined and have their own templates.

Modules can now implement their own URL dispatch mechanisms through `#dispatch{ }` notifications. They are triggered when the regular dispatch rules do not match.

A new major mode for Emacs has been developed, called `zotonic-tpl-mode`. See documentation at zotonic.com for details on how to install it.

Zotonic’s core routines which serve a greater purpose, like date and string manipulation and HTML sanitization, has been moved out of the main repository into a new subproject called `z_stdlib`.

New documentation

The documentation of everything Zotonic has been completely revamped. We now use Sphinx (<http://sphinx-doc.org/>) to generate HTML documentation. Other output forms (PDF, epub) are in the planning as well.

Source code for the documentation is in the main repository under `doc/`, so that it can be kept better in sync with new features in the code base.

The new documentation is up at <http://zotonic.com/docs/>

New tags/changed tags

`{% javascript %}...{% endjavascript %}` Adds javascript that will be run after jQuery has been initialized. In dynamic content it will run after the DOM has been updated with the template where the JavaScript was defined.

`{% image %}` new attribute: `mediaclass`. Image classes can be defined using property files in the template directory. Which image class definition is chosen depends on the user agent classification.

New and updated modules

mod_geomap New module: Provides mapping and geocoding.

mod_comment Added the possibility to have comments be moderated before submitting.

mod_survey `mod_survey` has been largely rewritten. Now uses the new ‘blocks’ structure for adding questions to the survey. Many new question types have been added, like file uploads and category selection. Also supports mailing the survey results and custom handlers.

mod_ssl New module: adds SSL support to sites. Previously only a single certificate could be used per Zotonic server. With this module each site can have its own HTTPS listeners and certificates. When you don’t supply a certificate then a self-signed certificate and private key will be generated using the `openssl` utility.

mod_backup Adds basic revision control for editing resources in the admin interface. Every time a resource is saved, a backup version is made. Using a GUI you can inspect differences between versions and perform a rollback.

mod_rest A new module to interact with Zotonic’s data model using a RESTful interface.

New filters

menu_subtree: Finds the menu below a particular resource id. Usage: `m.rsc.main_menu.menu_submenu_subtree:id`

escape_link: Escapes a text, inserts `
` tags, and makes anchor elements of all links in the text. The anchor elements have the ‘nofollow’ attribute.

sort: Sorts a list of resource ids based on their properties.

New custom tags

geomap_static Makes the HTML for a static map of a location. Uses the OpenStreetMaps tileset. Example usage:

```
{% geomap_static id=id n=3 %}
```

Show the location of the resource 'id' in a grid of 3x3 tiles.

Translations

Translations were added and updated for Dutch, German, Polish, Turkish.

Contributors

The following people were involved in this release:

Ahmed Al-Saadi, Alain O'Dea, Andreas Stenius, Arjan Scherpenisse, Artur Wilniewicz, Barco You, François Cardinaux, Grzegorz Junka, Hans-Christian Espérer, Ivette Mrova, Joost Faber, Kunthar Daman, Maas-Maarten Zeeman, Marc Worrell, Motiejus Jakštys, Piotr Meyer and Tom Bradley.

Release 0.9.1

Released on 2013-03-14 19:31 by arjan.

Andreas Stenius (21):

- core: add support for 'optional include' templates.
- doc: Add *build*: tag to list of prefixes.
- doc: Add *translations*: tag to list of prefixes.
- doc: add links to other versions of the docs.
- doc: fix version links.
- doc: removed a tag from the version being browsed.
- doc: rename *translations*: to *translation*:.
- doc: rename link to 0.9
- doc: renamed sidebar template to be more generic.
- doc: update frontend-growl cookbook entry.
- mod_admin: optionally include country options.
- zotonic-tpl-mode: add testsuite.
- zotonic-tpl-mode: bind C-M-to zotonic-tpl-indent-buffer.
- zotonic-tpl-mode: do not treat underscore as being part of a word.
- zotonic-tpl-mode: fix for indenting consecutive closing template tags.
- zotonic-tpl-mode: fix html tag highlighting when attr value contains //.
- zotonic-tpl-mode: indentation fixes.

Arjan Scherpenisse (49):

- Ensure iconv is started
- New z_stdlib module
- basesite skel: Add show_media filter to home template

- build: Explicitly build lager first
- controller_api: Fix throwing a 404 error when service module not is found
- core: Fix use of hostname in startup / management scripts
- core: Include the request hostname in a dispatch rewrite notification
- core: Keep file extension when it is allowed for the file's mime type
- doc: Add paragraph about multiple assignments to *with* tag docs
- doc: Add some documentation about startup environment variables
- doc: Document how to override TinyMCE options
- doc: Document mod_tkvstore
- doc: Document the “remove” action
- doc: Document the arguments of the “lazy” scomp
- doc: Explain API service naming in more detail
- doc: Explain the make_list filter to force model evaluation
- doc: Improve wording on replace filter
- doc: show_media filter: correct the default value for media dimensions
- mod_admin: Add button to run schema install from the site's module again
- mod_admin: Add inlinepopups TinyMCE plugin.
- mod_admin: Fix redirect to admin page when using admin logon form
- mod_admin: Fix some errors in new rsc dialog
- mod_admin: Fix view button class on readonly edit page
- mod_admin: Prevent id conflict between forms
- mod_admin: clicking the “move” indicator of an edge should not open the details dialog
- mod_admin_config: Truncate large config values in the config table
- mod_admin_identity: Show inactive users in a lightgrey color
- mod_admin_predicate: Fix escaping of translated text in predicate help popup
- mod_base: Add z_stream_restart() function to zotonic-1.0.js
- mod_base: CSS typo
- mod_base: Fix ‘width’ argument for Bootstrap modal
- mod_base: Fix remember password for Chrome
- mod_base: Fix rendering issue in TinyMCE when scrolling
- mod_base: Log error in widgetmanager when evaluating the JSON data attribute fails
- mod_base: Remove accidental console.log
- mod_base: make controller_page:get_id/1 return undefined when no id is found
- mod_signup: Add fold notification for determining which fields to validate
- mod_signup: Allow signups with email only
- mod_survey: New question type “multiple choice”
- mod_translation: Add zh to languages list, add Shuyu Wang to translators
- mod_translation: Make checkbox for setting language as part of the URL
- mod_twitter: Change URL where one can resolve a Twitter ID

- scripts: Fix typo in zotonic-update
- scripts: When zotonic is not running, echo the output of erl_call
- translation: Add complete Chinese(zh) translation
- z_media_identify: Fix typo in re-run command on identify fail

Feather Andelf (1):

- translations: Chinese(zh) translations by Shuyu Wang aka andelf

Grzegorz Junka (2):

- Update controller_static_pages documentation
- skel: Update default nodb skeleton config

Ilyas Gasanov (1):

- translation: Added more complete Russian translations

Maas-Maarten Zeeman (1):

- Check if ua_classifier is functioning at startup.

Marc Worrell (57):

- core: add option 'use_absolute_url' to args and context of url generation (pages and media)
- core: added 'runtime' option to 'include' tag. (Renamed from undocumented and legacy 'scomp' argument.)
- core: added option to force the protocol of user facing urls. This is useful when a proxy translates between https and http.
- core: added z:ld(modulename), z:restart(sitename) and z:flush(sitename)
- core: added z_render:update_iframe/3. Used to render a template and place it into an iframe.
- core: allow an included file to use extends/overrides within the scope of the included file.
- core: allow anonymous access to the page_url_abs property.
- core: better error messages for z_db:q and q1 in case of sql errors.
- core: cache result of check if a specific module is enabled.
- core: ensure resource ids passed to the ACL are integers. This makes ACL modules simpler.
- core: fix for forced host/protocol redirects by mod_ssl
- core: fix return value of protocol redirect.
- core: fix spec of function.
- core: m_rsc - sanitize the uri input
- core: merges z_user_agent modifications from master (e3a6605ab3f9c48bda7eaa2d37c12bc7ad58a67b)
- core: more error messages instead of match error.
- core: prevent sites dispatcher from crashing when redirect page dispatch rule is missing.
- core: psql: better error message for insert/delete
- core: remove start/stop of iconv (is now eiconv)
- core: removed some unused var warnings.
- core: return 'undefined' for m_rsc:rid([]) lookups.
- core: show a stack trace when a periodic task crashes.
- core: use z_context:abs_url/1 when redirecting to the main site url. Also make it a permanent redirect (301)
- deps: add eiconv as git submodule.

- doc: add google analytics code.
- doc: added some missing smtp options
- doc: fix in docs for split_in and vsplit_in filters.
- eiconv: fix for warning on linux about missing iconv library.
- facebook/twitter: fix redirects, page_url is now a binary.
- mod_admin_identity: added #identity_verified{ } notification. Used to mark a verified identity to an user.
- mod_admin_identity: added a control to add/verify e-mail identities instead of the e-mail address input for a person.
- mod_admin_identity: fix for update_done notifications other than insert and update.
- mod_authentication: fixes for password reset e-mail and some error feedback messages.
- mod_base/core: added 'qarg' argument to postback actions. Enables adding the value of input elements as query args to the postback.
- mod_base: don't handle onsubmit validations for 'nosubmit' input elements.
- mod_base: moved the e-mail validation regexp to z_stdlib's z_email_utils.erl.
- mod_base: remove all zmodal masks when removing existing zmodals.
- mod_base_site: add style for meta share button
- mod_export: added generic export controller and notifications. (docs will be added)
- mod_l10n: added translations for relative time descriptions in the core z_datetime.erl.
- mod_search: allow nested category lists, allowing for more flexible category specification.
- mod_seo: fix for showing the summary when there is no seo description for the page.
- mod_seo: let search engines only index pages with real translation.
- mod_seo_sitemap: added category specific setting for update frequency and indexing priority.
- mod_seo_sitemap: fix for url generation, should page_url_abs.
- mod_seo_sitemap: generate language version links when mod_translation is enabled.
- mod_seo_sitemap: removed extra template, now uses catinclude.
- mod_seo_sitemap: use 'all catinclude' to fetch the extra seo options.
- mod_survey: added 'answers' option to survey_start, allowing answers to be preset (or added)
- mod_survey: fix for yes/no and true/false questions with immediate submit.
- modules: check if table is present before trying to create it.
- smtp: fix edocs parsing problem on binary notation in comments.
- smtp: use eiconv instead of iconv. Add better bounce recognition and is_bulk/is_auto flags for received e-mail.

MgpId (1):

- mod_base: Modal html title in z.dialog

furiston (1):

- translations: Turkish translations by Evren Kutar aka furiston

Release 0.9.2

Released on 2013-07-02 20:53 by arjan.

Andreas Stenius (7):

- zotonic-tpl-mode: optimize font-lock regexp.
- zotonic-tpl-mode: optimize font-lock regexp #2.
- zotonic-tpl-mode: refactor tag soup indent calculations.
- zotonic-tpl-mode: autocomplete closing tag soup tags.
- zotonic-tpl-mode: improved auto close tag.
- zotonic-tpl-mode: fix bug in find tag soup open tag when encountering a self closed tag.
- doc: minor tweaks to the dispatch documentation.

Arjan Scherpenisse (19):

- mod_logging: Fix live display of log messages
- mod_base: Fix filter_capfirst to support i18n strings
- core: Add 'default_colorspace' global option to set the default ImageMagick colorspace
- mod_import_csv: Correctly treat .csv files as UTF-8
- Fix z_sites_dispatcher:split_host/2 for 'none' value
- zotonic_status: Add API for global site status check
- doc: Update preinstall instructions for Ubuntu
- mod_mailinglist: New dialog which lets you combine lists
- mod_base: Fix controller_static_pages's rendering of template files
- mod_mailinglist: Make bounce dialog nicer
- mod_mailinglist: Only show addresses in bounce dialog for the current mailing page
- mod_backup: Also prefix the periodic backups with the site name
- mod_admin: Resources in the meta category can have 'features'
- core: Fix startup script distributed detection
- core: eiconv should do a make clean, not a rebar clean
- core: zotonic CLI command now refers to website for help on subcommands
- code: Add m_edge.id[123].predicate[456] syntax to look up an edge id from a triple
- mod_acl_simple_roles: Fix bug where visible_for would be reset when it was not part of a rsc update call.
- mod_admin_identity: Add option to send username/password to the user over e-mail

Bryan Stenson (1):

- fix typo in default data

Marc Worrell (67):

- core: show info messages when a site starts/stops.
- core: erlydtl - allow lookup in a list of n-tuples (n>2) by first tuple element.
- core: i18n - don't replace assume an empty translation should be added as the original lookup string. This fixes an issue where an undefined translation overwrites an existing translation from another module.
- core: remove accidentally added notes about federated zotonic.
- core: i18n - removed warning

- docs: fix build on OS X (BSD sed is a bit different than on Linux)
- docs: simpler echo to file, remove temp file.
- docs: document #foo.id syntax
- core: added support for signing up with facebook for a known user. Moved some check functions to m_identity. Fixed problem where an email identity could be added multiple times (when signing up using FB)
- Fix for close button on persistent notices.
- mod_signup: fix spec
- mod_signup: fix typespec
- mod_facebook: fixes to redirects etc.
- mod_base: added 'only_text' option to alert. Corrected documentation.
- mod_authentication: remove reload of other open 'tabs' on log on/off. This gives a race condition in Chrome as Chrome doesn't close the Websockets channel during redirects (which are setting the new cookies)
- core: added z:log_level(debug | info | warning | error) to set the console backend log level.
- mod_development: show debug message when reloading a module.
- mod_base: cherry pick from 5d252cb. Issue #491
- mod_base: remove part of merge.
- mod_l10n: japanese translation for days and months.
- mod_base_site: remove scrollbars from share dialog
- core: don't crash when an identity has duplicates.
- mod_base_site: separate copyright template. Use tablet/desktop footer for (smart)phone.
- mochiweb: merged fix for missing summer time hour.
- mod_oembed: fixes for provider list. Added oembed_client:providers/2 call for debugging. Added lager logging.
- mod_import_csv: connected reset checkbox, now also forces re-import of items. Added support for 'medium_url' column name, downloads and imports an url for a medium item. Added lager logging. Added filtering of escaped quotes.
- core: added extra arg to m_media:replace/insert for m_rsc_update options.
- core: fix for upload from url.
- mod_base: simpler and better z.clickable.js
- mod_custom_redirect: merge of 0f8d234 from master
- mod_custom_redirect: fix delete buttons, and type of submit button
- mod_custom_redirect: check for access permission when saving configuration.
- core: fix for mimetypes where 'image/jpeg' returns preferred extension '.jpe' (instead of '.jpg')
- doc: added documentation about the site.bounce_email_override config.
- Fix for checking delivery status - header decode was using wrong function.
- core: smtp: use os:timestamp() instead of now(). Added guards and more strict tests for inc_timestamp/sent handling
- core: the hostname can be 'null', handle it as 'undefined'
- core: make more robust against illegal page paths from hacking attempts.
- mod_l10n: Added South Sudan. Needs additional translations

- mod_authentication: make the password reset available when logged on
- mod_authentication: fixed reminder link in error message.
- mod_base: fixes for wired id css selector support in combination with a postback. This hack should be replaced with proper 'selector' support for wired targets.
- core: don't change the language on internal page_path rewrites. Fixes #559
- smtp: fix e-mail receive.
- deps: new z_stdlib.
- mod_admin/mod_admin_frontend: made admin more responsive. Added first version of frontend editor based on mod_admin.
- mod_admin: fix '&' in doc description as it crashed the doc generation.
- core: fix failed merge.
- mod_base: copy if_undefined filter from master.
- mod_admin: fix body top padding for smaller screens.
- mod_admin: Make the modal popup in the admin_frontend as wide as in the admin.
- mod_menu: use short title; absolute positioning of item buttons. mod_admin_frontend: minor textual changes, add dutch translation
- mod_menu: show warning when a resource is unpublished.
- mod_admin/mod_menu: added option so pass tab to connect/new dialog.
- mod_admin_frontend: set default connect tab to 'new'
- mod_admin_frontend: preselect the 'text' category for page linking.
- mod_admin: make the logon more responsive (header row-fluid; some styling)
- mod_admin_frontend: fixes for editing blocks and language selections.
- mod_admin: nicer fixed save button bar
- mod_admin_frontend: see images in the tinymce. Nicer save/cancel bar
- mod_menu: small textual change
- mod_admin: fix tinymce image preview.
- mod_admin_frontend: remove debug javascript.
- mod_base/mod_translation: add filter 'filter'; added m.translation model for easy language access.
- mod_ssl: change behaviour of 'is_secure' setting; redirect to https when not specified otherwise.
- mod_ssl: added an 'is_ssl' configuration to force a site to be served over ssl. Use in combination with 'is_secure'.
- Merge pull request #579 from fenek/install-timeout-fix

Release 0.9.3

Released on 2013-09-21 09:37 by arjan.

The main focus of this release is to make it compatible with the Erlang R16 series. Other changes are below, grouped by developer.

Arjan Scherpenisse (24):

- core: Fix .ogg/.ogv to be correctly identified as audio/video.
- core: Implement "center of gravity" option for crops

- doc: Fix syntax error in generated sphinx conf.py file
- doc: Move the section on git commit messages to the “contributing source code” section.
- mod_acl_simple_roles: Fix typo
- mod_admin: Create interface to pick the cropping center on images.
- mod_admin: Do foldr over the blocks instead of foldl
- mod_admin: Speed up admin edit page JS performance
- mod_admin_identity: Also offer the “send welcome” option when changing the username / password
- mod_admin_predicate: Fix category tree for fluid layout change
- mod_base: Add optional tabindex to button tag.
- mod_development: Fix crashing file watcher process while editing templates
- mod_oembed: Better error reporting and “fix all” button
- mod_oembed: Fix setting media title when programmatically importing an oembed media item.
- mod_oembed: Prevent crash in media tag when no oembed metadata is found.
- mod_survey: Fix crashing survey results
- mod_survey: Improvements (placeholder texts for input fields, translateable validation msgs)
- mod_twitter: Fix Twitter logon and API access for changed 1.1 API

Marc Worrell (21):

- core: added ‘magick’ image filter options. Fixed mime lookup if lossless filter was inside a mediaclass definition.
- core: added z:ld/0 to load all re-compiled beam files.
- core: filenames might have a space, escape it before os:cmd/1
- core: fixed mediaclass lookup - now all lookups are done as documented with adherence to the module priority and the ua lookup.
- core: move rsc property access check inside p/3 instead of the model access.
- core: remove debug statement from z_acl.
- core: removed debug messages from z_media_tag.
- deps: newest z_stdlib.
- m_identity: ensure that non-unique identities have is_unique set to ‘undefined’ (instead of false)
- m_media: accept upload of ‘data:’ urls for replace_url and insert_url functions.
- mod_admin: fix replacing media items.
- mod_authentication: send e-mails to all matching accounts, if and only if a username is set.
- mod_authentication: fix input sizes for password, and e-mail/username fields.
- mod_base: added ‘is_result_render’ option to ‘moreresults’ action. Allows render of all results at once instead of per result item.
- mod_base: added touch punch to enable drag/drop and other touch events on mobile touch devices. Fixes #597
- mod_base: fix hybi00 websocket connections. Fixes #637
- mod_base: fix in LiveValidation for resetting error class on textareas. Fixes #601
- mod_base_site: add class ‘wrapper’ for content, allows for better styling. Hide breadcrumb and subnav when the current page is the only page in the breadcrumb or subnav.
- mod_geomap: Perform country lookups by iso code - works better with OpenStreetMaps API.

- `mod_geomap`: export lookup functions, add hard coded lookups for exceptions.
- `mod_geomap`: load cluster popups via template. Disable google geo lookups for half an hour if quota exceeded. Cleanup of some js code. Perform geo lookups in same process as pivot, prevents too many parallel lookups when re-pivoting.

Release 0.9.5

Released on 2014-04-18 16:55 by arjan.

Arjan Scherpenisse (20):

- `mod_admin`: Remove warning in `m_admin_blocks`
- `doc`: Document how to set `m_config` values in the site's config
- `mod_mailinglist`: Fix button order of delete confirmation dialog.
- `mod_base`: Match the websocket protocol on the WS handshake.
- `z_stdlib`: new submodule
- New `eiconv` submodule
- `mod_survey`: in printable overview, use fallback to column name when no column caption has been set.
- `mod_survey`: Add `m_survey:get_questions/2` function
- `mod_base`: `controller_page` - check if page path equals the request path
- `mod_base`: `controller_page` - make canonical redirect behavior configurable.
- `doc`: Clarify the doc page about custom pivots
- `core`: basesite skeleton: install `mod_acl_adminonly` before `mod_authentication`
- `core`: `z_utils:json_escape` now points to right function in `z_json`
- `core`: Update dispatch info on start and stop of a site.
- `core`: Show error message in admin when a module fails to start
- `doc`: Update `mod_twitter` to show how to do Twitter login
- `doc`: Explain what a *query resource* is
- `doc`: Explain all default fields in a resource
- `mod_mailinglist`: Do not crash when `recipientdetails` filter encounters an empty body

Cillian de Róiste (1):

- `doc`: Add cookbook item on creating custom content blocks

Grzegorz Junka (1):

- `core`: Fix in Zotonic after removing access to the `webmachine_request:send_response/2` function

Maas-Maarten Zeeman (4):

- `core`: Make sure js boolean values are converted correctly.
- `mod_base`: Make sure no messages are left in the request process's mailbox. Fixes #676
- `mod_base`: Limit the number of websocket reconnects. Fixes #675
- `core`: Prevent to detach comet controllers from page sessions which are already stopped.

Marc Worrell (19):

- `core`: periodic process to unlink deleted medium files (and their generated previews)
- `z_stdlib`: fixes for `escape/escape_check` of proplists.

- deps: new z_stdlib
- mod_import_csv: improved parsing of CSV files, now handles quoted fields and performs automatic translation from latin-1 to utf-8
- mod_import_csv: remove stray underscore character from module name...
- mod_import_csv: remove superfluous lager messages about escaped chars.
- mod_base: reapply change from d5604eca1f83491c37adf7981ff1cc598b7c57e2
- Fix on fix. start_timer returns a Ref, not {ok, Ref}.
- mod_base: 0.9.x doesn't know about z_ua.
- core: use lower case extensions when guessing mime or file extension.
- core: for exif lookups, add LANG=en for sensible exif program output (not for win32 as shell var is used))
- mod_base: use window.location.host for websocket stream, unless configured otherwise. Also add try/catch around websocket connect, to catch potential IE10 security exception.
- core: new state machine for maintaining websocket/comet push connections. Perpetual comet loop which only polls if there is no active websocket connection. Websocket connection checks with a periodic ping/pong if the connection works.
- core: fix difference between master and 0.9 regarding z_stream_start.
- mod_base: fix for subdomain polling loop.
- New gen_smtp from zotonic/gen_smtp/master
- mod_admin_identity: fix identity insert race condition by moving email identity check into the #rsc_update event instead of the async #rsc_update_done event.
- base: fix NS_ERROR_XPC_BAD_CONVERT_JS on Firefox. Fixes #718
- core: fix SVG mime type mapping from ImageMagick identify.

Release 0.10.0

Welcome Zotonic 0.10.0, released on April 18, 2014. These notes list the most important changes for this new feature release.

New core features

The minimum required Erlang version is now release R15B03. Zotonic is always supporting up to major 2 versions behind the current stable Erlang release.

We now use the de-factor Erlang package manager rebar for the management of the subprojects.

Websocket / comet (re)connection handling have been greatly improved. A state machine now manages the process on the server and keeps the connection running also in the long run.

The Google Summer of Code-sponsored student Mawuli Adzaku provided excellent work on the Zotonic CLI for management (search, installation, ...) of external Zotonic modules such as those from <http://modules.zotonic.com/>

The media handling subsystem was heavily modified in this release. It is now much easier to add custom hooks to do custom media processing when a media file has been uploaded. The following notifications were added as extension points:

- #media_identify_extension{} to make custom mappings from a mime-type to an extension
- #media_identify_file{} to override media file identification
- #media_update_done{} when the media metadata has been stored in the database

For SEO purposes, a new table *rsc_page_path_log* was added, that tracks old *page_path* names from *resources* so that old urls keep working when the page path changes. Old URLs will be redirected to the new location instead.

Handling legacy username/password authentication can now be done by leveraging the new `#identity_password_match{}` notification.

New and updated modules

mod_seo The two modules *mod_seo* and *mod_seo_google* have been merged into a single module. Support for Bing webmaster verification has been added.

mod_artwork This module now ships with the font-awesome icon set.

mod_admin A much requested feature to set the “center of gravity” for images has been added in the admin. The ‘crop’ filter takes this center into account.

mod_admin_frontend This new module will provide admin-alike editing of resources, menu-trees and collections for non-admin users in the frontend of the website, by the re-use of admin templates in the frontend site. This module was backported to the 0.9 series.

mod_admin_stats The admin stats module adds a new admin page presenting a live view of system statistics for things like number of requests, request processing times, database and cache queries etc.

mod_custom_redirect This module provides hooks to redirect unknown request paths or hosts to other locations. It adds a page to the admin to define and maintain these redirects. This module was backported to the 0.9 series.

mod_video A new module for video conversion and playback. It uses the *ffmpeg* tool to convert uploaded movies to h264 and generate a poster image. Included are default templates for movie playback.

mod_mqtt Interfaces to MQTT publish/subscribe topics. Adds publish/subscribe between the browser, server and modules. Includes access control and topic mapping for external MQTT applications.

Template system improvements

The template include system was made more powerful by adding options to force the runtime include of a template and to make the include optional (e.g. not crashing when a template is not found).

The option was added (to *mod_development* (page 364)) to generate HTML comments with the paths to the included files embedded for easy debugging.

Besides this, the *erlydtl_runtime* can now directly lookup values in *mochiweb* JSON structures, making template syntax more straightforward.

New template filters

trim Removes whitespace at the start and end of a string.

pickle Pickle an Erlang value so that it can be safely submitted with a form.

tokens Returns a list of tokens from input string, separated by the characters in the filter argument.

filter Filters a list on the value of a property.

sort Sorts lists in various ways

if_undefined Tests whether a value is undefined, returning the given argument.

menu_is_visible Filters a list of menu items on visibility.

menu_rsc Get a “flat” of menu parents

Translations

Translations were added and updated for Chinese (ZH), Russian, Turkish, Dutch, Polish, Portugese, French and German.

Contributors

The following people were involved in this release:

Alexander Stein, Andreas Stenius, Arjan Scherpenisse, Arthur Clemens, Bryan Stenson, Carlo Pires, Cillian de Róiste, Feather Andelf, furiston, Grzegorz Junka, Ilyas Gasanov, Jarimatti Valkonen, Maas-Maarten Zeeman, Marc Worrell, Mawuli Adzaku, Mgpld, Piotr Nosek, Simon Smithies and Steffen Hanikel.

Release 0.10.1

Released on 2014-06-17 20:27 by arjan.

Release highlights

New core features

The properties of the resource model (`m_rsc`) can now be easily be inspected with the ‘print tag’ and iterated over.

New core features

The properties of the resource model (`m_rsc`) can now be easily be inspected with the ‘print tag’ and iterated over.

Updated modules

mod_development Added dispatch debugging and explanation. Added checkbox to disable the api-service / `api/development/recompile`

mod_admin_modules Add “configure” button in the module manager for modules which contain a template called `_admin_configure_module.tpl`.

New notification

Added the notification `request_context`. This is a foldl with the *Context* and is called after the request’s query arguments are parsed using `z_context:ensure_qs/1`. It can be used to perform transformations or actions based on the query arguments.

Commits overview

Arjan Scherpenisse (11):

- build: Always use a downloaded rebar
- core: Add pivot columns for location lat / lng.
- core: Add `zotonic_dispatch_path` to `z_context` blacklist
- core: Fix comet/WS when site streamhost is undefined and `redirect = false`
- core: Let Travis build 0.10.x

- core: recode iso639 module to UTF-8
- doc: Clarify why Zotonic cannot directly use port 80
- mod_admin: Add option 'show_date' to 'latest resources' admin dashboard widget
- mod_admin: Correct vertical centering of connection dialog
- mod_admin_modules: Add "configure" button on module manager
- mod_base: Vertically center the z_dialog when it is opened

Arthur Clemens (19):

- doc: Add OS support, change name to Requirements
- doc: Improve docs sidebar layout a bit
- doc: Rename 'tutorials' to 'installation'
- doc: Tuning docs on services and mod_oauth
- doc: add example to filter on depiction
- doc: add link to manual dispatch
- doc: document restart command
- doc: elaborate conditional display of widget
- doc: fix 'button title' to 'button text'
- doc: show_media tag TLC
- doc: tweak highlight and see also colors
- doc: update Zotonic shell
- make new page button pass current category
- mod_admin: disable reindex button while indexing
- mod_admin: improve feedback
- mod_admin: live update of pivot queue count
- mod_editor_tinymce: introduce z_editor

Marc Worrell (24):

- core: added 'request_context' notification. Triggered after a request's query arguments are parsed.
- core: always set the user explicitly in the session. Don't copy the user from the current context.
- core: bump version to 0.10.0p1
- core: fix problem where z_db:insert can hang when running out of db connections.
- core: let the template {% print m.rsc[1] %} print all resource properties. Same for {% for k,v in m.rsc[1] %}.%
- core: more binary/list changes (due to z_string changes)
- core: set default schema for sites without dbschema config.
- core: set the option for Erlang 17.0 and later. Issue #764
- core: set the zotonic_dispatch_path to the path before an internal resource redirect.
- docs: add documentation for zotonic_dispatch_path_rewrite var
- docs: add more mod_development documentation.
- docs: add note about 0.10.0p1
- docs: added preliminary 0.10.1 release notes.

- `mod_admin_frontend`: fix for introduction of `z_editor`. Also sync the language tabs.
- `mod_development`: add access control to the dispatch rule debugger.
- `mod_development`: add checkbox to enable or disable the development/recompile api.
- `mod_development`: add human-readable form for `rewrite_nomatch`. Always show the final dispatch.
- `mod_development`: added request dispatching debug. This shows the steps of the dispatcher in matching a dispatch rule, including all bindings, rewrites etc.
- `mod_development`: added template lookup tool to check the template selection. Optionally add a resource category. Shows the selected template per user-agent class.
- `mod_development`: fix build of `edoc`.
- `mod_development`: fix problem where the dispatch-debugging interfered with ssl connection redirects.
- `mod_development`: show final dispatch after `#dispatch_rewrite`.
- `mod_oembed`: fix for `z_string:to_name/1` now returning a binary.
- `mod_survey`: fix display of likert. Values were reversed from real values.

Mawuli Adzaku (1):

- `mod_base`: Add 'md5' filter to translate a string to an md5 hex value

Release 0.10.2

Released on 2014-10-01 20:16 by arjan.

Arjan Scherpenisse (13):

- `mod_menu`: Add menuexport JSON service
- `mod_survey`: Return the (possibly generated) submission id, after insert
- `mod_admin`: Put the unicode quotes of the link message in binaries
- `core`: Flush `z_datamodel` after managing new categories
- `mod_twitter`: Fix twitter OAuth logon
- `mod_base`: Export `controller_logon:set_rememberme_cookie/2`
- `doc`: clarify using categories defined by other modules in `manage_schema`
- `core`: Always normalize rsc blocks
- `mod_survey`: Align ACL check of printable results page
- `core`: Fix name-to-id cache invalidation
- `mod_admin`: Add option to create outgoing edges in new page dialog.
- `mod_survey`: Fix dispatch rule ACL access
- `mod_oauth`: No ACL check on authorize URL; just check if logged in

Arthur Clemens (19):

- minor translation and typo fixes
- `docs`: small improvements to link and unlink action
- `doc`: mention named wire actions in wire doc
- Remove redundant `is_protected` check on resource
- `doc`: update menu documentation to reflect actual output of the module
- `mod_menu`: add param class to menu

- doc: add reference to to_integer
- core: increase maximum image size for previews
- doc: describe attributes removebg and upscale
- doc: document md5 filter
- doc: document filter default
- doc: fix module install command
- translation: typo
- Fix various NL translation inconsistencies
- Fix NL translation
- Only use pre block if error_dump is available
- Add clause for 404
- mod_acl_simple_roles: add module names to titles
- mod_oauth: change required access to mod_oauth

David de Boer (3):

- Fix edges not being added in manage_schema
- Allow to set is_protected from manage_schema
- mod_admin: Fix typo

Marc Worrell (12):

- mod_video_embed: correctly fetch preview images for youtube videos with a '#' in the url.
- mod_video: fixes for handling binaries returned from the z_string and other function (used to be lists)
- Merge branch 'release-0.10.x' of github.com:zotonic/zotonic into release-0.10.x
- core: minor fix in mime-lookup for files not recognized by 'identify'
- core: allow binary results from validations. For now map them to lists, later with the binary webmachine replace this with binary only.
- core: fix problem where rsc_gone insertion would crash if there is already a 'gone' entry for the rsc. This is in the category 'should not happen', but also happened in production.
- core: add infinite timeout to dispatcher calls, prevent crashes on busy or swapping machines.
- core: add DB_PROPS macro to ease transition from 0.10 to 0.11
- Merge pull request #799 from driebit/is-protected-schema
- mod_survey: better survey results display.
- core: fixes for tinyMCE initialization and language tab sync in the admin
- mod_admin_identity: let's stop sending the password in clear text.

Release 0.11.0

Note: For upgrading to this release from an older Zotonic release, please read the [Upgrade notes](#) (page 258) carefully, as some backward-incompatible changes have been made.

Welcome Zotonic 0.11.0, released on October 8, 2014. These notes list the most important changes for this new feature release.

Full timezone support

Timezone support was added to the core. All dates are now stored in UTC. Existing resources with old dates (in local time!) are converted on read, assuming the configured server timezone. You need to set the timezone in the `zotonic.config` file, for example:

```
{timezone, "Europe/Berlin"}
```

A site-specific timezone can be set with the `mod_l10n.timezone` configuration.

Database driver and pool

The database driver and pool has been replaced by the standard `epgsql` and `poolboy` Erlang applications. This means we no longer use a special forked Zotonic version of the `epgsql` Postgres driver. As a nice side effect, database queries run 5-10% faster on the new database code. See the test report here: <https://gist.github.com/arjan/70e5ec0ecaf98b19a348>

Global changes

The default place for user-defined sites and external modules has been changed to the defaults `user/sites` and `user/modules`, respectively.

Also, the global file `priv/config` has been obsoleted in place of a new global configuration file, `~/zotonic/zotonic.config`. Zotonic actually looks in several places for its global configuration file, including `/etc/zotonic/zotonic.config`. See *Global configuration* (page 624) for all information on this topic.

Updated modules

mod_l10n Added timezone support.

mod_development Added dispatch debugging and explanation. Added checkbox to disable the api-service / `api/development/recompile`

mod_mqtt The *live* (page 542) custom tag was added to *mod_mqtt* (page 381), allowing you to live update a part of the page on pubsub events. Documentation for *mod_mqtt* (page 381) has been improved and a debugging option has been added so you can see which topics are published and subscribed to.

mod_base The Zotonic logo is now included in the distribution as a Webfont, resulting in crisp logo's in the admin and on the default Zotonic status website.

New and updated filters

date An optional second argument for the timezone has been added.

date_range An optional third argument for the timezone has been added.

truncate An optional second argument is added to specify the text added where the text is truncated.

truncate_html Truncates a HTML text to a specific length, ensures that all open tags are properly closed.

New transport mechanism

The client-server communication is now based on UBF-encoded messages. It has become easier to send messages to specific pages (browser tabs) and to do bi-directional communication in general. See *Transport* (page 73) for all information on this topic.

Misc

User-defined Erlang dependencies It is now possible to add extra rebar deps to Zotonic, by adding them to the `zotonic.config` file.

Version-locking of dependent Erlang applications Zotonic now uses the Rebar `lock-deps` plugin to keep all included dependencies at the versions that they were at when Zotonic was released. This improves the longterm stability of the Zotonic release.

Rememberme cookie changes The *rememberme* cookie (used for automatic logon) is now based on a token instead of the user id. The token is reset if the user's password is changed. Cookies set using the previous scheme are invalidated.

Request context notification Added the notification `request_context`. This is a foldl with the *Context* and is called after the request's query arguments are parsed using `z_context:ensure_qs/1`. It can be used to perform transformations or actions based on the query arguments.

Contributors

The following people were involved in this release:

Alberto López, Arjan Scherpenisse, Arthur Clemens, David de Boer, Jeff Bell, jult, Maas-Maarten Zeeman, Marc Worrell, Mawuli Adzaku and Stephan Herzog.

Release 0.11.1

Released on 2014-10-20 22:47 by arjan.

Arjan Scherpenisse (7):

- core: Disable sendfile support by default and make a note of this
- core: Fix crashing make on initial build
- core: When using locked deps, still add the deps from `zotonic.config`
- doc: Fix preinstall notes about buggy erlang versions
- doc: Fix sidebar
- doc: fix sidebar link
- scripts: Add “-v” argument to zotonic command to print the version

Arthur Clemens (1):

- doc: update references to `priv/config`

Maas-Maarten Zeeman (8):

- build: Added delete-deps
- build: Fix make clean
- build: Update mochiweb and webzmachine in `.lock` file
- core: Fix `z_sites_dispatcher` so it accepts empty paths. Fixes #842
- core: Fixes IE problems in `zotonic.js`
- core: IE8 fixes for `ubf.js`
- core: Prune context and spawn notifier process only when needed
- core: Remove error and close handlers before ws restarts.

Marc Worrell (31):

- admin: force to select category when adding new content.

- base: refactor the moreresults action.
- core: For websocket, keep reqdata information for 'is_ssl' checks.
- core: add acl mime check to m_media:replace/3.
- core: add lager warnings when modules are stopped.
- core: add m.modules.active.mod_foobar to test if a modules is active. Remove checks with the code server if a module is running.
- core: add mime type exception for WMA files, they were recognized as video files.
- core: add sanitization of text/html-video-embed. Move #context handling out of z_html to z_sanitize.
- core: do not delete/insert edges when changing the order via mod_menu
- core: fix concatenating certain combined file streams.
- core: lager info with modules to be started.
- core: m_rsc:p_no_acl/3 request for a non-existing resource should return 'undefined', just like m_rsc:p/3.
- core: module start/stop progress messages are now debug level.
- core: remove debug statement from m_media
- core: remove nested transaction from the z_edge_log_server check.
- erlydtl: allow lookup of var.key for a list [{<key>, ...}]
- filestore: use filezcache:locate_monitor/1 to let the filezcache track z_file_entry processes.
- m.modules: replace usage of m.modules.info. with m.modules.active.
- m_identity: don't crash if #identity_password_match{ } doesn't match any observers.
- menu: edge menu sorter has a problem with sorting nested collections. Disable sorting for now.
- mod_admin: if date is not editable, display it as text.
- mod_admin: more generic css for .category spans.
- mod_admin: move validation inside the category_select block
- mod_admin: when replacing a media item, show the oembed/video-embed panels for embedded content.
- mod_admin_identity: prevent Safari autofilling username/passwords in new user-account forms. Fixes #811
- mod_menu/mod_admin_frontend: enable editing of collections as side-menu.
- mod_menu: fix for passing the item_template option to the server when adding items.
- mod_survey: evaluate empty jump conditions to 'true'
- mod_tkvstore: don't start as named module.
- smtp: Initialize e-mail server settings on startup. This is needed now that disc_copies are used.
- translation: added some missing nl translations.

Release 0.12.0

Welcome Zotonic 0.12.0, released on October 8, 2014.

This release has been released on the same day as [Release 0.11.0](#) (page 143). The main difference between these 2 releases is that in 0.12, Bootstrap has been upgraded on all templates. 0.11 was released as a feature release with all major improvements of the last 6 months excluding Bootstrap 3 support.

Bootstrap CSS version 3

Zotonic has been switched over to the latest version of the Bootstrap Framework, version 3.2.0. When you are using Zotonic's `mod_bootstrap` or when you have customized the admin templates, you will need to update your templates.

A full migration guide to upgrading from Bootstrap 2.x is here: <http://getbootstrap.com/migration/>, a tool to help you convert your Zotonic templates is located here: <https://github.com/arjan/bootstrap3-upgrader>.

Release 0.12.1

Released on 2014-10-20 17:01 by arjan.

Arjan Scherpenisse (6):

- core: Disable sendfile support by default and make a note of this
- core: Fix crashing make on initial build
- core: Use rebar locked on 0.12 as well
- core: When using locked deps, still add the deps from `zotonic.config`
- doc: Fix preinstall notes about buggy erlang versions
- scripts: Add “-v” argument to `zotonic` command to print the version

Arthur Clemens (1):

- doc: update references to `priv/config`

Maas-Maarten Zeeman (8):

- build: Added delete-deps
- build: Fix make clean command
- core: Fix `z_sites_dispatcher` so it accepts empty paths. Fixes #842
- core: Fixes IE problems in `zotonic.js`
- core: IE8 fixes for `ubf.js`
- core: Prune context and spawn notifier process only when needed
- core: Remove error and close handlers before ws restarts.

Marc Worrell (31):

- admin: force to select category when adding new content.
- base: refactor the `moreresults` action.
- core: For websocket, keep `reqdata` information for ‘is_ssl’ checks.
- core: add acl mime check to `m_media:replace/3`.
- core: add lager warnings when modules are stopped.
- core: add `m.modules.active.mod_foobar` to test if a modules is active. Major performance increase.
- core: add mime type exception for WMA files, they were recognized as video files.
- core: add sanitization of `text/html-video-embed`. Move `#context` handling out of `z_html` to `z_sanitize`.
- core: do not delete/insert edges when changing the order via `mod_menu`
- core: fix concatenating certain combined file streams.
- core: lager info with modules to be started.
- core: `m_rsc:p_no_acl/3` request for a non-existing resource should return ‘undefined’, just like `m_rsc:p/3`.

- core: module start/stop progress messages are now debug level.
- core: remove debug statement from m_media
- core: remove nested transaction from the z_edge_log_server check.
- erlydtl: allow lookup of var.key for a list [{<<key>>, ...}]
- filestore: use filezcache:locate_monitor/1 to let the filezcache track z_file_entry processes.
- m.modules: replace usage of m.modules.info. with m.modules.active.
- m_identity: don't crash if #identity_password_match{ } doesn't match any observers.
- menu: edge menu sorter has a problem with sorting nested collections. Disable sorting for now.
- mod_admin: if date is not editable, display it as text.
- mod_admin: more generic css for .category spans.
- mod_admin: when replacing a media item, show the oembed/video-embed panels for embedded content.
- mod_admin_identity: prevent Safari autofilling username/passwords in new user-account forms. Fixes #811
- mod_menu/mod_admin_frontend: enable editing of collections as side-menu.
- mod_menu: bootstrap3 change.
- mod_menu: fix for passing the item_template option to the server when adding items.
- mod_survey: evaluate empty jump conditions to 'true'
- mod_tkvstore: don't start as named module.
- smtp: Initialize e-mail server settings on startup. This is needed now that disc_copies are used.
- translation: added some missing nl translations.

Release 0.12.2

Released on 2014-12-18 17:01 by mworrell.

Major changes are:

- Addition of new notifications for importing media.
- Authentication using mod_twitter and mod_facebook is now via a popup window
- mod_twitter can now follow keywords and multiple users
- New admin menu item **Auth -> App Keys & Authentication Services**

The following commits were done

Arjan Scherpenisse (3):

- build: force rebar to build 'setup' app
- mod_admin: Fix link to query documentation
- mod_search: Use binary parsing of stored query resource

Arthur Clemens (79):

- admin: button dropdowns for admin overview filters
- admin: fix checkbox in language dropdown
- admin: fix erroneous rule
- admin: fix insert depiction dialog

- admin: fix negative margins of editor
- admin: fix nested links in table
- admin: fix nested treelist
- admin: fix tab order of form fields
- admin: formatting
- admin: handle multiple lined tree list items
- admin: hide redundant Category text
- admin: improve display of thumbnails
- admin: improve interface in various locations
- admin: increase y position of fake form fields
- admin: make long filenames visible
- admin: make muted a bit lighter
- admin: minimal design of help buttons
- admin: more horizontal forms
- admin: more reordering of less styles
- admin: multiple css fixes forms and edit blocks
- admin: NL translation fixes
- admin: prevent nesting modal-body
- admin: refactor tabs
- admin: remove @baseFont reference
- admin: remove extraneous space
- admin: remove periods from dialog headers
- admin: remove unused file
- admin: tweak admin login in release branch
- admin: use translated string
- admin: various markup improvements
- doc: add action set_class
- mod_acl_simple_roles: add titles to modules and category names
- mod_acl_simple_roles: fix div nesting
- mod_acl_simple_roles: improve modules list
- mod_acl_simple_roles: NL typo
- mod_acl_simple_roles: remove period from header
- mod_admin: allow table parts to be not clickable
- mod_admin: also update dashboard button row
- mod_admin: block menu is right aligned
- mod_admin: bootstrap 3 uses class text-muted
- mod_admin: break apart dashboard blocks for easier overriding
- mod_admin: consistent New Page dialog header
- mod_admin: css tweaks

- mod_admin: enable “All pages” button when qcat is defined
- mod_admin: handle empty cat param value
- mod_admin: improve stacked elements within widgets
- mod_admin: layout and translation of dialog Duplicate Page
- mod_admin: make button dropdown alignment configurable
- mod_admin: make white borders in media visible
- mod_admin: more horizontal forms
- mod_admin: NL typos
- mod_admin: organize less styles in separate files
- mod_admin: preserve query string when changing category
- mod_admin: prevent disappearing of media images after undo
- mod_admin: remove period from logged in user
- mod_admin: show category and query in window title
- mod_admin: show crop center in connection overview
- mod_admin: show unpublished state in connected media
- mod_admin: smaller crop center lines
- mod_admin: support em-based buttons in button row
- mod_admin: update table row hover, make colors consistent
- mod_admin: use close button without glyphicon element
- mod_admin_config: consistent field widths
- mod_authentication: remove period from header
- mod_backup: typo
- mod_backup: use bullet list for warnings
- mod_base: document params width and addclass in js source
- mod_base: fix dialog text
- mod_base: missing button style
- mod_base: remove unused line of code
- mod_base: support dialog option backdrop “static”
- mod_base: tweak debug info
- mod_editor_tinymce: add newest version
- mod_editor_tinymce: add option to always use the newest version
- mod_editor_tinymce: if no config, use newest
- mod_editor_tinymce: limit autoresize; fix resize handle bug
- mod_oembed: activate upload button
- mod_seo: optimize descriptions
- mod_signup: remove period from link

Maas-Maarten Zeeman (10):

- Changed the websocket implementation.
- core: Added recon application

- core: Fix: return default when there is no session.
- deps: Updated mochiweb, ip-log fix for R15
- deps: upgraded sendfile
- deps: Upgraded webzmachine and mochiweb
- mod_base: More careful websocket handshake.
- mod_base: Removed commented out code.
- mod_seo: Make noindex and notrack configurable from templates
- Removed comment to fix edoc generation

Marc Worrell (47):

- Add docs
- core: add exceptions for .xls and .xlsx files to z_media_identify. Fixes #893
- core: added comment explaining expire_1 and expire_n for sessions. Issue #881
- core: allow a non-integer category id to be passed to all_flat/2
- core: allow setting any rsc property that is 'undefined' to 'false'.
- core: ensure all db timestamp columns have a time zone.
- core: fix args for transport ack.
- core: fix problem where mod_signed_url could not keep the user logged on.
- core: fix problem where the custom redirects form was not saved
- core: fix specs in z_db.
- core: make session cookie name configurable (solves problems where old cookies might interfere, especially on Chrome)
- core: on context prune-for-scomp, leave an empty list for request headers. Normalize user-agent lookup.
- core: removed debug from z_pivot_rsc
- core: the { % script % } tag has now arguments.
- core: z_search: fix acl check sql query.
- Create database when starting site
- docs: adapt docs to changes in files.
- install: use openssl to generate the admin password, as tr/urandom combo hangs on OS X. Fixes #847
- Make menu_subtree compatible with names
- media/embed: fixes for twitter streaming, added notifications for importing and analyzing fetch url media-data.
- mod_admin/mod_admin_frontend: preparations to allow creation of resources via the edit page.
- mod_admin: remove api dispatch rule, also defined in mod_base.
- mod_admin: return the correct context in controller_admin_media_preview
- mod_admin_frontend: fix a problem where combining the nestedSortable.js lib with other js files will result in erroneous drag behaviour
- mod_authentication: export send_reminder/2 and lookup_identities/2.
- mod_authentication: fix logon_box form input "password"
- mod_authentication: Refactor twitter/facebook logon and signup code.
- mod_base: do not redirect if redirect id is set to undefined

- mod_base: fix js error in livevalidation.
- mod_base: for catinclude, don't assign passed categories to 'id'. Only assign a resource id to id.
- mod_base: in do_popupwindow use e.preventDefault() to play nice with multiple click event listeners.
- mod_base: remove extra </div> from phone/_navbar
- mod_email_receive: when adding recipients, catch references to non existing rsc
- mod_facebook: add delegate for saving settings.
- mod_menu/admin_frontend: final fix for new-page topic on menu insertion.
- mod_menu: correct pubzub.publish topic.
- mod_menu: remove console.log message.
- mod_mqtt: fix js error in for loops.
- mod_mqtt: fix problem where removing one topic listener removed all listeners. Cleanup live subscriptions for removed elemnts.
- mod_oembed: don't display the media twice in the admin
- mod_oembed: remove http: protocol from embed html, this enables content to be viewable on https: pages.
- mod_search: allow dates like 'now' and '+2 weeks' in search questions.
- mod_survey: allow printable overview if user has edit rights.
- mod_translation: add more rtl languages.
- mod_twitter: fix edoc problem.
- Remove is_integer check for cc5d94
- smtp: more relaxed error handling for spamd errors.

Release 0.12.3

Released on 2014-12-22 10:30 by mworrell.

Major changes are:

- Fix of the twerl git url in rebar.config.lock
- New z_stdlib, fixing issues with url metadata extraction
- Sanitization of oembed content
- Addition of Font Awesome 4
- Extended Zotonic logo font

The following commits were done

Arthur Clemens (20):

- admin: general layout fixes
- admin: use zotonic icons
- doc: update logo and link style
- fix gitignore
- mod_admin: remove unused files
- mod_admin: update with mod_base changes
- mod_artwork: add font awesome 4

- mod_artwork: add zotonic logos
- mod_artwork: rename logo folder to “zotonic”
- mod_base: allow other libs to import z.icons.less
- mod_base: bring back (predictable) circle icons
- mod_base: extend logo font to include general icons for Zotonic modules
- mod_base: make extra tag more compatible
- mod_base: replace some icons with FA versions, add button styles
- mod_base: simplify icon creation; add extending FA icons
- mod_base: update favicon
- Remove unused files
- zotonic_status: add less files
- zotonic_status: make responsive, update appearance
- zotonic_status: notification tweaks

Marc Worrell (8):

- core: add instagram js and urls to whitelist.
- core: lock new z_stdlib library. Fix twerl git url. Fixes #895
- mod_admin: always show media content, independent if size was defined.
- mod_oembed/mod_twitter: prefer our own ‘website’ extraction above oembed links. Pass the tweet url in the import_resource notification
- mod_oembed: add missing fallback _oembed_embeddable.tpl
- mod_oembed: sanitize received oembed code html and texts.
- mod_survey: fix problem where displaying the results did not work due to move sanitization functions.
- mod_twitter: fix import of tweets with special-html chars, was double escaped in title.

Release 0.12.4

Released on 2015-02-20 14:19 by arjan.

Arjan Scherpenisse (13):

- core: Add ‘preview_url’ to rsc export API call
- core: Fix edocs build
- doc: Clarify that finished/upcoming filters don’t perform sorting
- mod_acl_simple_roles: Fix ACL check when category of rsc cannot be found
- mod_development: Enable automatic recompilation on MacOS X
- mod_search: API: Add limit, offset, format arguments to search API
- mod_search: Add ‘finished’ search filter + documentation
- mod_search: Add {finished} search method
- mod_survey: Add ‘submit’ API service
- mod_twitter: Close login window when user denies login request
- mod_twitter: Set all context vars when rendering logon_done template
- mod_twitter: Show logon page in current context’s language

- mod_video: Make command line to ffmpeg/ffprobe calls configurable

Arthur Clemens (44):

- mod_admin: remove confusing text
- Social login: use FB compliant “Login with...”
- admin: fix zlink from editor, remove superseded template
- admin: include font-awesome version 4
- admin: tidy up Authentication Services page
- basesite: remove old fix for logo image
- doc: Document template models
- doc: add icons documentation
- doc: document ‘page_url with’ syntax
- doc: document more use cases for `if`
- doc: formatting
- doc: remove deprecated `%stream%`
- doc: restructure actions
- doc: restructure filters
- doc: reword solution
- doc: show css output when extending icons
- mod_admin: cosmetic fixes logon widget
- mod_admin: fix layout person form
- mod_admin: include icons css instead of recreating with less
- mod_admin: layout tweaks
- mod_admin: mobile tweaks
- mod_admin: tidy email table
- mod_base: add non-circle icons
- mod_base: add share icon
- mod_base: add social login icons
- mod_base: clean up documentation
- mod_base: hide original x char
- mod_base: include all font source files
- mod_base: make Z icons independent of FontAwesome
- mod_base: make icon extend cleaner
- mod_base_site: add icon css
- mod_base_site: better defaults
- mod_bootstrap: version 3.3.2
- mod_development: flush when translation file changes
- mod_facebook: typo
- mod_filestore: tidy up html
- mod_filestore: use 0 if archive size is undefined

- social login: use icons and update brand colors
- translation tweaks (NL)
- validation: provide message_after param to templates

David de Boer (1):

- Connect to “postgres” when creating database

Maas-Maarten Zeeman (4):

- core: Make sure the z_file_entry fsm uses correct timeouts
- core: prevent reconnecting a ws when the page is unloading
- mod_admin_identity: Fix for adding email addresses
- mod_base: Close websocket when unloading page. Fixes #898

Marc Worrell (67):

- New z_stdlib locked.
- core: add ‘expected’ option to m_rsc:update.
- core: add compile/0 and /1 to z.erl, for compiling without flush.
- core: added e.issuu.com and static.issuu.com to the sanitizer whitelist.
- core: allow binaries for some special keys.
- core: better handling of erroneous urls for the z_file/media routines.
- core: correct the language utf8 encoding for R16+
- core: correctly parse multipart/signed emails.
- core: extra utf8 sanitization of received email’s subject, text, html and from.
- core: fix for importing structured blocks (like during imports)
- core: fix in m_identity where fetching email identities could loop on a check if the email property was known as an identity.
- core: fix problem in m_rsc:update where modified was not set on save.
- core: fix problem where erlydtl_runtime crashed on fetching a value from a ‘time_not_exists’ atom.
- core: fix stacktrace shown in transport lager messages.
- core: m_rsc:update now converts non-tuple dates and handles creator/modified on import correctly.
- core: move erlang:get_stacktrace() outside of lager calls. Otherwise a stacktrace of lager will be shown due to the parse transforms.
- core: on startup z_dropbox moves now all processing files to unhandled.
- core: refactor database creation on site init.
- core: remove unreachable code.
- core: set the edge’s creator_id on insert
- core: truncate the slug at 78 characters.
- core: z_datetime:to_datetime/1 now also handles numerical timestamps.
- docs: add link to the pdf version.
- docs: added placeholders.
- docs: correct the {% call %} documentation.
- m_identity: fix spec of get_rsc_types/2
- mod_admin: add pubzub and some related javascripts. needed for live tags etc.

- `mod_admin`: also log stacktrace on a catch.
- `mod_admin`: fix a problem where quick-editing a rsc adds all enabled languages.
- `mod_admin_identity`: publish identity changes to the topic `~/rsc/1234/identity`.
- `mod_admin_identity`: some extra padding for the identity verification page.
- `mod_admin_identity`: typo in translation.
- `mod_authentication/mod_twitter/etc`: changes for new font-awesome, bs3 and some small tpl fixes
- `mod_authentication`: add authentication via LinkedIn. Add possibility to connect/disconnect accounts with FB/LinkedIn/Twitter. Fix redirects after using an external service for authentication. List connected authentication services in the password reminder email.
- `mod_authentication`: add special error message if there are cookie problems and the current browser is Safari 8. Issue #902
- `mod_authentication`: make logon form responsive, add optional `page_logon` with title/body texts.
- `mod_base`: added filter `trans_filter_filled/3` export.
- `mod_base`: added the filter `'trans_filter_filled'`
- `mod_base`: check dialog height repeatedly, account for rounding errors in height calculation.
- `mod_base`: filter-sort of undefined is undefined.
- `mod_base`: handle ping/pong websocket control frames, remove name conflict with zotonic ping/pong.
- `mod_import_csv/core`: fixes for importing categories, new properties, corrected basename in `#import_csv_definition{ }`
- `mod_import_csv`: added checks to the model creation.
- `mod_import_csv`: fix handling of blocks. Add support for `'blocks.name.field'` keys in `m_rsc:update`
- `mod_import_csv`: fixes for file handling and `medium_url` imports.
- `mod_import_csv`: major changes to `mod_import_csv`.
- `mod_instagram`: authenticate and import tags from Instagram
- `mod_instagram`: fix property name in comment.
- `mod_l10n`: adaptations for utf8 parsing changes in R17
- `mod_l10n`: add utf-8 encoding hints to source file
- `mod_linkedin`: modify template for bootstrap3
- `mod_linkedin`: seems LinkedIn doesn't like URL encoded secrets?
- `mod_linkedin`: try to workaround a problem where LinkedIn doesn't recognize the Access Token it just handed out.
- `mod_linkedin`: work around for a problem with access-tokens at linkedin.
- `mod_mqtt`: allow topics like `['~site', 'rsc', 1234]`.
- `mod_oembed/mod_video_embed`: fix problem with access rights if new media insert was done without admin rights.
- `mod_oembed`: don't crash on oembed connect timeouts.
- `mod_signup`: show external auth services for signup using the logon methods. Also always force the presence of an `username_pw` identity for signed up users.
- `mod_survey`: fix 'stop' survey button.
- `mod_twitter`: Fix twitter redirect url
- `rebar.config.lock`: lock new `z_stdlib`

- rebar.config.lock: new s3filez
- rebar.config.lock: new z_stdlib
- rebar.config.lock: new z_stdlib
- rebar.config.lock: new z_stdlib
- rebar.config.lock: new z_stdlib
- skel: add mod_mqtt to the base site, as it is needed by mod_admin

(2):

- emqtt_auth_zotonic issue would cause crashed when mqtt client try to connect onto it.
- Fix the emqtt client connection issue.

Release 0.13.0

Welcome Zotonic 0.13.0, released on July 3, 2015.

Major changes are:

- New ACL module: mod_acl_user_groups (see below)
- New module: mod_content_groups
- New model: m_hierarchy, for category, content_group and user_group hierarchies
- Category renumbering is now a lot faster and more incremental
- Addition of Material Design art work
- New module: mod_email_status
- New module: mod_component
- DNSBL checks for incoming emails
- More strict sanitization of content
- Social integration with Twitter, Facebook, LinkedIn, and Instagram
- *is_dependent* Resources
- Refactored automatic compile and load of changed files

This release also incorporates all fixes in the 0.12.x branch.

A complete *git shortlog* since the 0.12.0 release is added below these release notes.

Documentation

Unfortunately we still miss some documentation for the new modules. This will be corrected as soon as possible.

New Access Control module

The new *mod_acl_user_groups* module adds rule based access control.

- All resources (pages) are assigned a content group.
- All users are member of zero or more user groups.
- Content groups are arranged in an hierarchy
- User groups are arranged in an hierarchy

Rules are defined between the user groups and the content groups. Per rule the following properties can be set:

- User group
- Content group
- Category (or *all categories*)
- Privileges: view, edit, link, delete
- Deny flag (for a negative rule)

The collection of rules has an *edit* and *publish* version. The *edit* version can be tested with a special url. If all is accepted, then the *edit* version can be published.

The *edit* version can also be exported and imported. This includes the full hierarchies of all user- and content groups.

Categories and `m_hierarchy`

The category hierarchy tables have been replaced by *m_hierarchy*. This model defines named hierarchies of resources (pages).

If the categories are changed then the system needs to update the *pivot_category_nr* field of all resources. With the introduction of *m_hierarchy* this renumbering is much more efficient and will only affect a minimal number of resources.

The *m_hierarchy* module is also used for the content- and user group hierarchies, as used by the new *mod_acl_user_groups* module.

Email status

The new module *mod_email_status* tracks for all outgoing email addresses:

- If emails are successfully sent
- Number of emails sent
- Number of errors
- Number of bounces
- Latest error message

With this it will be much easier to get feedback on all outgoing email.

DNSBL checks

All incoming SMTP connections are now checked against two DNS block lists:

- `zen.spamhaus.org` (<http://www.spamhaus.org/zen/>)
- `dnsbl.sorbs.net` (<http://dnsbl.sorbs.net/general/using.shtml>)

The list of DNSBL servers checked can be configured using the `smtp_dnsbl` key in the `zotonic.config` file.

`mod_component`

This is an experimental module. It will be the basis of a new method of loading *components* in HTML pages. Components consist of HTML, javascript, and css. These parts can be dynamically loaded. Layout can/will be done using *mithril.js*.

This module will undergo significant changes, so use at your own risk. It is included to support some parties that use this in production.

Sanitization of content

`iframe` Elements and CSS are now sanitized. The *iframes* are sanitized using a whitelist of servers. Use the `#sanitize_embed_url{}` notification to add or remove servers from the whitelist. The current whitelist can be seen at the bottom of the `src/support/z_sanitize.erl` file.

Object tags referring to services like *youtube* are replaced with *iframe* tags.

This sanitization is also done on all *embed* codes of media items.

The CSS parser is strict and only allows well formed CSS. It also strips CSS that is known to be (potentially) *dangerous*:

- loading external content
- `position: screen`
- some classes that are used internally in Zotonic
- and some more

Check `deps/z_stdlib/src/z_css.erl` for the actual sanitizer code.

Social integration

The integration with social sites is completely redone. It is now possible to login (and signup) with Facebook, Twitter, LinkedIn, and Instagram.

It is also possible to import content from Twitter and Instagram. For this it is possible to define tags and/or users to follow. All matching content is imported into special categories. An optional image or embed code is also imported and added *as part of* the imported resource (so not as a separate depiction).

is_dependent Resources

A new flag is added to all resources: *is_dependent*

If this flag is set, and a connection to the resource is deleted, then Zotonic will check if there are any other *incoming* connections to the resource. If not then the resource will automatically be deleted.

This makes it possible to have resources (images, documents) that can only exist in the context of another resource. If that referring resource is deleted then the depending resources are deleted as well.

Automatic compile and load of changed files

The core system now starts either `inotify-tools` or `fswatch`, depending on which one is available. You have to install one of these to enable auto-compile and auto-load of changed files.

These are watching for any change to files in Zotonic and the sites.

If a file is changed then Zotonic will:

- Recompile if a `.erl` file changed
- Regenerate css if a `.sass`, `.less` or `.scss` file changed
- Regenerate js if a `.coffee` file changed
- Reindex the module index if a lib file changed (`.css`, `.js`, etc)
- Reload translations if a `.po` file changed
- Reload a module if a `.beam` file changed
- Reload dispatch rules if a dispatch file changed

If a Zotonic module is reloaded then it will be automatically restarted if the function exports or the `mod_schema` is changed.

Commits since 0.12.0

There were 586 distinct commits (more than 600 in total) since release 0.12.0.

The committers were: Alain O’Dea, Arjan Scherpenisse, Arthur Clemens, David de Boer, Jeff Bell, Maas-Maarten Zeeman, Marc Worrell, Marco Wessel, Paul Guyot, Paul Monson, Sergei, Witeman Zheng, imagery, and .

Besides these commits many people contributed to this release.

Big thanks to all of you!

Git shortlog

For clarity, some cherry-picks and development branch merges are removed from this list.

Alain O’Dea (2):

- Fix#891
- scripts: Fix activating core / user modules

Arjan Scherpenisse (63):

- core: Move master up to 0.13-dev
- mod_base_site: Small Bootstrap 3 fixes
- core: Fix crashing make on initial build
- core: When using locked deps, still add the deps from zotonic.config
- scripts: Add “-v” argument to zotonic command to print the version
- doc: Fix preinstall notes about buggy erlang versions
- core: Disable sendfile support by default and make a note of this
- Add release notes for 0.12.1
- Add release notes for 0.11.1
- Merge pull request #856 from Yozhig/patch-1
- mod_search: Use binary parsing of stored query resource
- Merge pull request #857 from Yozhig/master
- mod_base: Add JSON source of the Zotonic logo font
- build: Add 0.11 and 0.12 release branches to travis
- build: Try to coerce Travis-CI not to run rebar get-deps by itself
- core: Prefix z_db error messages with site name
- mod_admin: Fix link to query documentation
- mod_acl_simple_roles: Fix ACL check when category of rsc cannot be found
- mod_twitter: Close login window when user denies login request
- mod_development: Enable automatic recompilation on MacOS X
- mod_twitter: Set all context vars when rendering logon_done template
- mod_twitter: Show logon page in current context’s language
- mod_survey: Add ‘submit’ API service

- mod_video: Make command line to ffmpeg/ffprobe calls configurable
- core: Add 'preview_url' to rsc export API call
- mod_search: API: Add limit, offset, format arguments to search API
- core: use epysql 2.0.0 instead of master
- mod_search: Add 'finished' search filter + documentation
- core: Fix edocs build
- mod_search: Add {finished} search method
- doc: Clarify that finished/upcoming filters don't perform sorting
- doc: Fix sphinx configuration
- Add release notes for 0.12.4
- mod_survey: Month names in dutch are lower case
- mod_content_groups: Put in 'structure' admin menu; fix dutch wording
- core: Remove category information from rsc export
- mod_content_groups: Put content group category in 'meta'
- mod_content_groups: Add .nl translation strings
- mod_acl_user_group: Initial version
- mod_search: Allow filtering resources on content group
- Add acl rule model
- Add ACL rule editor for module and rsc rules
- mod_survey: Add 'allow_missing' request parameter
- mod_admin: Place upload form in a block, allowing it to be easier overruled
- mod_admin: (CSS) file input buttons have varying heights on different OSs
- build: Bump Travis OTP version
- m_hierarchy: Add parents/3 function
- mod_acl_user_groups: Make ACL rules overview more usable
- Add edit basics dialog for user with tab to set user groups + options
- Let search box work on users page
- Search placeholder text
- Users overview: show all persons/institutions or only those with accounts
- Create import/export function for rules
- Fix compilation error in ACL rule export controller
- mod_development: Don't let sass create a source mapping
- mod_development: Don't let sass create a source mapping
- doc: Remove reference to Ubuntu PPA
- mod_survey: Fix results/printable page when survey rsc has page path
- mod_admin_identity: Remove big 'show all' button when searching users
- mod_admin_identity: Start with showing only users, not persons
- core: z_media_preview: Do not add white space to small images when resizing
- mod_base: Allow to set response headers from service API calls

- mod_base: Correct ReqData attribute in API controller on processing POST

Arthur Clemens (160):

- doc: update references to priv/config
- admin: fix negative margins of editor
- admin: make long filenames visible
- admin: button dropdowns for admin overview filters
- admin: remove @baseFont reference
- admin: remove unused file
- admin: fix nested links in table
- admin: prevent nesting modal-body
- admin: improve display of thumbnails
- admin: hide redundant Category text
- admin: increase y position of fake form fields
- admin: NL translation fixes
- admin: use translated string
- mod_admin: bootstrap 3 uses class text-muted
- admin: make muted a bit lighter
- mod_acl_simple_roles: improve modules list
- admin: fix tab order of form fields
- admin: improve interface in various locations
- admin: handle multiple lined tree list items
- mod_signup: remove period from link
- mod_authentication: remove period from header
- mod_admin: organize less styles in separate files
- mod_admin: block menu is right aligned
- admin: remove extraneous space
- admin: more reordering of less styles
- mod_seo: optimize descriptions
- admin: fix checkbox in language dropdown
- mod_backup: use bullet list for warnings
- mod_backup: typo
- mod_editor_tinymce: add newest version
- mod_editor_tinymce: add option to always use the newest version
- mod_editor_tinymce: limit autoresize; fix resize handle bug
- mod_admin: NL typos
- mod_base: missing button style
- mod_admin: break apart dashboard blocks for easier overriding
- mod_admin: css tweaks
- mod_admin: remove period from logged in user

- `mod_editor_tinymce`: if no config, use newest
- `admin`: multiple css fixes forms and edit blocks
- `admin`: multiple css fixes forms and edit blocks
- `mod_admin`: prevent disappearing of media images after undo
- `admin`: formatting
- `mod_admin`: use close button without glyphicon element
- `mod_admin`: more horizontal forms
- `mod_acl_simple_roles`: remove period from header
- `mod_oembed`: activate upload button
- `mod_admin`: make button dropdown alignment configurable
- `mod_acl_simple_roles`: add titles to modules and category names
- `mod_base`: fix dialog text
- `admin`: remove periods from dialog headers
- `admin`: minimal design of help buttons
- `mod_acl_simple_roles`: fix div nesting
- `mod_admin`: allow table parts to be not clickable
- `mod_admin`: update table row hover, make colors consistent
- `admin`: more horizontal forms
- `mod_acl_simple_roles`: NL typo
- `admin`: refactor tabs
- `mod_base`: document params width and addclass in js source
- `mod_admin`: preserve query string when changing category
- `mod_base`: tweak debug info
- `mod_admin`: improve stacked elements within widgets
- `mod_authentication`: refactor logon templates
- `admin`: fix insert depiction dialog
- `admin`: various markup improvements
- `admin`: fix nested treelist
- `mod_admin`: layout and translation of dialog Duplicate Page
- `mod_admin`: handle empty cat param value
- `mod_admin`: enable “All pages” button when qcat is defined
- `mod_admin`: consistent New Page dialog header
- `mod_admin_config`: consistent field widths
- `mod_admin`: also update dashboard button row
- `doc`: add action `set_class`
- `mod_base`: remove unused line of code
- `mod_admin`: show unpublished state in connected media
- `mod_admin`: make white borders in media visible
- `mod_admin`: show crop center in connection overview

- mod_admin: smaller crop center lines
- mod_base: support dialog option backdrop “static”
- mod_admin: support em-based buttons in button row
- mod_artwork: add font awesome 4
- mod_artwork: add zotonic logos
- doc: update logo and link style
- Merge pull request #896 from pmonson711/readme_fixes
- mod_artwork: rename logo folder to “zotonic”
- mod_base: extend logo font to include general icons for Zotonic modules
- admin: use zotonic icons
- mod_admin: remove unused files
- admin: general layout fixes
- zotonic_status: make responsive, update appearance
- fix gitignore
- zotonic_status: add less files
- mod_base: update favicon
- Remove unused files
- mod_base: simplify icon creation; add extending FA icons
- mod_base: replace some icons with FA versions, add button styles
- mod_admin: update with mod_base changes
- zotonic_status: notification tweaks
- mod_base: make extra tag more compatible
- mod_base: allow other libs to import z.icons.less
- mod_base: bring back (predictable) circle icons
- doc: typo
- mod_base: add non-circle icons
- mod_base: make icon extend cleaner
- mod_admin: mobile tweaks
- mod_base: hide original x char
- doc: restructure actions
- admin: include font-awesome version 4
- doc: typo
- mod_base_site: add icon css
- doc: sort lines
- doc: restructure filters
- basesite: remove old fix for logo image
- mod_filestore: use 0 if archive size is undefined
- mod_filestore: tidy up html
- Don’t crash when email is undefined

- doc: formatting
- doc: Document template models
- mod_development: flush when translation file changes
- doc: remove deprecated `%stream%`
- mod_authentication: document template structure
- mod_authentication: remove superseded templates
- mod_authentication: better defaults for social login buttons
- mod_authentication: doc tweak
- mod_facebook: typo
- admin: tidy up Authentication Services page
- admin: fix zlink from editor, remove superseded template
- mod_admin: fix layout person form
- mod_admin: tidy email table
- Social login: use FB compliant “Login with...”
- mod_base: make Z icons independent of FontAwesome
- mod_base: include all font source files
- mod_base: add social login icons
- social login: use icons and update brand colors
- doc: add icons documentation
- mod_admin: remove confusing text
- mod_admin: layout tweaks
- translation tweaks (NL)
- mod_authentication: make login work in dialog
- mod_base: clean up documentation
- mod_admin: include icons css instead of recreating with less
- mod_authentication: shorter messages
- mod_authentication: layout tweaks
- mod_authentication: let admin logon use modal screens
- mod_authentication: default no box
- mod_base: add share icon
- mod_video_embed: add bootstrap responsive class
- mod_base_site: better defaults
- Revert “mod_video_embed: add bootstrap responsive class”
- doc: document ‘page_url with’ syntax
- doc: document more use cases for `| i f`
- mod_bootstrap: version 3.3.2
- doc: show css output when extending icons
- validation: provide message_after param to templates
- Authentication and signup: template refactoring

- doc: reword solution
- mod_artwork: add Material Design icons
- Fix log in verb
- mod_bootstrap: update update script
- mod_bootstrap: fix sourceMappingURL replacement
- mod_bootstrap: update to v3.3.4

David de Boer (10):

- Make menu_subtree compatible with names
- Remove is_integer check for cc5d94
- Create database when starting site
- Add docs
- Connect to “postgres” when creating database
- Fix page block links
- Fix modal signup
- mod_base: Allow to return 401 from API services
- mod_base: Decode JSON body in service calls
- Allow post-login redirect to be passed to logon modal

Jeff Bell (3):

- mod_admin: Flushed out ability to save [object, predicate] pairs when creating new rsc using the dialog_new_rsc action.
- CONTRIBUTORS: added myself to CONTRIBUTORS file
- mod_admin: action_admin_dialog_new_rsc fix

Maas-Maarten Zeeman (43):

- core: Fixes IE problems in zotonic js
- core: IE8 fixes for ubf js
- build: Added delete-deps
- core: Prune context and spawn notifier process only when needed
- core: Remove error and close handlers before ws restarts.
- build: Fix make clean
- core: Fix z_sites_dispatcher so it accepts empty paths. Fixes #842
- core: Added recon application
- mod_base: Removed commented out code.
- mod_seo: Make noindex and notrack configurable from templates
- Changed the websocket implementation.
- Removed comment to fix edoc generation
- mod_base: More careful websocket handshake.
- core: Fix: return default when there is no session.
- core: Export function to create a checksummed url for css and js lib files.
- mod_component: Added new module to make and use components on your page.

- fix for lazy loading css
- core: prevent reconnecting a ws when the page is unloading
- mod_base: Close websocket when unloading page. Fixes #898
- core: Fix a problem with mapping topic names to local names.
- Merge pull request #909 from witeman/emqtt_auth_issue
- mod_component: Fix bug in initializing an already loaded component. Typo
- core: Make sure the z_file_entry fsm uses correct timeouts
- mod_component: delegate onunload to the component controller
- mod_component: updated mithril.js
- mod_component: Ignore load requests for components not in pending state
- z_utils: Allow trusted js values.
- mod_mqtt: Add an ack callback to subscribe calls.
- core: Fix for postback triggered by mqtt events.
- mod_admin_identity: Fix for adding email addresses
- core: Removed compile warning
- mod_component: Make it possible to use iolist init scripts
- core: Cache generated link and script tags when mod_development is disabled
- core: Do a session_context fold to get the right values in the context. Fixes language problem in websockets.
- core: Make acceptor_pool_size configurable. Fixes #923
- core: change default number of acceptors to 75.
- mod_acl_simple_roles: Fix to prevent blanking out all acl settings when role rsc is changed.
- mod_base_site: Make configuring the site's navbar logo easier
- mod_base_site: Minor base template fixes.
- filewatcher: fix, file_changed/2 moved
- mod_base_site: Fixed typo
- mod_base: Make synchronous ajax requests when the page is unloading
- mod_search: Make it possible to pass rsc_lists as parameter to hasanyobject.

Marc Worrell (280):

- smtp: Initialize e-mail server settings on startup. This is needed now that disc_copies are used.
- mod_survey: evaluate empty jump conditions to 'true'
- mod_menu: fix for passing the item_template option to the server when adding items.
- core: do not delete/insert edges when changing the order via mod_menu
- core: remove nested transaction from the z_edge_log_server check.
- mod_admin: if date is not editable, display it as text.
- mod_admin: more generic css for .category spans.
- mod_menu/mod_admin_frontend: enable editing of collections as side-menu.
- mod_menu: bootstrap3 change.
- core: add mime type exception for WMA files, they were recognized as video files.
- core: remove debug statement from m_media

- core: fix concatenating certain combined file streams.
- filestore: use filezcache:locate_monitor/1 to let the filezcache track z_file_entry processes. Fix difference in keys for uploading and * downloading. Better debug/info messages.
- mod_admin_identity: prevent Safari autofilling username/passwords in new user-account forms. Fixes #811
- translation: added some missing nl translations.
- erlydtl: allow lookup of var.key for a list [{<key>, ...}]
- core: m_rsc:p_no_acl/3 request for a non-existing resource should return 'undefined', just like m_rsc:p/3.
- core: For websocket, keep reqdata information for 'is_ssl' checks. When pruning for contexts, keep socket info for the is_ssl check.
- core: add sanitization of text/html-video-embed. Move #context handling out of z_html to z_sanitize.
- core: add acl mime check to m_media:replace/3.
- mod_admin: when replacing a media item, show the oembed/video-embed panels for embedded content.
- base: refactor the moreresults action.
- m_identity: don't crash if #identity_password_match{ } doesn't match any observers.
- core: add lager warnings when modules are stopped.
- core: lager info with modules to be started.
- mod_tkvstore: don't start as named module.
- admin: force to select category when adding new content.
- menu: edge menu sorter has a problem with sorting nested collections. Disable sorting for now.
- core: module start/stop progress messages are now debug level.
- core: add m.modules.active.mod_foobar to test if a modules is active. Remove checks with the code server if a module is running.
- m.modules: replace usage of m.modules.info. with m.modules.active.
- core: on context prune-for-scomp, leave an empty list for request headers. Normalize user-agent lookup.
- core: fix args for transport ack.
- mod_email_receive: when adding recipients, catch references to non existing rsc
- core: make session cookie name configurable (solves problems where old cookies might interfere, especially on Chrome)
- Merge pull request #864 from driebit/fix-menu-subtree-name
- Merge pull request #865 from driebit/remove-is-int-menu-subtree
- core: z_search: fix acl check sql query.
- mod_survey: allow printable overview if user has edit rights.
- mod_base: remove extra </div> from phone/_navbar
- mod_admin: remove api dispatch rule, also defined in mod_base.
- mod_base: for catinclud, don't assign passed categories to 'id'. Only assign a resource id to id.
- mod_menu: remove console.log message.
- core: allow a non-integer category id to be passed to all_flat/2
- mod_admin/mod_admin_frontend: preparations to allow creation of resources via the edit page.
- core: allow setting any rsc property that is 'undefined' to 'false'.
- core: ensure all db timestamp columns have a time zone.

- mod_menu: correct pubsub.publish topic.
- mod_admin_frontend: fix a problem where combining the nestedSortable.js lib with other js files will result in erroneous drag behaviour
- mod_menu/admin_frontend: final fix for new-page topic on menu insertion.
- core: removed debug from z_pivot_rsc
- core: fix problem where the custom redirects form was not saved
- core: fix problem where mod_signed_url could not keep the user logged on.
- mod_mqtt: fix problem where removing one topic listener removed all listeners. Cleanup live subscriptions for removed elemnts.
- core: added comment explaining expire_1 and expire_n for sessions. Issue #881
- Merge pull request #880 from witeman/mod_authentication_logon_display_bugfix
- smtp: more relaxed error handling for spamd errors.
- install: use openssl to generate the admin password, as tr/urandom combo hangs on OS X. Fixes #847
- mod_base: do not redirect if redirect id is set to undefined
- mod_base: in do_popupwindow use e.preventDefault() to play nice with multiple click event listeners.
- mod_mqtt: fix js error in for loops.
- core: the { % script % } tag has now arguments.
- mod_authentication: Refactor twitter/facebook logon and signup code.
- mod_facebook: add delegate for saving settings.
- mod_base: fix js error in livevalidation.
- mod_authentication: export send_reminder/2 and lookup_identities/2.
- Merge pull request #872 from driebit/auto-create-db
- docs: adapt docs to changes in files.
- core: fix specs in z_db.
- mod_oembed: remove http: protocol from embed html, this enables content to be viewable on https: pages.
- core: add exceptions for .xls and .xlsx files to z_media_identify. Fixes #893
- media/embed: fixes for twitter streaming, added notifications for importing and analyzing fetch url media-data.
- mod_search: allow dates like 'now' and '+2 weeks' in search questions.
- mod_admin: return the correct context in controller_admin_media_preview
- mod_translation: add more rtl languages.
- mod_twitter: fix edoc problem.
- doc: add mod_component.
- mod_oembed: don't display the media twice in the admin
- docs: added 0.12.2 release notes
- mod_oembed: add missing fallback _oembed_embeddable.tpl
- mod_oembed/mod_twitter: prefer our own 'website' extraction above oembed links. Pass the tweet url in the import_resource notification
- mod_twitter: fix import of tweets with special-html chars, was double escaped in title.
- mod_admin: always show media content, independent if size was defined.

- mod_oembed: sanitize received oembed code html and texts.
- core: add instagram js and urls to whitelist.
- mod_survey: fix problem where displaying the results did not work due to move sanitization functions.
- core: lock new z_stdlib library. Fix twerl git url. Fixes #895
- docs: fix length of header-underline
- docs: add 0.12.3 release notes
- Merge pull request #897 from driebit/connect-postgres-db
- core: refactor database creation on site init.
- mod_authentication: add authentication via LinkedIn. Add possibility to connect/disconnect accounts with FB/LinkedIn/Twitter. Fix * redirects after using an external service for authentication. List connected authentication services in the password reminder email.
- mod_linkedin: modify template for bootstrap3
- m_identity: fix spec of get_rsc_types/2
- mod_admin_identity: some extra padding for the identity verification page.
- mod_authentication: add optional page_logon for logon-title and an alert box on the logon page.
- mod_authentication: add special error message if there are cookie problems and the current browser is Safari 8. Issue #902
- mod_signup: show external auth services for signup using the logon methods. Also always force the presence of an username_pw identity for * signed up users.
- mod_linkedin: seems LinkedIn doesn't like URL encoded secrets?
- mod_admin: also log stacktrace on a catch.
- mod_oembed/mod_video_embed: fix problem with access rights if new media insert was done without admin rights.
- core: set the edge's creator_id on insert
- mod_survey: fix 'stop' survey button.
- core: fix stacktrace shown in transport lager messages.
- core: move erlang:get_stacktrace() outside of lager calls. Otherwise a stacktrace of lager will be shown due to the parse transforms.
- mod_twitter: Fix twitter redirect url
- mod_admin: add pubsub and some related javascripts. needed for live tags etc.
- mod_authentication/mod_twitter/etc: changes for new font-awesome, bs3 and some small tpl fixes
- mod_oembed: don't crash on oembed connect timeouts.
- mod_instagram: authenticate and import tags from Instagram
- core: fix problem where erlydtl_runtime crashed on fetching a value from a 'time_not_exists' atom.
- core: truncate the slug at 78 characters.
- core: fix in m_identity where fetching email identities could loop on a check if the email property was known as an identity.
- mod_base: handle ping/pong websocket control frames, remove name conflict with zotonic ping/pong.
- mod_linkedin: try to workaround a problem where LinkedIn doesn't recognize the Access Token it just handed out.
- mod_linkedin: work around for a problem with access-tokens at linkedin.
- core: allow binaries for some special keys.

- mod_import_csv: major changes to mod_import_csv.
- mod_instagram: fix property name in comment.
- mod_import_csv: added checks to the model creation.
- mod_base: check dialog height repeatedly, account for rounding errors in height calculation.
- core: add 'expected' option to m_rsc:update.
- mod_110n: add utf-8 encoding hints to source file
- mod_110n: adaptations for utf8 parsing changes in R17
- core: fix for importing structured blocks (like during imports)
- core: on startup z_dropbox moves now all processing files to unhandled.
- core: z_datetime:to_datetime/1 now also handles numerical timestamps.
- core: m_rsc:update now converts non-tuple dates and handles creator/modified on import correctly.
- mod_import_csv: fixes for file handling and medium_url imports.
- core: fix problem in m_rsc:update where modified was not set on save.
- mod_base: added the filter 'trans_filter_filled'
- core: remove unreachable code.
- docs: added placeholders.
- mod_import_csv: fix handling of blocks. Add support for 'blocks.name.field' keys in m_rsc:update
- core: add compile/0 and /1 to z.erl, for compiling without flush.
- docs: added xelatex target to generate PDF.
- mod_mqtt: allow topics like ['~site', 'rsc', 1234].
- mod_admin_identity: typo in translation.
- mod_admin_identity: publish identity changes to the topic ~/rsc/1234/identity.
- core: added e.issuu.com and static.issuu.com to the sanitizer whitelist.
- mod_import_csv/core: fixes for importing categories, new properties, corrected basename in #import_csv_definition{ }
- doc: cleanup of pdf version.
- docs: add link to the pdf version.
- docs: better link text to the pdf version.
- docs: move the cookbooks to their own top level chapter.
- docs: correct the {% call %} documentation.
- skel: add mod_mqtt to the base site, as it is needed by mod_admin
- core: correct the language utf8 encoding for R16+
- mod_base: added filter trans_filter_filled/3 export.
- mod_admin: fix a problem where quick-editing a rsc adds all enabled languages.
- mod_base: filter-sort of undefined is undefined.
- core: correctly parse multipart/signed emails.
- core: better handling of erroneous urls for the z_file/media routines.
- core: extra utf8 sanitization of received email's subject, text, html and from.
- mod_content_groups: new module to categorize content into groups for the access control.

- `m_hierarchy`: merge `m_menu_hierarchy` and `m_category` into `m_hierarchy`. `m_category` now interfaces to `m_hierarchy`.
- Merge pull request #922 from `imagency/master`
- Merge pull request #921 from `CyBeRoni/configure-dbcreation`
- `core`: added support for 'is_dependent' flag.
- `core`: set longer timeout for `renumber_pivot_task` query.
- `core`: check if `rsc` existed before adding a `rsc_gone` entry.
- `mod_admin_identity`: correct the button class on the identity verification page.
- `mod_mailinglist`: more efficient query for polling scheduled mailings.
- `mod_mqtt`: allow 'live' wiring postbacks to mqtt topics, depending on the presence of an element-id.
- `mod_email_status`: new module to track email recipient status. Changes to the email bounce, sent and failure notifications.
- `docs`: fix length of header underline.
- `docs`: add `mod_email_status` placeholders.
- `mod_email_status`: fix problem with re-installing the model.
- `mod_email_status`: handle non-binary error status values.
- `mow_acl_user_groups`: first working version. To do: docs and test interface.
- `mod_menu`: breack cyclic dependency [`mod_acl_user_groups`,`mod_authentication`,`mod_admin`,`mod_menu`,`mod_content_group`]
- Merge pull request #933 from `driebit/fix-page-block-links`
- Merge pull request #936 from `CyBeRoni/override-pidfile`
- `mod_base`: remove 'render-update' observer, leave template rendering to modules for more control
- `mod_l10n`: add some extra country-name to country-code mappings.
- `mod_l10n`: fix iso lookup of Serbia and Montenegro
- `core`: add log message when adding `rsc.is_dependent` and `rsc.content_group_id`.
- `core`: add `is_meta/2` function to test if a category is a meta category, needed for determining the default content group in `m_rsc:get/2`
- `mod_content_groups`: use the `m_category:is_meta/2` to determine the default content group.
- `mod_content_groups`: convert tabs to spaces.
- `mod_acl_user_groups`: fix search restriction if user can't see any content groups.
- `core`: fix progressbar for form posts with body.
- `mod_content_groups`: fix adding `content_group_id` for `#rsc_get{ }` on an non-existing resource.
- `mod_base`: added the action `update_iframe`
- `mod_email_status`: add status dialog and simple templates to see the status of an email address.
- `mod_base`: missing part of the `update_iframe` addition
- `mod_base`: cross-platform fix for `update_iframe`
- `mod_survey`: add `is_autostart` option. Fix some minor template errors.
- `core`: use 'optional' includes for blocks.
- `mod_survey`: use the 'optional' include for the templates.
- `core`: normalize the name of the `z_update_iframe` JS function with its Erlang equivalent.
- `mod_import_csv`: Force convert to utf8 of the top row. Add test for ';' as a column separator.

- core: fix 'next' page number for queries returning a `#search_result{ }`.
- mod_admin: use search query 'admin_overview_query' for the overview page if it is defined.
- mod_admin: add admin_content_query, also displayed in the content menu.
- mod_admin: enable 'all pages' button if a qquery is defined.
- core: ensure that the compiled template's name equals the one found via an (optional) catinclude. Issue #938
- mod_content_groups: 17.x compatibility for string with unicode character (breaks 15.x)
- mod_survey: fix crash on prep_answer if question name is not unique
- core: fix sql query in category delete.
- mod_acl_user_groups/mod_content_groups: ensure the hierarchy if the content/user groups are changed.
- mod_admin: in tree-list, show the category in gray.
- core: escape filenames using a single quot. Fixes #924
- core: shortcut for lib file lookup without filters (don't check the file store)
- core: add mod_mqtt to the default installed modules (as it is a dependency of mod_admin).
- core: fix m_category/2 lookup. Thanks to Alvaro Pagliari.
- Merge pull request #946 from AlainODea/BUGFIX_issue891
- core: fix problem where a gmail autoreply was classified as a bounce. Fix tracking of some email errors. Started collecting email test data.
- mod_email_status: change info message about previous problems.
- mod_admin: pass all strings from the new-rsc dialog form. Fixes #948
- mod_acl_user_groups: when changing category/content-group, show the meta category if the id is a meta or the current user is 1 (the admin)
- Merge branch 'master' into acl-content-groups
- zotonic_status: use default 'post' method for the logon form to prevent showing the password if there is a js error.
- mod_linkedin: adaptations for LinkedIn API changes.
- mod_linkedin: fix for picture-url.
- mod_search: fix a problem where a 'hassubject=[...]' query term was incorrectly parsed. Fixes #950
- mod_survey: change the thank you text, remove the 'Be sure to come back for other surveys' text.
- mod_search: add cat_exact query argument.
- mod_base: fix html_escape function in zotonic-1.0.js
- mod_admin_identity: on the users page, only show those with an username_pw entry. Issue #747
- mod_admin_identity: show 'email status' buttons if mod_email_status is enabled.
- mod_search: add 2nd ordering on -publication_start to featured search.
- mod_search: fix search_query for 'hasobject=123'. Fixes #953
- core: do not create atoms from rsc names in catinclude.
- core: add tick_10m and tick_6h notifications
- core: fix a problem where ImageMagick identified PDF files as PBM.
- core: cleanup location_lat/lng update/pivot code.
- mod_base: in scomp_lazy, ensure the 'visible' argument to all 'morerresults' actions

- Merge pull request #955 from pguyot/patch-2
- Merge pull request #956 from pguyot/patch-3
- mod_admin: pass all strings from the new-rsc dialog form. Fixes #948
- Merge pull request #958 from driebit/fix-signup-delegate
- core: adding some test data for smtp tests.
- Merge pull request #960 from trigeek38/fix-dialog-new-rsc
- Merge pull request #961 from trigeek38/fix-my-fix
- mod_admin_identity: fix user query.
- mod_admin: make action admin_dialog_new_rsc more robust against illegal objects arg
- mod_admin_identity: small changes/fixes to the users query.
- mod_acl_user_groups: add 'block' option to access control rules. Update the view via a 'live' subscribe.
- Start removing R15 support.
- core: move more code into the m_hierarchy update to prevent race conditions.
- core: in m_hierarchy, lock the rows of a hierarchy when updating.
- core: trace module (re-)loads and restart modules if any exported functions are changed. Issue #964
- core: bit more silent start/stop of modules.
- core: add realtime block list checks for incoming email. Needs updated z_stdlib.
- mod_development: move fswatch/inotify to the core.
- core: add cli commands 'zotonic startsite stopsite restartsite'. Fixes #964
- core: fix delete of timer in fswatch.
- core: log warnings if trying to insert duplicate rsc name/uri/path
- Fixes for language handling. Allow undefined pref_language for user. Filter on valid language code on pref_language update. Show select list * with all valid languages in /admin/translation.
- core: add m_rsc 'is_linkable' lookup, also in z_acl and m_acl
- docs: add placeholders for ACL documentation.
- mod_search: allow search terms in texts without '=value' arg, map to '=true'. Fixes #970
- [docs] proposal for site-fsm
- [core] add possibility to fetch sub-trees from a hierarchy. Example: m.hierarchy.content_group[id].tree1
- [docs] A websocket connection is opened by the browser, accepted by the server.
- mod_menu: fix issue that on insertion of a new item it is added to all sub-menus. Fixes #971
- mod_search: add query term 'hasanyobject', to search using an 'or' on outgoing edges. Fixes #968
- mod_acl_user_groups: new import/export version of the acl rules. Include the content groups, user groups and hierarchies in the export.
- mod_acl_user_groups: move ensure_name to the core m_rsc module.
- mod_menu: in the menu-editor, add open/close buttons for submenus. This makes editing big lists easier.
- docs: fix documentation of smtp server settings. Also fix z_config settings.
- core: name correction, the bounce server is a complete smtp receive server.
- docs: clarification of the bounce addresses.
- core: allow pivot task to return {delay, DateTime}.
- core: fix a problem where m_rsc:ensure_name/2 could insert duplicate names. Fixes #974

- core: add reserved usernames to prevent users signing up as these. Fixes #929
- core: if some reserved username was taken, allow to update the password. Issue #929
- core: allow diff between Date and DateTime.
- core: add 'flickrit.com' to the iframe whitelist.
- core: copy tmpfiles if they are mailed, add periodic cleanup of the tmp folder. Fixes #972
- core: fix a problem where a fulltext search with an empty category list didn't return any results. Fixes #976
- core: fix mqtt login, don't start a session_page if there is no session or reqdata. Fixes #973
- mod_admin: fix problem where the lazy-loader for moreresults kept loading till no results were left.
- Set version number to 0.13.0
- Lock deps
- docs: 0.13.0 release notes and some extra (minimal) documentation.
- docs: add tentative 0.13.0 release date
- core: determine mime type of attachments if it was not given.
- core: use 'rebar' to compile zotonic from the command line. Move .yrl output to default location for rebar.
> Use skip_deps for 'make compile-zotonic'. Add 'make refresh-deps' and 'make list-deps'. Fixes #978

Marco Wessel (4):

- Allow configuration of db creation and installation
- Add commented dbinstall/dbcreate options + explanation
- Add warning message that db won't be created or installed
- Allow to override pidfile location with env var

Paul Guyot (3):

- Fix typo in _block_view_survey_thurstone.tpl
- Handle &# entities when splitting markers
- mod_survey: Fix multi & single HTML structure for thurstone checkboxes

Paul Monson (1):

- Update README link

Sergei (2):

- fix dependencies build order
- force rebar to build 'setup' app

Witeman Zheng (1):

- mod_authentication: fix logon_box form input "password"

imagency (12):

- TinyMce support for Russian

(2):

- emqtt_auth_zotonic issue would cause crashed when mqtt client try to connect onto it.
- Fix the emqtt client connection issue.

Release 0.13.1

Welcome Zotonic 0.13.1, released on July 29, 2015.

This is a maintenance release of Zotonic 0.13

A complete *git shortlog* since the 0.13.0 release is added below these release notes.

Commits since 0.13.0

There were 16 commits since release 0.13.0.

The committers were: Arthur Clemens, David de Boer, Fred Pook, and Marc Worrell.

Big thanks to the committers and all other people contributing to Zotonic!

Git shortlog

Arthur Clemens (4):

- admin_development: Add hostname to dropdown to indicate this does not need to be entered
- mod_base: Use dialog width param, fix vertical centering
- Remove debugging
- doc: Document action dialog parameters

David de Boer (4):

- Make module and site includes path-independent
- Support custom redirect page after social media logon
- Make post-logon actions dynamic
- Fix opening admin dialog by removing call to dialogReposition()

Fred Pook (2):

- Added dutch translations
- Added knightlab audio plugin to the whitelist

Marc Worrell (6):

- doc: fix sidebar versions
- Merge pull request #980 from driebit/compile-include-paths
- docs: fix title attr of docs sidebar link.
- mod_search: catch errors when evaluating stored search questions. Issue #988
- mod_base: filter to_integer returns now 'undefined' for invalid inputs.
- mod_oauth: new model functions to directly add a application (with generated tokens)

Release 0.13.2

Welcome Zotonic 0.13.2, released on August 11, 2015.

This is a maintenance release of Zotonic 0.13

The main change is that this version compiles correctly on Erlang 18.

Commits since 0.13.1

There were 15 commits since release 0.13.1.

Big thanks to all people contributing to Zotonic!

Git shortlog

Marc Worrell (15):

- docs: fix ref in 0.13.1 release notes.
- mod_oauth: fix user_id initialization on direct app creation
- mod_oauth: use the abs_url of the request path for checking the signature. This fixes an issue where OAuth access to https sites is denied.
- mod_content_groups: also add the content group as related data to the pivot.
- mod_base/mod_admin: add optional redirect to the set website, using 'is_website_redirect'. Allow overruling the controller's 'is_canonical' flag with 'is_page_path_multiple'. Add documentation for rsc properties 'is_dependent', 'content_group_id', 'is_page_path_multiple' and 'is_website_redirect'.
- New z_stdlib
- core: allow max 5 smtp connections per relay, set max parallel smtp senders to 100. Fixes #993
- mod_survey: open survey results from mod_admin_frontend in new window.
- mod_base: hide any files and directories whose name starts with a '.' Fixes #991
- core: fix compilation on Erlang 18.0. Fixes #979
- core: add erlang 18 as travis target.
- Locked new deps.
- Preliminary 0.13.2 release and release notes.
- New z_stdlib
- mod_content_groups: on pivot add the complete path for the content-group as related ids.

Release 0.13.3

Welcome Zotonic 0.13.3, released on September 9, 2015.

This is a maintenance release of Zotonic 0.13

Main changes are:

- refinements to the admin interface by Arthur Clemens
- included Bootstrap version has been upgraded to version 3.3.5
- fixes for some issues around the newly introduced content groups
- CORS support for the API (Thanks to @ghosthamlet!)
- fixes some issues with the websocket connection
- fixes an issue where the wrong language could be selected

Commits since 0.13.2

There were 103 commits since release 0.13.2.

Big thanks to all people contributing to Zotonic!

Git shortlog

Arjan Scherpenisse (1):

- mod_base: Support CORS API requests

Arthur Clemens (76):

- mod_admin: (small screen) prevent scrolling of body when menu open
- mod_admin: fix lost tree-list styling
- mod_admin: style tweaks
- mod_admin: position help buttons in widget headers
- mod_base: remove redundant div container in modal body
- Remove inadvertently created files
- mod_admin: create user dropdown in main nav
- mod_admin: improve sorting of page overview
- mod_admin: optional category-specific column
- mod_admin: update admin bootstrap
- mod_admin: prevent executing content_group query
- doc: Add dev note on translations; add batch update command
- mod_translation: Localize messages
- mod_base: make pager numbering more logical
- mod_base: update translation files
- mod_bootstrap: update to 3.3.5
- mod_base: move pager css overrides to mod_admin
- mod_admin: reorganize edit page header
- mod_admin: Make topbar elements fit on a smaller screen when country and account buttons are visible
- docs: release notes in reverse chronological order
- mod_editor_tinymce: keep text inside moved blocks
- mod_editor_tinymce: add translation files
- mod_editor_tinymce: Add latest TinyMCE 4.2.4, and updated codemirror plugin
- mod_admin: reorganize page edit header: keep image reference after saving
- mod_admin: optional category-specific column
- mod_admin: Fix unlinking media on Enter
- docs: Mobile friendly layout
- docs: Mobile friendly layout: fix search box layout
- mod_admin: revert “also check in .css files resulting from .less compilation”
- File watcher: read optional config file to run lessc with params and compile just the main less file
- mod_admin: reorganize page edit header: hide “by” if modified/creator does not exist
- mod_acl_user_groups: minor frontend tweaks
- mod_base: minor icon font tweaks
- mod_base: minor icon font tweaks

- Admin: straighten admin-header blocks
- Admin: add breadcrumbs to hierarchical pages
- File watcher: read optional config file to run lessc with params and compile just the main less file
- mod_translation: tidy up language settings dialog
- mod_admin: give logo a width to prevent jumping navbar
- docs: Add note about /etc/hosts on first site
- Fallback language
- Revert partial css files
- mod_admin: fix position of drag icon in connection item
- mod_admin: Update Dutch translations
- mod_menu: A couple of Dutch translations
- mod_menu: Improve text (and remove html markup)
- mod_menu: Fix fuzzy Dutch translations
- mod_admin_predicate: Fix string escaping in translation text
- mod_admin: Fix extraneous spaces; fix fuzzy Dutch
- mod_110n: typo
- mod_admin: tweak layout top header
- mod_admin: tweak widget colors on edit page
- mod_admin: tweak colors dashboard
- mod_admin: remove period from “No items.”
- Fix translations by unmatched spaces in text
- mod_admin: Edit category dialog: add button to pages
- mod_admin: improve layout edit form in dialog
- mod_admin: Fix markup in translation text
- mod_admin: tweak colors dashboard: couple of bug fixes
- mod_admin: add test query button
- mod_admin: tweak dashboard and dialog navs
- mod_translation: Give feedback when no languages can be copied
- mod_admin: update translations
- mod_admin: update translations with msguniq
- docs: update admin layout cookbook
- mod_admin: couple of translation fixes
- mod_admin_identity: update translation files; add NL translations
- mod_admin: add button container to widget header if it is not present in the template
- Small fixes
- docs: update cookbook for dynamic select boxes
- mod_admin: hide minimize button in dialog
- mod_admin: tweak menu item style
- mod_admin: make connections list reusable

- `mod_admin`: make connections list reusable: fix button label, shorten add link and update translations
- `mod_admin`: fix dialog header translation
- `docs`: update info about translations

Maas-Maarten Zeeman (7):

- `mod_base`: Only use the websocket when it is in OPEN state.
- `mod_backup`: Catch errors when restoring a resource. Better user feedback in case of problems.
- `authentication`: Set user preferences like language and tz during a login.
- `core`: Show an error when there is a problem rendering `error.tpl`
- `core`: Set `reqdata` in context when rendering `error.tpl`. Fixes #1013
- `core`: Make sure the error template is rendered with the expected variables and a good context.
- `core`: Removed error logging code on ws error handler.

Marc Worrell (19):

- `mod_acl_user_groups`: show default user group if user is not member of any group.
- `mod_acl_user_groups`: when displaying user's groups, check for displayed user being an user.
- `mod_admin`: in `dialog_edit_basics`, only show the 'full edit' button if the user can access either `mod_admin` or `mod_admin_frontend`
- `core`: refactored pivot queue polling, now will poll faster if anything was found in the queue.
- `core`: force reload of client if `page_id` in `z_msg` is undefined.
- `core`: fix a problem where a multipart post form could loop if the end-boundary was missing
- `mod_mqtt`: fix race condition in re-subscribe of mqtt listeners after a module restart.
- `mod_admin`: also check in `.css` files resulting from `.less` compilation.
- `mod_base`: fix `controller_api` context handling in `to_json`.
- `core`: allow '1' and 'true' as timezones (both map to UTC)
- `mod_admin`: also check in `.css` files resulting from `.less` compilation.
- `mod_base`: fix a problem where a pager on page with 'id' and was not using the `page_path` of the resource. Fixes #1005
- `core`: set default timeout of sql queries higher (from 5sec to 30sec)
- `mod_base`: fix a problem with checking if the current path is the canonical path. Issue #1010
- `mod_admin`: add generated `.css` files for systems without less installed.
- `core`: change logging notifications from `{log, ...}` to `#zlog{ }`. Issue #992
- `core`: fix typo in `os:timestamp()` call.
- Merge branch 'release-0.13.x' of [github.com:zotonic/zotonic](https://github.com/zotonic/zotonic) into `release-0.13.x`
- `core`: add 'www.google.com/maps/' to the sanitizer white list.

Release 0.13.4

Welcome Zotonic 0.13.4, released on September 16, 2015.

This is a maintenance release of Zotonic 0.13

Main changes are:

- fixed an issue where `mod_survey` answers could be shown without escaping

- fixes for admin css and template issues
- corrected handling of unsafe (escaped) characters in email addresses
- easier access rule definitions by special handling of 'meta' category
- fixed an issue where mod_video would crash if there was an empty ffmpeg configuration
- fixed an issue where the last row of an imported CSV file could be reversed

Commits since 0.13.3

There were 34 commits since release 0.13.3.

Big thanks to all people contributing to Zotonic!

Git shortlog

Arthur Clemens (17):

- Pass property is_dependent in datamodel; show status in edit page even if protected
- Apply tab style to all tabs within widgets
- Typo
- mod_admin: use relative date in event info
- docs: exemplify unique ids
- mod_admin: allow other modules to get result from connect dialog by passing option 'delegate'
- mod_admin: give first input field focus when switching tab
- mod_admin: improve layout of navigation bars
- mod_admin: improve layout of navigation bars: fix toolbar padding
- mod_admin: improve layout of navigation bars: improve some colors
- docs: add example of a postback action called from javascript
- mod_admin: add styling for justified tabs
- doc tweaks
- mod_admin: override style for code
- Merge remote-tracking branch 'origin/release-0.13.x' into release-0.13.x
- Revert "Merge remote-tracking branch 'origin/release-0.13.x' into release-0.13.x"
- mod_admin: override style for code: fix top bar

Marc Worrell (16):

- Merge pull request #1022 from millenaarg/release-0.13.x
- mod_import_csv: fix an issue where the last row is reversed iff there is no empty line at the end of the file.
- mod_backup: add option to make a database-only backup. Also make the backup list a live include.
- mod_backup: fix build by removing '&' from @doc
- core: fix a problem where email addresses couldn't have a ' (single quot) in them. Fixes #1023
- mod_video: use the default ffmpeg command if the configured ffmpeg command is empty.
- mod_video: use the default ffprobe and ffmpeg_preview command if the configured command is empty.
- New dependencies

- `mod_acl_user_groups`: ACL rules only apply to category 'meta' if either 'meta' or the system content group is mentioned in the ACL rule. Fixes #1030
- `mod_110n`: added Kosovo (code xk) to the country list.
- `mod_acl_user_groups`: add 'view-only' option. Fixes #1034
- `mod_acl_user_groups`: show error if access to the ACL rules view is denied.
- `mod_acl_user_groups`: remove 'not' that was left in for checking.
- `core`: notify `#hierarchy_updated` observers which ids have added and/or removed.
- `mod_menu`: show the category in menu-list items muted.
- `mod_survey/mod_base`: url escape google chart arguments; escape survey label names.

millenaarg (1):

- Added reference to `scomps-script`

Release 0.13.5

Welcome Zotonic 0.13.5, released on October 27, 2015.

This is a maintenance release of Zotonic 0.13

Main changes are:

- Updates to the bundled tinymce and code view plugins
- Fulltext search fixes, search terms could be stemmed multiple times
- Many fixes to the admin css and html
- Fixes an ACL user groups problem where the sudo permissions didn't reflect the admin
- Fixes a problem with the template compiler where a template could be compiled multiple times
- Translation option to force the default language of a site
- Fixes for the comet and postback controllers where push data was not sent with postback requests
- New admin filter 'temporary_rsc'
- Fix for a problem in `mod_video` where a video could be rendered with a color space that is incompatible with QuickTime

Commits since 0.13.4

There were 101 commits since release 0.13.4.

Big thanks to all people contributing to Zotonic!

Git shortlog

Alex Popov (1):

- Fixes `zotonic/zotonic#1027` issue with `zotonic-module` script

Arjan Scherpenisse (1):

- `mod_admin`: Fix media upload when no content groups configured

Arthur Clemens (41):

- `mod_editor_tinymce`: load the correct lib files
- `mod_admin`: improve reusability of connections interface

- mod_admin: fix background color of top menu on small screen
- Make fswatch work with spaces in paths
- mod_admin: prevent disconnecting when pressing enter
- mod_admin: optimize display of connection list
- doc: doc tweak
- mod_admin: tweaks
- mod_search: return multiple results for search by id
- mod_admin: improve layout of thumbnails in find dialog
- mod_admin: find dialog: visually distinguish existing connections for current predicate
- mod_admin: align connection dialogs to browser top
- mod_admin: remove redundant intro text
- mod_admin: find dialog: visually distinguish existing connections for current predicate
- mod_base: safer defaults
- mod_base: safer defaults: add type “submit” to submitting buttons
- mod_base: add bootstrap classname to invalid form
- mod_editor_tinymce: link to correct codemirror
- mod_editor_tinymce: use bootstrap classnames
- mod_admin: admin_connect_select: fix actions in cases the value undefined is passed from a template
- docs: refactor custom admin edit page
- revert: template comment not allowed
- mod_admin: remove margin from empty tree list
- Revert “mod_base: safer defaults”
- Revert “mod_base: safer defaults: add type “submit” to submitting buttons”
- doc: describe “s” formatting character
- mod_menu: allow nested divs inside the list item
- mod_admin: make labels in list items visible
- mod_admin: prevent emptied tree list taking up space
- mod_menu: position all dialogs at the top
- mod_admin: show default text when no features are enabled
- zotonic_status: fix error message
- doc: update zotonic_status site screenshots
- mod_admin: tweak colors
- mod_admin: tweak panel heading
- mod_admin: align variables with bootstrap config
- mod_admin: optimize layout of date fields
- mod_admin: tweak colors
- mod_admin: make growl messages less dark
- mod_editor_tinymce: update codemirror plugin
- mod_editor_tinymce: replace codemirror plugin with default code editor

Maas-Maarten Zeeman (3):

- core: Fix for using template rendered file requests
- core: Fix for uncollapsing lib files without a dirname
- mod_component: Refactor of component injection code

Marc Worrell (55):

- mod_email_status: styling of some buttons.
- core: better error returns for m_identity:set_username_pw/4
- mod_backup: fix periodic backup - broke because of added 'is full backup' flag in the backup list.
- Fix a problem where the connected depictions were not always updated when adding a depiction.
- Make the admin 'replace media' buttons an anchor element instead of a button.
- Ensure that newly created connected content is in the same content group as the subject of the connection.
- core: new pivot task return: {delay, Delay, NewArgs}
- mod_filestore: changes to the 'move to local/remote' queries. This fixes an issue where moving many files at the same time results in timeouts during move initialization.
- core: refactor the template compiler, use separate compilation process so that 'changed' checks can continue whilst compiling.
- mod_acl_user_groups: add type="submit" to rule-add button
- core: fix return value check of password change.
- mod_acl_user_groups: set the default user group for user 1 and sudo to the administrators-group. Fixes #1042
- mod_admin: fix a problem with uploading media if no subject_id is present
- mod_acl_user_groups: fix default manager-group membership for admin or sudo
- mod_contact: add form-group class to divs, this fixes the problem that validation errors were not flagged. Thanks to @fredpook
- mod_search: add query term 'match_objects'. This is similar to the {match_objects id=...} query.
- core: if postback is handled, also return the page message queue.
- mod_base: fix fetching queued transport messages
- mod_translation: add config 'mod_translation.force_default' to prefer the default language above the browser accept languages.
- acl: add option to check permission as-if the typical member has been logged in. Issue #1050
- core: better error message in z_notifier.
- core: fix find_value error in m_acl. Fixes #1052
- mod_acl_user_groups: merge fix for sudo user groups.
- core: fix a problem where the authoritative uri could be requested by the datamodel installer before the id dispatch rule was known.
- core: change stemming of the full text indexes.
- core: fix a problem with abs_url and urls starting with '//'. This now correctly adds the protocol.
- core: Stop email transmission if sender has been disabled or deleted. Fixes #1046
- core: defined an admin as someone who is either user 1, sudo, or allowed to use the mod_admin_config. Fixes #1033
- core: always return an error when looking up 'undefined' with name_to_id.

- `mod_search`: fix a problem where search texts were stemmed twice.
- `mod_search`: fix for dutch wildcard tsquery with dutch stemming; 'overstee' and 'oversteek' were mapped to 'overstee':* and 'overstek':*
- `erlydtl`: hack fix for escape filter application in arg lists. Like '{foo bar=xlescape}'. For now direct mapping to the `force_escape` filter.
- `core`: add dialogs and routines to move categories/content-groups/user-groups/predicates if one is deleted. Fixes #1041
- `mod_admin`: fix js problem with touch js
- `mod_atom`: fix test for new `z_stdlib:strip`
- New locked version for `z_stdlib` and `s3filez`
- `mod_base`: fix `controller_comet` for a problem where comet didn't flush on page data if a postback controller fetched the data before. (picked from d855b1abec4b55e0c0ab7c77f4a66c08da3194bd)
- `mod_base`: export the post-loop in `controller_comet`.
- `mod_search`: show growl error message on missing predicates and categories. Missing category matches the empty range. Fixes #998
- `mod_search`: ignore 'undefined' categories in search.
- `mod_rest`: fix a problem where a backup file could not be restored if the content group id was unknown. Fixes #1011
- `mod_search`: fix fulltext query sql. Fix searching for prefixes which are wildcards. Fixes #1061
- `core`: `z_utils:is_empty/1`, define `St. Juttemis` date as an empty value..
- `mod_base`: add parameter less filter 'filter' to remove empty values from a list.
- `erlydtl`: stricter definition of `gb_trees`, added lookup of `m_search_result` properties.
- `core`: adds `z_pivot_rsc/get_task/1,2,3` and `/4`. Used to check pivoted tasks
- `mod_admin`: add filter `temporary_rsc`. Fixes #1044
- `docs`: add some doc placeholders and documentation for the action `mask_progress`.
- `mod_video`: add '-pix_fmt yuv420p' for QuickTime compatibility
- `mod_email_status`: don't register a 'sender_disabled' error with the recipient's status.
- `core`: expose `pivot_location_lat`, `pivot_location_lng` and `pivot_geocode_qhash` as `m_rsc` properties.
- `core`: correct identify of MS Visio files.
- `core`: allow 'gray' and 'grey' for the grey image filter.
- `core`: set `Cache-Control-Allow-Origin: *` on all `controller_file` replies.
- Lock deps. Fixes a problem where the empty url was changed into '/'. Fixes #1066

Release 0.13.6

Welcome Zotonic 0.13.6, released on December 14, 2015.

This is a maintenance release of Zotonic 0.13

Main changes are:

- Fix the deps version lock by adding `USE_REBAR_LOCKED`
- New `bin/zotonic` command to start without a shell: *start-nodaemon*
- Overview of edges in the `mod_admin`
- New module `mod_admin_merge` to merge resources

- Retained messages for ~pagesession topics
- Performance fixes for api calls
- Significant better revision history in mod_backup
- Add `no_touch` option to `m_rsc:update/4`
- Many smaller fixes

The following 0.13.7 release will incorporate some performance and stability enhancements from the master (1.0-dev) branch.

Commits since 0.13.5

There were 96 commits since release 0.13.5.

Big thanks to all people contributing to Zotonic!

Git shortlog

Arjan Scherpenisse (1):

- core: Revert edown revision bump due to compilation breakage

Arthur Clemens (24):

- Tweak input border border to match buttons
- doc: typo
- mod_admin_predicate: fine-tune display of items
- mod_signup: update translation strings, update Dutch
- mod_authentication: update translation strings; update Dutch
- doc: move number formatting to strings
- doc: update examples so that they will work
- mod_admin: style disabled tabs
- doc: fix make script
- doc: fix make script
- mod_admin: add category name to category dialog on edit page
- mod_admin: add category preselection to connect-find dialog
- filewatcher: handle spaces filepath in less command
- filewatcher: handle spaces filepath in less command
- mod_admin: reverse created/modified
- Remove borders from connections list
- mod_admin: align category param between find and new tabs
- mod_translation: handle concatenated lists
- mod_admin: generate translation strings, update NL
- mod_acl_simple_roles: typo
- mod_admin: add dependency mod_mqtt
- doc: improve docs on CORS
- mod_editor_tinymce: remove codemirror init code

- `mod_admin_merge`: grammar and text tweaks

David de Boer (2):

- Add modules model docs
- Fix query export

Maas-Maarten Zeeman (6):

- `mod_mqtt`: Added a way to pass js arguments for mqtt wires.
- `mod_base`: Use beacon api to get good page close feedback for chrome/firefox and recent android users
- `mod_mqtt`: Added retained messages for local (~pagesession) topics
- `mod_component`: Updated mithril to version 0.0.2
- `mod_mqtt`: Just return the topic if it is not a site topic
- `core`: Make sure looking up translations always succeed. Fixes #1103

Marc Worrell (57):

- `mod_acl_user_groups`: fix acl check for module use rights. Fixes #1071
- Merge pull request #1064 from driebit/modules-model-doc
- Add `USE_REBAR_LOCKED` file to use the locked config
- Fix `erlware_commons` version, as the `erlware_commons` master requires `rebar3`
- Copy Travis config from master
- Lock version of new mochiweb and qdate
- `mod_admin_predicate`: add overview pages of edges, optionally filter by predicate. Issue #1075
- `core`: tuning the full text search. Add option to exclude certain predicates from adding the object's title to the pivot text. Set different full text ranking weights and behaviour
- `mod_admin_predicate`: rename 'connection' to 'page connection'
- `mod_admin`: add the 'all connections' button to the dashboard
- `mod_search`: fix for `rank_behaviour` arg in simple fullext query.
- `mod_base`: add convenience arg to lazy scomp. If 'template' argument is defined then all arguments are wrapped in an update action.
- `core`: add `controller_http_error` for handling HTTP errors in other controllers.
- Locked new webzmachine (needed for the error controller)
- `mod_survey`: add option to send answers to the respondent.
- `mod_survey`: change some texts to make options clearer.
- `filewatcher`: use `os:find_executable/1` instead of `os:cmd("which ..")` to find command line executables. Fixes #1078
- `core`: check for the presence of 'identify', 'convert' and 'exif' before execution. This gives better error messages. Issue #1078
- `core`: fix 'undefined' warning if `rsc` is inserted with `content_group_id` set to 'undefined'.
- `mod_admin_identity`: let all people with `use.mod_admin_identity` manage usernames and passwords. Fixes #1077
- `mod_acl_user_groups`: refine 'block' semantics, now blocks for the mentioned user group.
- Undo changes by commit `c10055bdd581877d9df113407d8083877a1bc6b6`

- erlydtl: change generated (image tag) code to never echo errors inline in the template. Instead call `lager` or `error_logger`.
- core: fix tag/viewer generation (typo)
- core: fix a problem in `z_email_server` with mailing attachments by rsc id.
- `mod_base_site`: add `z.clickable.js`
- `mod_base_site`: add email to the share page dialog.
- `mod_admin`: change 'replace media item' text to 'add media item' if no medium is present. Thanks to @fredpook.
- `mod_admin`: use live templates for the connection lists.
- core: fix order of subcategories returned by `is_a`
- core: cleanup text to pivot and search. Remove '-', '/', and tags. Unescape the remaining text.
- smtp: mark emails from the mailer-daemon as automatic emails.
- Merge changes by @CyBeRoni issue #1082
- `mod_search`: allow to check on 'is null' and 'is not null' using '=<>' and 'undefined'
- New `z_stdlib`.
- Lock `parse_trans` to newer version.
- Move to `uwiger/jobs` instead of `esl/jobs`
- New `z_stdlib`
- `mod_admin`: set 'inputmode' attributes. Remove 'type=url' to prevent Chrome from checking the input. Issue #1093
- `mod_admin`: allow changing edges on the 'linkable' permission instead of 'editable'. Fixes #1098
- `mod_base`: fix some performance regressions in `controller_api` and `z_service`. Add simple Techempower json benchmark.
- New `webzmachine` - fixes logging of 500 errors.
- `mod_backup`: redo user interface of revision history
- Merge pull request #1088 from driebit/fix-export-query
- core: add api to return edge properties. Add 'no_touch' option to `m_rsc_update`, and option to set `creator_id` and created on edge. Fixes #1087
- `mod_acl_user_groups`: add a 'published' check to SQL searches for non-admin users. Small cleanup in `z_search`. Fixes #1081
- core: remove sql errors when enabling modules in site (re)start.
- core: log errors returned from collecting the custom pivots.
- core: add `z_pivot_rsc:pivot_delay/1`, makes it possible to delay pivoting when performing many resource updates.
- core: trim pivot fields like name and city.
- core: new functionality to merge two resources. Adds module `mod_admin_merge` and `#rsc_merge{}` notification.
- docs: new placeholders for `mod_admin_merge` and some new controllers.
- core: fix `rsc_gone` lookup for new location
- core: when merging, also merge properties (todo: map languages). Fix a problem in the admin new-rsc dialog when the category selection is set to '*'
- `mod_admin(_merge)`: prevent form submit when pressing enter in the connect or merge search fields.

- `mod_content_groups/mod_admin_category/mod_acl_user_groups`: Delay pivoting whilst performing updates of many resources.
- Prepare for 0.13.6 release, preliminary release notes.

Marco Wessel (5):

- Allow properly starting up in the foreground
- Script should be executable of course.
- Reinstate heart
- Reverse logic
- Be correct about quoting variables

loetie (1):

- Typo inside `is_allowed` check `mod_import_csv.erl`

Release 0.13.7

Welcome Zotonic 0.13.7, released on 1 February, 2016.

This is a maintenance release of Zotonic 0.13.

Main changes are:

- This release is the first in the new [monthly release schedule](#).
- From now on, release tag names no longer have the `release-` prefix.
- Combine `if` and `with` tags in [one expression](#).
- Improved performance through dispatch compiler.
- New search options: `unfinished`, `unfinished_or_nodate`.
- New image option `lossless=auto` that prevents images from being converted to JPG.

Commits since 0.13.6

There were 60 commits since release 0.13.6.

Big thanks to all people contributing to Zotonic!

Git shortlog**Arjan Scherpenisse (2):**

- `z_media_preview`: Add `lossless=auto` option to keep images as PNG after resizing
- `z_media_preview`: Remove debug statement

Arthur Clemens (4):

- `mod_admin_merge`: improve error feedback
- `mod_admin_merge`: show error message when trying to delete the admin page
- `mod_admin_merge`: more typo fixes
- `mod_admin_merge`: prepare interface for “merge only”

David de Boer (6):

- Fix category preselection

- Fix template not found error
- Fix e-mail address
- Add cookbook entry for new user's group
- Merge pull request #1153 from DorienD/formerror
- Prepare release 0.13.7

Dirk Geurs (1):

- mod_base: Issue #1084: Force model queries into values more often

Dorien (5):

- [survey] added bootstrap 3 class support, added styles for narrative question
- Added config key for content group when user registers
- Added documentation for mod_signup content group config
- changed default content group to undefined
- removed has-error class from form

Maas-Maarten Zeeman (1):

- core: Removed usage of md5_mac and sha_mac.

Marc Worrell (39):

- Merge pull request #1108 from DorienD/fix-bootstrap-survey
- Merge pull request #1105 from driebit/fix-category-preselection
- mod_video: fix preview generation.
- docs: fix generation for modules stubs. Fixes #1091
- mod_base: fix reference for (in)validFormClass. Issue #1107
- mod_search: add option to define a filter term with 'or' query
- Merge pull request #1112 from driebit/fix-template-not-found
- mod_base: fix ubf.encode typo. Thanks to Werner Buchert
- core: fix crash in rsc duplicate.
- mod_admin: show unique name in overview, slight cleanup of overview loop code.
- core: fix a problem where a revert sql query could have twice the creator_id.
- core: fix a problem where old file contents would be served for gzip-encodings.
- mod_admin: fix category pre-selection for insert and connect dialog.
- core: better fallback for unexpected file 'identify' results.
- erlydtl: add optional 'as var' to the if/elif expressions. Fixes #1085
- docs: add 'if-with' documentation. Issue #1085
- erlydtl: add missing to_list/1 function from master.
- erlydtl: merge .yrl file from master.
- core: use dispatch compiler instead of dispatch server. Minor adaptations from the version in master. Issue #1121
- core: better error message on depickle problems.
- core/mod_development: fixes for dispatch-rewrites and dispatch tracing.
- mod_development: also accept path-tokens that are strings.

- `mod_search`: add 'edge.created (and friends) sort order option
- `mod_search`: if ordering by edge seq then add sub-order by id, so that the order is similar to the one used by the edge model
- `m_search`: log errors on invalid queries, don't crash.
- `m_search`: fix empty search result for erroring queries.
- `core`: fix a problem where the `sign_key` for `zotonic_status` could change between requests.
- `mod_search`: make the ranking behaviour configurable.
- `mod_acl_user_groups`: relax adding users to a user group. Fixes #1125
- `mod_acl_user_groups`: better check for 'hasusergroup' edge insert permission.
- `mod_search`: add 'unfinished' and 'unfinished_or_nodate' query terms.
- `mod_query`: let 'unfinished_or_nodate' check the `date_start` for `nodate`, as often the end date is not set.
- `mod_admin_predicate`: move predicates in batches of 100, take care of possible duplicate edges.
- `mod_translation`: set the session language if the language is changed. Fixes #1155
- `mod_survey`: replace some old bs2 size classes with bs3 classes.
- `core`: fix for `z_email_receive:get_host/1`
- `core`: fix return value for `z_sites_dispatcher:get_host_for_domain/1`
- `mod_admin_identity`: make the error-target element optional for identity-add events.
- `docs`: fix header for sending email

Paul Guyot (1):

- Fix issue with `ffprobe` returning unicode text

Witeman Zheng (1):

- add a encoder fun for the external mqtt client

Release 0.13.8

Welcome to Zotonic 0.13.8, released on 29 February, 2016.

This is a hotfix release of Zotonic 0.13 in which a leap year bug was fixed:

- [Fixed bug with leap year.](#)

Git shortlog**Lil' Dash (1):**

- Fix the leap year error(issue : #1202)

Release 0.14.0

Welcome to Zotonic 0.14.0, released on 7 March, 2016.

Main changes are:

- From now on, we follow Semantic Versioning by increasing [the minor instead of the patch number](#) for our monthly releases.
- Added [data model notification](#).
- Added [managed ACL rules](#).

- Added `zotonic wait` and `zotonic rpc` commands.
- Added new user category `configuration` parameter.
- Fixed redirect to login when user has no permissions by showing 403 page with login form.
- Fixed several import and export bugs.
- Fixed Dutch translations for `mod_acl_user_groups`.
- Fixed status page login button.
- Improved documentation.
- Improved m.req model.

Commits since 0.13.8

There were 65 commits since release 0.13.8.

Big thanks to all people contributing to Zotonic!

Git shortlog

David de Boer (17):

- `mod_import_csv`: Fix CSV import for CR files
- Add release preparation script
- Restructure documentation into guides and reference
- Add data model notification
- Increase resource URI length
- Move 0.13.7 release notes to new dir
- `doc`: Document how to build docs
- `mod_admin_identity`: Add config param for new user category
- Support managed ACL rules
- `mod_acl_user_groups`: Improve wording and add missing Dutch translations
- Prepare hotfix release 0.13.8
- `doc`: Improve access control chapter
- Move release notes to proper location
- Prepare release 0.14.0
- `doc`: Fix link
- `doc`: fix notification reference
- Add 0.14.0 release notes

Lil' Dash (1):

- Fix the leap year error(issue : #1202)

Maas-Maarten Zeeman (15):

- `core`: Added reference to active comet connection. Preparation for connection test page
- `mod_base`: Added connection test page
- `core`: Return 0 bytes when there is nothing to transport. Related to #1097

- core: Typo in refactored controller_comet
- mod_base: removed debug statement from comet controller
- admin: Make it possible to brand the admin console
- mod_base: Removed delegate macro, it was tripping edoc generation
- core: Fixes setting and resetting ws transport timeouts. Related to #1116
- mod_base: Let postback and comet return text/x-ubf mimetype.
- build: Make it possible to separate user beams
- build: Add user_ebin_dir to zotonic.config.in
- build: Fix for the case when there is no user work
- doc: Repair underline warning.
- mod_mqtt: Restart middleman subscriber processes. Fixes #1201
- mod_ssl: Create and use a file with dh parameters. Fixes #1198

Marc Worrell (31):

- mod_export: encode YMD only date as UTC, add fallback for other dates to handle illegal dates.
- mod_export: suppress ST_JUTTEMIS and year 9999 dates in the output.
- Lock new depcache. Fixes a problem where the wrong depcache could be used for lookups. Fixes #1172
- core: refactor transport of multipart posts and comet polls.
- core: fix specs of refactored z_parse_multipart functions.
- mod_base: only initialize callback forms after javascript delegate calls, not after other delegates. Issue #1159
- core: allow 'none' as crop argument.
- core: allow flexible 'visible_for' settings.
- docs: add doc for m.category.text.tree_flat
- core: use the 'image' dispatch rule to generate image urls.
- core: add more information and better docs for the m.req model.
- core: (m_rsc) allow template access to page_url and other 'public' properties of non accessible resources.
- core: show 403 page with option to login instead of redirecting to login controller. Fixes #1148
- core: fix 'provide_empty' in http error controller.
- core: fix erlydtl debug for binary filenames.
- mod_twitter: drop tweet id 0
- mod_translation: add interface to change the i18n.language_stemmer configuration
- mod_oembed/mod_video_embed: add the medium properties video_embed_service and video_embed_id. Fixes #941
- mod_base: added actions overlay_open/overlay_close
- mod_backup: fix acl check for revisions
- core: fix redirect returns within dispatcher for alternate hosts.
- mod_acl_user_groups: fix a problem with edge checks for hasusergroup edges. Fixes #1199
- mod_authentication: fix ru translation for 'Sign in' Fixes #1197

- core: allow '{lossless, false}' in mediaclass definitions.
- core: fixes for initialization and startup of new sites.
- mod_acl_user_groups: fix schema initialization. Fixes #1200
- core: fix for check of db schema exists.
- core: better check query for schema existence.
- core: filter the watch_dirs on dead links, this fixes a problem with inotify stopping on removed link targets
- core: fix a problem where the comet loop can crash if the page process dies.
- mod_survey: do not crash on unknown survey question types when preparing charts.

Marco Wessel (1):

- scripts: Add zotonic-rpc and zotonic-wait commands

Release 0.15.0

Welcome to Zotonic 0.15.0, released on 4 April, 2016.

Main changes are:

- Added `is_number` template filter.
- Added `persistent_set` template action. resource.
- Added `mod_media_exif`.
- Improved `mod_export`.
- Upgraded TinyMCE to 4.3.7.
- Fixed #1216 by changing `lossless` media option to `false` (instead of `auto`).
- Fixed #1148 by adding 403 page for logged in users that have no access to the current
- Fixed #1205: DPI for resized images.
- Fixed #1224: limit access to `mod_rest` and `mod_export` to users that have `use` permission.
- Fixed #1221: order of Growl messages.

Commits since 0.14.0

There were 60 commits since release 0.14.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

Arthur Clemens (3):

- mod_admin: remove extraneous closing div
- mod_admin: move footer to its own row
- mod_admin: add a bit of space to the footer

David de Boer (4):

- mod_admin: Fix link to search docs (#1220)
- doc: Fix ACL rule fixture syntax

- scripts: Fix comment
- Prepare 0.15.0

Marc Worrell (53):

- core: fix a problem where the comet loop can crash if the page process dies.
- mod_base: add 'is_number' filter.
- mod_backup: move the list of known resource properties to m_rsc.
- mod_export: refactor export, separate encoder functions. Added xlsx export encoder.
- mod_survey: remove controller_survey_results, mod_export is now used.
- docs: remove controller_survey_result docs.
- docs: add documentation for filter is_number
- mod_export: add vcalendar export (ics)
- mod_export: connect mod_export to the 303 handling of controller_id
- mod_export: remove #export_resource_data observe, this is already handled in export_encoder:do_body/2
- mod_base: new action 'persistent_set'
- docs: fix inline quote error.
- docs: add persistent_set to the doc tree
- core: also look into ~/.zotonic/<major-version>/.. directory for configuration.
- mod_export: filter tags and unescape html with spreadsheet exports. Add 'raw' option to not filter/unescape.
- mod_export: fix double reverse of xlsx rows
- mod_survey: also show the prompts in the answers exports.
- core: use Erlang 'exif' module and add mod_media_exif to extract resource properties (gps, orientation, crop, date)
- Run travis on 0.x
- docs: add simple documenttaion for mod_media_exif
- core: add mod_media_exif to the core modules.
- core: fix for focus point calculation.
- tinymce: add version 4.3.7. Add option to add captions to the body media. Also: * Add a template for the image options dialog * Move some named wires to _editor.tpl to prevent multiple initialization for every single editable body.
- mod_editor_tinymce: remove version 4.0.26 and 4.1.6
- mod_editor_tinymce: remove the deleted tinymce versions from the configure dialog
- mod_editor_tinymce: include the correct css version
- mod_export: simplified download buttons for the admin sidebar
- mod_export: add explanation for event download
- mod_authentication: add 403 page with logon form (or redirect button for secure page)
- Switch to nlfiedler/erlang-exif.git instead of our own branch
- Switch to nlfiedler/erlang-exif.git instead of our own branch
- mod_base: remove comet streamhost from zotonic js
- core: remove mentions of streamhost (which is unsupported)

- New mochiweb
- mod_filestore: remove GreenQloud - they transferred their business to another company.
- core: start using psql 'IN (SELECT(unnest(::int[])))' instead of concatenated id strings.
- mod_twitter: fix a problem where httpc sessions were not closed.
- New twerl library
- mod_base: restart ws/comet if navigating back to the page in iOS/Safari
- mod_base: tune stream restart on pageshow of persisted pages.
- mod_base: better session check on pageshow. Still a problem if tinymce is enabled and the page is re-visited for the 2nd time (twice back to the page).
- mod_mailinglist: add some useful shortcuts to the edit sidebar panel
- New tw* erl dep
- core: ensure that resized images have a density of 72DPI. Fixes #1205
- Fix media preview test for dpi forcing
- mod_base: show newer growl messages on top. Fixes #1221
- core: change media preview option 'lossless' default to 'false' (instead of 'auto'). Fixes #1216
- Fix a problem with filtering on content-group in searches.
- mod_acl_user_groups: fix a problem where the ACL tree expand could not find some entries.
- mod_acl_user_groups: fix problem adding new rules. Stabilize the order of rules by including the rule creation date and id into the sort order Split system content groups in pull-down, to clarify that 'all' doesn't apply to the system content groups.
- mod_export: limit exports to users with mod_export.use permission. Refactor export api, simple privacy filter for email address. Issue #1224
- mod_rest: add acl check for mod_rest.use. Issue #1224
- mod_logging: fix a problem with filtering on content-id and other-id.

Release 0.16.0

Welcome to Zotonic 0.16.0, released on 2 May, 2016.

Main changes are:

- Fixed #1099: added collaboration groups.
- #1227: Added [ip_whitelist configuration option](#) to block unwanted access when admin password is 'admin'.
- Serve status site with 503.
- Fixed #1229: users cannot update their person resource.
- Fixed #1236 by removing is_authoritative access check.
- Fixed #1245 by improving the 403 page.
- Fixed #1147 by passing `qargs` to search.
- Fixed #1230: Firefox file upload error.
- Fixed #1248: Dutch month abbreviation.
- Fixed #1250: view button in case of no update permissions.

Commits since 0.15.0

There were 39 commits since release 0.15.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

Arjan Scherpenisse (4):

- zotonic_status: Serve the root page with HTTP 503
- mod_acl_user_groups: Fix permission issue
- mod_acl_user_groups: Fix typo in edit check on the collab group itself
- scripts: Fix prepare-release.sh

David de Boer (5):

- mod_acl_user_groups: Automatically publish managed ACL rules
- doc: Add Sphinx extlinks extension
- mod_110n: Fix #1248 Dutch month abbreviation
- doc: Update for new lossless default (6e5692d0cab0eb5a125a05a3ab02933a0e6829e4)
- Prepare release 0.16.0

Marc Worrell (29):

- Add a whitelist of IP addresses that can use the default ‘admin’ password to login into sites.
- Add doc for ip_whitelist config. Fix ip6 netmask
- mod_search: add search argument ‘qargs’
- mod_admin: add ‘object_id’ as optional argument to the connect dialog. Remove template from the ‘update’ notification, as it is ‘_rsc_item.tpl’ anyway.
- modules: correction for some icons and small textual changes.
- mod_admin_predicate/category: fix a problem with moving categories and predicates.
- Add 169.254.0.0/16 and fe80::/10 to the default ip_whitelist
- core: add z_module_manager:activate_await/2, z_notifier:await/2 and /3.
- core: rename z_module_manager:activate_wait to activate_await
- mod_base: fix a problem with Firefox when uploading files via the postback controller. Fixes #1230
- mod_base: fix a problem with firefox and sortable. Fixes #1239
- mod_acl_user_groups: added support for collaboration groups. Issue #1099
- Fix a problem where an user can’t update their own ‘person’ resource. Issue #1229
- mod_acl_user_groups: fix a problem where ‘sudo’ was not handled properly during rsc updates. Issue #1229
- mod_acl_user_groups: remove default ‘view’ rights on collaboration group.
- core: do not require additional permissions to change the is_authoritative flag. Fixes #1236
- docs: remove mention of streamhost, as it is removed from the code. Fixes #1228
- mod_survey: changes for editing surveys and readable display of survey answers.
- mod_export: fixes for export.
- mod_export: fix header value lookup for {trans, _} tuples.

- `mod_base`: in filter temporary_rsc, don't crash on insert access error but return 'undefined'
- `mod_admin`: on edit page, make 'view' button into an anchor if no edit rights. Fixes #1250
- `mod_acl_user_groups`: do not give view rights on unpublished rsc, unless user has edit permission.
- `mod_authentication/mod_base`: better 403 page and translations. Fixes #1245
- `mod_acl_user_groups`: 'edit' should be 'update', use `m_rsc` version of `is_published_date`
- `core`: ensure in `m_rsc` that 'language' is a list of known languages.
- `mod_signup`: use `foldr` for `signup_form_fields`, let higher prio modules win.
- `mod_acl_user_groups`: tune access permissions for collaboration groups. All collab group members can view the collab group. If someone can update/link/delete a collab group, then that user can do the same on the collab group content. Rename the config `collab_group_edit` to `collab_group_update`.
- `mod_acl_user_groups`: members of a collaboration group can view each other.

Osei Poku (1):

- doc: Fix minor errors in documentation

Release 0.17.0

Welcome to Zotonic 0.17.0, released on 6 June, 2016.

Main changes are:

- Added #1274: SNI support on Erlang 18.3 and higher.
- Added #1284: default ACL rules.
- Added #1240, #1276 and #1283: documentation for ACL user groups, task queue and Google Analytics.
- Added #1265 and #1268: sanitise SVG uploads and link tags.
- Fixed #1285: be less verbose when inserting ACL rules.
- Fixed #1272: Vimeo embeds.
- Fixed #1271: protected media security.
- Fixed #1207 by giving `m_rsc.uri` property precedence over generated URI.
- Fixed #1132 by setting `Content-Security-Policy` header.

Commits since 0.16.0

There were 54 commits since release 0.16.0.

Big thanks to all people contributing to Zotonic!

Git shortlog**Arjan Scherpenisse (11):**

- scripts: Fix `prepare-release.sh`
- doc: Fix version nr; add missing rst files; update installation requirements
- doc: Update sidebar, update installation instructions
- admin: Show collaboration groups in content dropdown on overview
- `mod_admin`: Show content groups + page size filters on media overview
- `zotonic_status`: Show site name next to URL

- `mod_acl_user_groups`: Fix typo in edit check on the collab group itself
- `mod_email_dkim`: Add DKIM signing of outgoing emails
- `mod_base`: Add 'without' filter
- `doc`: Add `filter_without` to filter toc
- `mod_acl_user_groups`: Fix permission check for adding members/managers to group

Arthur Clemens (1):

- `core`: use short notation to include the header

David de Boer (14):

- `deps`: Lock `erlang_localtime`
- `doc`: Fix absolute/relative URL terminology
- `mod_admin_identity`: Fix verification e-mail URL
- `doc`: Update release branch name
- Add 0.16.0 release notes
- Clean up README and fix dead links
- `doc`: Document Google Analytics
- `doc`: Document media caption
- `base`: Fix `figcaption` tag
- `mod_acl_user_groups`: Insert default ACL rules (see #1131)
- `doc`: Document the task queue
- `mod_acl_user_groups`: Be less verbose when editing and publishing ACL rules
- `doc`: Document `mod_acl_user_groups`
- `doc`: Fix typos

Maas-Maarten Zeeman (2):

- `build`: Locked new mochiweb in order to support SSL on IE9 and 10 on OTP 18+
- `core`: Move ssl listeners to the core and support SNI.

Marc Worrell (26):

- `mod_signup`: use `foldr` for `signup_form_fields`, let higher prio modules win.
- `deps`: switch to original `erlang_localtime` from `dmitryme`. Issue #1036
- `mod_acl_user_groups`: tune access permissions for collaboration groups. All collab group members can view the collab group. If someone can update/link/delete a collab group, then that user can do the same on the collab group content. Rename the config `collab_group_edit` to `collab_group_update`.
- `mod_acl_user_groups`: members of a collaboration group can view each other.
- `core`: add support for '-' operator, extend support for '++' operator.
- `core`: add sanitization on the contents of uploaded SVG files. Issue #1265
- Lock new `dispatch_compiler`. Fixes #1261
- `core`: in `m_rsc`, always let the `uri` property take presendence above the local 'id' rule for non-informatonal uri generation. Fixes #1207
- `core`: fix a problem where a file could be downloaded iff the file is not stored via a filestore. Fixes #1271
- `mod_video_embed`: fix a problem where Vimeo embeds did not show a preview images. Fixes #1272

- `mod_admin`: in depiction upload dialog, enable the ‘upload’ tab by default.
- `core`: in html sanitize, add ‘noopener noreferrer’ to `<a/>` tags with a ‘target’ attribute.
- `mod_video_embed`: better handling of 404 when fetchin Vimeo thumbnail.
- `core`: added extra SVG sanitization.
- Lock new `z_stdlib` for SVG sanitization. Issue #1265
- `mod_survey`: also mail all uploaded files to the ‘survey_email’ email address.
- `mod_admin`: set X-Frame-Options: SAMEORIGIN header for admin pages. Issue #1132
- `mod_admin`: fix `is_authorized/2` in `controller_admin_edit`
- `mod_base`: set CSP sandbox header if an user uploaded file is served with `controller_file`. Issue #1132
- `mod_acl_user_group`: add option to move resources to collaboration groups the user is not member of.
- `mod_admin`: allow to select multiple connection in the connection-find dialog. Use the option ‘auto-close’ to change the text of the close button to ‘cancel’.
- `mod_acl_user_groups`: better overview of rules. Use dialog for editing rules.
- `mod_acl_user_groups`: in acl rule edit, clear collaboration group search when group is selected.
- `mod_acl_user_groups`: fix filtering on content groups when searching. This fixes a problem when an user is allowed to see all or specific collaboration groups via the ACL rules.
- `mod_survey`: fix layout of admin options.
- `mod_survey`: in the emails, also show any ‘injected’ fields. This allows the template to dynamically inject some answers without corresponding questions.*

Release 0.18.0

Welcome to Zotonic 0.18.0, released on 4 July, 2016.

Main changes are:

- Removed module `mod_acl_adminonly` from core (#1289). Please use `mod_acl_user_groups` (page 341) instead. If you need `mod_acl_adminonly`, the module is available in a [separate repository](#).
- Added compatibility with OPT 19 (#1323).
- Added page blocks to the full text search index (issue:1131).
- Added support for Google Universal Analytics (#1281).
- Added `testsandboxdb` site for integration tests against a database (#1235).
- Fixed multiple images captions (#1311).
- Fixed hierarchy tree not updating after edit (#1293 and #1309).
- Fixed inserted video in TinyMCE not shown (#1304).
- Fixed anonymous user editing.

Commits since 0.17.0

There were 33 commits since release 0.17.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

Arthur Clemens (4): mod_mailinglist: fix unsubscribe url syntax mod_admin: add param cat_exclude to overview mod_admin_identity: use same column headers as overview mod_admin: tweak Dutch translation

David de Boer (3): tests: Add testsandboxdb site core: Fix inotify statup log message tests: Make build fail if Zotonic fails to start

Maas-Maarten Zeeman (2): mod_ssl: Redirect http and https to the outside ports mod_base: Use utf-8 encoding for application/javascript files

Marc Worrell (22): core: fix z_module_manager:activate_await/2. Also fix a problem where upgrading a module could activate that module. Issue 1235 core: only start testsandboxdb during tests. mod_base: more generic escapejs filter. Issue #1281 mod_admin: fix is_session_alive status, thanks to @Dirklectisch core: let the ACL decide if anonymous users can edit something. Thanks to @Dirklectisch Locked new dep-cache. mod_editor_tinymce: fix a problem where setting the options of one media item changed the options of all media item. Fixes #1311 mod_video_embed: fix an issue where inserting a video inside tinymce doesn't show the video. Fixes #1304 mod_admin: prevent iframe overflowing from media preview on edit page. mod_admin: fix an issue where the menu editor was not updated after creating a new resource. Fixes #1309 Fixes #1293 mod_admin: in connect dialog, fix enabling upload tab without depiction predicate. core: also pivot texts in blocks. Fixes #1113 core: only install the skeleton modules on initial site install. Fixes #1279 core: remove debug from pivot docs: remove 'resource' from the contactform cookbook. Fixes #1166 core: fix return value of z_install:install_skeleton_modules/1 core: pivot block texts to priority B not A. mod_base_site: misc fixes for base site. mod_survey: set the language of result emails to the default system language. Fixes #1324 core: in activate_await, check the module process pid, and not the db. Fixes #1325 core: fix whereis check in activate_await. Issue #1325 core: OTP-19 compatibility changes for 0.x (merging from master). Issue #1323 (#1327)

Péter Gömöri (1): Fix xref issues (#1315)

Tah Teche (1): mod_seo: Switch to Google Universal Analytics

Release 0.19.0

Welcome to Zotonic 0.19.0, released on 1 August, 2016.

Main changes are:

- Removed module mod_acl_simple_roles from core (#1328). Please use [mod_acl_user_groups](#) (page 341) instead. If you need mod_acl_simple_roles, the module is available in a [separate repository](#).
- Added [site tests](#) (#1331).
- Added View dropdown menu in admin (#1345).
- Added possibility to return [JSON error objects](#) from API service controllers.
- Added `m.req.is_bot` (#1340).

Commits since 0.18.0

There were 22 commits since release 0.18.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

Arjan Scherpenisse (9):

- Implement running site-specific tests (#1332)

- core: Add `z_sitetest:{watch,unwatch}` to speed up test driven development
- API: Allow `process_{post,get}` to `throw()` error for shortcircuiting
- core: Only run sitetests when compilation succeeds
- `mod_acl_user_groups`: Documentation fixes; change 'edit' -> 'update'
- API: Fix logic bug in try/catch clause handling for `process_{get,post}`
- doc: Add note that testname cannot containing underscores
- API: Log crashes, serve HTTP 500 on uncaught exception
- `filewatcher`: reload module first when running all sitetests for a site

Arthur Clemens (3):

- docs: Describe error handling from API service
- Fix `include_lib`
- `mod_base: controller_api`: add option to pass JSON error object

David de Boer (4):

- core: Move `mod_acl_simple_roles` out of core into separate repo (#1328)
- doc: Add `.pot` generation function call
- admin: Add view dropdown menu (#1345)
- doc: Fix build

Marc Worrell (6):

- core: add forward-compatible `z_utils:name_for_site/2`. Issue #1333
- `mod_admin`: add possibility to disconnect connections via the `connection-dialog` Fixes #1339
- core: add 'is_bot' property for `m.req` Fixes #1340
- core: use the `ua_classifier` supplied `is_crawler` flag. Fallback to the hardcoded list for unknown user agents (example: curl). Fixes #1340
- core: in `z_user_agent` allow `WebSocket` and `websocket`.
- `z_user_agent`: make the upgrade comparison case insensitive

Release 0.20.0

Welcome to Zotonic 0.20.0, released on 5 September, 2016.

Main changes are:

- Added an MQTT notification when identity is deleted (#1376).
- Fixed menu separators in the admin menu (#1388).
- Fixed some tests not being run (#1392).
- Fixed memory leak in Comet en `WebSocket` connection handling.
- Renamed 'App Keys and Authentication Services' admin menu item to 'External services' (issue:1389).
- In the admin, moved CSV import to the Content admin menu and require only use `mod_import_csv` permissions (#1350).
- Removed Perl dependency (#1373).
- In many places, the resource's unique name and id can now be used interchangeably (#1356).
- Several documentation improvements, including an example configuration for SSL termination using Nginx (#1359).

Commits since 0.19.0

There were 36 commits since release 0.19.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

Arjan Scherpenisse (3):

- mod_acl_user_groups: Fix typo in ACL import
- core: Allow retrieving site config values from the OS environment
- mod_base: Log the stacktrace when an API method throws an error

Arthur Clemens (6):

- docs: Describe uploading files with API services
- dialog: reposition one tick after initialization
- dialog: make repositioning function accessible outside
- dialog: fix dialogs that do not have option 'center'
- dialog: add function to scroll to position
- Add style to LESS instead of CSS

David de Boer (20):

- doc: Fix build
- docker: Create lite Zotonic container (#1310)
- Add 0.19.0 release notes (#1358)
- doc: Fix typo
- doc: Document page template names (#1364)
- admin: Fix inline links (#1366)
- Add .editorconfig
- Remote trailing whitespaces from .erl and .tpl files
- core: Add MQTT notification for delete_by_type_and_key (#1376)
- mod_base: Fix Content-Security-Policy for inline PDFs (#1379)
- doc: Fix indent
- doc: Improve search docs (#1369)
- doc: Add a note that only unprotected Tweets will be imported (#1387)
- mod_import_csv: Fix untranslated cancel button
- core: For every resource id argument, allow resource name as well (#1356)
- mod_authentication: Rename menu to 'External Services' (#1389)
- mod_import_csv: Move CSV import to Content menu (#1391)
- mod_admin: Fix menu separators not shown (#1388)
- tests: Include db_tests in testrun (#1393)
- core: Remove rid from maybe_allowed (#1400)

Maas-Maarten Zeeman (1):

- core: Fix memory leak in comet and ws connection handling.

Marc Worrell (5):

- mod_menu: fix a problem where the conversion of old config-menu failed due to missing main_menu resource.
- core: allow m_rsc p_no_acl fetch of exists and id
- mod_survey: fix fetch of label names for truefalse questions.
- smtp: ensure that the logged error reason is a binary. This fixes a situation where the value {error, {error, timeout}} was returned from gen_smtp_client:send_blocking/2.
- smtp: ensure that the logged error reason is a binary. This fixes a situation where the value {error, {error, timeout}} was returned from gen_smtp_client:send_blocking/2.

Marco Wessel (1):

- Remove perl dependency (#1373)

Release 0.21.0

Welcome to Zotonic 0.21.0, released on 3 October, 2016.

Main changes are:

- Added an `acl_is_owner` notification (#1404).
- Added a Docker image for Zotonic development (#1425).
- Improved Docker images size by switching to Alpine (#1374).
- Improved password hashing by switching to bcrypt (#1390).
- Improved `is_visible` filter to support fulltext search results.
- Moved documentation to Read the Docs (#1454).
- Fixed removed observers not being detached (#1441).

Commits since 0.20.0

There were 31 commits since release 0.20.0.

Big thanks to all people contributing to Zotonic!

Git shortlog**Arjan Scherpenisse (1):**

- core: Add default config for 'setup' application

David de Boer (12):

- core: Remove rid from maybe_allowed (#1400)
- tests: Fix m_identity_tests on 0.x (#1415)
- Add 0.20.0 release notes (#1423)
- mod_admin: Fix spelling error
- doc: Add deeplink to custompivot
- doc: Fix ref
- docker: Negate paths in .dockerignore to make image even smaller (#1426)

- docker: Fix compilation by switching bcrypt dep (#1434)
- doc: Improve site creation (fix #1440)
- docker: Add dev Docker image (#1425)
- docker: Fix duplicate config files (#1442)
- doc: Build 0.x docs on Read the Docs (#1454)

Maas-Maarten Zeeman (1):

- core: Use bcrypt and erlpass for password hashes (#1390)

Marc Worrell (16):

- mod_acl_user_groups: make the is_owner check a notification (#1404)
- mod_acl_user_groups: fix is_owner check.
- mod_admin: pass args to connect dialog items. Add 'thumbnail-linkable' class.
- mod_admin: in connect-dialog search result, add the class 'unpublished' for unpublished items.
- mod_admin: filter connect-list on visibility
- mod_base: let is_visible filter also accept a list of tuples {Id,Score} as returned by the fulltext search
- core: add smtp headers from the #email message.
- mod_video: fix replacement of temp re-render placeholder.
- i18n: manual merge of #1437
- mod_oembed: don't embed data for 'link' type (our metadata sniffer is better at this)
- mod_email_status: export clear_status/2
- core: fix eaccess/eaccess mixup
- core: fix a problem where removing an exported observe function did not detach the observer. Fixes #1441
- core: less restrictive receive for module shutdown.
- core: skip identify errors when extracting mime format.
- mod_export: catch errors in xlsx export of illegal dates

Marco Wessel (1):

- docker: Use Alpine Linux for the Docker images (#1374)

Release 0.22.0

Welcome to Zotonic 0.22.0, released on 7 November, 2016.

Main changes are:

- Added support for managed collaboration rules (#1492).
- Added deletion interval for files stored on S3 (#1493).
- Fixed database connection recovery after PostgreSQL restart (#465).
- Fixed '53300 too many connections' PostgreSQL error (#1469).
- Fixed 'Add connection' button invisible for link permission (#1476).
- Fixed S3 delete requests being indefinitely repeated (#1488).

Commits since 0.21.0

There were 16 commits since release 0.21.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

David de Boer (8):

- doc: Add 0.21.0 release notes (#1461)
- docker: Fix log_dir sed selector
- core: Fix bcrypt dep on 0.x (#1465)
- mod_acl_user_groups: Add managed collab rules support (#1492)
- mod_filestore: Fix #1231 by adding ACL explanation
- mod_filestore: Add delete interval (#1493)
- mod_acl_user_groups: Fix #1505 erroneous foreign key
- Add build deps to environment variable

Dirk Geurs (1):

- mod_admin: Show add connection buttons if linkable (#1477)

Finn Kruyning (1):

- Remove TD from admin dashboard

Maas-Maarten Zeeman (2):

- mod_export: Fix, follow export_resource_visible spec (zotonic_notifications.hrl:847)
- mod_component: Added the possibility to unmount a component

Marc Worrell (4):

- core: sanitize Microsoft Office comments, classes and styles. Fixes #1464
- core: remove trim_all option from binary:split/3 as it is incompatible with OTP 17
- core: remove transport log messages about unknown page sessions.
- Close open db connections on a 'too may connections' error. (#1470)

Release 0.23.0

Welcome to Zotonic 0.23.0, released on 5 December, 2016.

Main changes are:

- Added delete interval (and completely disabling deleting) to files stored on S3 (#1493).
- Added search rank weight configuration parameter (#1538).
- Added editing of embedded collections in admin (#1520).
- Added support for custom Google Analytics parameters (#1518).
- Fixed consistency of log messages and log metadata (#1510).
- Fixed ACL checks in admin (#1514).
- Fixed embedding not allowed for non-admins (#1545).
- Fixed initialization error in mod_acl_user_groups (#1505).

- Fixed initial tab on connect modal (#1497).

Commits since 0.22.0

There were since 22 commits since release 0.22.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

Arjan Scherpenisse (1):

- scripts: Add -noshell to sitetest command

David de Boer (14):

- mod_filestore: Add delete interval (#1493)
- mod_acl_user_groups: Fix #1505 erroneous foreign key
- Add build deps to environment variable
- Prepare release 0.22.0 (#1503)
- mod_filestore: Upgrade version to add deleted column (#1506)
- admin: Switch position of translation and user menu (fix #1354)
- admin: Fix access checks (#1514)
- mod_seo: Support custom Analytics params (#1518)
- core: Improve log messages consistency (#1510)
- docker: Switch to Alpine PostgreSQL for smaller file size
- mod_menu: Fix menu and user groups not editable (#1531)
- mod_search: Quote search weight and document options
- mod_base: Validate resource name (#1504)
- mod_filestore: Fix an issue with setting deletion to 'never' (#1549)

Fred Pook (3):

- admin: Don't restrict initial tab for zmedia connect dialog (#1497)
- mod_search - Allow configuration of rank weight
- mod_editor_tinymce: translation for middle size changed to medium size (#1539)

Marc Worrell (4):

- Close open db connections on a 'too may connections' error. (#1470)
- mod_admin: edit collection in block for embedded collections (#1520)
- core: fix a problem with sanitizing classes in html. (#1526)
- Fix a problem where a non-admin could not add embedded media. Fixes #1545

Release 0.24.0

Welcome to Zotonic 0.24.0, released on 2 January, 2017.

Main changes are:

- Fixed deletion date 'never' in mod_filestore (#1549).

- Fixed handling of illegal Exif data (#1557).
- Fixed adding embedded media (#1545).
- Fixed live validation message_after error when id is invalid (#1543).

Commits since 0.23.0

There were since 16 commits since release 0.23.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

David de Boer (4):

- core: Support custom search total (#1537)
- mod_filestore: Fix an issue with setting deletion to 'never' (#1549)
- Add 0.23.0 release notes
- core: Fix filezcache gzip compress pattern match

Marc Worrell (11):

- Fix a problem where a non-admin could not add embedded media. Fixes #1545
- Lock new exif (and bcrypt). Issue #1556
- mod_media_exif: fix a problem with illegal exif dates. Issue #1557
- mod_base_site: fix order of includes.
- mod_base: add UBF support for floats, proplists and dates.
- Lock new z_stdlib (for z_ubf)
- mod_base: fix a problem where a transport before init of z_pageid would reload the page.
- core: add z_mqtt:payload_user/1 function.
- core: fix a problem with identify of EPS files. Also catch crashes in Exif library on unknown or corrupt exif data.
- Lock new exif
- core: fix a problem with pre-page init pubsub subscriptions.

Álvaro Pagliari (1):

- mod_base: fix livevalidation message_after error when invalid id (#1543)

Release 0.25.0

Welcome to Zotonic 0.25.0, released on 6 February, 2017.

Main changes are:

- Added `embedded_media` filter (#1591).
- Added Atom support to `mod_export` (#1394).
- Added `name` search query argument (#1574).
- Fixed image crop defined in `mediaclass` (#919).
- Fixed adding connection to page block (#1561).

- Fixed text blocks not considered by `without_embedded_media` filter (#1566).
- Fixed unique name validation for empty names (#1579).
- Fixed filewatcher reacting to changes in log files (#1588).
- Fixed pivot index names not unique (#1598).

Commits since 0.24.0

There were since 33 commits since release 0.24.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

Arthur Clemens (1):

- Add clarification in template

David de Boer (6):

- doc: Add custom controllers cookbook chapter (#1565)
- doc: Fix postback validator documentation (#1567)
- docker: Build Erlang from source (fix #1590)
- core: Raise timeout when starting site for sitetest
- doc: Fix `embedded_media` docs
- core: Make pivot index name unique (#1598)

Maas-Maarten Zeeman (1):

- core: Accept push queue names as binaries. Fixes #1575 (#1576)

Marc Worrell (24):

- `mod_search`: use `publication_start` as a sub sort for matching, prefer newer content.
- core: evaluate 'and'/'or' as 'andalso' and 'orelse'. Fixes #1561
- `mod_base`: fix a problem in `show_media` where utf8 characters were wrongly displayed. Fixes #1572
- `mod_search`: add a query term 'name' for searching on unique names. Fixes #1574
- `mod_export`: add export in application/atom+xml format. Fixes #1546 Fixes #1394
- Lock new `z_stdlib`
- core: let filewatcher ignore changes to log files. Fixes #1588
- core: also ignore changes to files in the mnesia directory
- `mod_base`: in `ubf.js` use `Object.keys()` to fetch the keys of an object. This fixes an issue where injected object.prototype functions were serialized as well.
- docs: fix build problem with 'less' code block by using 'text' instead
- `mod_base`: let filter `without_embedded_media` also consider text blocks. Add filter `embedded_media`. Fixes #1566
- core: rsc names with only a '_' are considered 'undefined'. Fixed #1579
- `mod_base`: let the unique name validation first use 'z_string:to_name/1'. Consider '_' as an invalid unique name. Issue #1579
- `mod_base`: allow empty unique name.

- core: use `crop_center` if `crop` is defined in the `mediaclass`. Fixes #919
- `mod_admin`: fix a problem with media preview for unknown resources.
- `mod_base`: fix `ubf` decode for `plist/map` types.
- `mod_admin`: change positioning of images in connection lists. Fixes #1570
- core: check `crop_center` syntax on `rsc` update.
- core: add email DNS Whitelisting support
- Lock new `z_stdlib`
- core: fix a string/binary conversion issue with the new `z_stdlib`
- New `z_stdlib`
- core: fix a problem where `next/prev_day` could result in a non-existent date. Fixes #1596

row-b (1):

- Minor typo in syntax fixed (#1569)

Release 0.26.0

Welcome to Zotonic 0.26.0, released on 6 March, 2017.

Main changes are:

- Added support for special characters in `z-media` caption (#1618).
- Added `facets` property to `#search_result{ } record` (#1606).
- Fixed `zmedia` dialog does not open in Firefox (#1620).
- Fixed hidden dropdown menu in admin (#1603).
- Fixed a problem with UBF-encoded date values.
- Fixed forbidden response from S3 filestore triggering retries.
- Moved `filezcache` data and journal directories outside the dependency directory (#1601).

Commits since 0.25.0

There were 19 commits since release 0.25.0.

Big thanks to all people contributing to Zotonic!

Git shortlog**David de Boer (5):**

- core: Make pivot index name unique (#1598)
- Add 0.25.0 release notes (#1600)
- core: Add `facets` property to `search_result` (#1606)
- core: Move `filezcache` data dir outside `dep` dir (#1621)
- core: Upgrade `filezcache`

Maas-Maarten Zeeman (1):

- core: Explicitly set the timezone of the db connection to UTC

Marc Worrell (13):

- `mod_base`: in `ubf.js` fix a problem with encoding date values
- `mod_admin`: fix a problem where in tree-lists the dropdown menu was hidden. Fixes #1603
- Upgrade `z_stdlib`.
- New `z_stdlib`
- `mod_base`: add `'is_danger'` flag to the confirm dialog. This will show the `'ok'` button with the `'btn-danger'` style.
- Add support for special characters in the z-media caption. (#1618)
- New `z_stdlib`
- `core`: allow `m_edge:get_id/4` calls with an non-existing predicate.
- `mod_filestore`: handle S3 forbidden as non retryable error.
- `core`: fix utf8 problem for R16
- `mod_editor_tinymce`: fix check for crop checkbox.
- `core`: the medium size is actually called `'middle'` (for 0.x)
- `mod_editor_tinymce`: fix zmedia for Firefox

Release 0.27.0

Welcome to Zotonic 0.27.0, released on 3 April, 2017.

Main changes are:

- Added `platform.twitter.com` to whitelist (#1647).
- Added config location override based on environment variable (#1627).
- Fixed resource creation ACL check missing content group selection (#1640).

Commits since 0.26.0

There were 10 commits since release 0.26.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

David de Boer (2):

- `core`: Upgrade `filezcache`
- Add 0.26.0 release notes

Maas-Maarten Zeeman (1):

- `core`: Add properties to `acl insert` to allow checks on content/collaboration groups

Marc Worrell (5):

- `core`: fix a problem with identifying certain m4a files.
- `mod_editor_tinymce`: fix zmedia for Firefox
- `mod_import_csv`: fetch more data analyzing the column labels.
- Facebook now returns JSON during OAuth handshake.
- Upgrade `z_stdlib`

Marco Wessel (1):

- Allow admin to override default config location also for rebar and such (#1627)

loetie (1):

- [core] Add platform.twitter.com to whitelist (#1647)

Release 0.28.0

Welcome to Zotonic 0.28.0, released on 1 May, 2017.

Main changes are:

- Added a temporary workaround for ErlyDTL compiler creating too many atoms (#1676).
- Added reqdata to #dispatch_host notification (#1664).
- Fixed non-admin users have no permission to upload file (#1665).
- Fixed depcache cleanup causes crash (#1671).
- Fixed uploading extremely large files crashes z_file_entry (#1650).

Commits since 0.27.0

There were 17 commits since release 0.27.0.

Big thanks to all people contributing to Zotonic!

Git shortlog**David de Boer (5):**

- Add 0.27.0 release notes
- mod_seo: Rename shorturl to shortlink (#1651)
- core: Upgrade depcache
- mod_acl_user_groups: Fix non-admins denied permission to upload file (#1665)
- scripts: Raise maximum number of atoms (#1676)

Dirk Geurs (1):

- Retain reqdata for use in #dispatch_host notification (#1664)

Marc Worrell (11):

- Do not crash if resulting image is too big (#1652)
- Lock new z_stdlib
- mod_oauth: cleanup templates, add option for 'anonymous' users with only the consumer key/secret. (#1635)
- mod_video: change log level of 'Video info' lager message.
- Upgrade s3filez.
- mod_base_site: add 'showmedia' filter
- mod_base_site: fix typo.
- core: remove some throws from z_db, return error tuple instead.
- mod_survey: add es translation for 'submit'

Release 0.29.0

Welcome to Zotonic 0.29.0, released on 5 June, 2017.

Main changes are:

- Added an Exometer metric for the number of depcache evictions (#1682).
- Added Erlang maps support in ErlyDTL (#1684) and `moreresults` action (#1689).
- Added ISO 8601 date support to `date` filter (#1702).
- Added a *Exometer metrics* (page 330) cookbook (#1679) and a *Server configuration* (page 85) chapter to the documentation (#1680).
- Fixed `erlang.config` not read during `sitetest` (#1672).
- Fixed race condition in notifications during database transaction (#1693).
- Fixed ErlyDTL compilation generating too many atoms (#1673).
- Fixed `wav` mimetype for Internet Explorer (#1661).

Commits since 0.28.0

There were 24 commits since release 0.28.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

David de Boer (12):

- core: Add ErlyDTL maps support (#1684)
- doc: Bump year
- core: Ignore `priv/filezcache`
- tests: Read `erlang.config` in `sitetest` command (#1672)
- `mod_base`: Support maps result in `moreresults` filter (#1689)
- doc: Fix build (#1694)
- doc: Fix RTD theme's CSS not loaded
- core: Record depcache evictions in exometer (#1682)
- doc: Add server configuration chapter
- `mod_base`: Support ISO 8601 dates in `filter_date` (#1702)
- core: Delay notifications during database transactions (#1693)
- `mod_editor_tinymce`: Don't override language if already set (#1697)

Maas-Maarten Zeeman (1):

- doc: Added information on increasing `nf_conntrack_buckets`

Marc Worrell (9):

- core: fix a problem where compilation of templates generates random atoms. Fixes #1673 (#1674)
- `mod_acl_user_groups`: fix a problem with insert checks without passed props.
- core: fix a problem with `.wav` mime type (#1678)
- core: don't crash in `rsc_gone` on race conditions.

- mod_oauth: use the user-id from the app-token, not the consumer.
- Upgrade z_stdlib
- Lock new z_stdlib
- Upgrade z_stdlib
- mod_email_status: better transient error handling.

Marco Wessel (1):

- doc: Add Exometer cookbook (#1679)

yorivanloo (1):

- translation: 'bedank' to 'bedankt' (#1681)

Release 0.30.0

Welcome to Zotonic 0.30.0, released on 3 July, 2017.

Main changes are:

- Fixed a problem with escaping characters in From/Reply-To e-mail headers.
- Fixed retrieving header info when there is no wm_reqdata.

Commits since 0.29.0

There were 7 commits since release 0.29.0.

Big thanks to all people contributing to Zotonic!

Git shortlog**David de Boer (1):**

- Revert "scripts: Raise maximum number of atoms (#1676)"

Maas-Maarten Zeeman (2):

- core: Fix retrieving header info when there is no wm_reqdata
- mod_mqtt: Provide a minified version of qlobber to save some kbs

Marc Worrell (4):

- smtp: fix a problem with escaping certain characaters in email from/reply-to headers.
- core: add max smtp connection per domain
- Lock z_stdlib on 0.x branch
- mod_acl_user_groups: use default content group if no content group defined.

Release 0.32.0

Welcome to Zotonic 0.32.0, released on 4 September, 2017.

Main changes are:

- Added support for edges in CSV data (#1690).
- Sort textual admin search results on relevance (#1799).
- Fixed a problem where validation crashes on inputs with a : in their name (#1785).

- Fixed unintentional password changes (#1801).

Commits since 0.31.0

There were 7 commits since release 0.31.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

David de Boer (3):

- core: Pass is_import to normalize (#1726)
- mod_admin_identity: Fix unintentional password changes (fix #1801)
- tests: Raise timeout to fix build failures (#1806)

Dirk Geurs (1):

- build: Install openssl in 0.x Docker container (#1792)

Marc Worrell (3):

- mod_import_csv: add support for incoming/outgoing edges in normal csv data. Issue #1690
- core: fix a problem where validation crashes on inputs with a ':' in their name (#1785)
- mod_admin: ensure that text search results are sorted on relevance (#1799)

Release 0.33.0

Welcome to Zotonic 0.33.0, released on 2 October, 2017.

Main changes are:

- Added support for multiple e-mail addresses in survey e-mail field.

Commits since 0.32.0

There were 3 commits since release 0.32.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

Marc Worrell (3):

- mod_base: add GitHub icons (#1794)
- mod_survey: support multiple email addresses in survey email field.
- mod_admin_merge: fix left/right display in confirm dialog.

Release 0.34.0

Welcome to Zotonic 0.34.0, released on 6 November, 2017.

Main changes are:

- Fixed admin password check.

Commits since 0.33.0

There were 2 commits since release 0.33.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

Maas-Maarten Zeeman (1):

- `mod_authentication`: Added foldl notification which can be used to customize the logon error page.

Marc Worrell (1):

- `core`: also check 'admin' password if password is set to 'admin'.

Release 0.35.0

Welcome to Zotonic 0.35.0, released on 4 December, 2017.

Main changes are:

- Added automatic pivot table re-creation (#1735).
- Added `match_object_ids` search term.
- Fixed RSS feeds by serving them with Content-Disposition inline (#1847).
- Fixed bug on non-SMP systems by enabling SMP by default (#1490).
- Fixed WebSocket error messages by correctly closing WebSocket connections (#1840).
- Fixed inconsistent data by re-pivoting both subject and object when edge is inserted/deleted (#1756).

Commits since 0.34.0

There were 14 commits since release 0.34.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

Dirk Geurs (1):

- Re-create custom pivot table when definition changes (#1735)

Fred Pook (1):

- `[mod_export]` Observe content disposition (#1847)

Maas-Maarten Zeeman (2):

- `core`: Carefully close websocket processes instead of killing them. Fixes #1840
- `mod_authentication`: Use sudo context to update password

Marc Worrell (10):

- Always enable `smp`. Fixes #1490
- `core`: remove dependency on `zotonic_release.hrl` from main `zotonic.hrl` include.
- `mod_export`: don't crash on dispatches that are not `export_rsc_query`.
- `core`: on edge change, pivot both object and subject. Issue #1756. Fixes #1722

- tinymce/core: add tinymce option to link to url. Manual merge of 9c8964f51fd84e00458a27d8838b0e0c2894ee92
- core: fix a problem in m_edge where edge-ids were given to a routine handling rsc-ids.
- mod_search: add option to match on given object-ids instead of only a subject.
- mod_search: add 'match_object_ids' search term.
- mod_search: add 'match_object_ids' search term.
- mod_editor_tinymce: remove quote at bottom of props dialog

Release 0.36.0

Welcome to Zotonic 0.36.0, released on 5 February, 2018.

Main changes are:

- Added support for BCE dates and era element (#1869, #1865).
- Added compatibility with Erlang/OTP 20 (#1864).
- Fixed parse error in search queries (#1854).
- Fixed empty image alt tag (#1868).

Commits since 0.35.0

There were 11 commits since release 0.35.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

David de Boer (2):

- deps: Fix BC date handling (#1865)
- docs: Incorporate changes from z_stdlib to improve date handling (#1869)

Maas-Maarten Zeeman (1):

- mod_base: Added filter_round from master to 0.x branch

Marc Worrell (8):

- mod_oauth: support OAuth signature check on POST to controller_api. (#1862)
- Make 0.x compatible with OTP20 (#1864)
- mod_atom_feed: fix a problem with search term selection for /feed/search
- mod_base: fix the static template lookup.
- docs: add docs for 'round' filter.
- Fix a problem with parsing custom filters. Fixes #1854 (#1866)
- core: fix a problem where an empty 'alt' attribute was suppressed. Fixes #1868
- core: image option 'removebg' has now also an optional color to be removed

Release 0.37.0

Welcome to Zotonic 0.37.0, released on 5 March, 2018.

Main changes are:

- Added color argument to `removebg`.
- Fixed static requests at root (/).

Commits since 0.36.0

There were 5 commits since release 0.36.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

David de Boer (1):

- `mod_base`: Fix `controller_static_pages` for requests at / (root)

Maas-Maarten Zeeman (1):

- `mod_component`: Remove the polling workaround for missing onload functionality. It is slowing down chrome

Marc Worrell (3):

- `core`: image option 'removebg' has now also an optional color to be removed
- `mod_menu`: fix a problem where `_menu_extra.tpl` is not shown if the normal menu is empty.
- `core`: Truncate texts for tsv indexes. Only index the first 30K

Release 0.38.0

Welcome to Zotonic 0.38.0, released on 2 April, 2018.

Main changes are:

- Fixed session cookies missing secure attribute when TLS is terminated by a proxy (#1889).
- Fixed `hassubject` and `hasobject` query term parsing (#1894, #1896).

Commits since 0.37.0

There were 10 commits since release 0.37.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

David de Boer (2):

- `mod_search`: Fix `hassubject` and `hasobject` query term parsing (#1894)
- `mod_search`: Fix #1895 by also splitting non-binaries (#1896)

Marc Worrell (8):

- `mod_admin_category`: fix a problem with deleting pages in a category.

- Fix messages on category delete error.
- Lock new z_stdlib
- Lock new z_stdlib
- Lock new z_stdlib
- core: sanitize pivot text before stemming/tsvector. This fixes an issue where illegal utf8 could crash the pivot process.
- mod_import_wordpress: better image handling. With thanks to @fredpook
- Set session cookie to secure if site protocol is https. Fixes #1889 (#1890)

Release 0.40.0

Welcome to Zotonic 0.40.0, released on 4 June, 2018.

Main changes are:

- Added translatable slug and SEO fields (#1911).
- Added JavaScript and CSS minification (#1903).
- Fixed surname prefixed not saved during signup (#1907).
- Fixed duplicate username error message not shown (#1908).

Commits since 0.39.0

There were 11 commits since release 0.39.0.

Big thanks to all people contributing to Zotonic!

Git shortlog

David de Boer (1):

- core: Fix comet error when logging in (#1877) (#1878)

Maas-Maarten Zeeman (2):

- jsmin (#1903)
- mod_component: Fix needed to work with automatically minified resources

Marc Worrell (8):

- mod_admin_category: fix a problem with moving resources to another category.
- core: also remove ascii 0 on text pivots.
- mod_atom: fix timezone issue in test.
- core: drop testsandboxdb schema on site start
- mod_signup: Fix #1755 by showing error on duplicate username (#1908)
- core: make the slug translateable. Issue #1095 (#1911)
- core: force edge delete/insert events on edge update of object_id.
- core: better version of m_edge:update_nth/5
- mod_signup: fix a problem with surname prefix on signup. (#1907)

Release 0.41.0

Welcome to Zotonic 0.41.0, released on 1 October, 2018.

Main changes are:

- Upgraded jquery from 1.11.1 to 3.3.1 (#1922).
- Added livereload to mod_development (enable via `/admin/development`).
- Fix a problem with default date handling in resource updates (#1885).
- mod_mailinglist now uses mod_email_status for bounce checks.

jQuery upgrade note:

Please note that if you manually include jquery-migrate in your templates, you will have to change your templates manually because jquery-migrate is also upgraded from jquery-migrate-1.2.1 to jquery-migrate-1.4.1.

Commits since 0.40.0

David de Boer (4):

- core: Don't default start date to now (#1885)
- core: Don't crash when files have been deleted (#1928)
- docker: Fix rebar failure during Docker build (#1930)
- core: Fix pivot failure during manage_data (#1932)

Maas-Maarten Zeeman (5):

- core: Fix, minify all concatenated js and css files, not just the first
- core: Compress svg files
- mod_mqtt: Support retained messages
- Added 0.41.0 release notes with an extra note about jquery migrate
- core: Call z_datamodel:manage inside a transaction to prevent inconsistencies. See #1927

Marc Worrell (17):

- mod_mailinglist: Fix typo in predicate name
- mod_admin_merge: add option to add missing translations to the winner. (#1915)
- mod_authentication: escape q arg
- Fix leap year problem
- Fix a problem with the slug when merging resources.
- mod_translation: Fix a problem with copying languages
- core: fix tz update for tables/columns that are reserved SQL keywords.
- Use folsom and bear from github.com/folsom-project. Fixes #1920
- mod_development: add livereload of css and js. Issue #1793 (#1924)
- base: Upgraded old jquery to 3.3.1 (#1926)
- Update mailinglist to work with email_status (#1929)
- mod_development: work around for server caching of livereload css
- core: Add option to use 'user_id' as controller_page id configuration.
- base: allow 'user_id' as configured id for controller_redirect.
- controller_file: fix a problem where IE11 could not download files.

- Prepare 0.41.0
- Fix issue for `jquery.find('#')`, upgrade bs.js. Issue #1934 (#1935)

loetie (1):

- Support `failure_message` in `email_unique` validator (#1931)

Release 0.42.0

Welcome to Zotonic 0.42.0, released on 5 November, 2018.

Main changes are:

- Prevent search engine indexing of error, logon, and language switch pages.
- Add Yandex site verification
- Support `user_id` as configured dispatch id in `controller_redirect`

Commits since 0.41.0

David de Boer (1):

- `mod_admin`: Pass content group id when inserting a resource (#1937)

Marc Worrell (9):

- `mod_seo`: add Yandex site verification
- `mod_base`: support 'user_id' as id value in `controller_redirect`.
- `mod_seo`: allow overrule of shortlink and cononical url
- `mod_translation`: do not index the translation urls, as this will end up in recursions for the crawlers.
- `core`: fix a problem where the pivot process could crash if a resource was deleted during pivot.
- `mod_authentication`: set 'noindex' and 'notrack' template vars on the logon and error pages.
- Add 0.42.0 release notes
- Set version to 0.42.0
- Merge branch '0.x' into 0.42

Release 0.43.0

Welcome to Zotonic 0.43.0, released on 19 December, 2018.

Main changes are:

- Allowed files in `mod_acl_user_groups` are now configurable
- Security fixes for reflected XSS in the admin and `skel/blog/archives.tpl`
- Hardened HTTP headers for securing Zotonic sessions and requests
- `mod_twitter` now uses polling for fetching tweets, stopped using deprecated streaming API

Security Advisory

If you have a blog site derived from the `skel/blog` then replace the `archives.tpl` file in your site with the one provided in `priv/skel/blog/archives.tpl`

Compatibility

If you include a page of your site inside a frame on another site, then set the `allow_frame` option on the affected dispatch rule.

Commits since 0.42.0

David de Boer (1):

- docker: Build on Erlang 19.3 (#1950)

Maas-Maarten Zeeman (1):

- Support binary data over websockets. Fixes #1953

Marc Worrell (4):

- `mod_editor_tinymce`: fix a problem where zmedia stopped parsing if a non zmedia image was encountered.
- Fix a problem with reusing ids for tinymce editors.
- `mod_twitter`: stream api has been removed, add poller instead. (#1955)
- Make acceptable mime types configurable per user group (#1956)

Michel Rijnders (3):

- Remove unnecessary call to internal function (#1947)
- Exclude `node_modules` from the file watcher (#1948)
- Remove empty check (#1957)

Release 0.44.0

Welcome to Zotonic 0.44.0, released on 18 January, 2019.

Main changes are:

- New module `mod_ratelimit`, which limits authentication requests and password resets
- Automatic session timeout, configure with `site.session_expire_inactive` (default 14400 seconds)
- See all your active sessions at `/logon/sessions`
- Mnesia files are now placed in a directory per node, e.g. `priv/mnesia/zotonic001@foobar/`

BC breaks

- The password reset dispatch rule was changed. If you override `email_password_reset.tpl`, you now need to pass the username too.

Before:

```
{% url logon_reset secret=secret use_absolute_url %}
```

After:

```
{% url logon_reset secret=secret u=username use_absolute_url %}
```

- The delegate and postbacks for authentication requests has been changed. Use now `controller_logon` and check the logon and password reset templates for the correct arguments. If you didn't add your own authentication templates then you don't need to do anything.

Commits since 0.43.0

David de Boer (1):

- core: Remove pivot_resource/2 export (#1967)

Maas-Maarten Zeeman (2):

- Prevent a crashing mqtt module subscriber to also crash the module. Fixes #1962
- Added the possibility to compile when running zotonic (#1954)

Marc Worrell (17):

- mod_twitter: fix coordinates lookup.
- mod_email_status: fix problem upgrading older email_status module.
- mod_twitter: fix a problem with the first page when following users.
- core: also trigger pivot_done on manual rsc pivot (#1966)
- Pivot done (#1968)
- core: Move mnesia files to a directory per node. (#1969)
- mod_facebook: Use https for the Facebook share link
- Ratelimit for authentication requests (#1970)
- Session timeout on inactivity and session overview (#1972)
- mod_base: make the z_logon cookie more secure.
- core: fix problem with setting x-frame-options
- Add 'Please try again in ...' to the password reset forms.
- mod_authentication: add nl translations.
- mod_authentication: pass wire-args on logon.
- mod_seo: use Google Analytics localStorage instead of cookies.
- Remove compile warnings.
- Upgrade webzmachine

Release 0.45.0

Welcome to Zotonic 0.45.0, released on 29 January, 2019.

Main changes are:

- ACL fixes, especially for upload restrictions

Commits since 0.44.0

Maas-Maarten Zeeman (1):

- Use new_request to initialize the context for a new request (#1976)

Marc Worrell (9):

- mod_seo: Fix a problem with Google Analytics settings.
- mod_authentication: check if user has an username on expired status.
- mod_base: better activity tracking
- mod_editor_tinymce: fix problem with tinymce init in dialog on firefox

- `mod_acl_user_groups`: fix a problem where the ACL settings for file-types were not visible.
- `mod_acl_user_groups`: fix a problem where the max-upload file size was not properly checked.
- `mod_acl_user_groups`: show default ACL mime setting if not set in group.
- `mod_acl_user_groups`: fix column width for mime-rule view.
- Set version to 0.45.0 (#1980)

loetie (1):

- Badmatch in expired logon on `get_username` value (#1977)

Release 0.46.0

Welcome to Zotonic 0.46.0, released on 15 February, 2019.

Main changes are:

- Added `mod_auth2fa` for two factor authentication support
- Added `mod_clamav` for virus scanning of media items
- Newly entered passwords can now be checked against a regex, configured in `mod_admin_identity.password_regex`
- New versions of tinyMCE, jQuery, and Bootstrap.

Commits since 0.45.0

David de Boer (6):

- core: Upgrade jquery-ui to 1.12.1, which fixed CVE-2016-7103 (#1992)
- core: Embed YouTube over HTTPS (#1983)
- `mod_admin_identity`: Check password regex for new users (#1997)
- core: Add `cors_headers` notification (#1993)
- `mod_bootstrap`: Upgrade to 3.4.0 (#1991)
- `mod_authentication`: Fix config check (#2003)

Marc Worrell (9):

- Add two-factor authentication (TOTP) (#1987)
- `mod_authentication`: Autocomplete off on password field
- `zotonic_status`: add optional ip whitelist for accessing the zotonic status site. (#1990)
- `mod_clamav`: Scan for viruses in uploaded files with ClamAV (#1994)
- `mod_editor_tinymce`: upgrade to 4.9.3 (#1999)
- `mod_authentication`: add config to enable reminder email to unregistered addresses. (#1998)
- `mod_twitter`: Add configurable poll delay. Issue #1985 (#2002)
- Lock new webzmachine
- `mod_clamav`: fix error message for tcp connect errors.
- `mod_signup`: check password against configured regex.

Rob van den Bogaard (1):

- [`mod_admin`] Don't let flush button in admin interface flush all sites (#2000)

Release 0.47.0

Welcome to Zotonic 0.47.0, released on 4 March, 2019.

Main changes are:

- Fix for a problem where many websockets could be opened
- Fix for a problem duplicating resources with custom slugs
- Fix for a potential xss issue in the mod_admin
- Unpublished resources are now created without an publication date
- New option per predicate to insert new edges as the first edge
- New option to force users to re-agree with updated terms or privacy documents

Commits since 0.46.0

David de Boer (3):

- doc: Explain anonymous context in task queue callback (#2007)
- core: Default publication_start to null (#2008)
- core: Fix publication_start default (#2015)

Dorien (1):

- Fix #2009: Add lists to plugins to support lists in editor (#2010)

Maas-Maarten Zeeman (2):

- core: Fix the accidental creation of a global variable
- core: Prevent too many WebSockets (#2014)

Marc Worrell (11):

- mod_editor_tinymce: remove branding.
- mod_admin: fix a problem with escaping values in admin_button_dropdown template. (#2006)
- mod_admin: rename 'selected_value' to 'selected_qvalue' to make clear it is an unescaped query arg.
- mod_seo: allow site config settings overrule the seo.noindex config.
- core: dump sql queries as string in errors.
- core: Edge insert top (#2011)
- mod_authentication: Add T&C re-agreement and restructure logon errors (#2012)

loetie (1):

- When duplicate reset custom_slug boolean (#2005)

Release 0.48.0

Welcome to Zotonic 0.48.0, released on 9 April, 2019.

Main changes are:

- mod_geoip to lookup Geo information for an IP address
- New connect dialog in the admin
- Option to let users agree to changed Terms & Conditions after logon

- Changes to `mod_clamav` virus scanning to allow uploads of files larger than the maximum file size for clamd.

Commits since 0.47.0

David de Boer (1):

- Split HTML head templates to make overrides possible (#1746) (#2017)

Dorien (1):

- admin: Add block around modal footer (#2018)

Maas-Maarten Zeeman (3):

- core: Template controller accidentally used the wrong context variable
- `mod_bootstrap`: Upgrade to 3.4.1 and add a upgrade description. Fixes #2020 (#2021)
- Locked new diffy and `z_stdlib`. Fixes diffing bug and adds options for encoding ubf

Marc Worrell (20):

- admin: Combine new-page, upload, and connect dialog (#2019)
- `mod_clamav`: fix default max size.
- Upgrade `z_stdlib`
- Lock new `z_stdlib`
- Fix a problem where the new find/new connect dialog failed with other ACL modules and with linking in texts.
- `mod_geoip`: new module to map IP addresses to country names. (#2025)
- `mod_geoip`: add `ip2info` filter to extract more information about an IP address
- Rename filter `ip2info` to `ip2geo`
- `mod_admin_identity`: let admins not switch to the 'admin' account.
- `mod_admin`: fix error message display on erroneous media upload.
- `mod_content_groups`: give feedback why a content group could not be deleted. (#2023)
- Tos agree on create (#2022)
- `mod_search`: fix a problem with specifying 'qargs' in a stored query. Fixes #2026
- Suppress backup files for dispatch rules.
- `mod_clamav`: fix spec
- `mod_acl_user_groups`: reconnect all websocket connections of an user if their permissions change. Fixes #2028 (#2029)
- Redesign of connect/new tab. (#2034)
- Fix a problem with the `preflight_check` of a query with 'qargs' term.
- `clamav`: set medium flags 'is_av_scanned' and 'is_av_sizelimit' (#2031)
- New `z_stdlib`
- Fix a problem with the menu editor and the new connect/find dialog. Fixes #2038

Release 0.48.1

Welcome to Zotonic 0.48.1, released on 12 April, 2019.

Main changes are:

- Fix for the new connect dialog if used from the resource blocks

Commits since 0.48.0

Marc Worrell (1):

- Fix for page block link dialog (#2040)

Release 0.49.0

Welcome to Zotonic 0.49.0, released on May 6, 2019.

Main changes are:

- **BC break:** Upgraded Bootstrap CSS and JavaScript in `mod_admin` to 3.4.1; `mod_admin` no longer ships with a separate Bootstrap instance but uses the one from `mod_bootstrap`. Any customizations you made to the Bootstrap CSS will therefore be visible in the admin, too.
- **Logging:** the `/admin/log` changed and now more useful. Events logged include module activation, config changes, rate limiting and ACL changes.
- `mod_clamav` now also rejects MS Office files with `externalLinks` directories. This can be turned off using the Zotonic config key `clamav_reject_msoffice_external_links`
- Uploaded CSV files are now given the mime type `text/csv` and properly sanitized
- Password reset tokens are now valid for two days.
- `mod_linkedin` now uses their v2 API

Commits since 0.48.1

David de Boer (2):

- `mod_admin`: Upgrade Bootstrap (#2041)
- Add password reset token expiration (#2055)

Marc Worrell (5):

- LinkedIn v2 (#2048)
- `mod_linkedin`: fix cherry-pick
- Add config option to force new password on password reset (#2056)
- Additional virus scanning for spreadsheet files (#2054)
- Logging of system/acl events in `mod_logging` (#2057)

Release 0.49.1

Welcome to Zotonic 0.49.1, released on May 8, 2019.

Main changes are:

- Fix a problem where search was not returning any results due to an access control check in `mod_logging`

Commits since 0.49.0

Marc Worrell (2):

- mod_admin: add error message text for av_external_links in media upload.
- Fix problem with mod_logging blocking searches unless admin user.

Release 0.49.2

Welcome to Zotonic 0.49.2, released on May 15, 2019.

Main changes are:

- Fix a problem with password resets on expired passwords
- Fix a problem with password resets when the new password matches the old one

Commits since 0.49.1

Marc Worrell (3):

- core: fix a problem with site.password_force_different where the user could be redirected if the passwords matched.
- mod_import_csv: allow device pid to be passed to parse_csv:scan_lines/2
- mod_authentication: fix a problem with resetting expired passwords. (#2060)

Release 0.50.0

Welcome to Zotonic 0.50.0, released on June 27, 2019.

Main changes are:

- Compatible with OTP 21, minimal OTP version is now 18
- View a log of Javascript UI errors in /admin/log/ui (disable with mod_logging.ui_log_disabled)
- Log more events like identity changes and applied rate limit throttling
- Only the admin can now change 2FA settings of the admin
- In the admin connect dialog a new filter on category is added
- Fix a problem with password resets on expired passwords
- Fix a problem with password resets when the new password matches the old one
- Fix a problem with wrong orientation of portrait video on Linux
- Accept resource updates with dmy formatted date values

The `exometer` application was wrongly initialized since it was split into `exometere_core` and other applications. The default erlang configuration had an entry for `exometer`, you should rename this in your `erlang.config` file to `exometer_core`.

Commits since 0.49.2

David de Boer (2):

- build: Test 0.x against Erlang/OTP 21 (#1936)
- mod_seo: Don't store noindex value in db when read from site config (#2081)

Dorien (1):

- mod_admin: Convert predicate to binary to fix matching (#2091)

Maas-Maarten Zeeman (1):

- Exometer core config update (#2095)

Marc Worrell (17):

- core: fix a problem where a filename with a space failed identify. (#2080)
- Log levels and extra checks for identity changes / ratelimit (#2078)
- Allow all categories if no predicate is defined. Fixes #2083 (#2085)
- mod_admin: make upload element id dynamic to prevent id clash.
- Log client ui errors (#2086)
- Fix race condition in rsc_gone (#2087)
- Only allow admin to change admin 2fa settings (#2077)
- In connect/create dialog, add extra filter on category.
- Fix link in body, hide 'link' when editing hierarchies.
- WIP: Ensure 'rsc' table ref on search sort terms (#2035)
- mod_logging: add config mod_logging.ui_log_disabled
- Fix a problem with searching on 'id:..' in the create/link dialog. Issue #2089 (#2092)
- admin: show 'unique name' field in quick edit for all meta resources. (#2097)
- mod_video: cleanup and fix rotation on linux (#2099)
- Support 'dmy' input format for dates. (#2098)
- mod_ratelimit: add 'm_ratelimit:delete_event/3' to remove a ratelimit
- mod_admin: do not toggle 'depiction' if image is embedded in tinymce (#2103)

Release 0.51.0

Welcome to Zotonic 0.51.0, released on August 26, 2019.

Main changes are:

- Added DTAP (Development, Testing etc.) status of site to http headers and `m.site.environment`
- Updated PostgreSQL driver `epgsql`
- Added *Copy download link* button in admin media panel
- Fixes to admin connect/create dialog

Commits since 0.50.0

Dorien (1):

- Download link (#2163)

Maas-Maarten Zeeman (1):

- Renamed all pgsq1 references to epgsq1 (#2100)

Marc Worrell (14):

- mod_logging: set {ssl, any} on client log url

- `mod_base`: fix json for `controller_error` http-status return.
- Use ubuntu trusty for OTP 18 on Travis (#2116)
- Use `epgsql` error record (#2122)
- `mod_admin`: in the new-rsc dialog, always show all categories. Fixes #2119
- Fix double translations in po files.
- More robust media insert of `#upload` with data
- Fix icon paths in `mod_artwork` font css files. (#2159)
- Show connected media in tinymce select media dialog. (#2152)
- Allow pager to accept a list for the 'result' arg. (#2125)
- `scomp` pager: fix args for list handling
- pager: fix html for 1 page result.
- `mod_admin`: simplify 'copy download link to clipboard' functionality.
- Add support for DTAP environment. (#2151)

Release 0.51.1

Welcome to Zotonic 0.51.1, released on September 13, 2019.

Main changes are:

- Updated `depcache` for compatibility with OTP18
- Upgrade `gen_smtp` to 0.14

Commits since 0.51.0

Marc Worrell (3):

- Upgrade `gen_smtp` to 0.14 (#2168)
- core: let `z_trans` handle `#m{ }` values coming out of models.
- Upgrade `depcache`.

Release 0.51.2

Welcome to Zotonic 0.51.2, released on September 26, 2019.

Main changes are:

- Fix for recognizing PDFs generated by Microsoft Word
- Fix for an issue where handling maps in templates could result in a crash

Commits since 0.51.1

Marc Worrell (2):

- Fix PDF identification for some special PDFs (#2192)
- Fix a problem with map values in templates (#2194)

Release 0.52.0

Welcome to Zotonic 0.52.0, released on November 6, 2019.

Main changes are:

- Add controller option `set_session_cookie`, set to `false` to not set a session cookie
- Add a *delete* button in the tinymce media edit dialog
- API for fetching the active language list `/api/translation/language_list`
- Add tinyMCE release 4.9.6
- Fix site install using the cli `addsite` command; replace `.dev` with `.test`
- Fix a problem where ui logging in `mod_logging` could not be disabled
- Fix a problem with resource merge where edges notifications were not sent
- Fix a problem where path names in template debug were truncated

Commits since 0.51.2

Dorien (1):

- Add dispatch so pager tpl can use this (#2197)

Maas-Maarten Zeeman (1):

- core: Implemented option to make setting the session cookie optional. (#2207)

Marc Worrell (11):

- `mod_logging`: no logging if `is_ui_ratelimit_check` returns false. (#2203)
- Add api to fetch language list (#2208)
- Fix widgetmanager fallback for non JSON data
- Show full filepath in template debug. Fix #2118 (#2218)
- core: merge edges by inserting new ones. (#2219)
- `mod_admin`: fix problem with crashes on undefined translations. Fix #2212
- Fix problems with blog skel installation (#2201)
- `mod_editor_tinymce`: add zmedia 'delete' button. Fixes #2105 (#2222)
- Fix `relpath` display in dtl compiler. Issue #2118
- Include `z.zeditor.js` widget in the admin. This enables use of 'do_zeditor' classes.
- TinyMCE 4.9.6 (#2224)

Release 0.53.0

Welcome to Zotonic 0.53.0, released on December 10, 2019.

Main changes are:

- jQuery updated to 3.4.1
- Fix a problem with handling email errors
- Remove a problem with Erlang module name clashes (thanks to @rl-king)
- Always repivot subject resource on edge delete/insert
- Fix a problem with displaying images with defined crop-center in the admin

- Fix a problem where the authors of resource revisions were visible on the revision list for all users

Commits since 0.52.0

Maas-Maarten Zeeman (4):

- Wip blackhole controller (#2236)
- mod_bootstrap: Remove sourceMappingURL properly
- mod_bootstrap: Ran updated script to remove the sourceMappingURL from the css files
- mod_base: Update jquery to 3.4.1

Marc Worrell (12):

- Correct 0.52.0 release date
- Fix a problem with smtp throws not caught. (#2230)
- Fix smtp retry logic
- Fix a problem with log email for cat templates. Fix #2232 (#2233)
- Merge pull request #2254 from zotonic/wip-upgrade-jquery-3.4.1
- Merge pull request #2252 from zotonic/wip-fix-bootstrap-update-script
- Repivot subject on edge delete/insert (#2239)
- Load google chart via https
- Fix erlang_localtime and qdate_localtime name clashes (#2263)
- Do not emit the link-... event if an edge changes (#2268)
- mod_admin: Fix crop center view in rsc_edge_media.tpl Fix #2269 (#2270)
- Only show revisions list if user has edit permissions. (#2273)

loetie (1):

- Add block around language options (#2271)

Release 0.54.0

Welcome to Zotonic 0.54.0, released on February 4, 2020.

Main changes are:

- Now fixed DHE parameters are used for SSL. If there is an existing DH file then that file will still be used.
- Options and configurations to set the HSTS headers
- Upgraded Locus (MaxMind Geo Location Library), now a license key is mandatory. See http://docs.zotonic.com/en/0.x/ref/modules/mod_geoip.html
- A scomp ignore_user_inactivity is introduced to disable activity tracking. This is useful for pages that should not timeout with their session.
- Generating the translation .pot files now needs *administrator* or *use.mod_development* rights.
- Fixes a problem with LinkedIn OAuth authentication
- Fixes a problem where password reminders could be sent to non-user accounts.

Commits since 0.53.0

Maas-Maarten Zeeman (4):

- core: Don't close the connection in a beforeunload, but later in an unload. Make sure all ajax requests are async (#2296)
- 1329 set hsts headers (#2297)
- 2290 session active (#2291)
- mod_ssl: Use rfc7919 dhe parameters (#2299)

Marc Worrell (8):

- mod_geoip: add support for MaxMind license key (#2283)
- mod_base: safe check on error stack object. Fix #2276
- mod_base_site: fix Twitter share page link.
- Fix problem with LinkedIn login. Fix #2277
- overlay: cancel js close click event.
- mod_authentication: Only send password reminders to resources with a password identity.
- mod_translation: demand admin or use.mod_development right to generate .pot files. Issue #2298
- Upgrade locus to 1.9.0

Release 0.54.1

Welcome to Zotonic 0.54.1, released on June 16, 2020.

Main changes are:

- Fix a problem with calendar functions on large dates.

Commits since 0.54.0

Marc Worrell (1):

- Prevent a problem with calendar functions on large dates.

Release 0.54.2

Welcome to Zotonic 0.54.2, released on August 4, 2020.

Main changes are:

- Fix a problem with embedding vimeo URLs

Commits since 0.54.1

Marc Worrell (1):

- OEmbed vimeo endpoint moved to vimeo.com

Release 0.54.3

Welcome to Zotonic 0.54.3, released on August 31, 2020.

Main changes are:

- Fix a problem with Custom validators and empty inputs (issue 2514)

Commits since 0.54.2

Marc Worrell (1):

- Allow error on empty input with Custom validator. Issue #2514 (#2515)

Release 0.55.0

Welcome to Zotonic 0.55.0, released on March 12, 2020.

Main changes are:

- All admin pages are now ssl only (if ssl enabled)
- Warning on admin/status page if the ssl app is not configured
- Option to start editing languages but not use them yet for the site itself
- Color coding of the admin depending on the current DTAP environment
- Show warning in the admin when editing system resources like categories and predicates
- Reorganization of the new-resource dialog
- Backoff for erroneous task-queue functions
- Added function `z:load_config()` to reload the `zotonic.config` file, this does not restart listeners but does change settings like the smtp proxy and DTAP environment.

Commits since 0.54.0

Maas-Maarten Zeeman (2):

- admin: Make all admin page accessible via ssl only. (#2310)
- 1337 ssl application configuration (#2304)
- core: Add an explicit table lock to prevent race conditions (#2333)

Marc Worrell (30):

- Remove X-DTAP-Environment header. Issue #2302
- Remove X-DTAP-Environment header. Issue #2302 (#2303)
- docs: Remove mention of x-dtap-environment header.
- Merge branch '2032-remove-dtap-header' into 0.x
- Fix bs3 class of progress-bar.
- mod_translation: add option to make languages available in de admin/edit but not yet enabled. (#2307)
- mod_admin: Show warning on edit page if editing system content. (#2311)
- mod_admin: change content warning link to datamodel
- core: add `z:load_config()` to reload the `zotonic.config` file. (#2313)
- mod_admin: reflect the environment in the admin colors. Issue #2319 (#2320)

- Hide the 'file upload' on initial show with preselected non-media category. Issue #2305 (#2322)
- mod_admin: use abs uri for copy of download link
- core: Add backoff for task queue errors.
- Add 'name.<name>' as selector for catincludes (#2316)
- email_status: Mark 'access denied' as unrecoverable.
- mod_admin: correct menu item text color.
- mod_admin: small correction in environment.less
- mod_email_status: add email status panel to email log.
- Add class 'data-table' which wraps td on any character
- Fix icons on email status view.
- mod_email_status: new nl translations
- mod_logging: new translations
- email: do not send any email to blocked addresses.
- mod_admin: new page dialog, swap columns and small changes (#2330)

Release 0.55.1

Welcome to Zotonic 0.55.1, released on March 16, 2020.

This is a patch releae to fix a problem with sending emails.

Commits since 0.55.0

Marc Worrell (1):

- Fix problem with sending emails due to string/binary conversions. Upgrade z_stdlib

Release 0.56.0

Welcome to Zotonic 0.56.0, released on April 23, 2020.

Possible breaking changes:

- We removed jquery-migrate from the standard template in modules/mod_base/templates/_js_include_jquery.tpl. If you need jquery migrate in your project then you can add this template to your site and re-add "js/apps/jquery-migrate-1.4.1.min.js" to the include library files.
- The `{% image %}` tag now includes `width` and `height` attributes. Ensure that you have width or height set to `auto` in your css if you are restricting one of the dimensions. Otherwise use the new `nowh` option to the image tag to suppress the width/height attributes.

Main changes are:

- Added tracing of database queries - switch on at the /admin/development and see SQL traces (and explain of slow queries) for your session only in the console log.
- Removed jquery-migrate from the standard jquery include
- Image tags now include width/height attributes
- Language preference is now stored in `z.lang` cookie instead of the database persistent storage. This greatly reduces the number of rows being added to the persistent storage table.
- SQL queries that timeout are now canceled, the query is now also monitored and logged in case of crashes or timeouts.

- If an email is received for an unknown hostname then a temporary error is returned if and only if there are enabled sites that are not running.
- Transport retransmission is now disabled. This turned out to cause more problems than solve, due to duplicate requests being sent in case of slow servers, making the server even slower.
- Pivot queue inserts are now more robust and handled by the pivot server process, this prevents race conditions.
- The site supervisor is reorganized for greater robustness against crashing of some services. These services are now organized in their own one-for-one supervisor and are independently restarted.
- Next/prev on the admin edit page is now evaluated when clicked. This keeps the correct category and speeds up page load of the edit page.
- Long connection lists on the admin/edit page are now truncated and can be viewed in full on the connections page.
- File modification times are now cached, this speeds up the template modification checks on busy servers.

Commits since 0.55.0

Maas-Maarten Zeeman (2):

- core: Let the pivot queue server insert new edges into the db (#2379)
- core: Add a database pool usage warning (#2382)

Marc Worrell (35):

- core: generate image width and height attributes. Issue #2306 (#2318)
- Remove dependency on \$.browser in zotonic.js
- Remove msie test in z.popupwindow.js
- Cleanup some js, remove jquery.ba-hashchange.js
- Remove deprecated '.load()' call from z.cropcenter.js
- Remove jquery migrate from default js jquery include
- Preliminary 0.56 docs
- Also document changed w/h attributes on the image tag
- Add 'loading' attribute to allowed image tag opts.
- Merge pull request #2352 from zotonic/jq-migrate-0x
- core: export z_email_receive:parse_email/4
- Fix problem with sending emails due to string/binary conversions. Upgrade z_stdlib
- Merge pull request #2356 from zotonic/email-patch-0.55
- Fix problem with sending emails due to string/binary conversions. Upgrade z_stdlib
- Release 0.55.1
- Merge pull request #2357 from zotonic/release-0.55.1
- Cleanup code.
- mod_translation: fix a problem setting the default language
- mod_base: sanitize redirect location for action redirect
- mod_admin: replace next/prev on admin with lazy search. Issue #2375 (#2376)
- Use z.lang cookie to store the language preference. Fix #2374 (#2378)
- SMTP: give 451 error if unknown host and not all sites are running

- Make pivot queue insert more robust and handle batches of ids.
- Cache file modification time lookups (#2384)
- Export pivot function, more conservative truncate of pivot title.
- With filter lower and upper, sanitize input on failure.
- Disable zotonic transport retransmission.
- Truncate long connections lists in the admin. Issue #2387 (0.x) (#2388)
- Trace database queries. Issue #2390 (#2391)
- Reorg site supervisor (0.x) (#2386)
- Allow to change category if category does not exist. Issue #2360 (0.x) (#2389)
- Catch throw on creation of pages or predicates.
- Merge pull request #2397 from zotonic/page-create-error-0x
- z_db_pgsql: fix types.
- Fix spanish translation.

Release 0.56.1

Welcome to Zotonic 0.56.1, released on June 16, 2020.

Main changes are:

- Fix a problem with calendar functions on large dates.

Commits since 0.56.0

Marc Worrell (1):

- Prevent a problem with calendar functions on large dates.

Release 0.56.3

Welcome to Zotonic 0.56.3, released on August 31, 2020.

Main changes are:

- Fix a problem with Custom validators and empty inputs (issue 2514)

Commits since 0.56.2

Marc Worrell (1):

- Allow error on empty input with Custom validator. Issue #2514 (#2515)

Release 0.57.0

Welcome to Zotonic 0.57.0, released on May 11, 2020.

Main changes are:

- Make compatible with OTP-22
- Fix a localtime problem for Theran/Iran timezone.
- Fix for autostart of survey

- Allow s3 filestore to retry on fetch errors
- Add option to create a s3 bucket if it is not present

For the timezone and OTP-22 fix the following dependencies need to be updated:

- mochiweb
- qdate_localtime

Easiest method is to delete the *deps* folder and let rebar refetch all dependencies.

Commits since 0.56.0

Arjan Scherpenisse (1):

- mod_survey bugfixes (#2429)

Maas-Maarten Zeeman (2):

- Use zotonic's qdate_localtime to fix a problem when using Iranian tz settings (#2424)
- Moved the qdate_localtime dep back to the original repo

Marc Worrell (8):

- Fix a problem with cache invalidation after a resource merge.
- Make compatible with OTP-22 (#2420)
- mod_filestore: retry upload / download errors. Fix #2187 (#2272)
- Allow to create s3 bucket when not present. (#2426)
- mod_development: fix layout.
- mod_filestore: set deleted date when deleting a file.
- mod_filestore: cleanup and better handoff of file to cache.
- mod_filestore: fix race condition. (#2427)

Release 0.57.1

Welcome to Zotonic 0.57.1, released on May 19, 2020.

Main changes are:

- Fix a problem with https connections and OTP-22
- Add a `is_disabled` flag to surveys

This updated mochiweb. Easiest upgrade method is to delete the *deps* folder and let rebar refetch all dependencies.

Commits since 0.57.0

Arjan Scherpenisse (1):

- mod_survey - option to hide survey form (#2432)

Marc Worrell (1):

- Upgrade mochiweb

Release 0.58.0

Welcome to Zotonic 0.58.0, released on November 2, 2020.

Main changes are:

- `{% image %}` tags now have a width and height attribute, use the `nowh` option to suppress generation of the width and height attributes
- Fix for a problem where an ACL deny rule could restrict too much
- Support to store JSON in PostgreSQL
- Fix a problem with Twitter/Facebook/LinkedIn authentication and Safari 14
- Fix a problem where very large dates could give unexpected results
- Better TLS cipher suite selection for SSL connections
- Allow custom formatters to show an error on empty input values

To fix sites that break on the width/height attributes on image tags there is an option added to the `zotonic.config` file: `media_tag_nowh`

If the `media_tag_nowh` option is set to `true` then all `{% image %}` tags have the `nowh` option added. Effectively suppressing the generation of width and height attributes.

Note that this option is *not* available on the 1.x version and later of Zotonic.

If you have css like:

```
img {
  max-width: 100%;
}
```

Then add one line:

```
img {
  max-width: 100%;
  height: auto;
}
```

Commits since 0.57.0

Arjan Scherpenisse (1):

- `mod_survey` - option to hide survey form (#2432)

Maas-Maarten Zeeman (5):

- Added encoding and decoding of json values (#2450)
- Use `sassc` instead of `ruby-sass` (#2499)
- Updated jquery to 3.5.1 (#2510)
- Added an extra timeout before the page reload to give the browser more time to synchronize the cookie between tabs (#2525)
- Bump mochiweb version (#2529)

Marc Worrell (16):

- Upgrade mochiweb
- Release 0.57.1 (#2436)
- ACL: Fix a problem where a specific deny rule restricted other content to be found. Fixes #1964
- Prevent a problem with calendar functions on large dates.

- OEmbed vimeo endpoint moved to vimeo.com
- Upgrade jsx to 3.0 and jsxrecord to 1.1. (#2484)
- Set jsx to 3.0.0
- Correct tag for jsx is v3.0.0
- Correct NL translation
- Fix a problem where changing language via javascript added an extra '?' to the url. Issue #2508 (#2509)
- Allow error on empty input with Custom validator. Issue #2514 (#2515)
- Fix a problem with dropbox file scanning for relative site directories.
- Add dropbox upload dispatch rule
- Add 0.x option to suppress wh attrs on image tags. Fixes #2527
- Make hreflang link elements abs urls.
- Fix a problem with OAuth flow on Safari 14. Issue #2536 (#2537)

Ruben Lie King (1):

- Log errors on resource update (#2523)

Release 0.59.0

Welcome to Zotonic 0.59.0, released on December 18, 2020.

This is a maintenance release and only includes fixes.

Main changes are:

- Fix an issue where illegal URLs might be accepted for preview generation
- Fix an issue where an export of a person did not include all expected properties

Commits since 0.58.0

Marc Worrell (8):

- Fix error message in m_rsc_update.
- Fix a problem where the nowh image option could be added to the image url.
- Fix a problem where generating a button-tag without text would omit the closing element.
- Be more strict on acceptable formats for resized images.
- mod_export: default disposition to 'attachment'
- Log transport timeouts.
- Small cleanup of media/file routines.
- Remove wrong zh translation.

Rob van den Bogaard (1):

- core: Fix m_rsc_export for person resources (#2548)

Release 0.60.0

Welcome to Zotonic 0.60.0, released on February 15, 2021.

This is a maintenance release and only includes fixes.

Main changes are:

- Added support for embedding Soundcloud using OEmbed
- Fixed a problem where no extension could be found for “application/pgp-keys” .asc files
- Made the maximum edge list length in the admin configurable using `mod_admin.edge_list_max_length`
- Use Github Actions for CI

Commits since 0.59.0

Marc Worrell (6):

- Test build using GitHub Actions. (#2561)
- Add build batch, remove Travis build batch
- Add mapping of application/pgp-keys to .asc Fixes #2581
- Make max edge list length configurable. Fixes #2580
- Fix typo.
- Fix Docs GH action.

Rob van den Bogaard (1):

- Add SoundCloud oembed provider (#2583)

Release 0.61.0

Welcome to Zotonic 0.61.0, released on June 7, 2021.

This is a maintenance release.

Main changes are:

- Changes `mod_seo_sitemap` module for sites with many pages and/or multiple sources of pages
- Fix for a problem where database connections were left idling if a timeout happened during a transaction
- Fix for a problem where the `zotonic stop` command could take a long time
- Support for Google Analytics new “G-” codes
- Support for Google Tag Manager
- Make it possible for an authenticated user to change their password on `/logon/change`

Commits since 0.60.0

Dorien (2):

- Refactor permissions to check use of `mod_logging` (#2606)
- Fix #1628: Use summary filter for meta description (#2605)

Marc Worrell (12):

- SEO GTM support and misc fixes (#2652)

- Add password change for current user (#2654)
- Info message if an url was dropped from a tweet.
- Add placeholders to password change input fields.
- New sitemap building using urlsets.
- Delete docs for sitemap.xml.tpl
- Add SEO sitemap docs.
- Add comments and license info. Delete empty SEO controllers.
- Better 'language' handling of seo_sitemap.
- Remove debug, fix urlset xml
- Fix a problem with connections lingering after SQL timeouts.
- Fix a problem where stopping Zotonic could take a long time. Fixes #2670

Rob van den Bogaard (1):

- Merge pull request #2659 from zotonic/sitemap-0x

Release 0.62.0

Welcome to Zotonic 0.62.0, released on July 7, 2021.

This is a maintenance release.

Main changes are:

- Fixed an issue where lib .tpl files were not searched inside the lib folder, the behavior is now corrected and the same as in the 1.x release
- Fixed a problem where large file uploads could give a timeout
- Fixed a problem where the progress bar was not shown during file uploads
- Fixed a problem where SEO sitemap routines could crash on long URLs
- Ensure that the default SEO canonical and shortlink URLs are absolute URLs

Incompatibilities:

- If you generated lib files using template files (.tpl) then ensure that those template files are present in the lib directory.

Commits since 0.61.0

Marc Worrell (7):

- Only update seo sitemap on insert/update actions.
- Fix a problem where the module indexer was not ready before the sitetests could start.
- Find lib .tpl files inside lib. This makes the location of lib .tpl files consistent with 1.x and also gives less surprises where those files can be found.
- Upgrade mochiweb
- Use abs uri for canonical and shortlink
- Make SEO sitemap loc 2000 chars long to accommodate longer URLs
- Fix a problem where the progress was not shown during file uploads. Fix a problem where a file upload could show a timeout alert.

Release 0.63.0

Welcome to Zotonic 0.63.0, released on Oct 29, 2021.

This is a maintenance release.

Main changes are:

- Fix for a problem where deactivating languages on an edit page could leave the deactivated language selected for editing
- Fix for a problem where the column of the UI log was too small for some IPv6 addresses
- Dutch translations added to mod_google and mod_facebook

Commits since 0.62.0

Dorien (1):

- Update link to facebook dev apps (#2761)

Maas-Maarten Zeeman (1):

- Fixed typo (#2739)

Marc Worrell (5):

- Fix a problem with activating languages on the edit page where an inactivated language might stay visible.
- Show 2FA key below QR code.
- Fix a problem where the ip address field of log_ui was too small for some IPv6 addresses.
- typo in nl translation
- Add FB translations

Release 0.64.0

Welcome to Zotonic 0.64.0, released on March 4, 2022.

This is a maintenance release.

Main changes are:

- Add Instagram Javascript to the sanitizer whitelist.
- Show username and last logon date in the admin on the user's page and in the user lists.
- Better support for additional URLs in the SEO sitemap table.

Commits since 0.63.0

Marc Worrell (7):

- Fix an issue where the allowed predicates of a subject did not change when its category was updated. Fixes #2791
- seo_sitemap: fix a problem where resources were not final-checked for inclusion in the sitemap.
- Allow www.instagram.com/embed.js in the sanitizer.
- In the admin show information about username and last logon.
- mod_filestore: add simple test function to test the filestore connection.
- seo_sitemap: add m_seo_sitemap:delete_before/3. (0.x)

- seo_sitemap: add index on modified. Add sql timeouts.
- Fix a problem where heart restarted Zotonic after a zotonic stop command. Fixes #2715

Rob van den Bogaard (2):

- Merge pull request #2874 from zotonic/0x-m_seo_sitemap-delete_before
- seo_sitemap: add notification at sitemap rebuild for custom indexing (#2894)

Release 0.65.0

Welcome to Zotonic 0.65.0, released on March 17, 2022.

This is a maintenance release.

Main changes are:

- Fix an issue where git protocol is not supported anymore by GitHub

Commits since 0.64.0

Marco (1):

- use https to access github after disabling git protocol (#2921)

Release 0.66.0

Welcome to Zotonic 0.66.0, released on May 24, 2022.

This is a maintenance release.

Main changes are:

- Ensure in sent emails that the ‘To’ field is properly formatted
- Do not crash in m_edge:objects/3 and subjects/3 on non existing resources
- The image template tag now supports the ‘decoding’ attribute

Commits since 0.65.0

Colin de Roos (1):

- Fix syntax error in zotonic.config.in (#2941)

Marc Worrell (4):

- core: on image tag, support the ‘decoding’ attribute
- core: log sass errors to the console log as info messages
- core: in m_edge, accept non existing resources for objects/3 and subjects/3
- core: ensure To email is properly formatted. (#2968)

Release 0.67.0

Welcome to Zotonic 0.67.0, released on July 5, 2022.

This is a maintenance release.

Main changes are:

- Fix an issue with the sendfile repository being removed from GitHub

- Better next/prev month calculation

Commits since 0.66.0

Marc Worrell (5):

- core: add z_datetime:next_month/2 and z_datetime:prev_month/2. (#3014)
- core: fix an issue where the email recipient was not properly escaped when an override was set. (#3018)
- mod_seo_sitemap: update sitemap on pivot, remove prio 0 resources from urlsets (0.x) (#3029)
- Update webzmachine, remove sendfile (#3034)
- Remove sendfile from .app.src

Release 0.68.0

Welcome to Zotonic 0.68.0, released on July 21, 2022.

This is a maintenance release.

Main changes are:

- Change z_search to return estimated number of rows instead of a count
- New zotonic config 'filewatcher_terminal_notifier' to disable desktop notifications
- Fix a problem in controller_static_pages with handling Unicode filenames
- Change signup-confirmation to require a click on a button before confirming, this fixes a problem where confirmations could accidentally be done.

Search changes

The search routines are now using the query planner to estimate the number of rows.

To support this, the following changes were made:

- The `all` field in the `#search_result` record is now deprecated.
- In `#search_result` there is a new flag `is_total_estimated`. This is set to `true` if the total was estimated by using the query planner.
- If the total is estimated then the pager does not show the last page.
- In the admin the (estimated) total items found is shown below the pager.

With this change some searches will be much faster as the database does not need to count to max 30K rows, like in the pre 0.68.0 implementation.

The 1.x version of Zotonic is also using the query planner to estimate the number of rows, so handling this correctly will help you in moving your site to Zotonic 1.x.

Commits since 0.67.0

Marc Worrell (10):

- mod_base: allow controller_static_pages to handle UTF-8 filenames. (#3038)
- core: error message on gettext po file problems.
- mod_facebook: use abs urls in head
- mod_admin: point query args doc to 0.x on Github

- core: add estimated row counts to search results. (#3041)
- mod_base: make next/prev pager is page count is undefined. (#3049)
- core: show more descriptive error on Ajax errors.
- mod_signup: always press a button to confirm.
- core: add 'all' to search_result.
- core: add config 'filewatcher_terminal_notifier' to enable or disable file update growl messages. (#3063)

Rob van den Bogaard (2):

- Merge pull request #3052 from zotonic/page-unloading-ajax-error
- Merge pull request #3053 from zotonic/signup-confirm-submit

Release 0.69.0

Welcome to Zotonic 0.69.0, released on August 30, 2022.

This is a maintenance release.

Main changes are:

- Security fixes by extra sanitization of redirect URLs in authentication and language switching
- Fix an information leak about possible usernames by adding more constant time password checks, even if there is no valid user.
- Update of jquery ui to 1.13.
- Show 2FA status in the user overview in the admin.
- Remove the APIs /api/base/persistent_get and /api/base/persistent_put.
- Option to add a SEO noindex to all pages of a certain category.

Incompatible changes

The persistent APIs are removed, as they could be used to store information on the server by anonymous users.

New configuration key

With the configuration key `site.email_images_noembed` it is possible to disable the embedding of images in emails. If this key is set to true value then the images will be URLs and downloaded from the webserver when the email is viewed.

This results in smaller emails, but the images might not be viewed by the recipient (as some email agents don't show image by default).

Commits since 0.68.0

Colin de Roos (1):

- Check for empty search string (#3074)

Marc Worrell (34):

- mod_seo: add all available hreflang and x-default links to head. (#3066)
- Set Docker files to Erlang 21.3
- Dockerfile: update keys

- Docker: add oidcc for docker images.
- docs: add note in search.rst about difference in syntax between tpl and stored query
- core: in emails, only embed images < 1MB. (#3082)
- core: fix site.email_images_noembed check
- core: fix an issue where the search_result page count was off by one.
- core: fix a problem where a search could result in illegal SQL
- Add 'delete username' button to change pw dialog (#3037)
- core: add option to exclude categories from the SEO indexing.
- Set docs ubuntu version to 20.04 - as 18.04 is being deprecated
- GH Action docs: upgrade ubuntu and python
- Add translations
- Update po files
- mod_base: upgrade jquery-ui to 1.13.1
- Merge pull request #3092 from zotonic/jquery-ui-1.13-0x
- Merge pull request #3089 from zotonic/seo-noindex-cat-0x
- Fix po files
- mod_base: add option to controller_template to set the http status code (0.x) (#3094)
- mod_menu: filter menu_is_visible now also checks for exists.
- mod_base: remove persistent_get and persistent_set services. Fix #3103 (#3104)
- core: add timer on pw checks against timing attacks for finding valid usernames (0.x) (#3097)
- mod_video: escape filename for ffmpeg preview cmd (#3114)
- mod_base: more restrictive CSP header in controller_file (0.x) (#3108)
- Filter the redirect args of logon/logoff to ensure that it is a local redirect (#3096)

Release 0.70.0

Welcome to Zotonic 0.70.0, released on September 16, 2022.

This is a maintenance release.

Main changes are:

- Option in mod_auth2fa to force setting 2FA on log on.
- Fix for sorting menus and category trees, this was caused by the earlier jQuery update.

Commits since 0.69.0

Marc Worrell (4):

- core: change lookahead limit of searches for better pager display.
- mod_auth2fa: add option to force setting 2FA before log on. (#3122)
- mod_auth2fa: small fixes.
- mod_menu: update nestedSortable.js (#3132)

Rob van den Bogaard (1):

- `mod_base`: Pager ellipsis link (0.x) (#3120)

Release 0.70.1

Welcome to Zotonic 0.70.1, released on September 26, 2022.

This is a patch release that fixes a problem where extra identities were not correctly added for repeated authentications with third-party services.

Commits since 0.70.0

Rob van den Bogaard (1):

- Fix identity set by type (#3138)

Release 0.71.0

Welcome to Zotonic 0.71.0, released on October 18, 2022.

This is a maintenance release.

Main changes are:

- Force 2FA for accounts with certain user groups (this used to be set to ‘nagging’).
- In the pager, round the number of estimated pages found. This prevents showing an arbitrary number that gives an exact impression. The number of pages is rounded to two significant digits.

Commits since 0.70.0

Marc Worrell (3):

- `mod_base`: add filter `round_significant` (0.x) (#3143)
- `mod_oauth2`: change the per UG 2FA setting to ‘3’ (force), instead of ‘2’ (nagging) (#3154)

Rob van den Bogaard (1):

- Fix identity set by type (#3138)

Release 0.72.0

Welcome to Zotonic 0.72.0, released on November 21, 2022.

This is a maintenance release.

Main changes are:

- New module `mod_cookie_consent`
- Better estimates for the number of found rows in searches

Commits since 0.71.0

Marc Worrell (5):

- Upgrade `z_stdlib`
- `core`: ensure that `lager` metadata is set for spawned processes and queries (#3167)
- `core`: in search allow undefined for `pagelen`. Issue #3171 (#3172)

- core: in sql use 'any' instead of 'select unnest' (#3181)
- mod_cookie_consent: implements cookie consent handling and settings (#3184)

Release 0.73.0

Welcome to Zotonic 0.73.0, released on December 8, 2022.

This is a maintenance release.

Main changes are:

- Fix an issue where the next was not set in m_search_result

Commits since 0.72.0

Marc Worrell (2):

- mod_cookie_consent: cosmetic changes (#3204)
- Fix an issue where the next was not set in m_search_result (#3206)

Release 1.0.0-rc.10

Released on 2022-10-31.

This is a release candidate for Zotonic 1.0.

The code is stable and used in production. Some features will be added before the 1.0 release.

For the issues and changes to be added for the final 1.0.0 release, see:

<https://github.com/zotonic/zotonic/milestone/3>

The 1.0.0 release is expected in Q4 2022.

Updating

After updating Zotonic, also update Cotonic with:

```
make update-cotonic
```

This release needs the newest (master) version of Cotonic.

For the 1.0.0 we will start tagging the Cotonic versions.

Erlang/OTP Version

This release needs Erlang/OTP 23 or later.

Commits since 1.0.0-rc.9

Marc Worrell (8):

- workflows: add manual trigger for publish workflow (#3149)
- Use ex_doc for the hex docs (#3150)
- mod_admin: add column published-on to overview list table (#3153)
- docs: add release notes 0.71.0

- `mod_translation`: change x-default url to use 'en' (#3159)
- Schema medium import (#3158)
- `mod_ssl Letsencrypt`: ensure that connections to LE are closed (#3160)
- `zotonic_mod_base`: fix for `controller_static_pages` with `{files, ...}` root

Michiel Klønhammer (5):

- New translations `zotonic.pot` (Russian) (#3147)
- New Crowdin updates (#3148)
- New Crowdin updates (#3151)
- New Crowdin updates (#3152)
- New Crowdin updates (#3155)

Radek Szymczyszyn (1):

- Fix `-include{,_lib}` attributes to unbreak doc chunk generation (#3146)

Release 1.0.0-rc.11

Released on 2022-11-11.

This is a release candidate for Zotonic 1.0.

The code is stable and used in production. Some features will be added before the 1.0 release.

For the issues and changes to be added for the final 1.0.0 release, see:

<https://github.com/zotonic/zotonic/milestone/3>

The 1.0.0 release is expected in Q4 2022.

Updating

After updating Zotonic, also update Cotonic with:

```
make update-cotonic
```

This release needs the newest (master) version of Cotonic.

For the 1.0.0 we will start tagging the Cotonic versions.

Erlang/OTP Version

This release needs Erlang/OTP 23 or later.

Commits since 1.0.0-rc.10

Marc Worrell (12):

- New pot
- `mod_admin_frontend`: support linking to specific tab in edit page (#3165)
- `mod_seo`: longer seo descriptions, `noindex` for `q.page > 1` (#3166)
- `core`: ensure metadata is set on spawns (#3168)
- `mod_filestore`: upgrade webdavfiles, fix upload with undefined mime (#3169)
- `mod_search`: support OR query for language search term (#3170)

- core: in search, allow pagelen to be undefined. Fixes #3171 (#3173)
- core: find site path for setup without umbrella apps (#3174)
- mod_video_embed: handle embed of youtube embed video links (#3175)
- mod_base: fix a problem with filter toc where a ‘>’ was shown before the texts (#3176)
- mod_filestore: truncate remote filenames (#3177)
- mod_seo_sitemap: exclude resources with a redirect from sitemap. Fixes #3157 (#3178)

Michiel Klønhammer (1):

- New Crowdin updates (#3163)

Release 1.0.0-rc.12

Released on 2022-11-17.

This is a release candidate for Zotonic 1.0.

The code is stable and used in production. Some features will be added before the 1.0 release.

For the issues and changes to be added for the final 1.0.0 release, see:

<https://github.com/zotonic/zotonic/milestone/3>

The 1.0.0 release is expected in Q4 2022.

Updating

After updating Zotonic, also update Cotonic with:

```
make update-cotonic
```

This release needs the newest (master) version of Cotonic.

For the 1.0.0 we will start tagging the Cotonic versions.

Erlang/OTP Version

This release needs Erlang/OTP 23 or later.

Commits since 1.0.0-rc.11

Marc Worrell (6):

- Upgrade zotonic_stdlib to 1.11.2
- mod_video_embed: fix a problem where embed could crash on illegal vimeo links (#3180)
- core: use SQL ‘any’ instead of ‘unnest’ (#3182)
- mod_base: fix postback test (#3183)
- mod_survey: set answer_user_id to user if not editing existing answer (#3185)
- Fix typo (#3186)

Release 1.0.0-rc.13

Released on 2022-12-05.

This is a release candidate for Zotonic 1.0.

The code is stable and used in production. Some features will be added before the 1.0 release.

For the issues and changes to be added for the final 1.0.0 release, see:

<https://github.com/zotonic/zotonic/milestone/3>

The 1.0.0 release is expected in Q4 2022.

Updating

After updating Zotonic, also update Cotonic with:

```
make update-cotonic
```

This release needs the newest (master) version of Cotonic.

For the 1.0.0 we will start tagging the Cotonic versions.

Erlang/OTP Version

This release needs Erlang/OTP 23 or later.

Commits since 1.0.0-rc.12

Dorien Drees (1):

- [Search] Refactor search view (#3196)

Maas-Maarten Zeeman (1):

- mod_wires: Remove 2 char limit on typeselect search (#3194)

Marc Worrell (9):

- doc: add 0.72.0 release notes
- mod_authentication: fix a problem with pw reset for 2FA enabled accounts (#3190)
- core: ensure that the media filename is always a binary (#3191)
- Update zotonic_stdlib, template_compiler (#3193)
- New pot file
- mod_base: add filter translation (#3195)
- Privacy billing (#3197)
- mod_admin: Add optional editing of the page's timezone (#3201)
- mod_base: Fix an issue where controller_page could enter in a redirect loop. (#3200)

Michiel Klønhammer (3):

- Update email_password_reset.tpl
- Merge branch 'master' of <https://github.com/zotonic/zotonic>
- Merge branch 'master' of <https://github.com/zotonic/zotonic>

Mikael Karlsson (1):

- mod_search: fix sorting for custom pivot (#3198) (#3199)

Release 1.0.0-rc.14

Released on 2022-12-23.

This is a release candidate for Zotonic 1.0.

The code is stable and used in production. Some features will be added before the 1.0 release.

For the issues and changes to be added for the final 1.0.0 release, see:

<https://github.com/zotonic/zotonic/milestone/3>

The 1.0.0 release is expected in January, 2023.

Updating

After updating Zotonic, also update Cotonic with:

```
make update-cotonic
```

This release needs the newest (master) version of Cotonic.

For the 1.0.0 we will start tagging the Cotonic versions.

Erlang/OTP Version

This release needs Erlang/OTP 23 or later.

Commits since 1.0.0-rc.13

MM Zeeman (1):

- core: Support wepb in image tag (#3226)

Marc Worrell (16):

- mod_admin: Better warnings and texts for time zone info. (#3207)
- New pot file
- docs: add 0.73 release notes.
- core: fix paging in m_search with payload (#3212)
- mod_search: add is_unfindable and custom query terms (#3213)
- New pot file
- Also update query to add '= true' for better index usage (#3216)
- mod_search_query: fix a problem with filtering facets (#3217)
- mod_admin: better user create email, fix race condition (#3218)
- New pot
- mod_search: fix a problem with joining search_facet table (#3220)
- mod_admin: add admin notes; mailinglist opt out (#3222)
- New pot
- core: allow send to recipient-id email address if recipient is visible. (#3227)
- mod_acl_user_groups: add notification to update an user's user groups on lookup (#3225)
- mod_mailinglist: allow multiple mailinglists subscribe forms on one page (#3223)

Michiel Klønhammer (4):

- New Crowdin updates (#3208)
- New Crowdin updates (#3214)
- New Crowdin updates (#3219)
- New Crowdin updates (#3224)

Release 1.0.0-rc.8

Released on 2022-09-13.

This is a release candidate for Zotonic 1.0.

The code is stable and used in production. Some features will be added before the 1.0 release.

For the issues and changes to be added for the final 1.0.0 release, see:

<https://github.com/zotonic/zotonic/milestone/3>

The 1.0.0 release is expected in Q4 2022.

Updating

After updating Zotonic, also update Cotonic with:

```
make update-cotonic
```

This release needs the newest (master) version of Cotonic.

For the 1.0.0 we will start tagging the Cotonic versions.

Erlang/OTP Version

This release needs Erlang/OTP 23 or later.

Commits since 1.0.0-rc.7

Dorien (1):

- Mod_admin: Add disconnect btn for page blocks (#3040)

MM Zeeman (2):

- Fixed the documentation, and refactored the code to make it more readable (#3021)
- Return not_found instead of no_session. Fixes #3057 (#3059)

Maas-Maarten Zeeman (2):

- mod_acl_user_groups: Also log the kind of acl rule which is published
- mod_wires: Prevent creation of global variables in zotonic-wired

Marc Worrell (81):

- core: add z_datetime:next_month/2 and z_datetime:prev_month/2. (#3013)
- core/oauth2: remove compile warning
- Remove OTP 22 from the CI test matrix (#3024)
- mod_search: adapt quick search presentation css (#3023)
- core: call middleware 'request' notification after cookies has been parsed (#3028)
- mod_seo_sitemap: update sitemap on pivot, remove prio 0 resources from urlsets (#3030)

- core: add `m_config:set_default_value/4`
- core: append name~<group> to make unique radio button groups with the same name. (#3031)
- docs: use test topic in examples.
- Fix an issue where `rebar3 app_discovery` could take a long time if there are many directories.
- core: fix a problem where the split of an email did not work if the email address did not have a domain. (#3033)
- core: fix a problem with splitting email addresses containing extended utf8 characters. (#3035)
- doc: add 0.67.0 release notes
- mod_admin: show estimated row count. (#3042)
- core: define min lookahead for searches (for correct pagination) (#3044)
- mod_l10n: Turkey is now called Türkiye (#3003)
- Rename Turkey to Türkiye in `country.pot`.
- mod_admin: fix a problem where the multi-lingual title of a page was lost on duplication. (#3047)
- mod_base: make next/prev pager is page count is undefined. (#3050)
- Upgrade `zotonic_ssl` to 1.3.0. This fixes an issue with LibreSSL where a self-signed key could not be generated. (#3051)
- mod_base: fix a problem where `controller_static_pages` could not handle unicode filenames (#3054)
- mod_admin: Fix a problem where the `pagelen` was reset if the sort was changed. (#3062)
- core: add config `'filewatcher_terminal_notifier'` to optionally disable terminal messages on file updates. (#3064)
- doc: add release notes of 0.68.0
- Update `rel_0.68.0.rst`
- mod_translation: include `hreflang` link to current translation (#3006)
- core: on `rsync` update, as `sudo`, do not allow to add edges. (#3067)
- core: add functions to add/sub N second/minute/hour/week/month/year to datetimes. (#3068)
- mod_base: fix spec of `z_list_of_ids_filter:filter` functions. (#3073)
- mod_oauth2: support client credentials grant type, misc fixes/changes (#3070)
- mod_oauth2: better error texts.
- mod_authentication: service error cancel cleanup
- mod_oauth2: added support for new error `'unexpected_user'`
- core: filter identities returned via the `m.identity` api. (#3076)
- Dockerfile: fix for untrusted keys.
- smtp: do not embed images > 1MB (#3083)
- core: fix an issue where the `search_result` page count could be off by one for certain searches. (#3085)
- core: fix a problem where a search could result in illegal SQL
- Merge pull request #3086 from `zotonic/sql-append-publish`
- GH Actions: upgrade ubuntu version for Docs action
- Merge pull request #3090 from `zotonic/gh-action-docs-ubuntu-upgrade`
- core: add option to exclude categories from the SEO indexing
- Merge pull request #3088 from `zotonic/l10n_master`

- Upgrade jquery-ui to 1.13.2
- Merge pull request #3093 from zotonic/jquery-ui-1.13
- Fixes from review by @robvandenbogaard
- Merge pull request #3091 from zotonic/seo-noindex-cat
- mod_base: add option to controller_template to set the http status code (#3095)
- mod_microsoft: change default scope, add comments. (#3100)
- core: add z_context:site_url/2 (#3099)
- mod_base: add csp_nonce to standard dispatcher args. (#3102)
- facebook/linkedin/ms: On connected services show an x in the disconnect button
- mod_menu: filter menu_is_visible now also checks for exists. (#3105)
- mod_translation: sanitize URI for controller_set_language (#3112)
- core: add timer on pw checks against timing attacks for finding valid usernames (#3098)
- mod_video: escape filename for ffmpeg preview cmd (#3115)
- mod_base: set the CSP for files to: default-src 'none'; sandbox (#3106)
- doc: add 0.69.0 release notes.
- mod_admin_identity: escape identity key
- mod_authentication: ensure that logoff redirect url is sanitized. (#3116)
- Upgrade template_compiler to 2.4.0
- mod_search: add array support to the search facets (#3113)
- core: change lookahead limit of searches for better pager display.
- Add crowdin to readme thanks
- Update links in readme
- Remove unused macro, update copyright
- mod_base: make the ... in the page clickable. (#3124)
- core: adapt markdown for utf8 and modern string functions. (#3126)
- mod_editor_tinymce: add zanchor plugin (#3127)
- mod_editor_tinymce: Restrict height of tinymce edit area.
- zotonic_core: fix for markdown conversion on Unicode strings with code points outside Ascii (#3128)
- zotonic_core: add page urls to the model/rsc/get/id result. (#3129)

Michiel Klønhammer (25):

- New Crowdin updates (#3015)
- New translations zotonic.pot (Portuguese, Brazilian) (#3026)
- New Crowdin updates (#3039)
- New Crowdin updates (#3043)
- New Crowdin updates (#3046)
- New Crowdin updates (#3048)
- New Crowdin updates (#3069)
- New Crowdin updates (#3075)
- New Crowdin updates (#3077)

- New Crowdin updates (#3078)
- New Crowdin updates (#3079)
- New translations zotonic.pot (Russian) (#3080)
- New Crowdin updates (#3081)
- New Crowdin updates (#3084)
- New translations zotonic.pot (Russian)
- New Crowdin updates (#3101)
- Update mod_auth2fa.erl
- Merge branch 'master' of <https://github.com/zotonic/zotonic>
- Update mod_auth2fa.erl
- Update _auth2fa_user_actions.tpl
- New Crowdin updates (#3107)
- New Crowdin updates (#3110)
- New translations zotonic.pot (Russian) (#3111)
- New translations zotonic.pot (Russian) (#3119)
- New Crowdin updates (#3121)

Release 1.0.0-rc.9

Released on 2022-10-12.

This is a release candidate for Zotonic 1.0.

The code is stable and used in production. Some features will be added before the 1.0 release.

For the issues and changes to be added for the final 1.0.0 release, see:

<https://github.com/zotonic/zotonic/milestone/3>

The 1.0.0 release is expected in Q4 2022.

Updating

After updating Zotonic, also update Cotonic with:

```
make update-cotonic
```

This release needs the newest (master) version of Cotonic.

For the 1.0.0 we will start tagging the Cotonic versions.

Erlang/OTP Version

This release needs Erlang/OTP 23 or later.

Commits since 1.0.0-rc.8

Marc Worrell (13):

- docs: release notes 0.70.0
- mod_authentication: add erljwt (#3134)

- `mod_oauth2`: fixes for fetching new certificates. (#3135)
- `mod_filestore`: add ftps support (#3137)
- docs: add 0.70.1 release notes
- `mod_zotonic_site_management`: add blocks templates to blog skel (#3109)
- `mod_backup`: less backups, upload to filestore (#3141)
- `mod_backup`: fix for periodic upload
- Upgrade `s3filez`
- `mod_base`: add filter `round_significant` (#3142)
- `mod_admin`: round estimated number of found rows in search results
- core: in logger show unicode binaries as strings (#3144)
- Upgrade `mqtt_sessions` to 2.1.0

Michiel Klønhammer (3):

- New Crowdin updates (#3136)
- New Crowdin updates (#3139)
- New Crowdin updates (#3145)

3.1.29 Upgrade notes

These notes list the most important changes between Zotonic versions. Please read these notes carefully when upgrading to a new major Zotonic version.

Upgrading from 0.x to 1.0

Before upgrading to 1.0, be sure to check the upgrade notes for the latest 0.x version, especially if you are upgrading from before 0.12. See [the upgrade notes on 0.x](#) for more information.

Erlang/OTP

Support for Erlang versions before 23 was dropped.

Zotonic modules are now separately published as packages to [Hex.pm](#), which allows you to build your own Zotonic distribution and to have each of your sites depend on the Zotonic modules (and other Erlang packages) it needs.

This was done by restructuring Zotonic into an [umbrella application](#). The `src/` directory was moved to new `zotonic_core` app.

Before:

```
-include_lib("zotonic.hrl").
```

After:

```
-include_lib("zotonic_core/include/zotonic.hrl").
```

The HTTP and SMTP listeners were moved to a new `zotonic_listen_http` and `zotonic_listen_smtp` app respectively.

A `zotonic_launcher` app was introduced for starting Zotonic.

All built-in modules, the `testsandbox` and the status sites are now to be found in the `apps/` directory.

Docker

Only the `zotonic/zotonic-dev` image is now available and automatically updated.

We have decided to drop the other Docker images as in practice everybody was creating their own production images anyway.

All files used by the Docker container are now placed in the `docker-data` directory.

See for more information [Docker](#) (page 14).

Sites and modules are now OTP apps

Both sites and modules now follow the standard [OTP directory structure](#), which means all Erlang files should reside in `src/` and all other files (templates, dispatch rules etc.) in `priv/`.

Before:

```
yoursite/
  models/
    m_some_model.erl
  templates/
    some_template.tpl
  yoursite.erl
  config
  ...
```

After:

```
yoursite/
  priv/
    zotonic_site.config
    templates/some_template.tpl
    ...
  src/
    models/m_some_model.erl
    yoursite.erl
    yoursite.app.src
    ...
  rebar.config
```

The `user_sites_dir` and `user_modules_dir` configurations have been removed. The default location for sites and modules is now the `apps_user` directory. With the `ZOTONIC_APPS` environment variable you can define an additional source directory outside the Zotonic umbrella `apps` directory.

To upgrade, move your `user/modules` and `user/sites` applications to the `apps_user` directory.

Resources

All resources are now *maps* with *binary* keys. Use of the previous 0.x proplists is deprecated, the fetch routines will always return maps, the update routines will convert property lists to maps before updating.

The `name_to_id_check/2` functions were removed from `m_category`, `m_predicate` and `m_rsc`.

Before:

```
Id = m_rsc:name_to_id_check(Value, Context) .
```

After:

```
{ok, Id} = m_rsc:name_to_id(Value, Context) .
```

Inserting or deleting an edge no longer modifies the last modified and modifier properties of the edge's subject resource.

There are extra access controls on rsc properties. The `privacy` property controls what is visible for whom.

The function `m_rsc:get_visible/2` has been removed. The function `m_rsc:get/2` now checks on visibility of properties. To fetch all properties, either use `m_rsc:get_raw/2` or call `m_rsc:get/2` as an administrator level user.

Media

The medium record is now a *map* with *binary* keys. Use of the previous 0.x proplists is deprecated, the fetch routines will always return maps, the update routines will convert property lists to maps before updating.

ACL

`mod_acl_adminonly` was replaced by *`mod_acl_user_groups`* (page 341). To create users that have access to the admin, add them to the 'Managers' user group.

The `visible_for` property semantics and the `acl_can_see` notification were removed. You can get similar functionality by adding users to user and collaboration groups. These are provided by `mod_acl_user_groups`. The `visible_for` rsc table property has been kept for BC. So if you're using `mod_acl_adminonly`, `mod_acl_simple_roles` or a custom ACL module you can still rely on the property.

The `acl_rsc_update_check` notification was removed.

Authentication

All auth notifications values were converted to records.

Before:

```
observe_auth_logon(auth_logon, Context, _Context) ->
```

After:

```
observe_auth_logon(#auth_logon{}, Context, _Context) ->
```

Configuration

Port configuration *environment variables* (page 94) were changed.

Before:

```
ZOTONIC_PORT=80 ZOTONIC_SSL_PORT=443 bin/zotonic start
```

After:

```
ZOTONIC_LISTEN_PORT=80 ZOTONIC_SSL_LISTEN_PORT=443 bin/zotonic start
```

Black/white-lists are now called block/allow-lists.

- `proxy_whitelist` is now `proxy_allowlist`
- `smtp_dnsbl` is now `smtp_dns_blocklist`
- `smtp_dnswhl` is now `smtp_dns_allowlist`
- `ip_whitelist` is now `ip_allowlist`

- `ip_whitelist_system_management` is now `ip_allowlist_system_management`

If an IP is on DNS allowlist then `z_email_dnsbl:status/2` returns now `{ok, allowed}`.

Errors

`m_edge`, `m_identity`, `m_rsc`, `m_rsc_import` and `m_rsc_update` no longer throw exceptions. Instead, they return an `{error, atom() }` tuple on failure.

Before:

```
m_edge:insert(Id, this_predicate_does_not_exist, UserId, Context).
%% crashes with an exception
```

After:

```
m_edge:insert(Id, this_predicate_does_not_exist, UserId, Context).
%% fails silently, so to make it crash:

{ok, _EdgeId} = m_edge:insert(Id, this_predicate_does_not_exist, UserId, Context).

%% alternatively:
case m_edge:insert(Id, this_predicate_does_not_exist, UserId, Context) of
  {ok, _EdgeId} ->
    "Everything fine!";
  {error, Reason} ->
    "Something went wrong!"
end.
```

Logging

See also:

Logging to Logstash (page 328)

The `lager` logger has been removed and replaced with the standard `erlang` logger application.

For this to work:

- Add the logger configuration to the `erlang.config` file. See *Logging* (page 80) for an example.
- Remove the `lager` definition from the `erlang.config` file.

Export

Modules `mod_atom` and `mod_atom_feed` were removed. You can export data in a variety of formats using *mod_export* (page 370).

JSON

Mochijson structures replaced with Erlang maps.

All JSON encoding/decoding now relies on JSX and goes through `z_json:encode/1` and `z_json:decode/1`.

`{trans, _}` tuples should now be unpacked by the client, before calling `z_json:encode/1` (previously `z_json:to_mochijson/2`).

Removed or deprecated functions

Deprecated functions have been removed from `z_utils`. Use the `z_url` and `z_json` modules instead.

Deprecated function `z_utils:name_for_host/2` has been removed; use `z_utils:name_for_site/2` instead.

The `{% stream %}` tag was removed, use MQTT websocket instead

Removed older TinyMCE versions 3.5.0 and 4.2.4.

`z_utils:combine/2` is removed, use `lists:join/2` instead.

`z_utils:combine_defined/2` is renamed to `z_utils:join_defined/2`.

Removed the `notify` action, use the MQTT topics instead.

Module schema and data initialization

The `#datamodel.data` field has been removed. The notifier `#manage_data` has also been removed.

Now the call to (the optional) `manage_schema/2` will be followed by a call to `manage_data/2`. Note that `manage_data` will be called if and only if you have a `manage_schema/2` function exported (and the `-mod_schema(..)` version changes or the module is installed).

The `manage_schema/2` function is called inside a transaction. The `manage_data/2` function is called after that transaction and also after all (optional) `#datamodel` changes are applied.

Templates

The `use_absolute_url` argument of the `url`, `image` and `lib` tags was renamed to `absolute_url`.

Templates are now stored in `yoursite/priv/templates/` instead of `yoursite/templates/`.

The `maxage` caching argument was renamed to `max_age`.

The models have now extra ACL checks.

The `m.config`, `m.site` and `m.sysconfig` models are only accessible as administrator. Use the models *owning* the various settings to access the configurations.

Exception is that the hostname and site-title information is publicly accessible using `m.site`.

Examples:

- `m.config.site.title.value` is now `m.site.title`
- `m.config.mod_editor_tinymce.version.value` is now `m.editor_tinymce.version`

Check the various models of the modules for the new lookups.

The `catinclude` for a resource with an unique name will not look for (assuming the unique name is `my_unique_name` and the template is `page.tpl`): `page.name.my_unique_name.tpl` and **not** anymore for `page.my_unique_name.tpl`. Rename your templates accordingly.

The category property `feature_show_address` property is now called `is_feature_show_address`. All feature properties should be called `is_feature_...` to obtain a proper boolean value after the category edit form is saved.

Port, proxies and SSL certificates

SSL/https support has been completely refactored.

- SSL self signed certificates have been moved into the core

- New modules *mod_ssl_ca* (page 395) and *mod_ssl_letsencrypt* (page 397)
- Deleted module *mod_ssl*
- Port configuration has been changed, see *Port configurations* (page 630)
- If you have a *priv/ssl* directory in your site, rename it to *priv/security*

For an overview of https support, see *HTTPS support* (page 89)

Erlang code, Controllers, Event handlers

If you made a site using custom controllers or request handling then you need to adapt your Erlang code. Zotonic is now using Cowboy under the hood for the http handling, previously this was MochiWeb.

The following changes are made:

- Binaries for all request variables and arguments.
- Events use binaries for strings in templates.
- Cookies are binaries.
- Request headers are binaries.
- Controllers initialization callbacks are removed.
- Controller callbacks have a single *Context* argument.
- Custom websocket handlers are removed, implement your own using Cowboy.
- The include file *include/controller_webmachine_helper.hrl* is removed (and not needed anymore).

Binaries for request variables

If you request an argument with *z_context:get_q/2* and related functions then you might need to adapt some code. Requesting a query argument using an *atom* or *binary* will return a *binary*. Requesting with a *string* returns a string, this is for backwards compatibility. The function *get_q_all* will return all arguments as binaries.

In short:

- *z_context:get_q(<<"arg">>, Context)* returns *<<"value">>*
- *z_context:get_q(arg, Context)* returns *<<"value">>*
- *z_context:get_q("arg", Context)* returns *"value"*
- *z_context:get_q_all(Context)* returns *[{<<"arg">>, <<"value">>}, ...]*

The binary name is the preferred way to request arguments.

Events like submit, postback and postback_notify

Strings in the *#submit{}*, *#postback{}* and *#postback_notify{}* events are now binaries. This is especially the case for the message, trigger, target, and form fields.

For example, replace *#submit{message="hello"}* with *#submit{message = <<"hello">>}*. Watch the space between = and the *<<" . . . ">>*, without the space you will get a syntax error.

Cookies

Use binaries for fetching and setting cookie names and values, don't use strings.

Request and response headers

All request and response headers now use binary names and values, do not use strings.

The request and response header names are normalized to lowercase names, so always use `<<"x-my-header">>` and *never* `<<"X-My-Header">>`.

The header values are passed as-is, and they are always binaries.

Controllers

The controllers are simplified and will need some adaptations.

The following callbacks are removed:

- `init`
- `ping`

All other callbacks have now a single *Context* argument, the *ReqData* argument has been removed. There is no need anymore for the `?WM_REQ` and `?WM_REPLY` macros, and they have been removed.

Other controller changes are:

- Content types are now binaries in *content_types_accepted* and *content_types_provided*
- Character sets are now binaries in *charsets_provided*
- Methods are now binaries in *allowed_methods* and *known_methods*
- Encodings are now binaries in *content_encodings_provided*
- The return value of *generate_etag* must be a binary

Search

Search argument *authoritative* was renamed to *is_authoritative*.

The *custompivot* has been removed. Pivot fields can now directly be addressed with `pivot.mypivotname.column`. Pivot tables are now joined automatically, removing the need for the *custompivot* search argument.

Notifications

The *admin_menu* notifications is now a tuple: `#admin_menu{}`. Update the *observe_admin_menu* functions in sites and modules.

Modules

Moved *mod_base_site* to https://github.com/zotonic/zotonic_mod_base_site

Menus

The storage format of the *menu* property is changed. Previously it was stored as a list of tuples:

```
{1234, [ {5678, [ ... ]}, ...]}
```

This doesn't allow for conversion to JSON, so the structure has been changed to use records:

```
#rsc_tree{ id = ..., tree = [ ... ] }
```

This allows for serialization to JSON using jsxrecord.

A collection of recipes that show you how to solve reoccurring problems.

4.1 Cookbooks

4.1.1 Admin cookbook

Creating a custom widget on the edit page

Why

For an imaginary webshop edit page, we want to add 2 more data fields: the affiliate URL and a note about free shipping.

Assumptions

Readers are expected to have experience with Zotonic templates. For reference, look at the Zotonic admin template directory `apps/zotonic_mod_admin/priv/templates/`.

Custom widget

Content widgets are created by creating a file `_admin_edit_content.my_cat.tpl`, where `my_cat` is the category name - webshop in our case.

Create a file `_admin_edit_content.webshop.tpl`:

```
{% extends "admin_edit_widget_std.tpl" %}
```

Extending will also give us a couple of default properties like `id` and `is_editable`.

We can also override `admin_edit_widget_i18n.tpl` if fields need to get translated.

Add these block definitions:

```
{% block widget_title %}
    { _ Webshop Properties _ }
{% endblock %}

{% block widget_show_minimized %}false{% endblock %}
{% block widget_id %}edit-webshop-properties{% endblock %}

{% block widget_content %}
    <fieldset class="form-horizontal">
        <div class="form-group row">
            <label class="control-label col-md-3">{ _ Affiliate URL _}</label>
            <div class="col-md-9">
                <input type="text"
                    name="affiliate_url"
                    class="form-control"
                    value="{{ id.affiliate_url }}"
                    {% if not is_editable %}disabled="disabled"{% endif %}
                />
            </div>
        </div>
        <div class="form-group row">
            <label class="control-label col-md-3">{ _ Free shipping note _}</label>
            <div class="col-md-9">
                <input type="text"
                    name="free_shipping_note"
                    class="form-control"
                    value="{{ id.free_shipping_note }}"
                    {% if not is_editable %}disabled="disabled"{% endif %}
                />
            </div>
        </div>
    </fieldset>
{% endblock %}
```

When loading a webshop edit page, the widget should now appear below the Basics block.

To customize the edit page further, see *Customizing the layout of the admin edit page* (page 268).

Word of caution

Admin templates will evolve over time. Custom changes you make now may not work after an update. Use these guidelines at your own risk.

Customizing the layout of the admin edit page

Why

After having created a custom widget (see *Creating a custom widget on the edit page* (page 267)), we want to hide widgets that we don't need.

Assumptions

Readers are expected to have experience with Zotonic templates. For reference, look at the Zotonic admin template directory `apps/zotonic_mod_admin/priv/templates/`.

Page structure

To hide default widgets from the page, we need to change the template that loads these widgets.

The edit page widgets are grouped into 2 main areas, main and sidebar. These groups are created by the files `_admin_edit_main_parts.tpl` and `_admin_edit_sidebar_parts.tpl`.

Content parts

We won't change the default Zotonic edit template; instead we will override it by adding the category to the filename.

Create a file `_admin_edit_main_parts.webshop.tpl` and give it the contents:

```
{% all catinclude "_admin_edit_basics.tpl" id is_editable=is_editable_
↪languages=languages %}
{% all catinclude "_admin_edit_content.tpl" id is_editable=is_editable_
↪languages=languages %}
{% include "_admin_edit_content_advanced.tpl" %}
```

This template will now load the Basic widget (Title, Summary, Short title), all content widgets (for now only `_admin_edit_content.webshop.tpl`), and the default Advanced widget.

Sidebar

For the sidebar area you can follow the same procedure. Of course you need to keep the submit buttons, so always include `_admin_edit_content_publish.tpl`.

Category and instance

To show a block on a category edit page, but not on an instance of that category, check for `id.is_a.meta`. To show the block “Features” on the category page we write:

```
{% if id.is_a.meta %}
{% include "_admin_edit_meta_features.tpl" %}
{% endif %}
```

Word of caution

Admin templates will evolve over time. Custom changes you make now may not work after an update. Use these guidelines at your own risk.

Customizing the style of an admin page

How to make style customizations to admin pages.

Assumptions

Readers are expected to have experience with Zotonic templates.

How

Creating a custom style for admin pages is best done using a module:

- Create a directory for your skin, for instance `apps/mod_skin_admin`.
- Populate the module directory with `src/mod_skin_admin.erl`, `src/mod_skin_admin.app`, `src` file, sub directories `priv/lib/css/` and `priv/templates/` (see also: *Directory structure* (page 60)).
- Put your custom CSS file (f.i. `skin-admin.css`) in `priv/lib/css/`.
- Create a template file `_html_head_admin.tpl` with the contents `{% lib "css/skin-admin.css" %}`.
- Enable the module in admin.

Todo

reference to making a module using rebar3 templates (`src/mod_skin_admin.erl`, `src/mod_skin_admin.app.src`)

Storing date/time fields

Some interesting tidbits about saving/updating a date/time field of a resource.

Why

The purpose of this guide is to explain how Zotonic stores date-time data so that you can work with it in your own modules.

Assumptions

Readers are expected to be interested in authoring Zotonic Modules. To be successful you should be fairly experienced with Erlang programming.

How

When updating a *resource* by passing a proplist to `m_rsc:update/3`, Zotonic has a way of looking for specific name patterns in the proplist that tells it what items are date/time fields and which ones are not. This allows Zotonic to split those off the prolists and use the name to derive information about the date, such as expected format, if the default time should be 00:00 or 23:59, and the actual field name that should be updated.

Here is an example format: `dt:ymd:0:date_end`

Literally this means ...

- the characters `dt:`, signifying the date-time marker
- followed by some year, month, date pattern or whatever combination
- followed by `”:`
- followed by 1 or 0 to designate default 00:00 or 23:59
- followed by `”:`
- followed by the field name to be updated.

Have a look at `_admin_edit_date.tpl` in the default site templates to see this and other formatting options.

For example passing `[{<<"dt:ymd:0:date_end">>, <<"2011-04-05">>}]` to `m_rsc:update/3` would update the `date_end` field.

Here is an example event callback:

```
event(#submit{message={edit_details, [{id, Id}]}, Context) ->
  End_date = z_context:get_q(<<"dt:ymd:0:date_end">>, Context),
  Props1 = [
    {<<"dt:ymd:0:date_end">>, End_date}
  ],
  case m_rsc:update(Id, Props1, Context) of
    {ok, _} ->
      z_render:wire({reload, []}, Context);
    {error, _} ->
      z_render:growl_error(<<"Could not update record.">>, Context)
  end.
```

Of course the input field of your form could be named whatever you wish, but when you assign it to the proplist you would use the naming convention above.

Internals

The *date recombination* is done if and only if the input for the update (or insert) operation is a list with key/value pairs.

It is assumed the list is from a form post. Form inputs are often separate for the date and the time parts and need to be recombined to full dates.

This recombination process is done in the `z_props` module on conversion from the list with key/value pairs to a (nested) map.

Admin template specific things

Common markup in admin templates.

Linking to edit pages

Dispatches to edit pages is done by `admin_edit_rsc`.

Linking:

```
{% url admin_edit_rsc id=my_id %}
```

Redirecting:

```
{% button text="edit" action={redirect dispatch="admin_edit_rsc" id=my_id} %}
```

Automatically add new users to a user group

Why

Note: If you merely want to change the default user group for *all* users, you can do so through a *configuration parameter* (page 393) and you don't need the solution below.

When you create a person, you usually need to add it to a user group as well. You may want to automate this, in particular if you need to differentiate the user group based on the person's category.

Solution

You can set the user group by adding a `hasusergroup` property to the resource. In your site's main `.erl` file, add a `rsc_update` observer:

```
-export([
    observe_rsc_update/3
]).

observe_rsc_update(#rsc_update{action = insert, id = Id}, {ok, Props} = Acc, _
    ↪Context) ->
    %% Where 'vip' is a subcategory of 'person'
    case m_rsc:is_a(Id, vip, Context) of
        false ->
            %% Do nothing
            Acc;
        true ->
            %% Add hasusergroup property
            {ok, Props#{ <<"hasusergroup">> => m_rsc:rid(acl_user_group_vips, _
    ↪Context)}}
    end;
observe_rsc_update(#rsc_update{}, Acc, _Context) ->
    %% Fall through
    Acc.
```

4.1.2 Frontend cookbook

These cookbook entries contain valuable nuggets of information regarding the frontend development of a site.

Implementing a simple contact form

This tutorial teaches you to create a form, validate it, submit it over Ajax and e-mail the results back to you.

Why

Making a simple contact form might seem difficult, but with the smart application of different Zotonic techniques you'll see that it's actually very easy.

1. Create the contact page URL dispatcher and template
2. Create the contact form
3. Create the contact-form handler Erlang file.

Assumptions

Readers are assumed to be comfortable both in developing Zotonic templates and in writing Erlang modules.

How

Create the contact page URL dispatcher and template

Note: Your actual site location might be different, see the *Zotonic user directory*.

The URL dispatcher is placed in `apps_user/your-site/priv/dispatch/dispatch`. Add this line:

```
{contact_url, ["contact"], controller_template, [ {template, "contact.tpl"} ]},
```

This says that the page at “/contact” will use the “contact.tpl” template. Let’s create this template, at `apps_user/your-site/priv/templates/contact.tpl`:

```
{% extends "base.tpl" %}

{% block content %}
<h1>Contact page</h1>
{% endblock %}
```

Now we have this, let’s try to see if it loads. Flush the Zotonic cache (to refresh the URL dispatchers) by going to “modules” -> “rescan modules” in the admin. Now, point your browser to <http://your-site:8000/contact>. This should show the contact page with the template you just made.

Create the contact form

Now you should write the actual contact form. You should decide what fields you want in the form, so for now, just put a name, e-mail and comment field:

```
{% wire id="contact-form" type="submit" postback={contact} delegate="my_contactform" %}
<form id="contact-form" method="post" action="postback">

  <label for="name">Name</label>
  <input type="text" name="name" id="name" />

  <label for="email">E-mail</label>
  <input type="text" name="mail" id="mail" />
  {% validate id="mail" type={presence} type={email} %}

  <label for="message">Message</label>
  <textarea name="message" id="message" cols="60" rows="8"></textarea>
  {% validate id="message" type={presence} %}

  <input type="submit" value="Send" />
</form>
```

This form has 3 fields, of which the message and the e-mail are required, and the e-mail input has to contain a valid e-mail address. The name field is optional.

Create the contact-form handler Erlang file

As you see in the *wire* (page 551) statement in the contact form, the *delegate* argument is set to `my_contactform`, which is the name of an erlang module which we still have to create. When the form submits, this module’s `event/2` function gets called. Create a file `apps_user/your-site/src/support/my_contactform.erl` with the following contents:

```
-module(my_contactform).
-export([event/2]).

-include_lib("zotonic_core/include/zotonic.hrl").

event(#submit{message={contact, []}}, Context) ->
```

```
?DEBUG(z_context:get_q_all(Context)),
Context.
```

This is the most simple version of a Zotonic form-handling function: it just dumps all form input it gets to the Zotonic console (using the `?DEBUG` macro). To compile this Erlang module, enter the following on the zotonic console:

```
z:m().
```

You'll see a notice that the file is recompiled, which should end with a `ok` message to indicate the success. This compiling is actually very important: Whenever you change the `.erl` file, you'll need to recompile it using this command.

E-mail the contents of the contact form to somebody

See also:

Sending E-mail (page 76)

Using Zotonic's email module, you can very easily send somebody an e-mail. Let's create a simple template to send the contents of the form to the site administrator.

Create the file `apps_user/yoursite/priv/templates/_email_contact.tpl`:

```
<html>
  <head>
    <title>Contact form</title>
  </head>
  <body>
    <p>Hello, the contact form of the site has been submitted.</p>
    <p>Name: {{ name|escape }}</p>
    <p>E-mail: {{ mail|escape }}</p>
    <p>The contents of the message was this:</p>
    <pre>{{ message|escape }}</pre>
    <p>Regards, your website.</p>
  </body>
</html>
```

This template will function as the message body that will be sent. Note: this template gets scanned for the `<title>` tag, which will double as the e-mail's subject, so be sure to include it!

Now we have to change our `event/2` function to render this template and e-mail it using `mod_emailer`. Change the event function to the following:

```
event(#submit{message={contact, []}}, Context) ->
  Vars = [{mail, z_context:get_q(<<"mail">>, Context)},
    {name, z_context:get_q(<<"name">>, Context)},
    {message, z_context:get_q(<<"message">>, Context)}],
  z_email:send_render(z_email:get_admin_email(Context), "_email_contact.tpl", Vars,
    Context),
  z_render:update(
    <<"contact-form">>,
    <<"<p>The form has been submitted! Thank you, we'll get in touch soon.</p>">>
    Context).
```

This loads the relevant values from the form, puts them in the `Vars` variable, and then calls the `z_email` module to mail the given template to the e-mail address of the site admin (which is defined in your site's config file). For more information on sending mails from Zotonic, please see the `mod_emailer` documentation.

Finally, this contact-form handler replaces the contact form with a `<p>` tag with a success message, using the `z_render:update` function.

How to add a custom Content Block

Zotonic comes with a number of standard content blocks: Header, Text and Embed page. Additional content blocks are provided by modules, for example `mod_survey` uses content blocks extensively for composing surveys. Content blocks allow a content manager to add sophisticated sections of content, perhaps with configuration options. They could be used for design reasons e.g. for multi-column layouts, image carousels or floating elements. This cookbook item describes how to add a simple custom content block.

In order to add a custom content block, we need to register it, by editing an erlang file and adding two templates. For this tutorial we'll add a content block which inserts the code for a music player from jamendo.com, a Creative Commons music hosting site.

To register the content block we add an `observe_admin_edit_blocks/3` function to the site's erl file e.g. `apps_user/mysite/src/mysite.erl`:

```
%%=====
%% support functions go here
%%=====

-export([
    observe_admin_edit_blocks/3
]).

observe_admin_edit_blocks(#admin_edit_blocks{}, Menu, Context) ->
[
    {100, ?__("Media", Context), [
        {music_player, ?__("Music Player", Context)}
    ]}
    | Menu
].
```

The interesting parts are "Media", "Music Player" and `music_player`. "Media" will appear as a section heading in the [+add block] drop down menu, below the Standard section. "Music Player" will appear below this, and is what the content editor clicks on to insert the block. The `music_player` atom is what Zotonic uses to figure out which templates to use, they will be created in `templates/blocks` and will be called `_admin_edit_block_li_music_player.tpl` and `_block_view_music_player.tpl`.

As the name suggests, `_admin_edit_block_li_music_player.tpl` is the template used on the edit form:

```
{% extends "admin_edit_widget_i18n.tpl" %}

{% block widget_title %}{_ Music Player _}{% endblock %}
{% block widget_show_minimized %}false{% endblock %}
{% block widget_id %}edit-block-music-player{% endblock %}
{% block widget_header %}{% endblock %}

{% block widget_content %}
    <div class="control-group">
        <label class="control-label">URL of the music widget</label>
        <div class="controls">
            <input type="text" id="block-{{name}}-url"
                name="block-{{name}}-url"
                value="{{ blk.url }}"
                class="input-block-level"
                placeholder="{_ Enter the url of the music widget _}"
            >
        </div>
    </div>
{% endblock %}
```

For the purpose of demonstration, we extend the `admin_edit_widget_i18n.tpl` template and use some template blocks.

Each time this block is added to the page, the form above will be displayed. Here, we want to allow users to easily embed a music widget from Jamedo, an online music community. The way to do this is by adding extra input fields to the page through the block templates.

Please note the values of the attributes of the input fields in the block template. For example, `name="block-{{name}}-url"` will expand to `name="block-mus56h-url"`. The `name` variable is a randomly generated string that is unique within every block template. The last portion of the value of the attribute is the attribute to be added to the resource. In this case, the attribute to be added will be `url`.

Also note the value of the input's value attribute. This ensures that if the attribute has already been saved against the resource being edited, then the input will be filled with the saved value. Without this, the data entered will not be saved.

This is all you need to be able to add a block to the edit form. If you update your site and restart, you should now be able to select the new block. It just doesn't display anything yet so let's add `_block_view_music_player.tpl`:

```
<iframe id="widget" scrolling="no" frameborder="0" width="400"
        height="284" style="width: 400px; height: 284px;"
        src="{{ blk.url }}"></iframe>
```

In the block's frontend view template, a special variable called `blk` is available. This `blk` variable holds the attributes or properties of the block. Since we added only one attribute, `url`, to the block's admin template, the `blk` variable will hold only two properties: `name`, and our custom attribute, `url`.

So if the user supplied `http://widgets.jamendo.com/v3/album/40728?autoplay=0&layout=standard&width=400`, as the url to the music widget, the block's frontend view template will expand to:

```
<iframe id="widget" scrolling="no" frameborder="0" width="400"
        height="284" style="width: 400px; height: 284px;"
        src="http://widgets.jamendo.com/v3/album/40728?autoplay=0&layout=standard&
↵width=400">
</iframe>
```

We can extend this custom block to allow the user to specify the widget's height, width, and frameborder.

All you have to do is add new input fields in the block's admin template.

Updating form field from a dialog

Ever wanted to update a form field from a dialog, possibly giving the user some list to choose from? Here's how to do it.

Why

Some interactions benefit from presenting a dialog for input. This guide explains how to take input in a dialog and have it apply to a form input value.

Assumptions

Readers are expected to understand DOM IDs for uniquely identifying form inputs and also understand advanced usage of Zotonic Template tags.

How

To be able to update a form field, you need to pass the id of the element you want to update to the dialog. It may look something like:

```
{% button text="Update field dialog"
  action={dialog_open
    title="Update form field value"
    template="my_dialog.tpl"
    target_id="input_element_id"}
%}
```

Then create your dialog template, `my_dialog.tpl` in this example, and wire a `set_value` action to update the input form element:

```
<a id="my_anchor" href="javascript:void(0)" href="javascript:void(0)">Click here_
↳to update form value</a>
{% wire id="my_anchor"
  type="click"
  action={set_value target=target_id value="My new value"}
  action={dialog_close}
%}
```

See also:

Auto-generated identifiers (page 58)

If you include a template like the one above into your dialog template many times (i.e. from a for loop), then having fixed ids are no good. To prefix the id with a unique value (per invocation of the template) prefix the id with a `#`-sign. so the a-tag becomes `` and the wire becomes `wire id=#my_anchor...` which will expand to something like `"ubifgt-my_anchor"`.

Enabling Growl Notifications

See also:

growl (page 427), *lib* (page 528)

Using growl outside admin requires some magic to make it work.

Why

Growls provide an unobtrusive way of notifying users of background events or the completion of tasks. This guide provides step-by-step instructions on how to enable it for your site.

Assumptions

Readers are expected to be familiar with template editing.

How

Although the magic is quite simple. The missing pieces are a couple of client side scripts, `/lib/js/modules/z.notice.js` and `/lib/css/z.growl.css` from *mod_base* (page 360).

In your `base.tpl` template, include these files using the *lib* (page 528) tag:

```
{% lib "js/modules/z.notice.js" %}
```

And the CSS:

```
{% lib "css/z.growl.css" %}
```

Now you should be able to use growl actions in your templates, example:

```
{% button action={growl text="hello world"} %}
```

Add Chat to Your Zotonic Site

Thanks to Michael Connor's `zchat`, it's easy to add chat system on your Zotonic site.

Why

It is often very useful for a site to have a live support line built-in. Another case for chat is on a social site where you'd like your members to be able to communicate in real time.

Assumptions

Readers are assumed to have a working Zotonic system running, and Mercurial (the `hg` command) installed.

How

Install the `mod_chat` module using `zotonic modules install`:

```
$ zotonic modules install mod_chat
```

This should download `mod_chat` and install it in the `priv/modules/` directory. Restart and rebuild zotonic to see the effect of these changes:

```
$ cd /home/zotonic/zotonic
$ zotonic stop
$ make
$ zotonic start
```

Now, log into your site admin page, select *System* and then *Modules*. Scan down to find `mod_chat`. Activate it.

Now you can either include chat into an existing page on your site or create a new page.

To include chat in an existing page, find the template that formats the page in the templates directory and somewhere within the content block enter:

```
{% include "_chat_box.tpl" %}
```

To create a new page:

Create a page called "Chat" with category Text. Give it the Unique name `page_chat`, click on the Published check box, and save.

Add the following dispatch rule to your dispatch list:

```
{chat, ["chat"], controller_page, [ {template, "chat.tpl"}, {id, page_chat} ]},
```

If this is the last rule in your list, omit the trailing comma.

Go to *System* and *status* in your admin menu and click on *rescan modules*.

Now you should be able to go to `<yoururl>/chat` and see your chat window. You'll see *Anonymous <long number>* in the right-most pane. That's you. If your chat buddy enters the chat page, a second Anonymous

<another big number> will appear. When you or your buddy leave the chat page, the respective Anonymous tag will disappear. Note that chat messages are not stored or logged. That may come with a future version of zchat.

Page-specific dynamic backgrounds

Use edges (*page connections*) to associate backgrounds with pages.

Contributed by: Dmitrii Dimandt

Why

This is a small use case with a lot of words pouring into it.

I'm doing a very small project for a couple of friends. It currently resides on <http://rusukr.dmitriid.com/> and will soon move to a proper domain.

What the project needed was different backgrounds for different pages. Compare, for instance, <http://rusukr.dmitriid.com/products> and <http://rusukr.dmitriid.com/investments>

Assumptions

Readers are expected to be familiar with where templates reside and how things like conditional tags work. To benefit most you should also be comfortable getting around and editing items in Zotonic's CMS admin interface.

How

Now, the most obvious way to implement these backgrounds would be to attach a media item to a page and something along the lines of:

```
{% if id.medium %}
  {% media id.medium %}
{% endif %}
```

Actually, if you create your site using the blog skeleton, you will see similar code in *_body_media.tpl*. Obviously, such an approach has a severe downside. What if you actually want to attach other media to the page? What if you want to display images and video in your text, not just a background?

This is where custom predicates come into play. Don't fret! it's much easier than you think

Predicate is nothing but a fancy way to say "relationship". Predicates define relationships between various places (objects, entities) within Zotonic. If you are into graphs and graph theory, a predicate is a directed labeled edge from one node to another (see Marko Rodriguez's amazing presentations on (among other things) graph theory and graph algebra here: <http://www.slideshare.net/slidak/presentations>).

What a predicate does, it defines how objects relate to other objects. Since predicates are (usually) one-way only, you helps to think about them as follows:

if I have a parent, how do I discern between its children? Or how do I define its children? if I have an object, how can I have different collections of objects related to it? So, if you have an article A, then you can have:

- a number of tags describing the article
- several authors who collaborated on it
- a few images to go with the article

Let's rephrase that:

- an article has tags
- an article has authors

- an article has images

Once you can make such a “has” connection, you can use a predicate:

```
article -----tag predicate---> tag1, tag2, tag3, etc.
article -----author predicate---> author1, author2, author3, etc.
article -----image predicate---> image1, image2, image3, etc.
```

Easy, isn't it?

The greatest thing about predicates is that you can define your own. Here's how you do it.

- Go to the “Predicates” section (under *Structure*) and click “Make a new predicate”.
- Name it “Background”, since its used to signify a dynamic background on a page.
- Click “Make predicate”.

On the following screen we have to choose the direction of our predicate. “From” is the entity that can have relations to other objects, and “To” is the entity that can be had . So, our Text has an Image as a background. This means that “From” is Text and “To” is Image. Click the corresponding checkboxes and then click save.

you're done!

Now if you create a new text item (an article or news) you will see that in the Page Connections section there's now a new option, “Background”. Let's try and make a new background:

- Go to Pages
- Click “Make a new media item”
- Name this item, for example, “test”
- Select an image to upload from your computer
- Click “Upload file”
- The page that opens is irreleveant to us right now. Just go back to “Pages”.
- Click “Make a new page”
- Name this page, for example, “test page”
- Click on “Background” in “Page connections section”
- Type in “test”
- Select “test” media item
- Click “Save and view”

You should now see your test page... with no background on it. This is ok, since we haven't told Zotonic how we want to display our background. To do this we have to edit a template. Let's try and display the background image first. First, we can make a checklist to see how we should proceed:

- We need to make sure the background exists
- We need to retrieve the resource associated with the background
- We need to retrieve path to the image stored on the server
- We need to make sure that this path is actually accessible from a web browser

The last step is especially important since Zotonic stores uploaded files in a directory that is not directly accessible from the web. However, for images we can use the *image_url* (page 526) tag to circumvent that.

So, the code to go with our checklist is as follows:

```
<div{% if id.background %}{# we check to see if background exists #}
  style="background: url( {% image_url id.background.medium.filename %}{# output_
↪web accessible URL to the image #} ) no-repeat"
  {% endif %}>
```

```
 
</div>
```

Now that we know how to retrieve the background image we can use it to our advantage. Our dynamic background will now look something like this:

```
<div{% if id.background %} style="background: url({% image_url id.background.
↳medium.filename %})" {% endif %}>
   
</div>
```

Retrieving the category of a page

Getting the category from a URL is somewhat involved, but not impossible. This is an example of what you can do with filters.

Why

It is often useful to use a page's category to present it on the page itself or to look up related content. This guide provides step-by-step instructions for getting a page's category in a template.

I have a page with url of the form /my_category, /my_category/:id, or /my_category/:id/:slug. How can I retrieve the category from the url?

Assumptions

Readers are assumed to be comfortable with template development.

How

Since category is a direct property of m_rsc we can directly access it:

```
{% with id.category as my_cat %}
  ...
{% endwith %}
```

Share variable binding across blocks

How to avoid having to call the same query inside several blocks of the same page

Why

In some situations, you may have to use the same query result inside several blocks of the same template. For instance, you may need to use it in the body of a page and in its JavaScript block. Intuitively, you would program your template as shown in these scripts:

```
{# base.tpl #}
<html>
<head>...</head>
<body>
{% block html_body %}
  <!-- This is the default body -->
{% endblock %}
```

```
</body>

<script>
{% block js %}
    // This is the default js
{% endblock %}
</script>

</html>
```

And a page:

```
{# mypage_1.tpl #}
{% extends "base_1.tpl" %}

{# THIS won't WORK BECAUSE %with% AND %block% TAGS CANNOT BE NESTED THIS WAY #}
{% with m.mymodule.myquery as myresult %}

{% block html_body %}
    Here is your result: {{ myresult|escape }}
{% endblock %}

{% block js %}
    alert('Do something with your result: {{ myresult|escapejs }}');
{% endblock %}

{% endwith %}
```

Unfortunately, this doesn't work, because Zotonic doesn't allow you to place a `{% with %}` tag around all `{% block %}` tags of *mypage_1.tpl*.

One solution consists in duplicating the `{% with %}` tag and moving it inside each block:

```
{% extends "base_1.tpl" %}

{% block html_body %}
    {% with m.mymodule.myquery as myresult %}
        Here is your result: {{ myresult|escape }}
    {% endwith %}
{% endblock %}

{% block js %}
    {% with m.mymodule.myquery as myresult %} {# AGAIN THE SAME QUERY #}
        alert('Do something with your result: {{ myresult|escapejs }}');
    {% endwith %}
{% endblock %}
```

However, this has a severe drawback: the same assignment will be done twice, and if it comes from an “expansive” database query, this query will be performed twice (or the query result will have to be cached, which increases the complexity of your system significantly).

Assumptions

Readers are assumed to be comfortable with the template syntax and with template inheritance, specifically the `{% extends %}` and `{% block %}` tags.

How

The solution is to move both blocks to a new template, *_html_body_and_js.tpl*:

```
{# _html_body_and_js.tpl #}

<body>
{% block html_body %}
    <!-- This is the default body -->
{% endblock %}
</body>

<script>
{% block js %}
    // This is the default js
{% endblock %}
</script>
```

Our base template includes the new template so its contents will be drawn on the page:

```
{# base.tpl #}

<html>

<head>...</head>

{% block html_body_and_js %}
    {% include "_html_body_and_js.tpl" %}
{% endblock %}

</html>
```

Now we make sure that `_html_body_and_js.tpl` gets data. In a base subtemplate `smart_base.tpl` we override the block with the query:

```
{# smart_base.tpl #}

{% extends "base.tpl" %}

{% block html_body_and_js %}
    {% with m.mymodule.myquery as myresult %}
        {% include "_html_body_and_js.tpl" %}
    {% endwith %}
{% endblock %}
```

The page that needs the data extends `smart_base.tpl`. Variable `myresult` is now accessible and can be used in both blocks:

```
{# mypage.tpl #}
{% extends "smart_base.tpl" %}

{% block html_body %}
    Here is your result: {{ myresult|escape }}
{% endblock %}

{% block js %}
    alert('Do something with your result: {{ myresult|escapejs }}');
{% endblock %}
```

Customizing the sign up and sign in form

You want to change parts of the form, or change its appearance.

Sign up form

The sign up form is called with dispatch rule `signup`, implemented by *mod_signup* (page 393). This module must be enabled to view the form.

The form is built from quite a number of sub templates. While this creates some level of complexity (to find out what goes where), this also provides the flexibility to change the resulting output without breaking the functional code.

For details, see “Template structure Sign up form” below.

Overriding templates

Some sub templates are more critical than others.

While it is technically possible to override the sub templates, this will likely end up with non-working pages. The Page Controller expects to receive certain values from the submitted form.

These form fields are expected to have a value:

```
email
name_first
name_surname
username
password1
signup_tos_agree
```

If you want to hide a part of the form that contains a critical value, for instance the terms and conditions checkbox, you cannot just create an empty file `_signup_form_fields_tos.tpl` in your project, because the form value `signup_tos_agree` will no longer be passed.

Instead you need to pass the value through the form using a hidden input field:

```
<input type="hidden" name="signup_tos_agree" value="1" />
```

Non-critical templates can be overridden safely. For instance `_signup_title.tpl` will only change the title above the form, so this can be done without affecting the functionality. You could decide to add a brand logo to the title, or remove the title altogether.

Changing the configuration

The template file `_signup_config.tpl` is made to quickly define what parts of the sign up page should be shown. Copy the file from `mod_signup` to your project and change the values.

For convenience, critical variables are not exposed in the config file.

By default, two form parts are set to hidden. Make them visible by changing the value from 0 to 1:

```
show_signup_name_prefix      // surname prefix ("van" Gogh)
show_signup_password2        // repeat password
```

Two other configuration values deal with appearance: `style_boxed` and `style_width` - see below.

Changing appearance

Form fields are displayed using the Bootstrap 3 HTML structure. If you need a radical different layout that cannot be managed with CSS alone, you will need to override the form field templates.

The width of the form can be set with configuration value `style_width`. This should be a CSS value like “300px”. If no value is set, the form will expand to maximum width - in which case the width needs to be set by CSS.

A form background can be set with configuration value `style_boxed`.

CSS styles are defined in `css/logon.css` (`mod_authentication`).

Password length

The minimum password length is 8 characters, and can be changed in config value `mod_authentication.password_min_length`.

Sign up form in a modal dialog

Instead of directing the user to the sign up page, the form can be shown in a modal dialog:

```
<a id="{{ #signup }}" href="#">{ _ Sign up _ }</a>
{% wire
  id=#signup
  action={
    dialog_open
    title="Sign up"
    template="signup.tpl"
  }
%}
```

However, this does not work well if the user wants to switch to sign in: clicking the link will load the sign in page, removing the dialog.

A better approach is to use the sign in templates and pass `logon_state`:

```
<a id="{{ #signup }}" href="#">{ _ Sign up _ }</a>
{% wire
  id=#signup
  action={
    dialog_open
    title="Sign up"
    template="logon_modal.tpl"
    logon_state="signup"
  }
%}
```

If the user now clicks on the link to sign in, the new form is shown in the same dialog.

Sign in form

For a general understanding of the templates and configuration, first read the section on Sign Up form above.

The sign in form is called with dispatch rule `logon`, implemented by *mod_authentication* (page 359). This module will normally be enabled.

Sign In covers a number of associated functions:

- Sign in (both public and admin)
- Request password reset
- Feedback after reset
- Create new password

- Feedback when account needs verification

This makes the template structure more complex than the sign up form.

Overriding templates

The templates are quite minimal and will probably not need structural changes. Most likely candidates for changing are titles and perhaps removing/adding extra links.

For details see “Template structure Sign Up form” below.

Changing the configuration

The two configuration values in template file `_logon_config.tpl` deal with appearance: `style_boxed` and `style_width` (see below). Copy the file from `mod_authentication` to your project and change the values.

Changing appearance

Form fields are displayed using the Bootstrap 3 HTML structure. To change the layout, look at the form field templates:

- `_logon_login_form_fields.tpl`
- `_logon_reminder_form_fields`
- `_logon_reset_form_fields`

The width of the form can be set with configuration value `style_width`. This should be a CSS value like “300px”. If no value is set, the form will expand to maximum width - in which case the width needs to be set by CSS

A form background can be set with configuration value `style_boxed`.

CSS styles are defined in `css/logon.css` (`mod_authentication`).

Reference: Template structure Sign up form

Template tree:

```
signup.tpl                                // sign up page
|-- _signup_config.tpl                   // template and field configuration
    |-- _signup.tpl                     // if signed in, redirects to user_
↪page
    |-- _signup_box.tpl                 // form box components
        |-- _signup_stage.tpl          // feedback message
```

The central form template `_signup_box.tpl` is further populated by sub templates:

```
_signup_box.tpl
|-- _signup_title.tpl                   // header "Sign up"
|-- _signup_extra.tpl                   // sign up with other auth modules_
↪(if activated)
|-- _signup_form_form.tpl               // HTML form
|   |-- _signup_form_fields.tpl         // 3 form parts plus submit button
|       |-- _signup_form_fields_email.tpl // name and email
|       |-- _signup_form_fields_username.tpl // username and password
|       |-- _signup_form_fields_tos.tpl  // terms of service
|-- _signup_support.tpl                 // left empty
```

```
|-- _signup_outside.tpl           // link (back) to sign in
|-- _logon_link.tpl              // shown below sign in form
```

Reference: Template structure Sign Up form

Template tree:

```
logon.tpl                       // sign in page
|-- _logon_config.tpl           // template and field configuration
|   |-- _logon.tpl or _logon_modal.tpl // logon module
|   |-- _logon_box.tpl           // form box components
|       |-- _logon_stage.tpl      // feedback messages
|       |-- _logon_expired_form.tpl // when pw is expired
|-- logoff.tpl                  // log off page, redirects to q.p or _
↪homepage
```

Two mechanisms handle the state to determine which sub templates should be read:

- For display on the page: the dispatch rule
- For display inside a modal: the `logon_state` value

Depending on the the state value, `_logon_box.tpl` is populated by different sub templates:

```
when logon_state is::
|== logon_reminder:                // request a pw reset
|   |-- _logon_box.tpl
|       |-- _logon_reminder_title.tpl
|       |-- _logon_reminder_form.tpl
|       |-- _logon_reminder_form_fields.tpl or _logon_reminder_admin_form_
↪fields.tpl
|       |-- _logon_reminder_support.tpl // backlink to logon form
|== logon_reset:                  // reset pw
|   |-- _logon_box.tpl
|       |-- _logon_reset_title.tpl
|       |-- _logon_reset_form.tpl
|       |-- _logon_reset_form_fields.tpl
|       |-- _logon_reset_support.tpl // backlink to logon form
|== admin_logon:
|   |-- _logon_box.tpl
|       |-- _logon_error.tpl
|       |-- _logon_login_form.tpl
|       |-- _logon_login_admin_form_fields.tpl
|       |-- _logon_login_support.tpl // link forgot password
|== signup (logon_state/modal only)
|   |-- _signup_config.tpl (see Sign Up form)
|   |-- _signup_support.tpl // backlink to logon form
|== else:
|   |-- _logon_box.tpl
|       |-- awaiting verification:
|           |-- _logon_stage.tpl // alternative content for logon box
|       |-- else:
|           |-- _logon_login_title.tpl // title "Sign in to ..."
|           |-- _logon_login_extra.tpl // all-include by other modules
|           |-- _logon_error.tpl
|           |-- _logon_login_form.tpl
|           |-- _logon_login_form_fields.tpl
|           |-- _logon_login_support.tpl // link forgot password
|           |-- _logon_login_outside.tpl // all-include _logon_link.tpl
|           |-- _logon_link.tpl // all-include by other modules
```

Here the sub templates ensure a consistent markup inside the box when going from state to state.

Managing redirection after login and signup

Configure `mod_signup` to redirect to something other than a member's home page.

Why

The default behavior of Zotonic is to redirect the user to his or her own page after logon (`/page`). If you want to change this, you need to modify the `sitename.erl` file, where *sitename* is the name of your site. Remember that this file is located in `apps_user/sitename/src/`.

Note: Your actual site location might be different, see the *Zotonic user directory*.

Assumptions

Readers are assumed to be comfortable with Erlang syntax and use of the command shell.

How

Open `apps_user/sitename/src/sitename.erl` with your favorite editor:

```
$ cd /path/to/zotonic
$ vim apps_user/sitename/src/sitename.erl
```

Example 1

Redirection to the `/welcome` page after signup and to the `/welcome-back` page after login

Add the following code:

```
-export([observe_signup_confirm_redirect/2, observe_logon_ready_page/2]).

%% @doc Set the page that must be opened after signup
observe_signup_confirm_redirect({signup_confirm_redirect, _UserID}, _Context) ->
    "/welcome".

%% @doc Set the page that must be open after login
observe_logon_ready_page({logon_ready_page, _}, _Context) ->
    "/welcome-back".
```

Example 2

Conditional redirection, where the site administrator is redirected to the `/admin` page, and all other users are redirected to the `/welcome` page:

```
-export([observe_logon_ready_page/2]).

%% @doc Set the page that must be opened after login
observe_logon_ready_page({logon_ready_page, _}, Context) ->
    case z_acl:user(Context) of
        ?ACL_ADMIN_USER_ID -> "/admin";
        _                  -> "/welcome"
    end.
```

Example 3

Conditional redirection with one option being the default behaviour:

```
-export([observe_logon_ready_page/2]).

%% @doc Set the page that must be opened after logon
observe_logon_ready_page({logon_ready_page, _}, Context) ->
    case z_acl:user(Context) of
        ?ACL_ADMIN_USER_ID -> "/admin";
        _                  -> undefined    % All users except the administrator_
    are redirected to their own pages
    end.
```

Example 4

Conditional redirection, where the site administrator is redirected to the /admin page and all other users are redirected to the /welcome page, unless another URL is specified:

```
-export([observe_logon_ready_page/2]).

%% @doc Set the page that must be opened after logon
observe_logon_ready_page({logon_ready_page, Url}, Context) ->
    case {z_acl:user(Context), Url} of
        {?ACL_ADMIN_USER_ID, _} ->
            "/admin";
        {_, []} ->
            "/welcome";
        _ ->
            Url
    end.
```

Save your changes and quit.

Recompile zotonic:

```
$ cd /path/to/zotonic
$ make
```

Now when logging in, you should see this redirect behaviour in action.

Displaying a site map

For the benefit of search engines and fans of tables of contents you can easily provide a site map.

Note: This article discusses HTML site maps. For the XML kind which Google uses, see [mod_seo_sitemap](#) (page 390).

Why

Some users prefer to navigate a site based on its underlying structure. Search engines also use this information to determine how pages relate to each other within your site. For these cases it is useful to have a site map.

Assumptions

Readers are expected to know where templates go in a site and how they basically work. To benefit the most you will also need experience in CSS to style the end-result effectively.

How

See also:

menu (page 543)

This is straight-forward except for the CSS (which isn't provided here), but that isn't entirely avoidable since it depends completely on how you want to display things. On the other hand this does give you a workable nested unordered list with links to the pages which certainly serves the basic functionality out of the box.

Create `site_map.tpl` containing:

```
{% extends "page.tpl" %}
{% block below_body %}
    {% menu %}
{% endblock %}
```

Create a dispatch rule to take `/site-map` to your Site Map page:

```
{site_map, ["site-map"], resource_page, [{template, "site_map.tpl"}, {id, page_
↪site_map}]}
```

Create a Page with the Admin interface and set its Unique Name to `page_site_map`. Whatever you put in the body will show above the site map.

I haven't discussed CSS, but the HTML generated by the *menu* (page 543) tag is pretty CSS friendly.

Site-specific signup actions

Performing additional, project-specific actions when a user signs up

Why

Note: Your actual site location might be different, see the *Zotonic user directory*.

When a user signs up, Zotonic verifies the validity of some information, such as the e-mail address and the password strength. If these verifications succeed, it stores the user data in the Zotonic database. If you have your own database with project-specific user data, you may want to insert additional information into this database on sign up. To do this, you need to modify the `sitename.erl` file, where *sitename* is the name of your site. Remember that this file is located in `apps_user/sitename/`.

Assumptions

Readers are assumed to be comfortable with Erlang syntax and use of the command shell.

How

Open `apps_user/sitename/src/sitename.erl` with your favorite editor.

Example 1: a gaming website with several point accounts for each user. Add the following code:

```
-export([observe_signup_done/2]).

%% @doc Perform additional actions when a user signs up
observe_signup_done({signup_done, UserId, _IsVerified, _Props, _SignupProps},
    Context) ->
    % Create the necessary accounts for the user:
    create_user_accounts(UserId, [chess, go, checkers]).

% Implementation not shown here
create_user_accounts(_UserId, _Context) ->
    todo.
```

Save your changes and quit

Recompile zotonic:

```
$ cd /path/to/zotonic
$ make
```

Now you'll see that when a signup is done successfully, your `create_user_accounts/2` function will be called.

Dynamic select options using a wired template

Why

Suppose you want to wire a change event for a select box to update a another select box, i.e. you want to wire the action to use the selected value when rendering the template.

Assumptions

Readers are assumed to be comfortable editing templates and have Zotonic Scomp knowledge.

How

We want to select an image from a page in 2 steps. First we select a page from the pages that have images. Then we select an image from the selected page.

This is how the main template will look:

```
<div class="form-group">
  {% wire id="pages" type="change" action={update
    target="media_list"
    template="_media_list.tpl"
  }%}
  <select id="pages" class="form-control">
    <option value="">--Select a page--</option>
    {% for page_id in m.search[{query
      hasobjectpredicate='depiction'
      sort='+rsc.pivot_title'
    }] %}
      <option value="{{ page_id }}">
        {{ m.rsc[page_id].title }}
      </option>
    {% endfor %}
  </select>
</div>
<div id="media_list">
```

```
<!-- contents will be updated -->
</div>
```

When an item is selected (by the change event), element with id “media_list” will be replaced by `_media_list.tpl` which looks like this:

```
{% with q.triggervalue as page_id %}
<div class="form-group">
  <select class="form-control">
    <option value="">--Select an image--</option>
    {% for media_id in m.rsc[page_id].media %}
      <option value="{{media_id}}">
        {{ m.rsc[media_id].title }}
      </option>
    {% endfor %}
  </select>
</div>
{% endwith %}
```

This template uses the `q.triggervalue` returned from the postback of the wire event, and it contains the value of the selected option.

We can go one step further to show the selected image. `_media_list.tpl` also gets a wire action:

```
{% with q.triggervalue as page_id %}
<div class="form-group">
  {% wire id="images" type="change" action={update
    target="image"
    template="_image.tpl"
  } %}
  <select class="form-control" id="images">
    <option value="">--Select an image--</option>
    {% for media_id in m.rsc[page_id].media %}
      <option value="{{media_id}}">
        {{ m.rsc[media_id].title }}
      </option>
    {% endfor %}
  </select>
</div>
<div id="image">
  <!-- contents will be updated -->
</div>
{% endwith %}
```

And we show the image using `_image.tpl`:

```
{% with m.rsc[q.triggervalue].medium as medium %}
<div class="form-group">
  {% image medium width=300 %}
</div>
{% endwith %}
```

4.1.3 Just enough...

These Cookbook items each represent a stage in some Zotonic users’ journeys to understand the workings of Erlang and related technologies in general.

Just enough Erlang/OTP and rebar, part 1

Zotonic source code have you scratching your head? Learn Rebar first.

Created Aug 8, 2011 by Lloyd R. Prentice

Why

Rebar is a relatively new set of Erlang/OTP development tools. Rebar makes it easier to develop and maintain Erlang/OTP applications and releases.

Zotonic is built on Erlang/OTP. An understanding of Erlang/OTP conventions is essential if you wish to read Zotonic source code, develop Zotonic modules, or contribute code or patches to Zotonic.

By following each step in this tutorial carefully, and referring back to the many excellent on-line Erlang documentation resources, you will accelerate your progress up the daunting Erlang/OTP learning curve. And more, you'll learn how to read and understand Zotonic source code while you're at it.

In this tutorial we'll use rebar to create, compile, and test two Erlang applications. One will include a simple `gen_server`.

In Part II, we'll generate documentation, run eunit tests, and create a release that can be copied and run on a suitable host system.

Assumptions

You have Erlang/OTP 23 or later installed on your system. You have Internet access and basic Bash command line skills. File editing tasks refer to vim. But you can use your code editor of choice.

This tutorial was tested on Ubuntu 11.04.

How

How can I install and learn Rebar?

Create a root directory for experimentation. Let's call it "learn":

```
$ mkdir learn
$ cd learn
```

Download the rebar binary

In the shell:

```
learn$ git clone https://github.com/erlang/rebar3.git rebar-src
learn$ cd rebar-src/
rebar-src$ ./bootstrap
rebar$ cd ..
learn$ cp rebar-src/rebar .
learn$ chmod u+x rebar
```

How can I create an application

In the shell:

```
learn$ ./rebar create-app appid=zzz
learn$ ls
>> rebar rebar-src src
```

Note that Rebar has created a directory named `src`:

```
learn $ ls src
>> zzz_app.erl zzz.app.src zzz_sup.erl
```

In *src*, Rebar has created three Erlang modules. Open them up and take a look in your favorite code editor:

Learn more about Erlang applications: http://www.erlang.org/doc/design_principles/applications.html <http://www.erlang.org/doc/man/application.html>

How can I add a `gen_server` template to my new application?

In the shell:

```
learn$ ./rebar create template=simplesrv srvid=zzz_srv
learn$ ls src
>> zzz_app.erl zzz.app.src zzz_srv.erl zzz_sup.erl
```

Open up `zzz_srv.erl` and look it over. For more info, study these `gen_server` resources:

http://www.erlang.org/doc/design_principles/gen_server_concepts.html http://www.erlang.org/doc/man/gen_server.html

How can I make my new `gen_server` do something?

In `src/zzz_srv.erl`, add two functions to the API `-export` directive as follows:

```
-export([start_link/0, say_hello/0, stop/0]).
```

Add the `say_hello/0` function as follows:

```
say_hello() ->
    gen_server:call(?MODULE, hello).
```

Replace `handle_call(_Request, _From, State)`:

```
handle_call(hello, _From, State) ->
    io:format("Hello from zzz_srv!~n", []),
    {reply, ok, State};
handle_call(_Request, _From, State) ->
    Reply = ok,
    {reply, Reply, State}.
```

Add the `stop/0` function:

```
stop() ->
    gen_server:cast(?MODULE, stop).
```

Add before `handle_cast(_Msg, State)`, put:

```
handle_cast(stop, State) ->
    {stop, normal, State};
```

NOTE: If your `gen_server` is under supervision, there's a better way to stop your server. See:

Section 2.6 of `gen_server` Concepts - Stopping: http://www.erlang.org/doc/design_principles/gen_server_concepts.html

You could compile this code with Rebar now, but let's defer.

To really get the hang, let's create TWO applications. We'll put them under a new directory, *apps/*:

```
learn$ mkdir apps
learn$ mkdir apps/zzz
learn$ mkdir apps/zzz_lib
learn$ ls apps
>> zzz zzz_lib
learn$ mv src apps/zzz/
learn$ ls apps/zzz
>> src
```

Now we'll create the `zzz_lib` application:

```
learn$ ./rebar create-app appid=zzz_lib
learn$ ls
>> apps rebar rebar-src src
```

And let's make it do something:

```
learn$ cd src
```

Create and save a module called `hello.erl` that does something:

```
-module(hello).
-export([hello/0]).
hello() ->
    io:format("Hello from zzz_lib!~n", []).
```

Back in the shell move the `src` directory to `apps/zzz_lib`:

```
src$ cd ..
learn$ mv src apps/zzz_lib/
```

How can I compile these two applications?

First, we need to create a `rebar.config` file in our project home directory. Create the file, add the following directive and save:

```
{sub_dirs, ["apps/zzz", "apps/zzz/src", "apps/zzz_lib", "apps/zzz_lib/src" ] }.
```

Back in the shell:

```
learn$ ls
>> apps rebar rebar-src rebar.config
```

Now compile:

```
learn$ ./rebar compile
```

If you see the following, pat yourself on the back:

```
=> zzz (compile)
Compiled src/zzz_app.erl
Compiled src/zzz_sup.erl
Compiled src/zzz_srv.erl
=> src (compile)
=> zzz_lib (compile)
Compiled src/hello.erl
Compiled src/zzz_lib_app.erl
Compiled src/zzz_lib_sup.erl
=> src (compile)
=> learn (compile)
```

Check out the ebin directories:

```
learn$ ls apps/zzz/ebin
>> zzz.app zzz_app.beam zzz_srv.beam zzz_sup.beam
learn$ ls apps/zzz_lib/ebin
>> hello.beam zzz_lib.app zzz_lib_app.beam zzz_lib_sup.beam
```

you're now ready to rock and roll!!

How can I test?

Start the Erlang shell:

```
learn$ erl -pa apps/*/ebin
1> zzz_srv:start_link().
{ok,<0.33.0>}
2> zzz_srv:say_hello().
Hello from zzz_srv!
ok
3> zzz_srv:stop().
ok
4> hello:hello().
Hello from zzz_lib!
ok
```

Troubleshooting

I got an error when I compiled. What now?

make sure your `rebar.config` directive, as shown above, is correct.

Make sure you have this directory structure:

```
learn$ tree
.
apps
|  - zzz
|  |  - _build
|  |  - src
|  |      - zzz_app.erl
|  |      - zzz_app.src
|  |      - zzz_srv.erl
|  |      - zzz_sup.erl
|  - zzz_lib
|  |  - _build
|  |  - src
|      - hello.erl
|      - zzz_lib_app.erl
|      - zzz_lib_app.src
|      - zzz_lib_sup.erl
- rebar
- rebar.config
```

Fix any source code errors, and recompile:

```
learn$ ./rebar compile
```

What you've learned

You've now had a good soak in basic Erlang/OTP conventions and Erlang. You can install Rebar, create Erlang/OTP applications, and compile them. You've also created a simple `gen_server`.

Where to go from here

Study the online and printed Erlang documentation upside and sideways. Skim to see what's there, then reread everytime you have a problem. You'll be an Erlang/OTP wizard before you know it.

References on the web

Getting Started: <https://github.com/erlang/rebar3/wiki/Getting-started>

Damn Technology: <http://damntechnology.blogspot.com/>

How to create, build, and run an Erlang OTP application using Rebar: <http://skeptomai.com/?p=56#sec-3>

Commands: <https://github.com/erlang/rebar3/wiki/Rebar-commands>

Erlang App. Management with Rebar: <http://erlang-as-is.blogspot.com/2011/04/erlang-app-management-with-rebar-alan.html>

Dizzy Smith – Building Erlang Applications with Rebar: <http://ontwik.com/erlang/dizzy-smith-building-erlang-applications-with-rebar/>

Rebar Demo using ibrowse: <http://vimeo.com/8311407>

rebar / rebar.config.sample: <https://github.com/basho/rebar/blob/master/rebar.config.sample?source=cc>

Books

Programming Erlang: Software for a Concurrent World: <http://www.amazon.com/Programming-Erlang-Software-Concurrent-World/dp/193435600X>

Erlang Programming: http://www.amazon.com/ERLANG-Programming-Francesco-Cesarini/dp/0596518188/ref=pd_sim_b_1

Erlang and OTP in Action: http://www.amazon.com/Erlang-OTP-Action-Martin-Logan/dp/1933988789/ref=pd_sim_b_1

Just enough Erlang/OTP and rebar, part 2

Building a `gen_server` to front the library and generating documentation.

Created Aug 17, 2011 by Lloyd R. Prentice

WHY

If you worked through the Cookbook item *Just enough Erlang/OTP and rebar, part 1* (page 292), you have created two Erlang applications under management of Rebar. You've used Rebar to create a `gen_server` template in the first application and added several functions. You've also created a library application with one function.

In this tutorial we'll call our library function from our `gen_server` and generate documentation.

ASSUMPTIONS

You have Erlang 20 or later installed on your system. You have basic Bash command line skills. File editing tasks refer to vim. But you can use your code editor of choice.

You've successfully compiled the `zzz` and `zzz_lib` applications from Part I under Rebar.

This tutorial was tested on Ubuntu 11.04.

HOW

How can I call a library function from `zzz_lib` in a `zzz` module?

In the shell:

```
learn$ cd apps/zzz/src
```

Edit `handle_call/2` in `zzz_srv.erl` as follows:

```
handle_call(hello, _From, State) ->
    hello:hello(),
    io:format("~nHello from zzz_srv!~n", []),
    {reply, ok, State};
```

While you're at it, check your `handle_cast/2` code. It should look like this:

```
stop() ->
    gen_server:cast(?MODULE, stop).

%% callbacks
handle_cast(stop, State) ->
    {stop, normal, State};

handle_cast(_Msg, State) ->
    {noreply, State}.
```

Now edit and save `init/1` in `learn/apps/zzz/src/zzz_sup.erl`:

```
init([]) ->
    {ok, {{one_for_one, 1, 60}, [?CHILD(zzz_srv, worker)]}}.
```

Now recompile:

```
learn$ ./rebar clean compile
```

And test:

```
learn$ erl -pa apps/*/ebin
...
1> zzz_sup:start_link().
{ok,<0.33.0>}
```

Note that you may see a different PID on your system. Try some functions:

```
2> zzz_srv:say_hello().
Hello from zzz_lib!
Hello from zzz_srv!
ok
```

Both our library function and `zzz_srv` have performed as we'd hoped.

While developing this tutorial, I introduced a bug in `zzz_srv:handle_cast/2`. Once it started, I couldn't stop. This was a good excuse to introduce a valuable debugging tool.

How can I kill a running process?

In the Erlang shell, start *pman*, the process manager:

```
3> pman:start().
<0.37.0>
```

You should see a graphic window open with a list of running processes. Find *zzz_srv* and *zzz_sup* under Name. Click on *zz_sup* to highlight. Now pull down the Trace menu item and click on *kill*. Note that you've not only killed *zzz_sup*, but *zzz_srv* as well. Of course. *zzz_sup* supervises *zzz_srv*. When it dies, so does *zzz_srv*.

You can confirm:

```
4> zzz_srv:say_hello().
** exception exit: {noproc,{gen_server,call,[zzz_srv,hello]}}
   in function  exit/1
   called as  exit({noproc,{gen_server,call,[zzz_srv,hello]}})
   in call from gen_server:call/2
```

You assassin, you!

But, no worry, you can start it back up again:

```
5> zzz_srv:start_link().
{ok,<0.48.0>}
```

Now let's test *zzz_srv:stop()*:

```
6> zzz_srv:stop().
ok
```

Seemed to worked. But...:

```
7> zzz_srv:say_hello().
Hello from zzz_lib!
Hello from zzz_srv!
ok
```

It turns out, *zzz_sup* started it up again. Try it again:

```
8> zzz_srv:stop().
** exception exit: shutdown
```

Wait! If you look at *pman*, you'll see that *zzz_sup* also died. What's going on here?

If you look at *zzz_sup.erl* in *learn/apps/zzz/src*, you'll note that *init/1* allows only one restart, the value following "one_for_one":

```
init([]) ->
    {ok, {{one_for_one, 1, 60}, [?CHILD(zzz_srv, worker)]}}.
```

For more details, checkout the Supervisor Behaviour: http://www.erlang.org/doc/design_principles/sup_princ.html

How can I generate documentation?

Add comments to *learn/apps/zzz/src/zzz_srv.erl* as follows:

```
%% @spec say_hello() -> none
%% @doc display greeting
say_hello() ->
```

And to learn/apps/zzz_lib/src/hello.erl:

```
%% @spec hello() -> none
%% @doc Library test function
hello() ->
```

Now execute:

```
learn$ ./rebar clean compile
...
learn$ ./rebar doc
==> zzz (doc)
==> zzz_lib (doc)
```

Check out your new doc directories:

```
learn$ ls apps/zzz
doc ebin src

learn$ ls apps/zzz_lib
doc ebin src
```

Now bring up your browser and point to file:

```
file:///home/learn/apps/zzz/doc/index.html file:///home/learn/apps/zzz_lib/doc/
↪index.html
```

TROUBLESHOOTING

I got an error when I compiled. What now?

make sure rebar.config in ../learn looks like this:

```
{sub_dirs,
  [ "apps/zzz",
    "apps/zzz/src",
    "apps/zzz_lib",
    "apps/zzz_lib/src"
  ]
}.
```

Make sure you have this directory structure:

```
learn$ tree
.
apps
|  - zzz
|  |  - ebin
|  |  - src
|  |      - zzz_app.erl
|  |      - zzz_app.src
|  |      - zzz_srv.erl
|  |      - zzz_sup.erl
|  - zzz_lib
|  |  - ebin
|  |  - src
|      - hello.erl
|      - zzz_lib_app.erl
|      - zzz_lib_app.src
|      - zzz_lib_sup.erl
- rebar
- rebar.config
```


WHAT YOU'VE LEARNED

You've now had a good soak in basic Erlang/OTP conventions and Erlang. You can install Rebar, create an Erlang/OTP application, compile it and create documentation.

WHERE TO GO FROM HERE

Study the online and printed Erlang documentation upside and sideways. Skim to see what's there, then reread everytime you have a problem. You'll be an Erlang/OTP wizard before you know it.

Now, dive into Zotonic source code. It should be much easier to follow.

REFERENCES

Getting Started: <https://github.com/erlang/rebar3/wiki/Getting-started>

Damn Technology: <http://damntechnology.blogspot.com/>

How to create, build, and run an Erlang OTP application using Rebar: <http://skeptomai.com/?p=56#sec-3>

Commands: <https://github.com/erlang/rebar3/wiki/Rebar-commands>

Erlang App. Management with Rebar: <http://erlang-as-is.blogspot.com/2011/04/erlang-app-management-with-rebar-alan.html>

Dizzy Smith – Building Erlang Applications with Rebar: <http://ontwik.com/erlang/dizzy-smith-building-erlang-applications-with-rebar/>

Rebar Demo using ibrowse: <http://vimeo.com/8311407>

rebar / rebar.config.sample: <https://github.com/erlang/rebar3/blob/master/rebar.config.sample?source=cc>

Just enough Postgres

Understand the primary data-store of Zotonic.

Why

Data persistence in Zotonic is provided by PostgreSQL, a mature feature-rich open-source relational database.

Since Zotonic provides both a data model and wrapper around PostgreSQL queries, the Zotonic user is substantially insulated from routine Postgres operation. But now and again, for issues ranging from installation, backup/restore, and debugging, familiarity with a few PostgreSQL commands can save considerable time.

Sadly, PostgreSQL user documentation is abundant, but not as well organized as one might hope.

Assumptions

These commands have been tested on Ubuntu 16.04.

Familiarity with SQL and relational databases advised.

NOTE: The string “VERSION,” as used below, refers to your PostgreSQL version.

How

Where can I find PostgreSQL documentation?

If PostgreSQL is installed: `/usr/share/doc/postgresql-common/README.Debian.gz`.

For on-line documentation: <https://www.postgresql.org/docs/VERSION/static/>

For instance, in Ubuntu 18, look for: <https://www.postgresql.org/docs/10/static/>

Is PostgreSQL installed?

In a shell:

```
ls /usr/lib | grep postgresql
```

You should see:

postgresql

What version?

In the shell:

```
ls -l /usr/lib/postgresql/
```

You should see:

```
drwxr-xr-x 4 root root 4096 2011-06-23 14:13 VERSION
```

How can I install PostgreSQL?

In the shell:

```
apt-get update
apt-get install postgresql postgresql-client
```

Or, from source: <http://www.postgresql.org/docs/9.4/static/install-short.html>

Where are Postresql files located?

Configuration files: `/etc/postgresql/[version]/[cluster]/` Binaries: `/usr/lib/postgresql/[version]` Data files: `/var/lib/postgresql/[version]/[cluster]`

Is the PostgreSQL server running?

In the shell:

```
/etc/init.d/postgresql status
```

You should see something like:

```
postgresql.service - PostgreSQL RDBMS
Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset:
➔enabled)
Active: active (exited) since Fri 2021-04-02 10:24:08 CEST; 4 days ago
Process: 871 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
Main PID: 871 (code=exited, status=0/SUCCESS)
```

How can I stop the PostgreSQ server?

In the shell:

```
/etc/init.d/postgresql stop
```

You should see something like:

```
* Stopping PostgreSQL 10.16 database server      [ OK ]
```

How can I start the Postgres server?

In the shell:

```
/etc/init.d/postgresql start
```

You should see something like:

```
* Starting PostgreSQL 10.16 database server      [ OK ]
```

How can I restart the PostgreSQL server?

In the shell:

```
/etc/init.d/postgresql restart
```

You should see something like:

```
* Restarting PostgreSQL 10.16 database server
```

How can I switch to database 'zotonic_blog' in psql?

In the shell:

```
zotonic@host $ psql
zotonic=# \c zotonic_blog
```

You should now be on psql for the zotonic_blog database:

```
You are now connected to database "zotonic_pcc".
zotonic_blog=#
```

How can I enter the PostgreSQL interactive terminal?

In the shell:

```
psql
```

You should now be on the interactive terminal:

```
psql (10.14)
Type "help" for help.

postgres=#
```

How can I list databases?

From psql:

```
\l
```

Or directly from the Zotonic User's shell:

```
psql -l
```

You should see a list of databases like the following:

List of databases					
Name	Owner	Encoding	Collation	Ctype	Access privileges
postgres	postgres	UTF8	en_US.UTF-8	en_US.UTF-8	

```

template0      | postgres | UTF8      | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
               : postgres=CtC/
↪postgres
template1      | postgres | UTF8      | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
               : postgres=CtC/
↪postgres
zotonic         | zotonic  | UTF8      | en_US.UTF-8 | en_US.UTF-8 |
zotonic_blog    | zotonic  | UTF8      | en_US.UTF-8 | en_US.UTF-8 |
(5 rows)

```

How can I see if a database exists?

In the shell:

```
psql test
```

If the database doesn't exist:

```
psql: FATAL: database "test" does not exist
```

If the database exists, you'll see something like:

```

psql (10.14)
Type "help" for help.

test=>

```

How can I tell if the database for site 'blog' exists in the local postgres installation?

In the shell:

```
psql -l | grep blog
```

You should see something like:

```
zotonic_blog | zotonic | UTF8      | en_US.UTF-8 | en_US.UTF-8 |
```

How do I list the relations (tables, views, sequences) in a database?

In the shell:

```

psql zotonic_blog
zotonic_blog=# \d

```

You should see something like:

List of relations			
Schema	Name	Type	Owner
public	category	table	zotonic
public	comment	table	zotonic
public	comment_id_seq	sequence	zotonic
public	config	table	zotonic
public	config_id_seq	sequence	zotonic
public	edge	table	zotonic
{...etc. }			

If psql displays this in a pager (prompt is a :) you can escape by hitting q.

How can I create a table in a database?

NOTE: Many fine books and tutorials are available to help you learn SQL, the standard query language for relational databases. See references below.

The follow queries are for illustration only:

```
postgres=# CREATE TABLE books (
postgres=# title text NOT NULL);
CREATE TABLE
```

How to add a column to a table:

```
postgres=# ALTER TABLE books
postgres=# ADD author text NOT NULL;
ALTER TABLE
```

How to examine the structure of a table:

```
postgres=# \d books
      Table "public.books"
  Column | Type   | Modifiers
-----+-----+-----
 title  | text   | not null
 author | text   | not null
```

How to insert a record into a table:

```
postgres=# INSERT INTO books ( title, author )
postgres=# VALUES ('Programming Erlang', 'Joe Armstrong');
INSERT 0 1
```

How to examine records in a table:

```
postgres=# SELECT * FROM books;

      title      | author
-----+-----
Programming Erlang | Joe Armstrong
(1 row)
```

How to select a record from a table:

```
postgres=# SELECT title FROM books
postgres=# WHERE author = 'Joe Armstrong';
      title
-----
Programming Erlang
(1 row)
```

How to create a database user:

```
postgres=# CREATE USER myuser WITH PASSWORD 'userpassword' LOGIN;
CREATE ROLE
```

How to create a database:

```
postgres=# CREATE DATABASE testdb WITH OWNER = myuser ENCODING = 'UTF8';
CREATE DATABASE
postgres=# GRANT ALL ON DATABASE testdb TO myuser;
GRANT
```

How to initialize a database:

<http://www.postgresql.org/docs/10/static/app-initdb.html>

How can I back-up a database:

— Method 1: Use Backing up your site.

— Method 2: Dump can be created on the source machine with the following command (replace `zotonic_blog` with your site's db name):

```
pg_dump zotonic_blog > zotonic_blog.sql
```

How to delete a database named 'test' and all its contents:

```
pg_dump test > test.sql  
dropdb test
```

How can I restore the contents of a database from backup

See [Restore/upgrade content db from backup](#) (page 337)

Zotonic Conveniences that avoid direct Postgres interaction

How can I create a database for my first Zotonic?:

```
zotonic createdb blog  
zotonic addsite -d zotonic_blog blog
```

How can I create a database for an additional Zotonic site?:

```
zotonic createdb blog  
zotonic addsite -d zotonic_blog blog
```

Notice the pattern ;)

How can I open the Zotonic shell?

In the terminal:

```
zotonic shell
```

How can I select records from the Zotonic shell?

In the zotonic shell:

```
1> m_rsc:get (page_home, z:c(blog)).  
#{  
  <<"category_id">> => 104,  
  <<"created">> => {{2011,6,8},{22,21,55}},  
  <<"creator_id">> => 1,  
  <<"id">> => 313,  
  <<"is_authoritative">> => true,  
  <<"is_featured">> => false,  
  <<"is_protected">> => false,  
  <<"is_published">> => true,  
  <<"modified">> => {{2011,6,8},{22,21,55}},  
  <<"modifier_id">> => ,1,  
  <<"name">> => <<"page_home">>,  
  <<"page_path">> => <<"/">>,  
  <<"publication_end">> => ,{{9999,8,17},{12,0,0}},  
  <<"publication_start">> => ,{{2011,6,8},{22,21,55}},  
  <<"slug">> => <<"home">>,  
  <<"uri">> => undefined,  
  <<"version">> => 1,  
  <<"visible_for">> => 0,  
  <<"title">> => <<"Home">>,  
  <<"summary">> => <<"Welcome to your blog!">>,  
  <<"managed_props">> => #{  
    <<"title">> => <<"Home">>,  
    <<"summary">> => <<"Welcome "...>>,  
    <<"page_path">> => <<"/">>}}},
```

```
<<"installed_by">> => <<"z_install_defaultdata">>
}
```

Troubleshooting

Pay GREAT attention to permissions. Your tables and sequences should be owned by the user specified in the site's config file. GRANT may not be enough. So, if you see Zotonic trying to recreate tables or if Zotonic fails with a 3D000 error (database object doesn't exist) even if you are positive already exist, it means your permissions are wrong.

Problem:

You try to get an psql shell:

```
psql
```

And it refuses to work:

```
psql: FATAL:  Ident authentication failed for user "postgres"
```

Solution:

You need to configure `pg_hba.conf`

Note: For maximum security, correct configuration of `pg_hba.conf` is essential.

See *Enabling trust-authentication in PostgreSQL* (page 623) in this manual, or look at the PostgreSQL docs:

<https://www.postgresql.org/docs/10/interactive/client-authentication.html> <https://www.postgresql.org/docs/10/interactive/auth-pg-hba-conf.html>

Problem:

In postgres, you get the following:

```
testdb=> CREATE USER testdb WITH PASSWORD 'testdb' LOGIN;
ERROR:  permission denied to create role
```

Solution:

You need to create a database user. Retry as the Postgres superuser:

```
sudo su postgres psql
```

And it will work:

```
postgres=# CREATE USER testdb WITH PASSWORD 'testb' LOGIN;
CREATE ROLE
```

Problem:

In the shell:

```
cd /etc/postgresql
```

Outputs:: bash: cd: /etc/postgresql: No such file or directory

Solution:

This is evidently a bug in certain Debian Lenny installs when `/etc/postgresql` is inadvertently deleted. Uninstalling `postgresql-client` (`apt-get --purge remove postgresql-client`) is supposed to fix it. But it won't if the system has an older version of `udev`.

See: <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=517389>

Need updated version of udev

Problem:

Erratic performance of database

Solution:

Examine PostgreSQL installation files. Expect trouble if, by happenstance, you have more than one instance of PostgreSQL server running. You may have to back-up your data, uninstall all PostgreSQL files and reinstall.

Note: On some Lenny installations `apt-get --purge remove postgresql` will *NOT* remove all configuration files. And, `apt-get install postgresql` will not replace missing a missing `/etc/postgresql` directory and files.

Resources

Howto: Debian / Ubuntu Linux Install PostgreSQL Database Server <http://www.cyberciti.biz/faq/linux-installing-postgresql-database-server/>

psql: FATAL: Ident authentication failed for user "username" Error and Solution <http://www.cyberciti.biz/faq/psql-fatal-ident-authentication-failed-for-user/>

PostgreSQL for Beginners <http://www.postgresqlforbeginners.com/2010/11/interacting-with-postgresql-psql.html>

PostgreSQL 10 Documentation <http://www.postgresql.org/docs/10/static/index.html> <http://www.postgresql.org/docs/10/static/reference-client.html>

Howto Backup PostgreSQL Databases Server With `pg_dump` command <http://www.cyberciti.biz/tips/howto-backup-postgresql-databases.html>

How To Use `pg_dump` and `pg_restore` with Postgres Plus Tutorial for Linux [http://www.enterprisedb.com/resources-community/tutorials-quickstarts/linux/how-use-pgdump-and-pgrestore-postgres-plus-tutorial-](http://www.enterprisedb.com/resources-community/tutorials-quickstarts/linux/how-use-pgdump-and-pgrestore-postgres-plus-tutorial-postgresql-clustering-and-Debian)

<http://www.progsoc.org/~wildfire/notes/postgresql-cluster.html>

Books

Momjian, Bruce, PostgreSQL: Introduction and Concepts, 2001, Addison-Wesley, Upper Saddle River, NJ, 462 pp

Worsley, John C. and Joshua D. Drake, Practical PostgreSQL, 2002, O'Reilly & Associates, Inc., Sebastopol, CA, 618 pp

Just Enough Regular Expressions (in Erlang)

Learn how to manipulate string data with the `re` module.

Lloyd R. Prentice August 31, 2011

WHY

The Erlang standard library `re` provides a powerful suite of functions to excute regular expressions to find, replace, and manipulate substrings within a string or Erlang binary.

Re functions are particularly useful for form validation, decomposing and modifying urls, e-mail addresses, and other common web elements represented as strings or binaries.

They are found throughout Zotonic source files.

See: <http://www.erlang.org/doc/man/re.html>

ASSUMPTIONS

You have a current version of Erlang installed on your system.

Examples have been tested on Ubuntu 11.04.

HOW

Bring up an Erlang terminal:

```
$ erl
Erlang R14B03 (erts-5.8.4) [source] [64-bit] [smp:3:3] [rq:3] [async-threads:0]
↳[kernel-poll:false]

Eshell V5.8.4 (abort with ^G)
1>
```

and follow along with the following examples. Modify and re-execute each example until you feel comfortable with what's going on.

What is a regular expression?

A regular expression is a pattern that is matched against a subject string from left to right.

The power of regular expressions comes from the ability to include alternatives and repetitions in the pattern. These are encoded in the pattern by the use of metacharacters.

What is a pattern?

Most characters stand for themselves in a pattern.

The pattern “ick”, for instance, would match the first occurrence of “ick” in the string “The quick brown fox.”

We'll use `re:run/2` to illustrate. This function checks a string against a regular expression: `run(Subject, RE) -> {match, Captured} | nomatch`.

Example:

```
6> re:run("The quick brown fox.", "ick").
{match, [{6, 3}]}
```

The atom “match” is self-explanatory. The tuple {6,3} in the list provides the start position and length of the pattern in the subject string.

```
7> re:run("The brown fox.", "ick").
nomatch
```

`re:run/2` will also work with a binary:

```
8> re:run(<<"The quick brown fox.">>, "ick").
{match, [{6, 3}]}
```

If we wish to find all instances of “ick” in a string, we need to use `re:run/3`, `run(Subject, RE, Options) -> {match, Captured} | match | nomatch`.

Example:

```
9> re:run("The sick quick brown fox.", "ick", [global]).
{match, [[{5, 3}], [{11, 3}]]}
```

For documentation of `re:run/3` options, see <http://www.erlang.org/doc/man/re.html>

How can I replace a substring in a string?

Use `re:replace/3`: `replace(Subject, RE, Replacement) -> iodata() | unicode:charlist()`.

Example:

```
10> re:replace("The quick brown fox.", "brown", "red").
[<<"The quick ">>, <<"red">> | <<" fox.">>]
```

Hmmm... `re:replace/3` returns an odd-looking binary, which is called an *iolist*: an efficient data structure, used like this to prevent copying of data in memory.

But, let's use `re:replace/4` to provide an option: `replace(Subject, RE, Replacement, Options) -> iodata() | unicode:charlist()`:

```
11> re:replace("The quick brown fox.", "brown", "red", [{return, list}]).
"The quick red fox."
```

The `{return, list}` does the trick of returning a “regular” string. Erlang documentation is generally thorough, but often not that easy to follow. That's why I created this Cookbook item. I wanted to learn this stuff myself.

Regular expressions can deliver much much more, however, with shrewd use of metacharacters.

What is a metacharacter?

Metacharacters are interpreted in special ways. For instance, the metacharacter `.` matches the first instance of any character in a string except newline.

Example:

```
13> re:run("The quick brown fox.", ".").
{match, [{0, 1}]}
```

You'd usually use `.` in a more elaborate pattern:

```
14> re:run("The quick brown fox.", "qu.").
{match, [{4, 3}]}

15> re:run("The quack brown fox.", "qu.").
{match, [{4, 3}]}
```

The metacharacter `^` asserts start of string.

Examples:

```
16> re:run("The quack brown fox.", "^The").
{match, [{0, 3}]}

17> re:run("The quack brown fox.", "^qua").
nomatch
```

Similarly, the metacharacter `$` asserts the end of a line:

```
18> re:run("The quick brown fox is sick.", "ick.$").
{match, [{24, 4}]}
```

The metacharacter `*` matches zero or more characters.

Examples:

```
19> re:run("The quick brown fox.", "i*").
{match, [{0,0}]}

20> re:run("The quick brown fox.", "T*").
{match, [{0,1}]}

21> re:run("TTTTThe quick brown fox.", "T*").
{match, [{0,5}]}
```

The metacharacter `+` matches one or more characters:

```
22> re:run("TTTTThe quick brown fox.", "z+").
nomatch

23> re:run("TTTTThe quick brown fox.", "T+").
{match, [{0,5}]}
```

The metacharacter `|` alternate patterns. Think of it as “or”:

```
24> re:run("The quick brown fox.", "fox|pig").
{match, [{16,3}]}

25> re:run("The quick brown pig.", "fox|pig").
{match, [{16,3}]}
```

You can also match generic character types. `s`, for instance matches any whitespace character.

Examples:

```
26> re:run("The quick brown fox", "\s", [global]).
{match, [{3,1}], [{9,1}], [{15,1}]}
```

How can I match non-printing characters?

See: Non-printing characters <http://www.erlang.org/doc/man/re.html>

Note that the metacharacters `[` and `]` have special meaning, they enclose “character classes.” A character class is the set of characters in a character class match, if any found, one character in the subject string.

Examples:

```
24> re:run("The quick brown fox.", "[qui]").
{match, [{4,1}]}

25> re:run("The quick brown fox.", "[ui]").
{match, [{5,1}]}

26> re:run("The quick brown fox.", "[qui]", [global]).
{match, [{4,1}], [{5,1}], [{6,1}]}
```

You can combine characters, meta-characters, and other regular expression elements into extended patterns that can search, match, and replace nearly any substrings you can imagine.

Example:

```
27> re:run("E-mail: xyz@pdq.com", "[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-z]{2,3}").
{match, [{8,11}]}
```

Note: DO NOT use this pattern in production. It needs more refinement and much more testing.

What other goodies does `re` offer?

`split(Subject, RE) -> SplitList` and `split(Subject, RE, Options) -> SplitList`.

Examples:

```
28> re:split("this/is/my/path", "/").
[<<"this">>, <<"is">>, <<"my">>, <<"path">>]
```

If you wish to use a pattern multiple times and boost performance, you can compile it with `re:compile/1`.

Example:

```
29> {_, P} = re:compile("[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-z]{2,3}").
{ok, {re_pattern, 0, 0,
      <<69, 82, 67, 80, 164, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 64,
      ...>>}}
30> re:run("E-mail: xyz@pdq.com", P).
{match, [{8, 11}]}
```

How are regular expressions used in Zotonic source?

For one of many examples, look at `zotonic/src/markdown/get_url/1`:

```
get_url(String) ->
  HTTP_regex = "(H|h) (T|t) (T|t) (P|p) (S|s)*://",
  case re:run(String, HTTP_regex) of
    nomatch    -> not_url;
    {match, _} -> get_url1(String, [])
  end.
```

Where can I go from here?

Study and experiment with all the metacharacters and other regular expression constructs in:

<http://www.erlang.org/doc/man/re.html>

Do further research on the web. Everytime you see an interesting regular expression, test it in `re:run/2`. You may well have to edit to get it to run on `re:run/2`. But if you understand the basics, it won't be difficult.

TROUBLESHOOTING

CAUTION: Complex regular expression patterns are hard to read and error prone. Break them down into short segments and test each segment. Then build them back up.

The hard part is confirming that your pattern will match all possible instances of the string segments you're interested in.

RESOURCES

<http://www.erlang.org/doc/man/re.html>
[regular-expressions.info/examples.html](http://www.regular-expressions.info/examples.html)

<http://langref.org/erlang/pattern-matching>

<http://www.>

Just enough Erlang shell

An indispensable tool for both learning and programming Erlang.

Submitted by: LRP; July 30, 2011

WHY

You don't need to know Erlang or use the Erlang shell to create simple Zotonic websites. But for Zotonic application development and debugging, Erlang programming skills are essential.

Erlang is not that hard to learn. Several excellent books and any number of web tutorials will get you well underway.

The Erlang shell is an indispensable tool for both learning and programming Erlang.

The easiest way to learn how to use the Erlang shell is to fire it up and play.

This Cookbook recipe provides all you need to get started.

ASSUMPTIONS

You have a recent version of Erlang installed on your system.

HOW

What is the Erlang shell?

The Erlang shell lets you test Erlang expression sequences both locally and remotely. It also lets you work with Erlang records and manage Erlang jobs.

How can I enter the Erlang shell?

From your terminal command line:

```
$ erl
```

(and press enter). You'll see something like:

```
Erlang R13B03 (erts-5.7.4) [source] [64-bit] [smp:3:3] [rq:3] [async-threads:0]
↳[hipe] [kernel-poll:false] Eshell V5.7.4 (abort with ^G)
1>
```

1> is the Erlang shell command line. You are now able to execute Erlang expressions.

How can I get help?

help(). – list of shell functions How can I execute an expression?

Type now `now()` . and then hit ENTER.

NOTE: the period at the end of the expression is necessary. It terminates the Erlang expression sequence and initiates evaluation.

How can I edit a line?

- Ctrl+A – go to beginning of line
- Ctrl+E – go to end of line
- `li <Tab>` – tab completion; for instance, `li tab` will return `lists: * li <Tab> <Tab>` – Erlang terms that contain the characters “li”

For more shell editing commands, refer to Section 1.3.1 in: http://www.erlang.org/documentation/doc-5.1/doc/getting_started/getting_started.html

How can I manage Erlang jobs?

- Ctrl+G – user switch command; job control options
- Ctrl+G h – help user switch options

How can I leave the shell?

`q()` . – quits Erlang

Are there other ways to quit?

- `halt()`. – exits the Erlang runtime system
- `Ctrl+C Ctrl+C` – Display break options, then quit

BEWARE: You don't want to bring a running Erlang system down just to quit the shell. Use `q()` until you get the hang of stuff.

When I enter `Ctrl+C`, I get a bunch of choices. What do they mean?

Refer to: <http://www.erlang.org/doc/man/shell.html>

REFERENCES

http://www.erlang.org/documentation/doc-5.1/doc/getting_started/getting_started.html

<http://www.erlang.org/doc/man/shell.html>

http://www.erlang.org/doc/getting_started/seq_prog.html

Erlang: Starting Out: <http://learnyoussomeerlang.com/starting-out>

4.1.4 Other cookbooks

Create a custom action

Todo

This chapter

Create a custom filter

See also:

Templates (page 28) guide

See also:

Filters (page 454) reference

Create custom *template filters* (page 29) to change the way variables are rendered in your templates. By following some simple rules, Zotonic will automatically find the filter for you:

1. Create a file in the *src/filters/* (page 63) directory of your site or module.
2. Prepend the filter filename with `filter_`.
3. Export a function with the name of your filter that corresponds to the filename.

So, let's say you need to sort a list of items and remove duplicate values from it:

Listing 4.1: `mod_yourmodule/src/filters/filter_uniquesort.erl`

```
-module(filter_uniquesort).  
-export([uniquesort/2]).  
-include_lib("zotonic_core/include/zotonic.hrl").  
  
uniquesort(List, _Context) ->  
    lists:usort(List).
```

The custom `uniquesort` filter is then available in your templates:

```
{% for thing in list_of_things|uniquesort %}
    {{ thing }} is now sorted and unique!
{% endfor %}
```

Create a custom model

See also:

[models section](#) (page 30) in the Developer Guide

See also:

list of [all models](#) (page 555).

In this chapter we will look at how to implement a model around the [The Open Movie Database \(OMDB\) API](#).

We will touch useful background information at the same time.

Model modules

Models are Erlang modules that are prefixed with `m_` and stored in your main module's subdirectory `models`. For example, the model to access Zotonic resources (syntax `m.rsc.property`) is written in `models/m_rsc.erl`.

Each model module is required to implement one function (as defined behavior in `zotonic_model.erl`):

- `m_get/3`

`m_get`

This function fetches a value from a model. Because there are quite some different variation how to use this function, it is good to understand a bit more about the inner workings of data lookup.

Let's start with the parsing of a template expression:

```
{{ m.rsc[1].is_cat.person }}
```

If you have done some work with Zotonic, you will be familiar with this syntax where we can find out if the resource with id 1 (the Administrator) is of category Person.

This expression contains 4 parts (separated by dots).

At the very start, the template parser resolves `m.rsc` to module `m_rsc`. The parser then calls `m_rsc:m_get(Keys, Context)` to fetch the value (where `Keys` in our expression has the value of `[1, <<"is_cat">>, <<"person">>]`).

This is the function specification of `m_get`:

```
-spec m_get(Keys, Msg, Context) -> Return when
    Keys :: list(),
    Msg :: zotonic_model:opt_msg(),
    Return :: zotonic_model:return(),
    Context :: z:context().
```

It takes the dotted expression list and returns the looked up value and the unprocessed part of the dotted list (if any).

In this example, the `m_get` is called as:

```
m_rsc:m_get([ 1, <<"is_cat">>, <<"person">> ], _Msg, Context)
```

And will return:

```
{ok, {true, []}}
```

Where `true` is returned because resource id 1 is indeed a person, and `[]` is returned because the call consumed all parts of the dotted expression.

Example: Setting up m_omdb

All data calls to the OMDB go through the url `http://www.omdbapi.com/?`, with query string appended. We can pass the movie id, title, and pass a type (movie/series/episode). OMDB offers more parameters but we don't need them now.

Template interface

Let's define how will we use the data in templates.

To get all data for a particular ID:

```
m_omdb["tt1135300"]
```

... so that we can get properties like the movie title:

```
{{ m_omdb["tt1135300"].title }}
```

Find an item by title:

```
{{ m_omdb["Alien"].year }}
```

Get all data from a movie:

```
{% for k,v in m_omdb.movie["Alien"] %}{{ k }}: {{ v }}{% endfor %}
```

Get data from a series:

```
{{ m_omdb.series[query title="Dollhouse"].plot }}
```

or from an episode:

```
{{ m_omdb.episode[query title="Dollhouse"].plot }}
```

Model skeleton

We will write our model in module `models/m_omdb.erl`. Let's first get the mandatory elements out of the way:

```
-module(m_omdb) .
-behaviour(zotonic_model) .

-export([
    m_get/3
]).

-include_lib("zotonic_core/include/zotonic.hrl").

% ... We will add our m_get functions here
```



```
-spec m_get( list(), zotonic_model:opt_msg(), z:context() ) -> zotonic_
↳model:return().
m_get([ _ | Rest ], _Msg, _Context) ->
    {ok, {undefined, Rest}};
m_get(_, _Msg, _Context) ->
    {ok, {undefined, []}}.
```

Querying the API

Before diving into the lookup functions, let's see what we want to achieve as result.

1. Using `m_get` we will generate a list of query parameters, for example `[{type, "series"}, {title, "Dollhouse"}]`
2. And pass this list to a “fetch data” function
3. That creates a URL from the parameters,
4. loads JSON data from the URL,
5. and transforms the JSON into a property list

The `fetch_data` function:

```
-spec fetch_data(Query) -> list() when
    Query:: list().
fetch_data([]) ->
    [{error, "Params missing"}];
fetch_data(Query) ->
    % Params title or id must be present
    case proplists:is_defined(title, Query) or proplists:is_defined(id, Query) of
        false -> [{error, "Param id or title missing"}];
        true ->
            % Translate query params id, title and type
            % into parameters that OMDb wants
            QueryParts = lists:map(fun(Q) ->
                make_query_string(Q)
            end, Query),
            Url = ?API_URL ++ string:join(QueryParts, "&"),
            % Load JSON data
            case get_page_body(Url) of
                {error, Error} ->
                    [{error, Error}];
                Json ->
                    % Turn JSON into a property list
                    JsonData = z_json:decode(Json),
                    lists:map(fun(D) ->
                        convert_data_prop(D)
                    end, JsonData)
            end
        end
    end.
```

It is important to know that we will pass a list, and get a list as result (for other template models this may be different).

Lookup functions

To illustrate the simplest `m_get` function, we add one to get the API url:

```
-define(API_URL, "http://www.omdbapi.com/?").
```

```
% Syntax: m.omdb.api_url
m_get([ <<"api_url">> | Rest ], _Msg, _Context) ->
    {ok, {?API_URL, Rest}};
```

The functions that will deliver our template interface are a bit more involved. From the template expressions we can discern 2 different patterns:

1. Expressions with 1 part:

- `m.omdb["Dollhouse"]`
- `m.omdb[{query title="Dollhouse"}]`

2. Expressions with 2 parts:

- `m.omdb.series["Dollhouse"]`
- `m.omdb.series[{query title="Dollhouse"}]`

When an expression is parsed from left to right, each parsed part needs to be passed on using our `m` record. For instance with `m.omdb.series["Dollhouse"]` we first transform “series” to `{type, "series"}`, and then “Dollhouse” to `{title, "Dollhouse"}`, creating the full query `[{type, "series"}, {title, "Dollhouse"}]`.

To parse the type, we add these functions to our module:

```
% Syntax: m.omdb.movie[QueryString]
m_get([ <<"movie">>, QueryString | Rest ], _Msg, Context) when is_
↳binary(QueryString) ->
    Query = [ {type, movie}, {title, QueryString} ],
    {ok, {fetch_data(Query), []}};

% Syntax: m.omdb.series[QueryString]
m_get([ <<"series">>, QueryString | Rest ], _Msg, Context) when is_
↳binary(QueryString) ->
    Query = [ {type, series}, {title, QueryString} ],
    {ok, {fetch_data(Query), []}};

% Syntax: m.omdb.episode[QueryString]
m_get([ <<"episode">>, QueryString | Rest ], _Msg, Context) when is_
↳binary(QueryString) ->
    Query = [ {type, episode}, {title, QueryString} ],
    {ok, {fetch_data(Query), []}};
```

Notice the `| Rest` in the patterns. This is needed for expressions like:

```
m.omdb.series["Dollhouse"].title
```

Which calls our `m_get` function as:

```
m_get([ <<"series">>, <<"Dollhouse">>, <<"title">> ], _Msg, Context)
```

We can also pass:

1. The movie ID: `m.omdb["tt1135300"]`
2. The title: `m.omdb["Alien"]`
3. A search expression: `m.omdb[{query title="Dollhouse"}]`

Luckily, the movie IDs all start with “tt”, so we can use pattern matching to distinguish IDs from titles.

For the ID we recognize 2 situations - with or without a previously found value:

```
% Syntax: m.omdb["tt1135300"]
m_get([ <<"tt", _/binary>> = Id | Rest ], _Msg, Context) ->
    Query = [ {id, Id} ],
```

```

    {ok, {fetch_data(Query), []}};

% Syntax: m.omdb.sometype["tt1135300"]
m_get([ <<"sometype">>, <<"tt", _/binary>> = Id | Rest ], _Context) ->
    Query = [ {type, sometype}, {id, Id} ],
    {ok, {fetch_data(Query), []}}.

```

We need to place these two patterns above the title searches we already wrote

fetch_data will return a property list, so we can write this to get all values:

```

{% for k,v in m.omdb["tt1135300"] %}
    {{ k }}: {{ v }}
{% endfor %}

```

Handling the title is similar to the ID. Title must be a string, otherwise it would be a property key (atom):

```

% Syntax: m.omdb["some title"]
% If no atom is passed it must be a title (string)
m_get([ Title | Rest ], _Context) when is_binary(Title) ->
    Query = [ {title, Title} ],
    {fetch_data(Query), []};

```

To parse the search expression, we can simply use the readymade property list:

```

% Syntax: m.omdb[{query QueryParams}]
% For m.omdb[{query title="Dollhouse"}], Query is: [{title,"Dollhouse"}]
m_get([ {query, Query} | Rest ], _Context) ->
    {fetch_data(Query), []};

% Syntax: m.omdb.sometype[{query QueryParams}]
% For m.omdb.series[{query title="Dollhouse"}],
% Query is: [{title,"Dollhouse"}] and Q is: [{type,"series"}]
m_get([ <<"series">>, {query, Query} | Rest ], _Context) ->
    {fetch_data([ {type, series} | Query ], []);

```

If we want to fetch the year of the first result we use:

```
m.omdb["Alien"].year
```

... we get called as:

```
m_get([ <<"Alien">>, year ], _Msg, Context).
```

Which (after a search on the title “Alien”) returns:

```
{ok, {SomeSearchResultList, [ year ]}}.
```

The [year] will then be used to lookup the year property of the found result.

We won’t do any validity checking on the parameter here, but for most modules it makes sense to limit the possibilities. See for instance how m_search:get_result is done.

Full source code

The source code of the documentation so far can be found in this gist: [Zotonic 1.0 - Template model for the OMDB movie database - source code to accompany the documentation.](#)

Possible enhancements

For a complete model for this API, I would expect:

- Data caching to speed up identical calls
- Support for all API parameters
- Better error handling (the service might be down or return wrong data)

Create a custom controller

See also:

Controllers (page 20) in the Developer Guide

Zotonic comes with a large collection *controllers* (page 20) that cover many use cases, so you'll probably have to resort to custom controllers less often than you may be used to from other web frameworks. Still, the time may come when you need to process HTTP requests in your own way.

You can do so by creating a custom *controller* (page 20). Create a module in your site's `src/controllers/` directory and prefix it with `controller_`:

Listing 4.2: `yoursite/src/controllers/controller_say_hello.erl`

```
-module(controller_say_hello).  
  
-export([  
    process/4  
]).  
  
%% This function renders some HTML when the controller is called  
process(_Method, _AcceptedContentType, _ProvidedContentType, Context) ->  
    {<<"Hello world and all the people in it!">>, Context}.
```

To be able to handle HTTP requests with this controller, you need to *define a dispatch rule* (page 22) that maps some request URL to this controller:

Listing 4.3: `yoursite/priv/dispatch/dispatch`

```
[  
    {say_hello, ["hello"], controller_say_hello, []}  
].
```

Now, if you go to `https://yoursite.test:8443/hello` in your browser, you will see the output of the `controller_say_hello` controller.

Handling POST requests

Now you've seen how to handle GET requests, let's turn to POST requests. First, your controller must show that it can handle POSTs. You do so by adding an `allowed_methods/1` function:

Listing 4.4: `yoursite/src/controllers/controller_say_hello.erl`

```
-module(controller_say_hello).  
  
-export([  
    %% ...  
    allowed_methods/1  
]).  
  
%% ...  
  
%% This controller will handle only GETs and POSTs  
allowed_methods(Context) ->  
    {[<<"GET">>, <<"POST">>], Context}.
```

The `process/4` function will be called with the POST data, so define it:

Listing 4.5: `yoursite/src/controllers/controller_say_hello.erl`

```
-module(controller_say_hello).

-export([
    allowed_methods/1,
    process/4,
]).

allowed_methods(Context) ->
    [{<<"GET">>, <<"POST">>}, Context].

process(<<"GET">>, _AcceptedContentType, _ProvidedContentType, Context) ->
    [<<"Hello world and all the people in it!">>, Context];
process(<<"POST">>, _AcceptedContentType, _ProvidedContentType, Context) ->
    % Process the POST data
    Name = z_html:escape( z_context:get_q(<<"name">>, Context, <<>>) ),
    [<<"Thank you posting, ", Name/binary>>, Context].
```

Try it out on the command line:

```
$ curl -k -v -X POST -d 'name=David' https://yoursite.test:8443/hello

# prints:
Thank you posting, David
```

Custom pivots

See also:

[pivot.name](#) (page 49) search argument for filtering on custom pivot columns.

See also:

[Pivot Templates](#) (page 324) to change the content of regular pivot columns and search texts.

[Search](#) (page 44) can only sort and filter on [resources](#) (page 26) that actually have a database column. Zotonic's resources are stored in a serialized form. This allows you to very easily add any property to any resource but you cannot sort or filter on them until you make database columns for these properties.

The way to take this on is using the “custom pivot” feature. A custom pivot table is an extra database table with columns in which the props you define are copied, so you can filter and sort on them.

Say you want to sort on a property of the resource called `requestor`.

Create (and export!) an `init/1` function in your site where you define a custom pivot table:

```
init(Context) ->
    z_pivot_rsc:define_custom_pivot(pivotname, [{requestor, "varchar(80)"}],
    Context),
    ok.
```

The new table will be called `pivot_<pivotname>`. When you change the column names in the table definition, the table will be recreated and **the data inside will be lost**.

To fill the pivot table with data when a resource gets saved, create a notification listener function `observe_custom_pivot/2`:

```
observe_custom_pivot(#custom_pivot{ id = Id }, Context) ->
    Requestor = m_rsc:p(Id, requestor, Context),
    {pivotname, [{requestor, Requestor}]}.
```

This will fill the ‘requestor’ property for every entry in your database, when the resource is pivoted.

Recompile your site and restart it (so the `init` function is called) and then in the admin under ‘System’ -> ‘Status’ choose ‘Rebuild search indexes’. This will gradually fill the new pivot table. Enable the logging module and choose “log” in the admin menu to see the pivot progress. Once the table is filled, you can use the pivot table to do sorting and filtering.

To sort on ‘requestor’, do the following:

```
{% with m.search.paged[{query cat='foo' sort='pivot.pivotname.requestor'}] as _  
  ↪result %}
```

Or you can filter on it:

```
{% with m.search.paged[{query filter=["pivot.pivotname.requestor", `=` , "hello"]}]  
  as result %}
```

Create a custom tag

Custom tags, internally called *scomps*, are module-defined tags, which are used when the logic is too complex to be executed in templates.

Custom tags add logic to templates or generate HTML/Javascript constructs that are too difficult for templates. A good example is the [menu](#) (page 543) scomp which implements the menu, including sub menus -and highlighting of the current menu item.

Implementing a custom tag

A scomp module lives in a module under the `scomps/` folder, and has the following naming convention: `scomp_modulename_scompname`.

So a fictitious scomp called `likebutton` in the `mod_facebook` module would be defined in the Erlang module called `scomp_facebook_likebutton.erl`.

You can implement scomps by using the *scomp* behaviours. An example scomp that does *absolutely nothing* is implemented as follows:

```
-module(scomp_facebook_likebutton).  
-behaviour(gen_scomp).  
-export([vary/2, render/3]).  
-include_lib("zotonic_core/include/zotonic.hrl").  
  
vary(_Params, _Context) -> nocache.  
render(Params, _Vars, Context) ->  
    %% FIXME: render like button here  
    {ok, Context}.
```

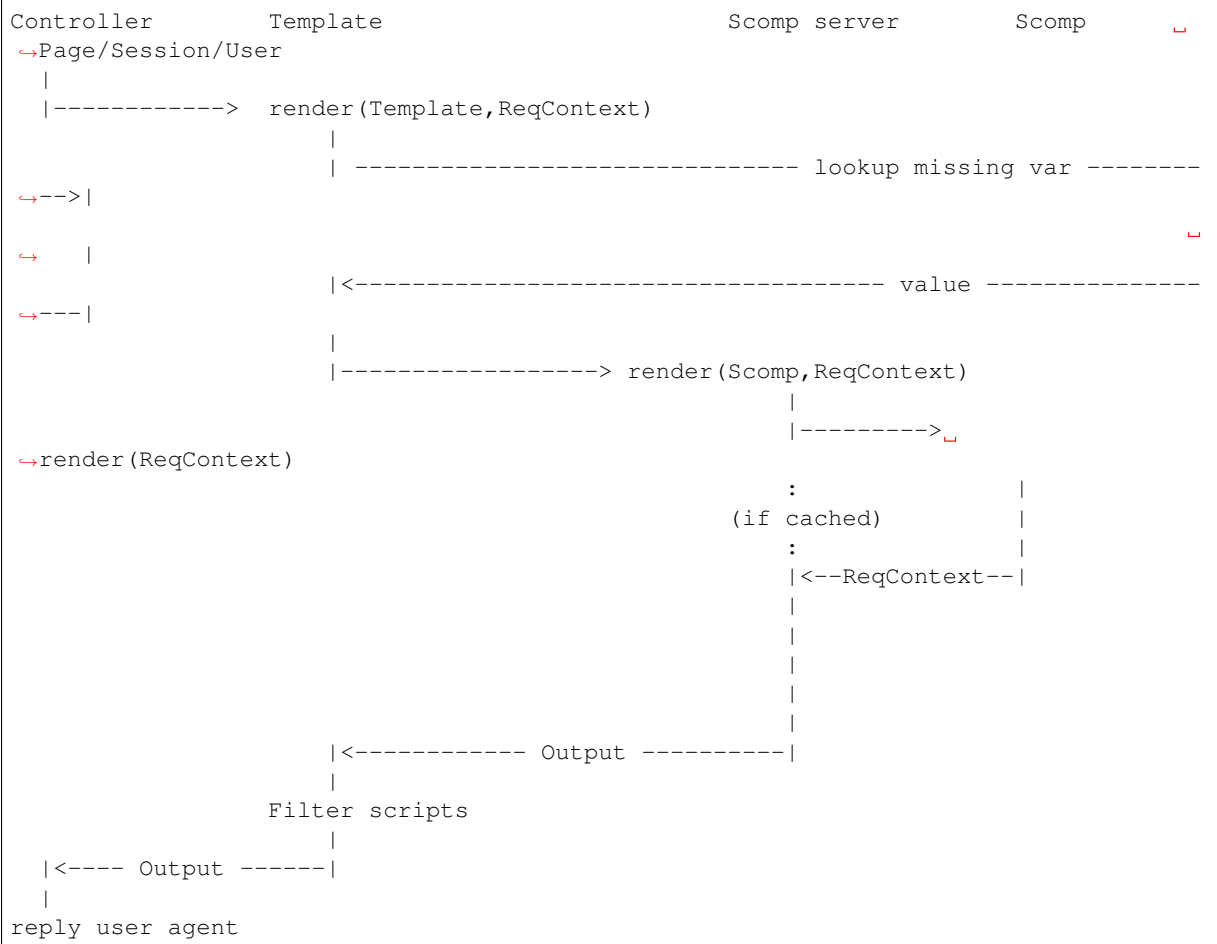
Scomp internals

During the evaluation of scomps we are able to:

- wire actions
- add scripts
- add validators
- add context variables (accessible only by other scomps)

After rendering a template it is inspected to find all scripts, actions and validators. They are placed as #context records in the resulting iolist, the #contexts should be replaced by the 'render' record-attribute. There will be a special 'script' atom to signify the place where all collected scripts will be inserted. When the script atom is not found, then the result of the template is a context, unless there are no collected scripts.

Process state diagram, how all processes work together to render a scomp:



The scripts/actions/validators are similar to the ones defined with Nitrogen, though the record structure is redefined to accomodate easy construction by the template compiler.

Custom search

Implement a custom search by observing the [search_query](#) (page 619) notification in your module. Imagine you want to search cookies in your database that either have chocolate or do not:

```

-include_lib("zotonic_core/include/zotonic.hrl").

-export([
    observe_search_query/2
]).

observe_search_query(#search_query{search = Search, offsetlimit = OffsetLimit},
    Context) ->
    %% Pass on to an internal function.
    search(Search, OffsetLimit, Context).

search({cookies, [{chocolate, Boolean}]}, _OffsetLimit, Context)
    %% Handle search queries of the cookie type.
    #search_sql{

```

```
select="c.id",
from="foo c",
where="c.chocolate = $1",
order="c.name desc",
args=[Boolean],
tables=[{cookies_table, "c"}]
};
search(_Other, _OffsetLimit, _Context) ->
  %% Let other modules handle other types of search queries.
undefined.
```

Do not forget to add the last function that matches and returns `undefined` for any other search request. If you forget this, the notification fold will crash when using any other search query.

This can then be used in your template like this:

```
{% for id in m.search[{cookies chocolate=true}] %}
  Looping over all ids in the 'cookies_table' table for which chocolate = true
{% endfor %}
```

Pivot Templates

Search (page 44) uses database indices on special pivot columns and full text fields.

These columns and text fields are extracted in a process called *pivoting*. This is done a short period after a resource has been updated.

The content of the pivot columns is determined by a template. This template is called `pivot/pivot.tpl` (in `mod_base`) and is rendered using a *catinclude*. This makes it possible to have your own unique indexing per category.

The `pivot/pivot.tpl` template consists of multiple blocks. Each block corresponds to a pivot column. The block is rendered for determining the pivot's content.

The following blocks are defined:

a, b, c and d

These blocks are used for the full text index. Block *a* contains more important texts than block *b*, *c* or *d*. Per default the title and other essential texts are shown in block *a*, and block *d* is used for related texts (think titles of connected resources).

The default pivot template uses the following *catincluded* templates for the different blocks:

- `pivot/_title_text.tpl` for block *a*
- `pivot/_main_text.tpl` for block *b*
- `pivot/_block_text.tpl` for block *c*
- `pivot/_related_text.tpl` for block *d*

title

The title used for database sorting etc. Per default the lowercased title in the selected pivot language or the default site language.

related_ids

The ids of all related resources and categories. This is a full text index used for finding resources that are similar to some other resource. All matching resource ids should be prefixed with `zpo` and category ids with `zpc` (for example `zpo1234`).

The *catincluded* template `pivot/_related_ids.tpl` is used for extracting these ids. This template also adds the id of the content group as a related resource.

address_street, address_city, address_postcode, address_state, address_country

The address to be used for searches. Defaults to the related `address_...` properties with a fallback to the `mail_address_...` properties.

The country should be the two letter ISO code, not the country's descriptive text.

name_first, name_surname

If the resource is a person then this should contain the first name and surname.

gender

The gender of the person, currently a single letter. For example: `f`, `m` or `?`.

date_start, date_end

The start or end date associated with the resource. This should be in a parseable format, which will be done automatically when echoing a date directly. Make sure UTC is used as the timezone (note that the timezone will be set to UTC when pivoting).

date_start_month_day, date_end_month_day

A four digit number representing the month and day for the date start and date end. Think of an index for birthdays, without exposing the birth year.

Defaults to `{{ id.date_start|date:"md" }}` and `{{ id.date_end|date:"md" }}`.

location_lat, location_lng

The latitude and longitude of the resource location. This should be empty or a floating point number.

Defaults to `{{ id.computed_location_lat|default:id.location_lat }}` and `{{ id.computed_location_lng|default:id.location_lng }}`

date_repivot

The date and time this resource should be repivoted. This is useful if some resources (like events) should be indexed with lower priorities after a certain date. A pivot task will be scheduled at the specified date. Per default no repivot will be scheduled.

See [Custom pivots](#) (page 321) for adding your own custom pivot columns.

Writing your own module

Todo

Write this cookbook chapter

Overriding Zotonic

This chapter describes how to override the templates, styling and logic provided by Zotonic.

Overriding works by adding a site or module that has a higher *priority* (page 63) than Zotonic's built-in modules. In your module/site, you add templates, assets and create notification observers.

Overriding templates

See also:

[How to customise error pages](#) (page 330)

[Override templates](#) (page 30) by adding a template with the same name to your module/site.

Overriding assets

If you wish to fully override a CSS or JavaScript file, do so in the same way as you do templates: create a file with the same name in your module/site. Alternatively, *add* your own CSS file and selectively override CSS styles.

Overriding logic

Observe notifications (page 70) to influence the decisions that Zotonic makes. You can change or add properties before a resource is persisted,

Execute tasks asynchronously using the task queue

The Zotonic task queue lets applications perform tasks asynchronously.

Let's say you have some external HTTP API that you want to update whenever a resource in Zotonic is changed. You can do so by queuing a task after each resource update. The HTTP request to the external API will be executed asynchronously, so your application and its users do not have to wait for it.

Add a task to the queue

To add a task to the queue, provide a module and function that should be called when the task is popped from the queue. So to add a task that will call the `external_api_client:update_external_rsc()` as a callback function:

```
z_pivot_rsc:insert_task(  
    external_api_client,  
    update_external_rsc,  
    Context  
) .
```

You can also supply arguments that will be passed to the function. So to have `external_api_client:update_external_rsc(RscId)` called:

```
RscId = 123,  
z_pivot_rsc:insert_task(  
    external_api_client,  
    update_external_rsc,  
    undefined,  
    [RscId]  
    Context  
) .
```

If you want to queue the task whenever a resource is changed, add this code to an *rsc_update_done* (page 604) observer in your *site module* (page 60):

```
%% yoursite.erl  
module(yoursite).  
  
-export([  
    observe_rsc_update_done/2  
) .  
  
observe_rsc_update_done(#rsc_update_done{id = RscId}, Context) ->  
    z_pivot_rsc:insert_task(  
        external_api_client,  
        update_external_rsc,  
        undefined,  
        [RscId]
```

```
Context
).
```

Execute queued tasks

Note: Your callback function receives an anonymous Context, so if it performs actions that depend on access checks, make sure to use either `m_rsc:p_no_acl/3` or `z_acl:sudo(Context)`.

Add the callback function `update_external_rsc/2` referenced above to your module:

```
%% external_api_client.erl
-module(external_api_client).

-export([
    update_external_rsc/2
]).

update_external_rsc(RscId, Context) ->
    %% Fetch resource properties
    {ok, JSON} = m_rsc_export:full(Id, Context),
    Data = jsxrecord:encode(JSON),

    %% Execute HTTP POST to external API
    {ok, Response} = httpc:request(
        post,
        {
            "https://some-external-api.com",
            [],
            "application/json",
            Data
        },
        [],
        []
    ),

    %% Return anything to signal the task was executed successfully
    ok.
```

Handle failing tasks

The `update_external_rsc` function above assumes that the HTTP request will return successfully. Of course, this is not always the case. To handle failing tasks, you can return a `{delay, NumberOfSeconds}` tuple that will retry the task later:

```
update_external_rsc(RscId, Context) ->

    case httpc:request(
        ...
    ) of
        {ok, Response} ->
            ok;
        {error, Error} ->
            %% Try the task again in one minute
            {delay, 60}
    end.
```

Prevent duplicate tasks

We decided above that the task should run whenever a resource is changed in Zotonic. However, if a resource is quickly edited multiple times in a row, we only need to send the latest changes once to the external API. In other words, we want to coalesce the tasks into one. You can do so by providing a unique key when queueing the task:

```
UniqueKey = "external-api-" ++ z_convert:to_list(RscId),
z_pivot_rsc:insert_task(
  external_api_client,
  update_external_rsc,
  UniqueKey,
  [RscId],
  Context
).
```

Icons in templates

See also:

Icons (page 34) reference

Zotonic provides a couple of ways to show icons in templates:

- *mod_artwork* (page 357) gives access to FontAwesome and Material Design icons. It also has a number of other icon collections, mostly PNG images. Activate the module and follow the instructions on the doc page.
- Zotonic icons provided by *mod_base*. This is explained on the current page.

To create a certain amount of consistency across modules, Zotonic comes with a small set of commonly used icons and CSS classes (edit, help, close, etcetera) plus the Zotonic logo.

Use cases:

- You create your frontend from scratch, but you also have pages in your site that are provided by other modules, for instance the login screens. It would be good if the social login icons show up.
- You are writing a template or module and like to take advantage of ready available icons.
- You are writing frontend styles in LESS and you would like to extend Zotonic / FontAwesome / Material Design icons.

Include the Zotonic icons CSS file in your template:

```
{% lib
  "css/z.icons.css"
%}
```

Then use this syntax in your template HTML:

```
z-icon z-icon-<name>
```

For instance:

```
<span class="z-icon z-icon-off"></span>
```

Logging to Logstash

See also:

the *Logging* (page 80) chapter in the Developer Guide.

Logstash is often used for log centralization and analysis. This cookbook describes how to set up Zotonic for logging to Logstash over UDP. As mentioned in the *Logging chapter* (page 80), Zotonic uses *Logger*.

So we will change Zotonic's Logstash configuration in order to send messages to Logstash.

Step 1: add a Logstash handler

Zotonic comes with the *logstash handler* `logstasher_h` for logger. The handler will be started automatically if it is configured as a logger handler in *The erlang.config file* (page 625):

Listing 4.6: erlang.config

```
{kernel, [
  % Minimum log level for all loggers below.
  {logger_level, info},

  {logger, [

    %% To use logstash:
    %% - Enable the logstasher_h handler
    %% - Configure logstasher (see below the kernel config)
    %%
    {handler, logstash, logstasher_h,
      #{
        level => info
      }
    },

    %%% Other logger configs here
    ...
  ]}
]},
```

Step 2: configure the Logstash handler

The next step is to tell the Logstash handler where it should send its messages to. The configuration of `logstasher` in the *The erlang.config file* (page 625) can be found below the kernel section or else added there:

Listing 4.7: erlang.config

```
%% Logstash configuration.
%% If a logger handler with 'logstasher_h' is defined then zotonic_core will start_
↳the
%% logstasher application.
{logstasher, [
  {transport, udp},      % tcp | udp | console
  {host, "localhost"},  % inet:hostname()
  {port, 5000}          % inet:port_number()
]},
```

Replace `logger` with the hostname or IP address of your logger. IP addresses can be configured as an Erlang tuple, for example: `{127,0,0,1}`

If the console output is shipped to Logstash, then use `console` as the transport. The transport `console` ignores the configurations `host` and `port`.

After you changed the `erlang.config` file you will need to restart Zotonic.

You should now find all Zotonic log messages in Logstash. To test this, just call:

```
logger:error_msg("Just testing the Logstash setup here!").
```

How to customise error pages

Specific error pages

Zotonic's *controller_http_error* (page 443) first tries to find an error page template that is specific for the HTTP status code, named `templates/error.status_code.tpl`. So, to override the 404 template, create a file `templates/error.404.tpl`.

Generic error page

If no specific error page template exists, Zotonic falls back to the generic `templates/error.tpl`.

Exometer metrics

Zotonic comes with a system for collecting and exporting metrics (such as how much memory is used, how many database requests were made, etc.) called Exometer. This cookbook details how to make use of that system to report on your Zotonic server's activity.

Step 1: get a Time Series Database (TSDB)

At the time of writing, Exometer in Zotonic supports sending metrics to Graphite, OpenTSDB and StatsD, as well as being able to send them to another system using AMQP or polling through SNMP. The installation of such a system is outside the scope of this document, but if you want to experiment without setting up a TSDB for yourself, [Hosted Graphite](#) has a 14-day free trial to play with.

Step 2: configure Exometer to report to your TSDB

After deciding on a TSDB, you need to configure Exometer to connect to the TSDB. This is done by configuring a *reporter* in your `erlang.config`.

The default `erlang.config` comes with a number of pre-defined exometer metrics:

```
{exometer_core, [{predefined, [
    {[erlang, memory], {function, erlang, memory, [], value, []}, []},
    {[erlang, system_info], {function, erlang, system_info, ['$dp'], value,
↳ [process_count]}, []},
    {[erlang, statistics], {function, erlang, statistics, ['$dp'], value, [run_
↳ queue]}, []},
    {[erlang, io], {function, erlang, statistics, [io], match, {{['_', input}, {'_',
↳ output}}}}, []}
  ]}
]},
```

To configure a reporter, add a block called `{reporters, []}` and a block inside that for the reporter. For example, to report to a graphite server, the following configuration could be used:

```
{reporters, [
  {exometer_report_graphite, [
    {connect_timeout, 5000},
    {prefix, "zotonic"},
    {host, "graphite.server.com"},
    {api_key, ""},
  ]}
]}
```

```
{port, 2003}
  }}
}}
```

For testing, it is also possible to simply send the output to the console. For this, use the `exometer_report_tty` reporter, like so:

```
{reporters, [
  {exometer_report_tty, []}
]}
```

Other reporters will require different configuration. For more examples, see the [Exometer documentation](#). It is possible to configure multiple reporters at the same time; you can then select what metrics are sent to what reporter with subscriptions.

Step 3: tell Exometer what to report, and how often

Now that Exometer knows *where* to send our metrics, we need to tell it *what* to send, and when to do it. This is done with a *subscription* and, like reporters, can be configured in `erlang.config` in a block called `{subscribers, []}`. The general format of a subscription is:

```
{reporter, metric, datapoint, interval}
```

Metrics have a hierarchical name consisting of a list of terms plus a datapoint name. For example, the number of requests made to Zotonic's status page is named:

```
[zotonic, zotonic_status, webzmachine, requests]
```

and the datapoint name is 'value'. So to report this metric to graphite every minute, we would add a subscription as follows:

```
{subscribers, [
  {exometer_report_graphite, [zotonic, zotonic_status, webzmachine, requests],
    ↪value, 60000}
]}
```

Since the metric name includes the site name, it may be preferable to set up generic subscriptions instead, that would apply to all sites in the system. This can be done by using a special syntax in place of the metric name:

```
{select, [{matchspec}]}
```

Where `matchspec` is an [ETS Match Specification](#). The specifics of the Match specification can be quite complex, but we can use simple ones to get going. To select all zotonic metrics of type counter and send them to graphite every minute, you would enter the following:

```
{exometer_report_graphite, {select, [{ {[zotonic | '_'], counter, '_'}, [], ['$_']
    ↪}}]}, value, 60000}
```

To send only metrics for a specific site, you can add the site name:

```
{exometer_report_graphite, {select, [{ {[zotonic, sitename | '_'], counter, '_'},
    ↪[], ['$_'] }]}}, value, 60000}
```

Using the Zotonic Shell

It is possible to set up exometer from the Zotonic shell. As this run-time configuration is lost upon restart, it is best to only use this for testing or to enact configuration file changes without a restart. A number of uses for the shell are shown below.

To test a match specification. Take the matchspec part of the metric name and feed it to `exometer:select()`. For example:

```
(zotonic001@server)2> exometer:select([{{ [zotonic, zotonic_status | '_'], counter,
↳ '_'], [], ['$_'] }}]).
[[[zotonic,zotonic_status,db,requests],counter,enabled], ...
```

To find out what datapoints a metric contains, you can ask for its information:

```
(zotonic001@server)2> exometer:info([zotonic,zotonic_status,webzmachine,requests],
↳datapoints).
[value,ms_since_reset]
```

To test a subscription, you can manually add it:

```
(zotonic001@server)2> exometer_report:subscribe(exometer_report_tty, {select, [
↳{{[zotonic, zotonic_status | '_'], counter, '_'], [], ['$_'] }}}, value, 60000).
ok
```

Complete configuration example

A complete configuration might look like the following:

```
{exometer, [
  {predefined, [
    {erlang, memory}, {function, erlang, memory, [], value, [], []},
    {erlang, system_info}, {function, erlang, system_info, ['$dp'], value,
↳[process_count]}, [],
    {erlang, statistics}, {function, erlang, statistics, ['$dp'], value, [run_
↳queue]}, [],
    {erlang, io}, {function, erlang, statistics, [io], match, {{['_', input], {
↳['_', output]}}}, []}
  ]},
  {reporters, [
    {exometer_report_graphite, [
      {connect_timeout, 5000},
      {prefix, "zotonic"},
      {host, "graphite.server.com"},
      {api_key, ""},
      {port, 2003}
    ]}
  ]},
  {subscribers, [
    {exometer_report_graphite, {select, [{{ [zotonic | '_'], counter, '_'], [],
↳ ['$_'] }}}, value, 60000},
    {exometer_report_graphite, {select, [{{ [zotonic | '_'], gauge, '_'], [], [
↳ '$_'] }}}, value, 60000},
    {exometer_report_graphite, {select, [{{ [erlang, memory, '_'], '_'], [], [
↳ '$_'] }}}, value, 60000},
  ]}
]},
```

4.1.5 Shell cookbook

Activate/deactivate modules

Rescuing a dysfunctional site from the Zotonic shell.

Why

Sometimes it happens that disabled or enabled a module by accident and your whole site is now dysfunctional. You can easily rescue your site, from the Erlang command line.

How

First launch the Zotonic EShell:

```
~/zotonic/bin/zotonic shell
```

On the Zotonic EShell: Deactivate a module:

```
z_module_manager:deactivate(mod_modulename, z:c(yoursitename)).
```

On the Zotonic EShell: Activate a module:

```
z_module_manager:activate(mod_modulename, z:c(yoursitename)).
```

On the Zotonic EShell: Restart a module:

```
z_module_manager:restart(mod_modulename, z:c(yoursitename)).
```

Where *mod_modulename* is the name of your module and *yoursitename* is the name of your site.

Check Your Hostname

If you can't launch the shell chances are that your hostname is not a fully-qualified domain name (FQDN). This complicates things since Erlang is picky about allowing connections to unqualified nodes. For example, the host-name should be `myprimarysite.example.com` rather than something random and local like `coolserver`.

Filter and convert characters

Applying Erlang Binary syntax to get fast character manipulation.

- *Author: Lloyd R. Prentice*
- *Co-Author: Andreas Stenius*

Why

Erlang bit syntax is extraordinarily powerful and well worth learning.

Review documentation here: http://www.erlang.org/doc/programming_examples/bit_syntax.html

Here's an experiment. Follow along in your handy Erlang shell:

```
1> A = "The cat on the mat".
"The cat on the mat"
2> B = <<"The cat on the mat">>.
<<"The cat on the mat">>
3> io:format(A, []).
The cat on the matok
4> io:format(B, []).
The cat on the matok
So, big deal. What's the difference?
```

Memory consumption, that's what. A is represented internally as a list. Each character is a memory cell with a character and a pointer to the next cell. The list format is a memory hog. List representation of text strings is both bane and blessing for Erlang programmers. B, on the other hand, is mapped in memory as a contiguous array of characters, thus consuming much less memory.

For details on working with Erlang Binaries, look here: <http://www.erlang.org/doc/man/binary.html>

When you want to filter and convert characters in an Erlang Binary, however, things get dicey. It's all there in the Erlang documentation, but examples are few and far between.

The purpose of this particular recipe is to prepare text for conversion from a *.txt file to *.tex, a file marked up for input to the truly wonderful open source typesetting program LaTeX.

In our case, we don't have control of what gets typed into the source text file. So we need to both filter for unwanted characters and sequences as well as convert characters and sequences into LaTeX markup.

What does this have to do with Zotonic?

You'll just have to wait and see.

Assumptions

All you'll need here is a text editor to copy and paste an Erlang module and an Erlang shell.

A passing understanding of Erlang list comprehensions will also be useful. Find more here: http://www.erlang.org/doc/programming_examples/list_comprehensions.html

How

Study latexFilter/1 below:

```
-module(filterz).
-export([latexFilter/1]).

%% Binary filters
%% with many thanks to Andreas Stenius
latexFilter(B) ->
    %% Delete control characters and extended ASCII
    B1 = << <<C>> || <<C>> <= B, (C == 10) or (C == 13) or (C >= 31),
    (C <= 127) >>,
    %% Filter binary for conversion to *.tex format
    %% NOTE: Partially tested, but needs more
    F = [{"\r", "\n"}, % Convert returns to new lines
         {"\n *", "\n"}, % Delete spaces following newline
         {"^\\s*", ""}, % Delete all whitespace
         {"^\\n*", ""}, % Delete new lines at
         {"\\n+$", "\n"}, % Delete excess new lines
         {" +", " "}, % Delete successive spaces
         {"\\n{3,}", "\\n\\\\\\bigskip\\n"}, % Convert three or more
         {"[&#$%~^{}]", "\\&"}, % Escape reserved Latex character
         {"<i>", "\\emph{"}, % Convert HTML tag to Latex tag
         {"</i>", ""}, % Convert HTML tag to Latex tag
         {"\"(.*)\"", "\\1"}, % Convert opening double quotes (")
    ]
    to Latex conventions (``)
    {"' '", "\\1"}, % Convert opening single quotes (')
    B2 = lists:foldl(fun({Pattern, Replacement}, Subject) ->
        re:replace(Subject, Pattern, Replacement,
```

```
[global, {return, binary}]] end,
    B1, F),
    {ok, B2}.
```

The first thing that happens in `latexFiler/1` is an Erlang binary comprehension to delete pesky control and extended ASCII characters:

```
B1 = << <<C>> || <<C>> <= B, (C == 10) or (C == 13) or (C >= 31), (C <= 127) >>,
```

Note that the syntax is very similar to list comprehensions. As you can see, a single binary comprehension can chug out a lot of work. Define a few binaries in your Erlang shell and play around with the conditionals. You'll catch on pretty quick.

For our purposes, we need to also search and replace a bunch of substrings. For this we've enlisted `lists:foldl/3`, a powerful list function that just happens to work with binaries. See: <http://www.erlang.org/doc/man/lists.html#append-1>

Expect a headache when you study this function. It's not that easy to understand.

The `fold` function exists in two versions, `foldl` and `foldr` (for left and right, described shortly). It takes a list, and calls a function for each item in the list, along with an accumulator, or state. The function can operate on the item and state, producing a new state for the next round. The fold function also takes an initial state to use for the first item. The left and right mentioned previously determines in which order the list is traversed. So `foldl` moves through the list from left to right, i.e. takes the head off of the list for each iteration moving towards the tail; whereas the `foldr` starts with the tail and moves towards the head, i.e. from right to left. Regular expressions can also get hairy, but they're invaluable if you're working with text strings. For more information:

- <http://www.troubleshooters.com/codecorn/littpperl/perlreg.htm>
- <http://www.erlang.org/doc/man/re.html>

Erlang tab completion

Get quicker access to Zotonic code on the shell.

Contributed by: Maas-Maarten Zeeman

Why

When you are working on the shell chances are you are looking to be fast. A way to be faster is to leverage the EShell's tab completion.

Assumptions

Readers are assumed to be comfortable using the EShell to run Erlang code interactively. This feature this guide describes only works with Zotonic modules only from `zotonic debug`. With the `zotonic shell` command, the Zotonic modules need to be explicitly loaded to work with tab-completion.

How

The Erlang shell has tab-completion. It is very handy for using in Zotonic, especially because of Zotonic's naming scheme. Just type:

```
1> z_<tab>
```

And you get a list of zotonic library modules. Type:

```
2> mod_<tab>, type m_<tab> a list of models.
```

You get the idea. The really nice thing is that it works for function names too.

Debugging db (query) issues

Techniques for finding root cause when queries are involved.

Why

When you face unexpected behavior as a result of some database query (z_db:q et al), you either have to hunt down the queries and re-run them by hand, which is really time consuming and painful, or, which this cookbook will detail, have postgresql log the queries for you.

The principal documentation and with more details, may be found here: <https://www.postgresql.org/docs/9.5/static/runtime-config-logging.html>

Assumptions

A working system where you want to inspect the db queries being run.

How

Edit your `postgresql.conf` file, enabling `logstatement = 'all'`. This file has a lot of options commented out, with comments which you may review. Options of interest in this scenario all begin with “log” (or similar).

By having `log_destination = 'stderr'` and `logging_collector = on`, you can capture your logging output to file.

Reset a user's password

Emergency password reset when you can't get into the admin interface.

Why

Sometimes it happens that you want to reset an user's password from the Erlang shell.

Assumptions

Readers are expected to be familiar with the EShell for running Erlang code interactively.

How

You can do this from the Erlang shell without using the `/admin` or the `reset-password` mail/dialog.

First go to the Erlang shell:

```
marc$ ./bin/zotonic shell
```

And then from the Erlang command prompt:

```
(zotonic@host) 1> m_identity:set_username_pw(1234, "username", "password", z_
↪acl:sudo(z:c(yoursitename))).
```

Where *1234* is the id of your user (this must be an integer), *yoursitename* is the name of your site.

Troubleshooting

If you get the error:

```
{error, admin_password_cannot_be_set}
```

That means you are trying to change the password for the admin (user 1). The admin password is not set in the database: you need to define your admin password in the site's config file, use the property `admin_password`. For more info on this, see *Anatomy of a site* (page 18).

Restore/upgrade content db from backup

Contributed by: Scott Finnie

How to protect your Zotonic content and bring it back in case of disaster.

Why

Restore / upgrade the content for a site from a database dump.

Assumptions

You have a dump of the content to restore as a `.sql` file, which was created by *mod_backup* (page 360).

How

Dumps from postgres include both database structure (tables, sequences, constraints, etc.) as well as content. Therefore it's not possible to simply import the dump directly into an existing Zotonic database populated with content. The target database must be empty before the dump can be imported. There are (at least) 2 ways to do this:

1. Drop contents of the target db and import the dump file;
2. Create a new database, import contents, rename old database to something temporary, rename new database to match site, and finally delete the old database.

Option 2 seems more involved but is safer and quicker: it's non-destructive. The old db remains intact until after the new one is activated. If anything goes wrong, you can fall back to the original.

It also means less downtime. The site can stay live on the old db while the new db is being created. Downtime is restricted to the rename operations (which are quick and db size independent).

For small databases the downtime difference will be minimal. However the safety is appealing. Hence this cookbook uses the second approach.

Instructions

These assume the following naming conventions:

- Your Zotonic site is named *yoursite*.

- The postgres dump file to load is named zotonic-dump.sql.
- Zotonic is installed in ~zotonic/zotonic

Replace these assumptions as appropriate in all commands below. Ensure the db dump file is available on target machine (see pre-requisites).

Create a new database in postgres:

```
zotonic:~$ sudo su -l postgres
postgres:~$ ~zotonic/zotonic/bin/zotonic createdb yoursite_new
CREATE DATABASE
GRANT
You are now connected to database "zotonic_yoursite_new".
CREATE LANGUAGE
postgres:~$
```

Import the dump file into the newly created db:

```
postgres:~$ psql -U zotonic -d zotonic_yoursite_new -f /path/to/zotonic-dump.sql
[...lots of SQL statements...]
postgres:~$
```

Note: the final sql commands shown will likely say ‘no privileges could be revoked for “public”’. You can ignore this.

Stop Zotonic (if running):

```
postgres:~$ ~zotonic/zotonic/bin/zotonic stop
Stopping zotonic zotonic001@hostname
Stop:'zotonic001@hostname'
postgres:~$
```

Rename the databases:

```
postgres:~$ psql
postgres=# ALTER DATABASE zotonic_yoursite RENAME TO zotonic_yoursite_old;
ALTER DATABASE
postgres=# ALTER DATABASE zotonic_yoursite_new RENAME TO zotonic_yoursite;
ALTER DATABASE
postgres=# \q
postgres:~$ exit
zotonic:~$
```

Restart Zotonic:

```
zotonic:~$ ~/zotonic/bin/zotonic start
```

Browse to your site & test it's now serving updated content.

(Optional) Drop the old database:

```
zotonic:~$ sudo -u postgres psql
postgres=# DROP DATABASE zotonic_yoursite_old;
DROP DATABASE
postgres=# \q
zotonic:~$
```

Best practices as formulated by the Zotonic core team.

5.1 Best Practices

5.1.1 Creating sites

Initialize your site with a proper data model and some resources through `manage_schema`.

5.1.2 Media Best Practices

- Use *Media classes* (page 33) to define image properties.

5.1.3 Template Best Practices and Pitfalls

This chapter lists some preferred solutions to common tasks and pitfalls you may encounter while developing with templates.

Variable Naming

Name your variables for what they represent. If you are searching for articles, name the search result variable `article` to make things clear.

In particular, if you are iterating over a list or other iterable variable called `images`, then your item variable should be named `image`. This makes generative templates easier to follow.

Use `id` instead of `m.rsc`

For the most frequently used model *m_rsc* (page 571), use a shortcut syntax: write `{{ id.title }}` instead of `{{ m.rsc[id].title }}`. So this:

```
<li>
  <h3>{{ m.rsc[id].title }}</h3>
  {% for image_id in m.rsc[id].o.depiction %}
  <figure>
    {% media image_id width=100 link=id %}
    {% if m.rsc[image_id].summary %}
    <p class="image-caption">{{ m.rsc[image_id].summary }}</p>
    {% endif %}
  </figure>
  {% endfor %}
</li>
```

becomes:

```
<li>
  <h3>{{ id.title }}</h3>
  {% for image_id in id.o.depiction %}
  <figure>
    {% media image_id width=100 link=id %}
    {% if image_id.summary %}
    <p class="image-caption">{{ image_id.summary }}</p>
    {% endif %}
  </figure>
  {% endfor %}
</li>
```

Pitfalls

Using ‘m’ as a template variable blocks model access

Avoid using the name `m` for a variable. It has a special meaning in templates for accessing models like `m_site` (page 579) and `m_rsc` (page 571) as `m.site` and `m.rsc`. Effectively, all Erlang modules with names starting with `m_` are made available in templates through the `m` variable.

It is particularly important since using `m` as a variable name will disable model module access for the entire scope within which that variable is defined. This can lead to very confusing template errors.

Using ‘id’ as a template variable blocks access to the current page

When rendering a page through `controller_page` (page 447), Zotonic sets the `id` variable based on the `resource` (page) being rendered. It is also conventionally used by dispatch rules to supply the id of the page to render.

Note, however, that there are legitimate cases for using `id` as a template variable. It is a good idea to reuse another template to render a section for one page and treating related content as the “current page” for that template by assigning `id` in a for loop or with context. Using `id` otherwise will likely confuse other developers trying to read your templates.

A complete and detailed overview of all Zotonic components.

6.1 Reference

All the nitty gritty details when the big picture has settled.

6.1.1 Modules

Zotonic comes with a considerable number of *modules* (page 59) that add functionality to your website. This section describes all of them. To use a module, you need to *activate it* (page 60) first. Some modules come with *configuration parameters* (page 60) that you can influence the module's behaviour with.

mod_acl_mock

Mocking module for access control during tests.

Do not use in production.

mod_acl_user_groups

See also:

Access control (page 58)

See also:

mod_content_groups (page 362)

This module adds rule-based access control.

- All resources (pages) are put into a content group.
- All users are member of zero or more user groups.
- Content groups are arranged in a hierarchy.
- User groups are arranged in a hierarchy.

ACL rules are defined between user groups and content groups. Each single rule gives some user group one or more permissions on some content group.

Managing user groups

By default, Zotonic has four user groups:

- anonymous (anonymous visitors of the website)
- members (logged in users of the website)
- editors (content editors)
- managers (manage users)

These user groups are arranged in a hierarchy, so that each group has the permissions its parent plus more. So, the permissions of users in the members group include those of anonymous users; and editors have the permissions of both anonymous users and members plus more.

To add or remove user groups, go to *Auth > User groups* in the admin.

Collaboration groups

Collaboration groups are a special type of user groups. They are most useful when you have groups of users that together collaborate on content. All content belongs to the group. Each collaboration group has one or more managers. So if you have groups of students working together and being supervised by teachers, you can define them as collaboration groups with the teachers as managers.

Managing content groups

By default, Zotonic has two content groups:

- default (all site content)
- system content (categories, predicates and user groups)

To add or remove content groups, go to *Structure > Content groups* in the admin. Just like user groups, content groups are ordered in a hierarchy. So the permissions that apply to the parent content group also apply to all of its children.

Defining access rules

You can add ACL rules in two ways:

1. in the admin web interface at `http://yoursite/admin/acl/rules`
2. in your site's or module's code; see *Managed rules* (page 345) below.

Let's start by defining rules in the web interface.

Content access rules

Each content access control rule grants some user group one or more permissions on some content group. So, for each rule you must specify the following properties:

- User group
- Content group
- Resource category (or 'all categories')

- Permissions: ‘view’, ‘insert’, ‘update’, ‘delete’, ‘link’, ‘manage own’.

If you wish to narrow down the rule, you can select a single resource category it applies to. The default is ‘all categories’.

The content group dropdown contains:

- all your *Managing content groups* (page 342)
- all your *Collaboration groups* (page 342)

The permissions include simple resource permissions that determine whether users in the group are allowed to view, insert, update or delete resources. The ‘link’ permission is about creating outgoing edges from resources in the content group to other resources.

Some rules may be greyed out and have a note saying ‘This rule is managed by module ...’. These are *Managed rules* (page 345) that you cannot edit in the web interface.

Collaboration group rules

Collaboration rules are special content access rules that apply to content in *collaboration groups* (page 342) only. Each rule applies to all collaboration groups.

Allowed media

For each user group it is possible to define:

- Maximum size of uploaded files
- Allowed types of files

In the admin, go to *Auth > Access control rules > File Uploads* tab to edit them.

The file size and allowed types is inherited along the user-group hierarchy. For example, if *Managers* is a specialized subgroup of *Editors* then all settings of *Editors* also apply to all *Managers*. Not the other way round.

The file types are entered using the mime type or extension of the allowed files.

The following are allowed types:

- Any mime type (e.g. image/png)
- Wildcard mime type (e.g. image/* or */*)
- A file extension (e.g. .txt)
- msoffice for Microsoft Office files
- openoffice for Open Office files
- embed for media imported with mod_oembed and mod_video_embed
- none

The default is: image/*, video/*, audio/*, embed, .pdf, .txt, msoffice, openoffice

The default mime types can be changed with the `site.acl_mime_allowed` key. The default upload size is 50MB.

If an user group is not allowed to upload any files then enter none.

Access control on properties

Some private sensitive resource properties are protected by the ACL rules. The *privacy* property defines who can see these properties.

The default privacy for category *person* is *collaboration group members*. For other categories the default is *public*.

The privacy property can have the following values:

- 0 public
- 10 members
- 20 member of same user group (except default members group)
- 30 collaboration group members
- 40 collaboration group managers
- 50 private

Increments are 10 so that more refined options can be added by custom modules.

The protected properties are:

- email
- phone
- phone_mobile
- phone_alt
- address_street_1
- address_street_2
- address_postcode
- address_city
- date_start
- date_end
- location_lat
- location_lng
- pivot_location_lat
- pivot_location_lng
- pivot_geocode
- pivot_geocode_qhash

Module access rules

Each module access rule grants some user group *use* permissions on some module. In the admin, go to *Auth > Access control rules > Modules* tab to edit them.

Deny rules

By default, rules grant some permissions. But sometimes you want to deny some permissions that are granted by another rule. For instance, if you have a rule that allows anonymous users to view all content groups, but you have a special content group ‘Top secret’ that you want to hide from anonymous users, add a rule to deny access:

Deny	ACL user group	Content group	Category	Permissions
	Anonymous	Top secret	All	View

Publishing rules

When you're editing rules, they are not effective immediately: you have to publish them first. Click the 'Publish' button to do so.

You can test out your rules before publishing them by clicking the 'Try rules...' button.

Managed rules

Above you've seen how you can add rules through the web interface. Using *module versioning* (page 64), you can also write rules in your code. These rules are called 'managed rules' because they are defined in the code of modules, including your own *site module* (page 18).

While editing a simple set of ACL rules in the web interface is easier for end users, developers may prefer to manage more complex rules in code. Managed rules have two important advantages:

- they are equal between all environments (such as development, acceptance and production)
- when developing and deploying new features, ACL rules and code often belong together. By defining the rules in your code, you can commit and store them along with the feature code.

If you haven't yet done so, set up *module versioning* (page 64) in `yoursite.erl` or `mod_your_module.erl`. Then, in the `manage_data/2` function, call the `m_acl_rule:replace_managed/3` function to add your new ACL rules.

Note that you always need a `manage_schema/2` function, even if it only returns `ok`. Otherwise the `manage_data/2` function will not be called:

```
%% yoursite.erl
-module(yoursite).

-mod_title("Your Site").
-mod_description("An example module for the docs").
-mod_depends([ mod_acl_user_groups ]).
-mod_schema(1).

-export([
    manage_schema/2,
    manage_data/2
]).

%% .... more code here...
manage_schema(install, Context) ->
    #datamodel{
        %% your resources...
    };
manage_schema({upgrade, 2}, Context) ->
    %% code to upgrade from version 1 to 2
    ok.

manage_data(install, Context) ->
    Rules = [
        %% A resource ACL rule is defined as {rsc, Properties}
        {rsc, [
            {acl_user_group_id, acl_user_group_members},
            {actions, [view, link]},
            {is_owner, true},
            {category_id, person}
        ]},
    ],
```

```
%% A module rule is defined as {module, Properties}
{module, [
    {acl_user_group_id, acl_user_group_editors},
    {actions, [use]},
    {module, mod_ginger_base}
]},
%% A collaboration group rule is defined as {collab, Properties}
{collab, [
    {is_owner, true},
    {actions, [view, insert, update, link]},
    {category_id, text}
]}
],
m_acl_rule:replace_managed(Rules, ?MODULE, Context);
manage_data(_Upgrade, Context) ->
ok.
```

Compile the code and restart your module to load the managed rules. They will be added and immediately *published* (page 345).

The set of rules added with `m_acl:replace_managed/3` is *declarative* and *complete*. That is to say, you declare the full set of rules that you wish to define. Any changes or deletions that you make to the rules in your code, will propagate to the site's rules. To protect the set's completeness, managed rules cannot be altered in the web interface.

Exporting and importing rules

To back up your rules, go to *Auth > Access control rules* and click the 'Export edit rules' button. The backup will include the full hierarchies of all user and content groups.

You can import a previous backup by clicking the 'Import edit rules...' button.

mod_admin

Todo

Finish documentation

Extending the admin menu

See *m_admin_menu* (page 557) on how to extend the admin menu.

Extending the admin edit page

There are several special templates names that will be automatically included into the */admin/edit/xxx* page from when you create these specially named templates.

`_admin_edit_basics.tpl` Will be automatically included into main (left) div (at top).

`_admin_edit_content.tpl` Will be automatically included into main (left) div (at bottom).

`_admin_edit_sidebar.tpl` Will be automatically included into right sidebar (near middle/bottom).

These templates are included using the *all catinclude* (page 511) tag; so if you need something in the sidebar just for persons, create a `_admin_edit_sidebar.person.tpl` file in your project.

Overriding TinyMCE options

If you need to override TinyMCE options; adding plugins, or setting other settings; you can create an `_admin_tinymce_overrides.js.tpl` file which can contain extra settings for the TinyMCE editors in the admin.

The template must contain JavaScript which modifies the `tinyInit` variable just before the editor is started. For example, to tweak the “paste” options you can put the following in the template:

```
tinyInit.theme_advanced_blockformats = "p,h1,h2"
tinyInit.paste_auto_cleanup_on_paste = true;
tinyInit.paste_remove_styles = true;
tinyInit.paste_remove_styles_if_webkit = true;
tinyInit.paste_strip_class_attributes = true;
tinyInit.paste_text_sticky = true;
tinyInit.paste_text_sticky_default = true;
```

TinyMCE Zotonic options

Zotonic provides extra init options:

z_insert_dialog_enabled Set this to false to prevent the insert media dialog from showing. Default true.

z_properties_dialog_enabled Set this to false to prevent the media properties dialog from showing. Default true.

Writing admin widget templates

This section contains examples of templates to create widgets for the /admin. Each of these examples extends basic several widget templates from `mod_admin`. To write your own you need to drop example content and fill holes in these example widgets.

You can use them as basis for your site admin-related tasks.

_admin_dashboard_example.tpl Very simple example widget for admin dashboard. Contains blocks for title and body. Look at /admin to see several dashboard widgets (latest events, pages, media, etc).

_admin_widget_std.tpl Slightly more complex widget example. Same templates are used into /admin/edit/N for main content and sidebar widgets. These widgets do not provide any localization abilities. Also note that there are several special widget names:

_admin_widget_i18n.tpl Complex widget example. Is used to edit localized rsc properties. It will be rendered as tabs. See /admin/edit/N top left to see the tabs. If `mod_translation` disabled, then i18n-widgets are displayed same as `_admin_widget_std.tpl`.

Making an admin widget conditionally visible

See also:

template-admin_edit_widget_i18n, *inherit* (page 527)

To make an entire admin widget visible or not, depending on some condition that you want to calculate inside the widget’s code, you can use the `widget_wrapper` block (which sits around the entire widget) in combination with the *inherit* (page 527) tag, wrapping that with a condition.

For instance, *mod_backup* (page 360) uses this technique to display the import/export sidebar widget. Excerpt from `mod_backup’s _admin_edit_sidebar.tpl`:

```
{# Make the widget conditional, based on the config value mod_backup.admin_panel #}
{% block widget_wrapper %}
    {% if m.config.mod_backup.admin_panel.value %}
        {% inherit %}
    {% endif %}
{% endblock %}
```

In this example, when the condition is true, the wrapper is rendered normally (content is inherited from the extended template); when false, the wrapper block is overridden by new (but void) content.

Extending the admin overview page

The overview page at `admin/overview` shows a table with links to all pages. It can be filtered and sorted. This extension deals with the view when a category filter has been selected.

The goal is to add more specific information that helps to distinguish pages.

The *Category* column, second from the left, can be extended to carry category-specific information. As an example, pages of category Event show the start date for each event, grayed out when the event is in the past. Something like this:

Title	Starts	Created on
Great event	2016-08-27 00:00:00	25 Aug 2015, 22:19
Past event	2014-01-02 00:00:00	01 Jan 2014, 13:10

Instead of dates, the information can be anything - from color coding labels to location data, the number of comments or the completeness state of product descriptions.

Setting up templates

To make it work we are using 3 templates (where *category_name* is the lowercase name of your category):

`_admin_overview_list.category_name.tpl`

Overrides the overview with a field variable for our custom sort. If we are using *an existing resource property* (page 571) such as `date_start`, we write:

```
{% include "_admin_overview_list.tpl"
    field="pivot.date_start"
%}
```

`_admin_sort_header.category_name.tpl`

The sort header caption. For a non-sortable header, just write the caption as text. For a sortable header, include the sort functionality in `_admin_sort_header.tpl` and pass the caption as variable:

```
{% include "_admin_sort_header.tpl"
    caption="My caption"
    type="date"
%}
```

`type="date"` indicates that sorting should start descending, from new to old.

`_admin_overview_list_data.category_name.tpl`

Contains the category-specific information. This is a freeform Zotonic template. The Events example checks if the end date is in the past:

```
{% if id.date_end|in_past %}
    <span class="text-muted">{{ id.date_start|timesince }}</span>
{% else %}
```



```
<span>{{ id.date_start|timesince }}</span>
{% endif %}
```

Custom sort properties

To sort on values that are not stored in the default Zotonic resources, you will need to create a *custom pivot* (page 321). This will create an additional database table with the values to sort on.

Let's take the (unlikely) example where we want to display the summary of each page (and sort on it as well). The summary data is not stored in an easily accessible way (at least for sorting), so we need to add 2 pivot methods to our Erlang module:

```
-module(mymodule).

-export([
    init/1,
    observe_custom_pivot/2
]).

init(Context) ->
    z_pivot_rsc:define_custom_pivot(?MODULE, [{summary, "text"}], Context),
    ok.

observe_custom_pivot({custom_pivot, Id}, Context) ->
    case z_trans:lookup_fallback(m_rsc:p(Id, summary, Context), Context) of
        undefined ->
            none;
        Summary ->
            {?MODULE, [{summary, Summary}]}
    end.
```

For this example we are just grabbing the default language text.

The field name in `_admin_overview_list.category_name.tpl` now just needs to contain the pivot column name:

```
{% include "_admin_overview_list.tpl"
    field="pivot.mymodule.summary"
%}
```

And the sort header template `_admin_sort_header.category_name.tpl` adds the custom pivot variable:

```
{% include "_admin_sort_header.tpl"
    caption="Summary"
%}
```

Resource meta features

Resources in the meta category can have ‘features’: certain resource properties (usually in the form of checkboxes) that decide what to show or hide on certain pages in the admin. To use this, create a `_admin_features.category.tpl` in your module.

For instance, the module `mod_geomap` defines the following `_admin_features.category.tpl` to create an extra checkbox so that per category can be defined whether or not the geodata box should be shown:

```
<div class="controls">
    <label class="checkbox">
        <input value="1" type="checkbox">
```

```
        name="feature_show_geodata"
        {% if id.is_feature_show_geodata|if_undefined:true %}checked{%_
↪endif %}
    />
    { _ Show geo data on edit page _ }
</label>
</div>
```

And on the edit page there is this check to conditionally include the geodata box:

```
{% if id.category_id.is_feature_show_geodata|if_undefined:true %}
```

The filter *if_undefined* (page 510) is used so that the default value can be true when the checkbox has never been touched.

Configuration keys

For the admin there are the following configuration keys:

- `mod_admin.rsc_dialog_tabs`
- `mod_admin.rsc_dialog_is_published`
- `mod_admin.rsc_dialog_is_dependent`
- `mod_admin.rsc_dialog_hide_dependent`

The `mod_admin.rsc_dialog_tabs` key defines which tabs are shown in the new resource, media-upload, and image-link dialogs. Per defaults these dialogs show all the possible tabs, with this configuration key it is possible to change that.

Available tabs are: `find`, `new`, `upload`, `url`, `embed`, `oembed`, `depiction`. More can be defined by modules.

Tab `depiction` is used for the TinyMCE image-link dialog; it shows all media connected using the `depiction` predicate.

The configuration `mod_admin.rsc_dialog_is_published` defines the default *is_published* state for new resources. Setting this key to *1* will check the *is_published* checkbox.

The configuration `mod_admin.edge_list_max_length` defines the maximum number of connections shown per predicate in the connection list sidebar. If there are more connections then the list truncated, and the message `_Too many connections, only the first and last are shown._` is displayed. The default is 100 connections.

The configuration `mod_admin.rsc_dialog_is_dependent` defines the default *is_dependent* state for new resources. Setting this key to *1* will check the *is_dependent* checkbox.

With the configuration `mod_admin.rsc_dialog_hide_dependent` the *dependent* checkbox can be hidden for non admin users.

mod_admin_category

Note: This module requires the presence of *mod_menu* (page 381) for the required JavaScript files which make up the menu editor.

Add support for editing *Categories* (page 7) in the admin, by presenting an editable category tree at `http://yoursite.com/admin/category`.

ACL permissions

The following *ACL permissions* (page 59) are required:

- to view the page, *use permission* (page 344) on the ‘mod_admin_category’ module
- to view the list of categories, *view permissions* (page 342) on category ‘category’
- to edit and re-order the categories, *edit permissions* (page 342) on category ‘category’.

Todo

Add more documentation

mod_admin_config

Add support for editing the site’s configuration values, as accessed through *m_config* (page 560).

The page in the admin is a list of every configuration module, key and textual value. Entries can be added, removed, and edited, if the user has the permission to do so.

When a value is large than 65 characters, it will be truncated in the admin config list view. To view the whole value, click the row.

SSL certificates configuration

Via the menu System > SSL Certificates all available certificates can be viewed.

The SSL modules add a panel here using a template `_admin_config_ssl_panel.[module_name].tpl`. Only the SSL modules providing a certificate are listed.

Here the module `mod_ssl_letsencrypt` lets you request a free certificate from the LetsEncrypt service.

Email configuration

See also:

m_config (page 560)

Configuration of outgoing email. This module provides a settings page via the admin menu System > Email configuration, where all email related settings are grouped.

On this page it is possible to send a test email.

Email modules like `zotonic_mod_mailgun` provide additional settings panels on this page. This is possible by adding a template with the name `_admin_config_email_panel.tpl`.

mod_admin_frontend

Note: This module is in active development. In the future live-editing and push-updates will be added. This might change the way the form and menu are handled.

Adds editing of resources, menu-trees and collections for non-admin users.

With many sites it is needed to let “normal” users edit content. For those users the `/admin` interface is overwhelming and also gives too many options that only administrators should be using.

For this use-case the `mod_admin_frontend` is created. It reuses some templates and parts of the normal admin, but in such a way that only the most needed options and fields are present. It also enables editing a menu-tree or collection, side-by-side with the content in the menu-tree.

Interface and dispatching

The interface is a one-page interface which uses postbacks to update the currently edited resource and forms. The editing history is maintained by using hashes in the url.

The URL of the edit page is generated by the dispatch rule `admin_frontend_edit`:

```
[
  {admin_frontend_edit, ["edit"], controller_page, [{acl, is_auth}, {template,
    ↪ "page_admin_frontend_edit.tpl"}]},
  {admin_frontend_edit, ["edit", id], controller_page, [{acl_action, edit},
    ↪ {template, {cat, "page_admin_frontend_edit.tpl"}}]}
].
```

The template uses the `menu_rsc` filter to find the contextual menu for the resource being edited. Per default the `menu_rsc` filter will fallback to the resource with the name `main_menu`. Hook into the `#menu_rsc{} notification` to change this behavior.

To edit a specific resource in the context of a menu that is non-standard for the resource, use the following code:

```
<a href="{% url admin_frontend_edit id=my_menu %}#edit_id={{ id }}">Click to edit {
  ↪ { id.title }}</a>
```

Customizing and templates

The `mod_admin_frontend` makes heavy use of `catinclude` to find the correct templates.

The column with the menu is rendered using `{% catinclude "_admin_menu_menu_view.tpl" tree_id ... %}`. The resource-column with the main editing form is rendered using `{% catinclude "_admin_frontend_edit.tpl" id ... %}`. This column is loaded lazily via a postback.

Customizing the menu column

Extra content can be added above or below the menu view by overruling the blocks `above_menu` and `below_menu`.

Customizing the main edit column

The main edit template `_admin_frontend_edit.tpl` provides the main edit-form, javascript initializations and fields for basic publication status editing.

The main edit block `edit_blocks` is defined as follows and can be overruled for specific categories:

```
{% block edit_blocks %}
    {% catinclude "_admin_edit_basics.tpl" id is_editable=is_editable_
    ↪languages=languages %}

    {% if id.category_id.feature_show_address %}
        {% catinclude "_admin_edit_content_address.tpl" id is_editable=is_editable_
    ↪languages=languages %}
    {% endif %}

    {% all catinclude "_admin_edit_content.tpl" id is_editable=is_editable_
    ↪languages=languages %}

    {% if id.is_a.media or id.medium %}
        {% include "_admin_edit_content_media.tpl" %}
    {% endif %}

    {% catinclude "_admin_edit_body.tpl" id is_editable=is_editable_
    ↪languages=languages %}
    {% catinclude "_admin_edit_blocks.tpl" id is_editable=is_editable_
    ↪languages=languages %}
    {% catinclude "_admin_edit_depiction.tpl" id is_editable=is_editable_
    ↪languages=languages %}
{% endblock %}
```

If any content needs to be added on top of the page, between the publication checkbox and the main edit fields, then overrule the block `meta_data_after`.

If you click on the *cog* icon on the right, then a meta data panel is shown (for access-control options and language settings). This panel can be extended with extra tabs using the blocks `meta_tabs` and `meta_panels`.

See also:

[mod_admin](#) (page 346), [menu_rsc](#) (page 484)

mod_admin_identity

Provides identity management in the admin - for example the storage of usernames and passwords.

It provides a user interface where you can create new users for the admin system. There is also an option to send these new users an e-mail with their password.

Create new users

See the *User management* (page 8) chapter in the User Guide for more information about the user management interface.

Configure new user category

To configure the category that new users will be placed in, set the `mod_admin_identity.new_user_category` configuration parameter to that category's unique name.

Password Complexity Rules

By default Zotonic only enforces that your password is not blank. However, if you, your clients or your business require the enforcement of Password Complexity Rules Zotonic provides it.

Configuring Password Rules

After logging into the administration interface of your site, go to the *Config* section and click *Make a new config setting*. In the dialog box, enter `mod_admin_identity` for Module, `password_regex` for Key and your password rule regular expression for Value.

After saving, your password complexity rule will now be enforced on all future password changes.

Advice on Building a Password Regular Expression

A typical `password_regex` should start with `^.*` and end with `.*$`. This allows everything by default and allows you to assert typical password rules like:

- must be at least 8 characters long `(?=.{8,})`
- must have at least one number `(?=.*[0-9])`
- must have at least one lower-case letter `(?=.*[a-z])`
- must have at least one upper-case letter `(?=.*[A-Z])`
- must have at least one special character `(?=.*[!@#$%^&+=])`

Putting those rules all together gives the following `password_regex`:

```
^.*(?!.{8,})(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%^&+=]).*$
```

To understand the mechanics behind this regular expression see [Password validation with regular expressions](#).

Migrating from an older password hashing scheme

When you migrate a legacy system to Zotonic, you might not want your users to re-enter their password before they can log in again.

By implementing the `identity_password_match` notification, you can have your legacy passwords stored in a custom hashed format, and notify the system that it needs to re-hash the password to the Zotonic-native format. The notification has the following fields:

```
-record(identity_password_match, {rsc_id, password, hash}).
```

Your migration script might have set the `username_pw` identity with a marker tuple which contains a password in MD5 format:

```
m_identity:set_by_type(AccountId, username_pw, Email, {hash, md5, MD5Password},
↳Context),
```

Now, in Zotonic when you want users to log on using this MD5 stored password, you implement `identity_password_match` and do the md5 check like this:

```
observe_identity_password_match(#identity_password_match{password=Password, hash=
↳{hash, md5, Hash}}, _Context) ->
  case binary_to_list(erlang:md5(Password)) == z_utils:hex_decode(Hash) of
    true ->
      {ok, rehash};
    false ->
      {error, password}
  end;
observe_identity_password_match(#identity_password_match{}, _Context) ->
  undefined. %% fall through
```

This checks the password against the old MD5 format. The `{ok, rehash}` return value indicates that the user's password hash will be updated by Zotonic, and as such, this method is only called once per user, as the next time the password is stored using Zotonic's internal hashing scheme.

mod_admin_merge

Adds functionality to merge two pages together into a single page.

In an interface the *winner* and *loser* can be selected. All connections and properties from the loser are merged into the winner. Properties of the winner are unchanged.












After merging the loser will be deleted. Visits to the deleted page will be redirected to the winner page using a *410 Gone*.

On the admin page editor a side panel is added for opening the merge page.

mod_admin_modules

Adds support in the admin for activating and deactivating *modules*.

The module overview in the admin presents a list of all modules, ordered by module status and priority. An activate/deactivate button allows the module to be (de)activated.

ZOTONIC					Dashboard	Content	Structure	Modules	Auth	System	Search...
Modules											
Zotonic is a modular web development framework. Most functionality is encapsulated inside modules. A set of basic modules are shipped with the Zotonic distribution, while others are externally developed. This page shows an overview of all modules which are currently known to this Zotonic installation.											
Title	Description	Prio	Author								
 zanymeta zotonic site zanymeta	An empty Zotonic site, to base your site on.	10	Arjan Scherpenisse	Deactivate							
 ACL Admins Only mod_acl_adminonly	Simple access control module, all users are site administrators. Use this for a simple site.	500	Marc Worrell <marc@worrell.nl>	Deactivate							
 Admin category support mod_admin_category	Support editing and changing the category hierarchy.	600	Marc Worrell <marc@worrell.nl>	Deactivate							
 Admin config support mod_admin_config	Allow admins to edit the system configuration.	800	Marc Worrell <marc@worrell.nl>	Deactivate							
 Admin identity/user supports mod_admin_identity	Adds support for handling and verification of user identities.	500	Marc Worrell <marc@worrell.nl>	Deactivate							
 Admin module mod_admin	Provides administrative interface for editing pages, media, users etc.	1000	Marc Worrell <marc@worrell.nl>	Deactivate							
 Admin module support mod_admin_modules	Manages modules. Adds an admin interface to activate and deactivate modules.	700	Marc Worrell <marc@worrell.nl>	Deactivate							
 Admin predicate support mod_admin_predicate	Adds support for editing predicates to the admin.	600	Marc Worrell <marc@worrell.nl>	Deactivate							
 Authentication mod_authentication	Handles authentication and identification of users.	500	Marc Worrell <marc@worrell.nl>	Deactivate							
 Base Site mod_base_site	Simple site building blocks.	550	Marc Worrell <marc@worrell.nl>	Deactivate							
 Bootstrap framework mod_bootstrap	Bootstrap provides simple and flexible HTML, CSS, and Javascript for popular user interface components and interactions.	900	Andreas Stenius <git@astekk.se>	Deactivate							

Module configuration dialog

When activate, some modules have a “configuration” button next to the activate/deactivate button. In that case, you can click the button to pop up a configuration dialog where you can set options which are required to configure the module.

To create such a dialog yourself, include a template called `_admin_configure_module.tpl` in your `templates/` folder in your module.

mod_admin_predicate

Add support for editing *predicates* (page 8) in the admin, by presenting a list of all defined predicates on `http://yoursite.com/admin/predicate`.

Predicates can be added, removed and edited, just like regular *resources*.

ACL permissions

The following *ACL permissions* (page 59) are required:

- to view the page, *use permission* (page 344) on the ‘mod_admin_predicate’ module
- to edit and delete predicates, *edit and delete permissions* (page 342) on category ‘predicate’.

Todo

Add more documentation

mod_admin_statistics

Todo

Not yet documented.

mod_artwork

This module contains many useful icons and images.

Included CSS icons

lib/material-design/ Material Design Iconic Font.

lib/font-awesome-4/ Font Awesome 4: fontawesome.io.

lib/font-awesome/ Font Awesome 3.2.1.

How to use Material Design icons in your templates

- Include the CSS file: `{% lib "material-design/css/material-design-iconic-font.css" %}`
- Follow the [examples](#)
- Find the class names in the [icons overview](#)

How to use Font Awesome icons in your templates

- Include the CSS file: `{% lib "font-awesome-4/css/font-awesome.css" %}`
- Follow the [Font Awesome examples](#)
- Find the class names in the [cheatsheet](#)

Included images

lib/images/emotes/ A collection of emoticons. These are from the Tango Icon Library and in the Public Domain.

lib/images/flags/ A collection of country flags. They are coded using the two letter ISO-3166 code of the country. For example `lib/images/flags/flag-nl.png` refers to the Dutch flag. This collection is from Wikimedia Commons and in the Public Domain. The PNG files are created by the Open Icon Library.

lib/images/zotonic/ Some Zotonic logos, with space for more.

lib/images/mimeicons/ Zotonic can't generate a preview of all files. If it can't generate a preview then an icon representing the mime type of the file is used. This is a collection of images for some mime types. This collection is from the Tango Icon Library and in the Public Domain.

lib/images/noun/ A selection of icons from The Noun Project. There are many black&white icons in this directory. This collection is licensed with the Creative Commons Attribution 3.0 Unported (CC BY 3.0) License, though about half of the icons are in the Public Domain or licensed using CC 0. When you use an icon in your project, check the license on <http://thenounproject.com/> and the proper attribution as described in <http://thenounproject.com/using-symbols/>

lib/images/social/round/ Round icons for various social sites. Think of Youtube, Twitter, Facebook etc. This collection is made by Veodesign (<http://veodesign.com/>) It is licensed under the Creative Commons Attribution Share-Alike 3.0 Unported License. This means you are not allowed to sell these icons and you need to properly attribute Veodesign.

lib/images/social/square/ Square icons for various social sites. This collection comes from the Open Icon Library (homemade) and is in the Public Domain. These icons are only 64x64 pixels (all others are 256x256).

How to use images in your templates

Most of the icons are in 256x256 PNG format. That is too large for normal usage. Best is to resize the images in your templates using the *image* (page 520).

For example, to display a 64x64 pixel image:

```
{% image "lib/images/social/round/email.png" width=64 %}
```

mod_audio

See also:

mod_video (page 402), *mod_video_embed* (page 403), *media* (page 529)

Adds support for viewing and handling audio medium items.

This module parses audio files and extracts tags and an optional image from the audio file.

Note: *mod_audio* uses the command-line utilities *ffmpeg* and *ffprobe*. For *mod_audio* to function correctly they must be present in the search path of Zotonic.

Uploading

The audio module hooks into the media model to intercept any audio file upload.

If an audio file is uploaded then the file is parsed using *ffprobe* to fetch:

- The bitrate
- Audio duration
- Tags, like artist, album
- Album art image, extracted as png using *ffmpeg*

The optional album art is used as the preview image. This image is visible when the audio media item is rendered using a `{% image ... %}` tag.

Viewing

The audio module extends the `{% media %}` tag for viewing `audio/*` videos.

It uses the template `_audio_viewer.tpl` for viewing.

Editing

The following properties are extracted from the audio file, and can be edited in the admin:

- artist
- album
- album_artist
- composer
- genre
- track
- copyright
- is_compilation (boolean flag)
- org_pubdate, this is extracted from the `creation_time` tag

If there is no resource title given when creating the audio page then the title from the tags is used. If that one is empty then the file's basename is used.

mod_auth2fa

Todo

Not yet documented.

mod_authentication

This module contains the main Zotonic authentication mechanism. It contains the logon and logoff controllers, and implements the various hooks as described in the [Access control](#) (page 58) manual.

Configuration keys:

mod_authentication.password_min_length The minimum length of passwords. Defaults to 8, set this to an integer value.

mod_authentication.is_rememberme Set this to 1 to check the *remember me* checkbox per default

mod_authentication.is_one_step_logon Normally a two-step logon is used, first the username is requested, then the password is requested. In between the server checks the username and is able to show alternative authentication methods based on the username. Set this to 1 to show the username and password field at once, and disable the display of alternative authentication methods.

mod_authentication.is_signup_confirm Set to 1 to force user confirmation of new accounts. This is useful when using 3rd party authentication services. If a new identity is found then a new account is automatically added. With this option set the user will be asked if they want to make a new account. This prevents duplicate accounts when using multiple authentication methods.

mod_authentication.reset_token_maxage The maximum age of the emailed reset token in seconds. Defaults to 48 hours (172800 seconds). This must be an integer value.

mod_authentication.email_reminder_if_nomatch On the password reset form, an user can enter their email address for receiving an email to reset their password. If an user enters an email address that is not connected to an active account then we do not send an email. If this option is set to 1 then an email is sent. This prevents the user waiting for an email, but enables sending emails to arbitrary addresses.

mod_authentication.auth_secret The secret used to sign authentication cookies. This secret is automatically generated. Changing this secret will invalidate all authentication cookies.

mod_authentication.auth_anon_secret The secret to sign authentication cookies for the anonymous user. This secret is automatically generated. Changing this secret will invalidate all authentication cookies for anonymous users.

mod_authentication.auth_user_secret The secret to sign authentication cookies for the identified users if there is no database to store individual secrets.

mod_authentication.auth_autologon_secret The secret to sign *remember me* cookies. This secret is automatically generated. Changing this secret will invalidate all *remember me* cookies for all users.

Related configurations:

site.password_force_different Set to 1 to force an user picking a different password if they reset their password.

site.ip_allowlist If the admin password is set to `admin` then logon is only allowed from local IP addresses. This configuration overrules the `ip_allowlist` global configuration and enables other IP addresses to login as `admin` if the password is set to `admin`.

mod_backup

`mod_backup` serves two different purposes: it makes a nightly backup of your files and database, and can also backup/restore individual *resource* items.

Daily backup of database and files

Losing data is bad for business. This applies to your customers as well if you are building sites for them. It is critical to keep backups of any Zotonic sites you develop.

After enabling `mod_backup`, it will make a backup of the site's data every night at 3 AM. It keeps the last 10 copies of the data, so you have always a backup to roll back to.

The backups are stored under `backup` in the files directory of your site. Check in the admin under System > Status to see where the site files directory is located.

The site's media files are stored as a `.tar.gz` file, while the database is stored as an uncompressed `.sql` file.

We advise to add a `cron` script to the server for copying the data to remote storage.

Per-resource backup/restore

Todo

Not yet documented.

mod_base

See also:

dispatch rules.

`mod_base` is the base module, which acts as a container module holding most of Zotonic basic dispatch rules, *Actions* (page 403) and *Module tags* (page 533).

Note that the amount of *templates* has been kept to a minimum in this module, so that sites are free to implement whatever templates they want.

Todo

Add more documentation

mod_bootstrap

Adds support for the [Twitter Bootstrap](#) CSS / JavaScript framework.

Todo

Add more documentation

mod_clamav

Uses `clamd` to scan all uploaded files for viruses.

The scanning happens after the mime type and access control is checked, but before the sanitization. If a file is infected then the error `infected` will be returned. The admin will display a growl message telling that the file was infected.

Clamd has a maximum size for checks, above that size the error `sizelimit` will be returned.

Configure in the `zotonic.config` file where `clamd` is listening.

The following configs are available:

clamav_ip IP address of `clamd`, default to "127.0.0.1"

clamav_port Port of `clamd`, default to 3310

clamav_max_size The `StreamMaxLength` of `clamd`, default to 26214400 (25M)

clamav_reject_msoffice_external_links Reject Microsoft Office documents containing `externalLinks` information. If the Zotonic config is set to `false` then rejection can be forced by setting the site's config key `mod_clamav.reject_msoffice_external_links` to 1. Defaults to `true`.

All clamav results are logged, any infected files or other errors are logged to the `error.log`.

Every hour the module checks if it can reach `clamd` using the configured settings. It will log an error if `clamd` can't be reached, and an info message if it can be reached.

mod_comment

Implements a basic commenting system, enabling commenting on *resources*.

The module has an admin comment overview, allowing the administrator to review the comments or delete them.

To enable commenting in your site, include `_comments.tpl` on your resource's page template, passing an *id* parameter for the resource on which you want users to be able to comment.

mod_contact

Implements a basic contact form, which gets emailed to the configuration value `mod_contact.email`, when submitted.

Todo

Add more documentation

mod_content_groups

See also:

mod_acl_user_groups (page 341)

Todo

Not yet documented.

mod_cookie_consent

Wrap external content in such a way that it is only loaded if the user consented to the inclusion of the content (and subsequent cookies).

The consent is one of:

- `functional` this is always allowed
- `stats` if consent for statistics tracking is given
- `all` for any other kind of cookies

For elements this defaults to `all`. This means that they are only rendered if all consent is given.

How to use

Ensure that your base template has an all-include of `_html_head.tpl` and `_html_body.tpl`.

Also, if you are using IFRAMEs, JS or CSS that sets non-functional cookies, check the changes below.

HTML

Media embedded via `mod_oembed` or `mod_video_embed` are automatically wrapped according to this method:

```
<figure class="cookie-consent-preview do_cookie_consent mediaclass-..." data-
↪cookie-consent="all">
  
  <figcaption>Please consent to cookies to display external content.</figcaption>
  <script type="text/x-cookie-consented">
    {% filter escape %}
      <iframe width="560" height="315" src="https://www.youtube.com/embed/
↪IdIb5RPabjw" title="YouTube video player" frameborder="0" allow="accelerometer;
↪autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"
↪allowfullscreen></iframe>
    {% endfilter %}
  </script>
</figure>
```

If there is no script tag then the page is reloaded after cookies are accepted.

IFRAME

Use the `data-cookie-consent-src` attribute to define the `src` if the cookie consent has been given:

```
<iframe width="560" height="315" data-cookie-consent-src="https://www.youtube.com/
↪embed/..."></iframe>
```

JAVASCRIPT

Use the special `type="text/x-cookie-consent"` and optionally the `data-cookie-consent="..."` attribute:

```
<script type="text/x-cookie-consent" data-cookie-consent="stats" src="https://...">
↔</script>
```

CSS

Use the special `type="text/x-cookie-consent"` and optionally the `data-cookie-consent="..."` attribute:

```
<link type="text/x-cookie-consent" data-cookie-consent="stats" href="https://...">
↔</script>
```

mod_cron

Provides periodic events.

Current this module sends *tick* notifications at periodic intervals. These ticks can be observed to implement periodic tasks.

tick

For periodic tasks the system has various periodic tick events. They are named after their interval *s* for seconds, *m* for minutes, and *h* for hours.

- tick_1s
- tick_1m
- tick_10m
- tick_15m
- tick_30m
- tick_1h
- tick_2h
- tick_3h
- tick_4h
- tick_6h
- tick_12h
- tick_24h

Example

Check something every hour:

```
-include_lib("kernel/include/logger.hrl").

observe_tick_1h(tick_1h, Context) ->
    ?LOG_INFO("And another hour has passed..."),
    do_something(Context).
```

The return value is ignored.

The *tick* observers are called one by one in a separate process. So a slow handler can delay the other handlers.

mod_custom_redirect

See also:

[Dispatch rules](#) (page 22), [mod_base](#) (page 360)

Enables redirects from unknown hosts and paths to other locations. The other location can be a known path or another web site.

Redirect unknown domains

If the site dispatcher encounters an unknown host name then it notifies all modules with the `#dispatch_host` notification.

The Custom Redirect module observes this notification and checks against a configurable list of domains and redirects. If a domain is matched then the site dispatcher will redirect the user agent to the new location.

The list of domains and their redirects can be configured in the admin: Modules -> Domains and redirects.

The domain must be like what is typed in the browser. Examples of domains are `www.example.org` and `mypc.local:8000`.

Domain names are case insensitive, that is `WWW.EXAMPLE.COM` and `WwW.Example.coM` will both be matched with `www.example.com`. Contrary to the domain name, the path is case sensitive. That is `/ABOUT` and `/About` are two different paths and will need their own redirect rules!

The redirect location can be a complete URL (for example `http://www.example.com/foo/bar.html`) or a path (for example `/about`).

Redirect unknown paths

After the site has been selected, the dispatcher matches the path to the dispatch rules.

When no dispatch rule matches, then the `#dispatch` notification is sent. The [mod_base](#) (page 360) module observes that notification to check the path against the `page_path` properties of all resources. If [mod_base](#) (page 360) didn't find match then [mod_custom_redirect](#) (page 364) will check all custom redirect with an empty domain and a matching path. The visitor will be redirected to the corresponding redirect location.

Permanent or temporary redirects

A redirection can be permanent or temporary. A permanent redirect will be remembered by the visiting browser (and search engines), replacing any occurrence of the redirected location. A temporary redirect will not be remembered and be retried on every visit.

mod_development

Presents various tools for development.

Admin page

After the development module is enabled a menu item *Development* is added to the *System* menu in the admin.

On the development page it is possible to set debugging options, trace template compilation, and test dispatch rules.

Options

This can toggle various development options:

Show paths to included template files in generated templates Checking this will add comments in the compiled templates. The comments will list the exact file included at that point.

Show defined blocks in generated templates Checking this will add comments in the compiled templates. The comments will show the start and end of any template `{% block %} ... {% endblock %}`.

Download css and javascript files as separate files (ie. don't combine them in one url). Checking this will generate separate `<link/>` and `<script/>` tags for all files mentioned in a single `{% lib %}` tag. This makes debugging those files easier but makes loading pages slower as more requests will be done per page.

Enable API to recompile & build Zotonic The `api` on `/api/model/development/get/recompile` can be accessed to trigger a full compilation and cache flush of Zotonic. This checkbox must be checked to enable this api.

Template debugging

The template selection mechanism is quite complicated. It takes into account all modules, their priority, the user-agent class (desktop, tablet, phone or text) and optionally the category of a resource.

With this debugging tool you can optionally select a category, and fill in the name of the template. Per user-agent class the selected template will be shown.

Template debugging

Find a template, check which template will be selected

Optional category for catinclude

Module	Template path
mod_site_tufte	/Users/marc/Sites/zotonic-master/user/modules/mod_site_tufte/templates/page.tpl

[Show which files are included in a template compilation](#)

At times it can be confusing which templates are actually used during a template compilation. Here you can see which files are included whilst compiling a template.

The second debug option is a page with a live display of all templates being compiled. With this it is possible to get greater insight in the template selection and compilation.

Dispatch rule debugging

With this it is possible to see for a request path which dispatch rules are matched and/or how it is rewritten.

Dispatch rule debugging

Match a request url, display matched dispatch rule.

http

Path	Action	Bindings
/en/test	#dispatch_rewrite{} notification New path: /test	z_language en
/test	Start matching dispatch rules	zotonic_dispatch_path ["test"] z_language en
/test	No matching dispatch rule found	
/test	#dispatch{} notification, checking first return	
	#dispatch{} found resource id: 22946 New path: /en/page/22946/redirect-test	
/en/page/22946/redirect-test	#dispatch_rewrite{} notification New path: /page/22946/redirect-test	z_language en
/page/22946/redirect-test	Start matching dispatch rules	zotonic_dispatch_path ["page","22946","redirect-test"] z_language en
/page/22946/redirect-test	Found matching dispatch rule: page Controller: controller_maxclass_page Options: [{"template","cat","page.tpl"}]	slug redirect-test id 22946 zotonic_dispatch_path ["page","22946","redirect-test"] z_language en

Automatic recompilation

Note: The system can only scan for changed files if either `inotify-tools` or `fswatch` is installed.

The core Zotonic system starts either `inotify-tools` or `fswatch`, depending on which one is available. You have to install one of these to enable auto-compile and auto-load of changed files.

See below for platform-specific installation instructions.

If a changed file is detected then Zotonic will:

- If an `.erl` file changes then the file is recompiled.
- If a `.scss` or `.sass` file changes then `sassc` is called to compile it to its `.css` equivalent. If the changed `.sass` file starts with a `_` then all `.sass` files without a `_` will be compiled.
- If a `.less` file changes then `lessc` is called to compile it to its `.css` equivalent.
- If a `.coffee` file changes then `coffee` is called to compile it to its `.js` equivalent.
- If a lib file changes then the module indexer will be called so that any removed or added templates will be handled correctly.
- If a template file changes then the module indexer will be called so that any removed or added template will be handled correctly.
- If a dispatch file changes then all dispatch rules are reloaded.
- If a beam file changes then the module will be loaded. If the beam file is a Zotonic module then it will be automatically restarted if either the function exports or the `mod_schema` changed.

- If the `.yrl` definition of the template parser changes, then the `.erl` version of the parser is regenerated. (This will trigger a compile, which triggers a beam load).

Linux installation

On Linux this feature depends on the *inotifywait* tool, which is part of the `inotify-tools` package. For displaying notifications, it uses `notify-send`:

```
sudo apt-get install inotify-tools libnotify-bin
```

Mac OS X installation

On Mac OS X (version 10.8 and higher), we use the external programs `fswatch` and `terminal-notifier`:

```
sudo brew install fswatch
sudo brew install terminal-notifier
```

Configuration options

`mod_development.libsep` Boolean value. If true, *lib* (page 528) files will be included separately instead of in one big concatenated file.

`mod_editor_tinymce`

Adds wysiwyg tinyMCE editors to the admin.

The module packs various versions, the version to be used can be selected on the `/admin/modules` page.

Two additional tinyMCE plugins are added:

- The plugin `zmedia` for selecting images, videos and other media items. In a dialog the centering, size, and optional caption can be chosen. The selected media are added to the page with a `depiction` connection.
- The plugin `zlink` for adding links (`...`) to Zotonic pages in the edited text. The title and URL to the page are automatically added.

Media are added as HTML comments in the body text. The filter *show_media* (page 469) is used to translate the HTML comment into a figure or other tags. This is done using the template `_body_media.tpl`, though other templates can also be used.

Usage

Use TinyMCE by including `_editor.tpl` in your template.

This will, after page load, check all `textarea` tags with the class `z_editor-init`. These `textarea` tags are then augmented with a TinyMCE editor.

Call the JavaScript function `tinymce.triggerSave()`; to save all the TinyCME editor instances to their `textarea` tags. A wired submit action will save before submitting the form.

Options

All TinyMCE options are set in the global object `tinyInit`. This is done by the `tiny-init.js` file inside the version of TinyMCE. For example `priv/lib/js/tinymce-5.10.2/tiny-init.js`

This file is included by `editor.tpl`, which should be included in your template if you want to use the editor.

Supply your own `tinyInit` object to change the default settings. This can be done before the include of `_editor.tpl`, as the `tiny-init.js` will not replace any existing `tinyInit` object.

Stylesheet

The default TinyMCE `tiny_init.js` included by `_editor.tpl` includes a stylesheet from `/lib/css/tinymce-zotonic.css`.

By adding a file `priv/lib/css/tinymce-zotonic.css` in your site or module it is possible to use your own styles for the editor.

Takes care of adding the styles for the images managed by the `zmedia` plugin. The default image styles are:

```
img.z-tinymce-media {
    position: relative;
    border: 1px solid #667;
    height: auto;
    cursor: pointer;
    width: 270px;
}

img.z-tinymce-media.z-active:hover {
    border: 1px solid #039ed4;
}

img.z-tinymce-media-align-block {
    display: block;
    margin: 1rem 0;
    clear: both;
}

img.z-tinymce-media-align-right {
    float: right;
    margin: 0.5rem;
}

img.z-tinymce-media-align-left {
    float: left;
    margin: 0.5rem;
}

img.z-tinymce-media-size-large {
    width: 90%;
}

img.z-tinymce-media-size-middle {
    width: 50%;
}

img.z-tinymce-media-size-small {
    width: 25%;
}
```

mod_email_dkim

Signs outgoing e-mails with DomainKeys Identified Mail Signatures ([RFC 6376](#)).

DKIM (DomainKeys Identified Mail) is an important authentication mechanism to help protect both email receivers and email senders from forged and phishing email.

How does it work?

DKIM works by signing each e-mail that Zotonic sends with a private key. The public key part is exposed through a DNS TXT record, with which email receiver can check whether the email actually originated from the domain that it claimed to come from.

Note: The generating of the keypair depends on the `openssl` utility to be available in `$PATH`.

This RSA keypair is generated automatically when the module is installed, and the private/public keys are put in the directory `sitename/dkim/`. When the module is active and the keypair has been generated, all outgoing e-mail will be signed.

DNS configuration

The receiving e-mail server checks the validity of the signature by doing a DNS lookup. To configure DKIM, you will need to add this DNS entry to your domain where you send the mail from.

In the admin, the page `/admin/email/dkim`, available under (“Modules” / “DKIM e-mail setup”) provides information how to configure this DNS entry, including the text to copy-paste into the DNS record.

DKIM selector

By default, the DKIM selector is set to the string `zotonic`. This will result in DNS lookups to the `zotonic._domainkey.yoursite.com` domain. You can change the selector name by adding a config value called `site.dkim_selector`.

mod_email_receive

See also:

[mod_email_relay](#) (page 369), [E-mail handling](#) (page 74).

Enables the Zotonic site to receive emails for the site’s users. The user’s email address is `username@hostname`, where the hostname is the hostname as configured in the [site’s config file](#) (page 18).

Any email that has no valid recipient is rejected.

Todo

Add more documentation

mod_email_relay

See also:

[mod_email_receive](#) (page 369), [E-mail handling](#) (page 74).

Enables the Zotonic site to *relay* emails for the site’s users to their real email addresses.

The user’s email address is `username@hostname`, where the hostname is the hostname as configured in the [site’s config file](#) (page 18). Any mails to those addresses get forwarded to the user’s email address, as configured in the user [resource](#).

Any email that has no valid recipient is rejected.

Todo

Add more documentation

mod_email_status

This module tracks for all outgoing email addresses:

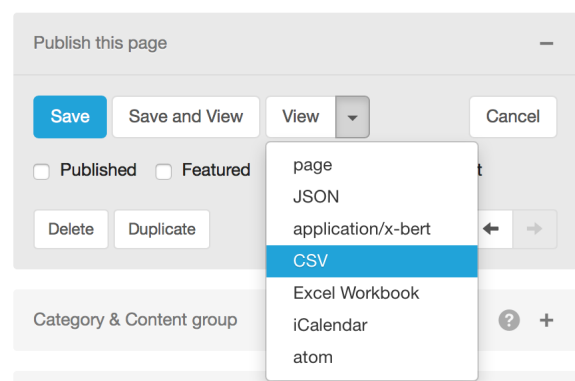
- If emails are successfully sent
- Number of emails sent
- Number of errors
- Number of bounces
- Latest error message

With this it will be much easier to get feedback on all outgoing email.

mod_export

Provides a generic framework to export *resources*.

Admin interface



When *enabled* (page 60), this module adds two things to each admin edit page:

- extra content types to the ‘View’ dropdown menu
- an ‘Export’ block.

Both single pages and *query resources* (page 53) can be exported. For a query, all resources matching it will be included in the export.

Customizing exports

To customize data selection and the properties that are exported, observe one or several of the *export notifications* (page 594).

mod_facebook

See also:

mod_linkedin (page 377), *mod_twitter* (page 401)

The mod_facebook module plugs into the *authentication system* (page 58) to enable Facebook login on your site.

Configuration

Activate (page 60) `mod_facebook`, then head to 'Auth' > 'External services' in the admin interface to enter your Facebook app ID and secret. Enable Facebook login by checking the 'Use Facebook authentication' box. This will add a 'Log in with Facebook' button to the logon form on your site.

If you need extended permissions, add them to the 'Scope' textbox. Note that the module needs the 'email' permission for login to work.

`mod_filestore`

See also:

m_filestore (page 564).

Support for storing uploaded and generated images and documents on external services.

Overview

This module stores uploaded files and generated preview-images on an external S3-compatible service. It listens for medium and file related notifications for any newly uploaded or generated files.

If a file is added then the file is queued in an upload queue. After a delay a separate process polls this queue and will upload the file to the external service.

If a file is needed and not locally available then the `mod_filestore` module will check its file registry to see if the file is stored on an external service. If so then then a *filezcache* process is added and a download of the file is started.

The file is served from the filezcache whilst it is being downloaded.

The filezcache will stop the entry after a random amount of time—if the entry was not recently used.

Configuration

S3 configuration

Configure the following permissions on your S3 service for `mod_filestore` to work correctly:

Resource	Permissions
/	<ul style="list-style-type: none"> s3:ListBucket
/-zotonic-filestore-test-file-	<ul style="list-style-type: none"> s3:GetObject s3:PutObject s3>DeleteObject
/preview/*	<ul style="list-style-type: none"> s3:GetObject s3:PutObject s3>DeleteObject (if file deletion is enabled)
/archive/*	<ul style="list-style-type: none"> s3:GetObject s3:PutObject s3>DeleteObject (if file deletion is enabled)

mod_filestore configuration

After the mod_filestore is enabled an extra menu entry 'Cloud File Store' is added to the 'System' menu in the admin.

Selecting the menu will show the configuration panel for the Cloud File Store.

Cloud File Store Configuration

Zotonic can store uploaded and resized files in the cloud. Here you can configure the location and access keys for the cloud service.

Currently Zotonic supports services that are compatible with the S3 file services API. These include:

- [GreenCloud](#)
- [Amazon Simple Storage Service \(S3\)](#)
- [Google Cloud Storage](#)

S3 Cloud Location and Credentials

Base Url

API Key

API Secret

☒ Upload new files to the cloud

Before the settings are saved they will be checked by uploading (and removing) a small file.

[Save Settings](#)

S3 Cloud Utilities and Statistics

	Media	Local Files	Cloud Files
Files	1160	1157	16
Storage	1.7 GB	1.6 GB	32.1 MB

Upload Queue	Download Queue	Delete Queue
0	0	0

Actions

All local uploaded and preview files can be moved to the cloud. This will queue the files for later asynchronous upload.

[Move all files to S3](#)

[Move all files from S3 to local](#)

Here you can define where the files should be stored and give the credentials to access the storage.

If you save the url and credentials then the system will try to upload a small file to the remote storage. If it succeeds then the configuration is saved. If it does not succeed then an error message will be displayed and the configuration will not be changed.

It is possible to (temporarily) disable uploading new files by unchecking the checkbox *Upload new files to the cloud*.

File deletion

You can also configure file deletion behaviour, i.e. what should happen when a file is removed from Zotonic. You can choose to immediately remove the file from the filestore, not delete it at all (to make your store immutable) or delete the file after a certain delay (to be able to restore accidentally deleted files).

Statistics

The system shows statistics:

Media All medium records and a sum of the sizes. A single medium record can have 0, 1 or 2 files attached.

Local Files These are all files found in the `files` directory, this includes files that won't ever be uploaded.

Cloud Files All files registered to be on any cloud service. This is extracted from the database and not by scanning the remote cloud service.

Queues These are the queues being processed by `mod_filestore`. On a quiet (stable) system they are usually empty.

Moving files

It is possible to move (almost) all files from the local file system to the cloud. And vice versa, from the cloud to the local file system. This is useful when starting or changing the cloud storage location.

If a file is moved to the cloud then it is first placed in the filezcache. The filezcache will start purging the files if the cache is bigger than configured in the filezcache application (default 10GB for all sites combined).

The system waits 10 minutes before a queued file is uploaded. This period is meant for a *cool down* of the file, as in the first moments after an upload some resize and preview operations will take place. The delay makes it less probable that a freshly uploaded file vanishes (to the cache) whilst a preview-generation is starting.

Notifications

The `mod_filestore` hooks into the following notifications, whose definitions can be found in `zotonic_file.hrl`:

#filestore{} Hooks into the Zotonic file management notifications to upload, delete or lookup files. This will trigger downloads of external files and interfaces to the filezcache.

#filestore_credentials_lookup{} Maps a local path and optional resource id to a service, external location and key/password for that external service. This can be used to store different resources on different external services.

#filestore_credentials_revlookup{} Maps a cloud file service and location to a key, password and request location.

#medium_update_done{} Queues newly inserted medium files into the upload queue.

#admin_menu{} To add the *Cloud File Store* menu to the admin.

Applications

The filestore uses the *s3filez* and *filezcache* Erlang applications.

s3filez

This application is used for uploading, downloading and deleting files on S3 compatible services. It provides asynchronous services and is compatible with the *filezcache* application. It is also able to stream files to and from the external S3 service, this makes it possible to have start serving a file before it is downloaded to the *filezcache*.

filezcache

This application manages a cache of downloaded files. The cache is shared between all sites. Every cache entry is managed by its own process, which can stream newly received data directly to any requesting processes.

The *filezcache* keeps a persistent *disk_log* with a description of all files in the cache. This log is read on startup to repopulate the cache with already present files. For each file the size and a hash is stored to check cache consistency.

The *filezcache* has a garbage collector. It keeps a pool of randomly selected cache entries, from which it will elect randomly processes to be garbage-collected. The processes themselves will decide if they will stop or not.

After a cache process stops it will keep running for a short period to handle late incoming requests.

Filezcache entries are started by the *mod_filestore* and filled by either moving a local file to the cache or by *s3filez* download processes.

Todo

The statistics are generated dynamically, which is not a good idea with many files. This will be changed.

mod_fileuploader

Upload files using a parallel web worker.

This module provides a web worker and server logic to quickly upload files.

The web worker sends files in 128KB parts to the server, the server recombines all parts to a single file. The web worker is using multiple parallel uploaders to make optimal use of the available bandwidth.

Files are sent to the webworker on the topic `model/fileuploader/post/new`:

```
let f = document.getElementById("upload_file").files[0];

let msg = {
  files: [
    {
      name: "resultname",    // The name used in the ready message
      file: f                // Must be a File object
    }
  ],
  ready_msg: {
    foo: "bar",             // Anything you want to send if all files are_
↪uploaded
    files: []               // This will be set by the worker
  },
  ready_topic: "bridge/origin/foo/bar",

  progress_msg: {
```

```
    form_id: "some-element-id",
    is_auto_unmask: true,
    percentage: 0 // Will be set by the uploader, 100 when ready
  },
  progress_topic: "zotonic-transport/progress"
};
cotonic.broker.publish("model/fileuploader/post/new", msg);
```

The files received with the `ready_msg` on the `ready_topic` will be:

```
{
  name: "resultname", // Name provided in the upload
  upload: "WZkhXoaMwrK2StUHmdpp" // Unique name to use with z_
  ↪ fileuploader:status/1
}
```

mod_geoip

Used to map IP addresses to geographical locations.

This modules uses the freely available MaxMind databases for the IP mapping. The database is updated automatically.

Note: As the database is only once downloaded per Zotonic instance the license key is shared between all sites. It is not possible to have a specific per-site key.

For downloading the database you need a License Key from MaxMind. The License Key can be generated after [registering with MaxMind](#).

The obtained license key can be configured in the `erlang.config` file with:

```
{locus, [
  {license_key, "YOUR_LICENSE_KEY"}
]}
```

The above license key can be overruled in the `zotonic.config` file using the `maxmind_license_key` key.

mod_image_edit

See also:

[image](#) (page 520), [Arguments](#) (page 521)

Non destructive edits of images.

This adds a *Edit image* button on the media panel in the admin.

Clicking that button opens an image editor where the following editing parameters can be set:

- Rotation: in 90 increments, used to correct the orientation of an image
- Crop: crop any or all sides of an image
- Brightness: make an image more or less bright
- Contrast: enhances or lowers the contrast
- Roll: correct the rotation of an image (between 45 and -45)
- Tilt: rotate an image in the Y direction, as if the viewer moves up or down
- Pan: rotate an image in the X direction, as if the viewer moves left or right

- Lossless flag: use PNG or GIF for the output image, used for clip art and logos
- Crop center: mark a point that should stay in view when using automatic crops

All these operations are non-destructive. They are stored in the `medium_edit_settings` properties and applied whenever an image is resized using the `{% image %}` tag.

The original image stays untouched. Downloads will download the original image.

If the original image should be used for a preview then pass the `original` argument to the `{% image %}` tag options.

mod_import_csv

Module which adds “import CSV” button to the admin status screen.

The dropbox folder of the site is also watched for CSV files.

To determine whether it can import a file, it uses a notification:

```
#import_csv_definition{basename, filename}
```

If the notification is not returning anything, it tries to map the columns found in the first row of the CSV file to *resource* column names.

Todo

Add more documentation

mod_import_wordpress

Import WordPress .wxr files in your site.

[WordPress](#), the famous PHP blog system, is able to export all of its contents into a .wxr XML file. With this module, you can import such a .wxr file in Zotonic, when you want to migrate your blog. The import routine will import all posts, keywords, categories, authors and images.

Once you enabled the module, the import button lives in the admin in the “status” tab.

The import module works incrementally: if you import a .wxr file twice, it will update the changed entries and add new entries. If you delete posts, a next import will not re-create the entries: it will remember that you have deleted them. You can override this behaviour by checking the “import previously deleted content” button.

Note: YMMV with importing 2 .wxr files from different blogs: as the unique keys of the entries are based on the WordPress numeric ids, it is possible that content from the second one will overwrite content from the previous import.

mod_l10n

Localization of Zotonic. Provides lookups for country, month, week names.

Todo

Add more documentation

mod_linkedin

The `mod_linkedin` module plugs into the *authentication system* (page 58) to enable LinkedIn login on your site.

Configuration

Activate (page 60) `mod_facebook`, then head to ‘Auth’ > ‘External services’ in the admin interface to enter your LinkedIn app ID and secret. Enable LinkedIn login by checking the ‘Use LinkedIn authentication’ box. This will add a ‘Log in with LinkedIn button to the logon form on your site.

See also:

- *mod_facebook* (page 370)
- *mod_twitter* (page 401)

mod_logging

See also:

For regular application logging, use *Logger* (page 80) instead.

Logs messages to the database and adds log views to the admin.

Logging messages to the database

To persist a log message in the database, enable `mod_logging` in your Zotonic site. Then, in your code, send the `#zlog{}` notification:

```
-include_lib("zotonic_core/include/zotonic.hrl").

some_function() ->
    %% do some things

    z_notifier:notify(
        #zlog{
            user_id = z_acl:user(Context),
            props=#log_email{
                severity = ?LOG_LEVEL_ERROR,
                message_nr = MsgId,
                mailer_status = bounce,
                mailer_host = z_convert:ip_to_list(Peer),
                envelop_to = BounceEmail,
                envelop_from = "<>",
                to_id = z_acl:user(Context),
                props = []
            },
        },
        Context
    );
```

E-mail log

The e-mail log is a separate view, which lists which email messages have been sent to which recipients. Any mail that gets sent gets logged here.

Todo

Add more documentation

mod_mailinglist

This module implements a mailing list system. You can make as many mailing lists as you like and send any page to any mailing list, including confirm mail and unsubscribe page.

Mailing lists are pages of the category *mailinglist*, which is installed by the module upon activation. In the admin there is support for managing these mailing lists where you can import, export, add or delete recipients.

For details on configuring e-mail sending and receiving in Zotonic, see *E-mail handling* (page 74).

Including the subscribe custom tag on your pages

The module includes the *signup tag* (page 543) tag (and template) that you can use on your site.

When you want to add a subscribe template to your page then you will need the following scomp include in your template:

```
{% mailinglist_subscribe id=mailing_id %}
```

Where `mailing_id` should be set to the page id of your mailing list. This scomp includes the template `_scomp_mailinglist_subscribe.tpl`. The subscription form itself can be found in the template “`_mailinglist_subscribe_form.tpl`”. You should overrule the latter template when you want to add or remove fields from the subscription form.

Pages for the mailing list category

The mailinglist module predefines the following dispatch rules for the mailinglist:

```
/mailinglist/1234  
/mailinglist/1234/my-mailinglist-slug
```

Where 1234 stands for the id of your mailinglist (which is obviously another id).

The mailinglist page is very simple. It shows the title, summary and body of the mailinglist, followed by a subscribe form and finally a list of other mailing lists.

Template used for the e-mails

All e-mails use the `template-mailing_page` template. It is a very simple template that just tells why the recipient received the e-mail, refers to the sent content page on the Internet and finally shows the title, the summary and the body of the sent page.

In the footer there is a link to the unsubscribe page.

All e-mail templates extend from the `template-email_base` template. The following templates are used for e-mails:

Template	Description
<code>template-email_mailinglist_confirm</code>	Sent after subscribing to a mailing list, requests to click on an url to confirm the subscription.
<code>template-email_mailinglist_goodbye</code>	Sent after unsubscribing from a mailing list.
<code>template-email_mailinglist_welcome</code>	Sent after subscribing and confirming the subscription.

Sending mailings in the admin

On the resource edit page of any page (remember: the mailinglist module can send *any* resource as a mailing!) there is an link in the right column called *Go to the mailing page*.

The mailinglist recipients page

Each mailinglist has a special page in the admin that lets you view the recipients that are part of the list. On that page you can perform various functions on the recipients of the list.

- *Add new recipient* - Opens a dialog to add a single recipient to the list.
- *Download recipient list* - Downloads a `.txt` file with all the e-mail addresses and name details of all recipients.
- *Upload recipient list* - Upload a new file with recipients. Each e-mail address goes on its own line. There is a checkbox which lets you clear the list before the import, effectively overwriting all recipients in the list.
- *Clear recipient list* - After a confirmation, this removes all recipients from the list.
- *Combine two lists* - this opens a dialog which lets you combine two lists. Using this new dialog, the recipients of two lists can be combined according to the three set operations union, subtract and intersect.

The mailing status page

From the mailing status page you can send the current resource to a mailing list, to the test mailing list or to an email address.

Sending mailings

The status page lists every mailing list in the system. On each row you see how many recipients the list has, and the status, e.g. if the mailing has already been sent to this list or not.

Todo

finish description of the mailing status page

Mailings are only sent when the to be send page is published and publicly visible. The page should also be within its publication period.

You can schedule a mailing by publishing the page but setting its publication start date to the date and time you want your mailing to be sent. The mailing list module checks every hour if there are any mailings ready for sending.

An exception is made for the test mailing list, mailings to that mailing list are always sent.

mod_media_exif

When uploading a file, this module extracts properties from the uploaded media and sets them in the resource.

Properties extracted from the uploaded file are:

- GPS location
- Crop center (derived from the focusing area)
- Orientation
- Date (start, end and original publication date)

mod_menu

See also:

The filters [menu_flat](#) (page 483), [menu_subtree](#) (page 484) and [menu_trail](#) (page 484).

Create nested navigation menus for your site.

Activating the module in the admin enables a “menu” item in the admin navigation under “content”, which lets you define a simple menu. Every item in the menu references a Zotonic page and can be looked up using the autocomplete widget.

This menu can be rendered in the frontend with the [menu](#) (page 543) custom tag.

It will use the `_menu.tpl` template which is by default able to render a Twitter Bootstrap compatible menu structure using nested `` elements.

To implement a different navigation menu, override the `_menu.tpl` in your project and create new markup.

Domain model

The *domain model* for this module is the following:

The module creates a new category named *menu*. This allows one to create multiple menus in a single site. Its edit page in the admin contains the hierarchical menu editor.

The menu resource that is accessible from the admin page (*Content > Menu*) is the resource with the unique name `main_menu`.

mod_microsoft

Adds logon using the Microsoft identity platform.

If enabled then on `/admin/authentication-services` a panel is added for the Microsoft identity platform.

Here the following can be configured:

- Application ID: as found in the app registration
- Client Secret: as found in the app registration
- Scope: space separated list of scopes that you want the user to consent to. Examples are: `email`, `offline_access` and `profile`. The `openid` scope is always added automatically. Defaults to `email profile`.
- Tenant: Control who can sign in. Allowed values are: `common`, `organizations`, `consumers`, and `tenant identifiers`. Defaults to `common`.

On the [Azure Portal App registrations](#) an App can be registered and configured.

The redirect path for the app is shown on top of the `/admin/authentication-services` screen and is of the format: `https://example.com/oauth-service/redirect`

mod_mqtt

MQTT is a machine-to-machine (M2M)/“Internet of Things” connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios

MQTT uses a simple message broker to route messages from publishers to multiple subscribers.

A quick overview of MQTT

Publish/subscribe

With MQTT messages are published to topics. All subscribers to a topic will then receive the message. A topic is a string, much like a file-path, for example: `truck/0001/temperature`

A subscriber can directly subscribe to a topic, or can use wildcards to subscribe to related topics. For this the wildcards `+` and `#` can be used. `+` matches exactly one word between the slashes, and `#` can be used at the end of a pattern to match all sub-topics.

Examples of subscription patterns:

- `truck/0001/temperature` matches the temperature publications of truck 0001.
- `truck/+/temperature` matches all temperature publications for all trucks.
- `+/+/temperature` matches all temperature publications for all trucks and other things with a temperature
- `truck/0001/#` matches all publishes to `truck/0001` and all its sub-topics

Retained messages

A publisher can publish a *retained* message to a topic. When publishing all current topic subscribers will receive the message. Above that, if a new subscription is made to the topic, then all retained messages are sent to the new subscriber.

Quality of service

MQTT has three levels for the quality of message delivery. These are used when sending messages between machines. The levels are:

- Level 0: send the message, no reception acknowledgments are reported.
- Level 1: on receipt a single ack is sent back to the publisher
- Level 2: a double handshake is performed

For most communication level 0 is used.

Wills

A client can set a *last will* message and topic. This is a message that will be published to the topic at the moment the client is unexpectedly disconnected.

MQTT in Zotonic

Zotonic has a central MQTT message broker. Optionally clients can connect to this broker using the normal MQTT protocol.

The broker is used for internal publish/subscribe support.

Each open HTML page can also have a local (simplified) broker. The system can relay messages between the brokers on open pages and the central broker in Zotonic. In this way it is possible for HTML pages to have their own local publish/subscribe system and also subscribe or publish to topics on the central broker.

As the central broker is shared between sites it is even possible to publish/subscribe between different sites. In the future it will be possible to bridge the brokers between servers.

Predefined topics

Currently the following topics are defined:

Topic	Description
public	Freely accessible topic, both for subscribe and publish
test	Test topic. If you publish here then mod_mqtt will log a debug message.
user	Topic available for any authenticated user
user/UserId	Topic available for a specific user of the site
bridge/ClientId	The topic forwarding to the client with id ClientId

Topics and namespaces

To make it easier to write generic software, without changing topic names, some namespace conventions and mappings are introduced.

The following topics are expanded:

Topic	Expansion	Description
~client	bridge/vWCUKL9QKmfLxotWorZv	The bridge topic that forwards to the user agent
~user	user/1234 <i>or</i> user/anonymous	The topic for the current user

Note that there are not automatic subscriptions for user topics. All subscriptions need to be added explicitly.

Access control

All topics have access control added. For this an extra ACL object `#acl_mqtt{}` (page 584) is defined, with the actions `publish` and `subscribe`. Modules can observe the usual `acl_is_allowed` (page 583) notification to allow access to MQTT topics:

Listing 6.1: your_site.erl

```
observe_acl_is_allowed(#acl_is_allowed{object = #acl_mqtt{topic = [ <<"my">>, <<
↳ "topic">> ]}}, _Context) ->
    %% Allow anonymous access on this topic
    true;
observe_acl_is_allowed(#acl_is_allowed{}, _Context) ->
    undefined.
```

Subscribing modules

Modules can automatically subscribe to topics. This is done by adding specially named functions.

For example, the following function subscribes to the topic `test/#`:

```
-include_lib("kernel/include/logger.hrl").

-export([
    'mqtt:test/#'/2
]).

-spec 'mqtt:test/#'(map(), z:context()) -> ok.
'mqtt:test/#'(Message, Context) ->
    ?LOG_DEBUG("mqtt:test on site ~p received ~p", [ z_context:site(Context),
↳ Message ]),
    ok.
```

Here *Message* is a map with the received MQTT message (of type `publish`):

```
#{
  type => publish,
  pool => Pool,                % A MQTT pool (and topic tree) per site
  topic => Topic,              % Unpacked topic for the publish [ <<"foo">>, <<
  ↪ "bar">> ]
  topic_bindings => Bound,     % Variables bound from the topic
  message => Msg,              % The MQTT message itself
  publisher_context => PublisherContext
}
```

The *Context* is the context of the process/user that subscribed to the message. Use the `publisher_context` for the *Context* (and ACL permissions) of the publisher.

Erlang API

Subscribe a function *F* in a module *M* to a topic:

```
-spec subscribe(z_mqtt:topic(), mfa(), pid(), z:context()) -> ok | {error, eaccess_
  ↪ | term()}.
z_mqtt:subscribe([ <<"my">>, <<"topic">>, '#' ], {M, F, []}, self(), Context)
```

This will subscribe the function, with the current process (`self()`) as the managing process. If the process exits then the subscription is removed.

Access control applies and the result `{error, eaccess}` will be returned if access is denied, `ok` will be returned on a successful subscription.

Subscribe the current process to a topic:

```
-spec subscribe(z_mqtt:topic(), z:context()) -> ok | {error, eaccess | term()}.
z_mqtt:subscribe(Topic, Context)
```

When the process stops it will automatically be unsubscribed. The process will receive messages `{mqtt_msg, map() }`, where the `map()` is like the `map()` in the section above.

Subscribe another process to a topic:

```
-spec subscribe(z_mqtt:topic(), pid(), z:context()) -> ok | {error, eaccess | _
  ↪ term()}.
z_mqtt:subscribe(Topic, Pid, Context)
```

To unsubscribe, use `z_mqtt:unsubscribe` with the same arguments as during subscription.

To publish a message:

```
-spec publish(z_mqtt:topic(), term(), z:context()) -> ok | {error, term()}.
z_mqtt:publish(Topic, Payload, Context)
```

With options (qos or retain):

```
-spec publish(z_mqtt:topic(), term(), z_mqtt:publish_options(), z:context()) ->_
  ↪ ok | {error, term()}.
z_mqtt:publish(Topic, Payload, #{ qos => 1, retain => true }, Context)
```

Or, with a complete MQTT message:

```
-spec publish(mqtt_packet_map:mqtt_packet(), z:context()) -> ok | {error, term()}.
Msg = #{
  type => publish,
  qos => 0,
  topic => [ <<"~client">>, <<"public">>, <<"hello">> ]
  payload = #{ key => 1, foo => <<"bar">> }
```

```
},
z_mqtt:publish(Msg, Context)
```

JavaScript API

See also:

live tag (page 542), which uses MQTT topics.

There is a separate topic tree in the browser. To be able to send message from/to the browser there are special *bridge* topics on both ends.

The browser receives an unique client and routing id on connecting to the server. On the server those ids can be used to route messages back to the client using a bridge topic.

For example the server side topic:

```
bridge/MyClientId/browser/topic
```

Is mapped on the client to:

```
browser/topic
```

It is possible to send messages to the server, or subscribe to topics on the server. For this there is a special *bridge/origin* (the *bridge to origin*, ie. the server serving the page) topic.

Any subscribe or publish action on this topic is relayed to the server. For example, to access the server side topic *my/server/topic*, use the client side topic *bridge/origin/server/topic* (both for publish and subscribe).

The JavaScript API uses callback functions:

```
cotonic.broker.subscribe("bridge/origin/test/#", function(msg, bindings, options)
→{ console.log(msg); });
cotonic.broker.publish("bridge/origin/test/foo", "hello world");
```

The received message is an JSON object:

```
{
  type: "publish",
  qos: 0,
  payload: "hello world",
  properties: {
    ...
  },
  ...
}
```

The transport between the server and the browser uses a websocket connection and binary encoded MQTT v5 messages.

Connection will

Currently a simple version of the *lastwill* is available for JavaScript. This sets a topic and message to be sent when the page process stops.

Multiple wills can be set. Currently it is not possible to remove a will, though that will change in the near future.

Example:

```
var will_id = pubzub.lastwill("~site/goodbye", "thanks for the fish");
```

Quality of service

Currently there is no quality of service implemented for the JavaScript API and relay. The server side page process will buffer all messages till the browser connects to the page session. This happens on connects with comet, WebSocket, and postbacks.

On the browser all messages are queued and sent one by one to the server. This uses either the WebSocket connection or the postback interface.

Enabling the MQTT listener

MQTT can listen on a port for incoming connections. Per default the listener is enabled.

Configuration

The MQTT listener is configured in the `zotonic.config`. Use `bin/zotonic configfiles` to see where this file is located.

If this file is missing then it can be copied from `~apps/zotonic_launcher/priv/zotonic.config.in`.

Per default it listens on MQTT port 1883 and MQTT with TLS on port 8883:

```
%%% IP address for MQTT connections - defaults to 'listen_ip'
%%% Use 'none' to disable.
    %% {mqtt_listen_ip, any},

%%% IPv6 address for MQTT connections - defaults to 'listen_ip6'
%%% Use 'none' to disable.
    %% {mqtt_listen_ip6, any},

%%% Port number for MQTT connections
    %% {mqtt_listen_port, 1883},

%%% Port number for MQTT ssl connections
    %% {mqtt_listen_ssl_port, 8883},
```

Authentication

All connections must authenticate using an username and password. The username is prefixed with the hostname of the user's site, for example: `foobar.com:myusername`. In this way Zotonic knows which site the user belongs to.

If no matching site can be found, or if no hostname is given, then Zotonic will try to authenticate against the default site.

mod_oauth2

Todo

Not yet documented.

mod_oembed

See also:

mod_video_embed (page 403), *mod_video* (page 402), *mod_audio* (page 358), *media* (page 529)

Makes media *resources* from embeddable URLs through the **OEmbed** protocol.

This module, if activated, checks the pasted URLs in the *create media / page* dialog of the admin. It will show an embed option for services like YouTube, Vimeo or any other service which supports OEmbed.

A saved media resource has a thumbnail image which is downloaded from the OEmbed service and embedded in the resource. Furthermore, the resource's *medium* record has an `oembed` field which contains the full JSON response of the request. The `oembed` field looks like this:

```
"oembed": {
  "type": "video",
  "version": "1.0",
  "provider_name": "Vimeo",
  "provider_url": "http://vimeo.com/",
  "title": "Heli Filming Showreel",
  "author_name": "Hot Knees Media",
  "author_url": "http://vimeo.com/hotknees",
  "is_plus": "1",
  "html": "<iframe src=\"http://player.vimeo.com/video/20898411\" width=\"640\"
↪height=\"362\" ...",
  "width": 640,
  "height": 362,
  "duration": 106,
  "description": "description..",
  "thumbnail_url": "http://b.vimeocdn.com/ts/138/106/138106290_640.jpg",
  "thumbnail_width": 640,
  "thumbnail_height": 362,
  "video_id": 20898411
}
```

So, to display the HTML of an OEmbedded medium, you would do the following in a template:

```
{{ id.medium.html }}
```

The module also supports the use of the *media* (page 529) tag:

```
{% media m.rsc[id].o.deipction.medium %}
```

Note however, that setting dimensions on this media tag is not supported for OEmbed, as the embed width/height is always taken from the provider.

Configuration options

The following *m_config* (page 560) options are supported:

`mod_oembed.embedly_key`

The API key for Embedly. This configuration can be added set in the admin via the menu **Auth -> External Services**.

`mod_oembed.maxwidth`

Requests the OEmbed service to return an HTML embed code with the requested maximum width. Defaults to 640.

`mod_oembed.maxheight`

Requests the OEmbed service to return an HTML embed code with the requested maximum height. Not set by default.

mod_ratelimit

Implements rate limiting for various resources.

After activation rate limiting will be added to the login and password reset flow.

The rate limiting is done on username or e-mail address. After five attempts the username (or e-mail address) will be blocked for an hour.

If an username is used for a successful login, then a special *device-id* cookie is placed on the user-agent. This device-id ensures that that particular user-agent is allowed its own five tries. This prevents an username to be blocked on known devices by tries on other devices. The device-id cookie is only valid for a single username.

mod_search

See also:

[Search](#) (page 44), [Custom search](#) (page 323)

mod_search implements various ways of searching through the main resource table using [m_search](#) (page 578).

Configuration

There are two *site configuration variables* (page 626) to tweak PostgreSQL text search settings.

mod_search.rank_behaviour

An integer representation to influence PostgreSQL search behaviour.

Default: 37 (1 | 4 | 32)

mod_search.rank_weight

A set of four numbers to override relative weights for the ABCD categories.

Default: {0.05, 0.25, 0.5, 1.0}

The following searches are implemented in mod_search:

Name	Description
query	Very powerful search with which you can implement almost all of the other search functionality. See query .
facets	Performs a query (see above) and then calculates facets. The facets are defined using the pivot function.
subfacets	Performs a query (see above) and then calculates facets. The facets are defined using the pivot function.
facet_values	Returns an empty result with the facets field set to all possible values per facet, or the min/max range.
featured	List of pages, featured ones first.
featured	List of pages in a category, featured ones first.
featured	List of pages in a category having a certain object, featured pages first.
latest	The newest pages.
latest	The newest pages within in a category.
upcoming	Selects pages with future date_end .
finished	Selects pages with a past date_end .
ongoing	Pages with past date_start and future date_end .
autocomplete	Full text search where the last word gets a wildcard.
autocomplete	Full text search where the last word gets a wildcard, filtered by category.
fulltext	Full text search. Returns { id , score } tuples.
fulltext	Full text search, filtered by category. Returns { id , score } tuples.
referrers	All subjects of a page.

Name	Description
media_category_image	All pages with a medium and within a certain category. Used to find category images.
media_category_depiction	All pages with a depiction edge to an image. Used to find category images.
media	All pages with a medium, ordered by descending creation date.
all_bytitle	Return all {id, title} pairs for a category, sorted on title.
all_bytitle_featured	Return all {id, title} pairs for a category, sorted on title, featured pages first
all_bytitle	Return all {id, title} pairs for a category without subcategories, sorted on title.
all_bytitle_featured	Return all {id, title} pairs for a category without subcategories, sorted on title, featured pages first
match_objects	Returns a list of pages with similar object ids to the objects of the given resource with the given id. I
match_objects_cats	Returns a list of pages with similar object ids or categories. Returns {id, rank} tuples. Accepts
archive_year	Returns an overview on publication year basis, for a specified category. Every row returned has parts
archive_year_month	Return a grouped “archive” overview of resources within a category. The result is a double list, cons
keyword_cloud	Return a list of {keyword_id, count} for all resources within a given category. The list is ord
previous	Given an id, return a list of “previous” ids in the given category. This list is ordered by publication d
next	Given an id, return a list of “next” ids in the given category. This list is ordred by publication date, o

mod_seo

Adds basic search engine optimization to the base templates and provides an admin interface for configuring SEO options and Google Universal Analytics.

SEO data

mod_seo adds metadata to your pages to improve your website’s display and ranking in search engine (e.g. Google) results. This metadata includes [structured data](#):

Google uses structured data that it finds on the web to understand the content of the page, as well as to gather information about the web and the world in general.

Following Google’s recommendations, mod_seo adds data in the [Schema.org](#) vocabulary, using the JSON-LD format. This enables [search features](#) such as rich results, carousels and the sitelinks searchbox.

Tip: If you wish to alter the structured data, you can do so by overriding the `schema_org/schema.*.tpl` templates. When you do so, make sure to validate your changes with Google’s [structured data testing tool](#).

Configuration

Disable search engine indexing

To prevent search engines from indexing a page, check the ‘Ask Google not to index this page’ checkbox in the admin. Programmatically, you can set the `seo_noindex` flag on a resource to do the same.

To disable indexing for the site as a whole, go to Modules > SEO in the Admin (<https://yoursite/admin/seo>) and tick the ‘Exclude this site from search engines’ checkbox.

Google Analytics

To enable [Google Universal Analytics](#) tracking on your Zotonic website, go to <https://yoursite/admin/seo> in your browser and enter your Google Analytics tracking code.

Zotonic does not automatically supports the [User ID Analytics feature](#). You have to [enable User ID](#) in your Analytics account and override the `_ga_params.tpl` template to add the parameter:

```
{#
    Override this template to provide extra Google Analytics parameters.
    See https://developers.google.com/analytics/devguides/collection/analyticsjs/
    ↪field-reference
#}
{
    {% if m.acl.user %}
        "userId": "{{ m.acl.user|escapejs }}",
    {% endif %}
}
```

Extra parameters

If you wish to add extra Google Analytics parameters, override the `_ga_params.tpl` file and add the parameters:

```
{#
    Override this template to provide extra Google Analytics parameters.
    See https://developers.google.com/analytics/devguides/collection/analyticsjs/
    ↪field-reference
#}
{
    {% if m.acl.user %}
        "userId": "{{ m.acl.user|escapejs }}",
    {% endif %}
    "sessionControl": "start"
}
```

Todo

Add more documentation

mod_seo_sitemap

Creates a `sitemap.xml` file for your site, containing links to all publicly accessible pages.

`mod_seo_sitemap` creates a `sitemap.xml` file for which is used by the Google bot to index your site. By default, its generated `sitemap.xml` lists all pages of the categories text, event, location, collection and person in the system which are publicly viewable.

The sitemaps follow the specification at <https://www.sitemaps.org/protocol.html>

Maintaining the sitemap index

The `seo_sitemap` model maintains an index of all pages (and their URLs) that are publicly readable.

There is a button in the System > Status menu to rebuild the sitemap index. This is needed if there are (big) changes in the access control affecting the publicly viewable pages or if languages are changed.

If a resource is edited then all URLs of that resource are automatically added to the sitemap index tables.

Structure of the sitemap

The generated sitemap is an index file with links to urlset files.

The urlsets links are generated from a combination of two sources:

1. All resources and URLs in the sitemap index tables maintained by model `seo_sitemap`
2. Urlsets for by the `#seo_sitemap_index{}` notification.

The `#seo_sitemap_index{}` is a map notification where all modules can return 0 or more maps with information about their pages. The map should be like:

```
#{
  source => <<"wherefrom">>,           % Unique code
  count => 1234,                        % Number of URLs
  size => 5000,                         % (optional) number of URLs per urlset
  lastmod => {{2021,5,12}}, {0,0,0}}    % (optional) last modification date
}
```

The sitemap urlset is generated using the `#seo_sitemap_urlset{}` notification:

```
#seo_sitemap_urlset, {
  source => <<"wherefrom">>,           % Unique code from the ``#seo_sitemap_index
  ↳ {} ``
  offset => 1,                         % Offset for search, 1 based.
  limit => 50000                       % Number of URLs to be returned
}
```

This is a *first* notification and should return a list of maps. Each map contains a single URL:

```
#{
  loc => <<"https://example.com/...", % The URL for the urlset
  lastmod => {{2021,5,12}}, {0,0,0}}, % (optional) last modification date
  priority => 0.5,                     % (optional) priority, between 0.0..1.0
  changefreq => <<"weekly">>          % (optional) change frequency, for recrawls
}
```

Creating a sitemap without using `mod_seo_sitemap`

If you have a site with mostly static URLs, it might be easier to create your own sitemap XML file. If you create a dispatch rule like this:

```
{sitemap_xml, ["sitemap.xml"], controller_template, [{template, "mysitemap.tpl"},
  ↳ {content_type, "text/xml"}]}
```

You will be able to write your own `sitemap.xml` file (in a `mysitemap.tpl` template) without using `mod_seo_sitemap`. This might be handy if you serve mostly static files.

`mod_server_storage`

Server side storage for the client (aka browser) and server.

The storage is tied to the `sid` in the authentication token of the client. This `sid` is unique and changed if the user logs on or logs off.

Model functions

`model/server_storage/post/key` store a key, with the value of in the payload of the message.

`model/server_storage/get/key` - fetch a key.

`model/server_storage/get` - fetch all keys.

`model/server_storage/delete/key` - delete a key.

`model/server_storage/delete` - delete all values from the server storage.

Example in JavaScript from the client:

```
cotonic.broker.publish("bridge/origin/model/server_storage/post/foo", { "hello":  
  ↪ "world" });  
  
cotonic.broker.call("bridge/origin/model/server_storage/get/foo")  
  .then( (msg) => console.log(msg) );  
  
{  
  dup: false  
  packet_id: null  
  payload: {result: {hello: "world"}, status: "ok"}  
  properties: {content_type: "application/json"}  
  qos: 0  
  retain: false  
  topic: "reply/page-3-0.4080048813145437"  
  type: "publish"  
}
```

Example in a template:

```
The hello key of foo is now: {{ m.server_storage.foo.hello|escape }}
```

Note that we **must** escape the value, as it originates from the client and contains arbitrary values, including HTML.

Example in Erlang:

```
m_server_storage:store(<<"foo">>, #{ <<"hello">> => <<"world">> }, Context),  
  
case m_server_storage:lookup(<<"foo">>, Context) of  
  {ok, Value} -> % Found the value  
  {error, no_session} -> % No session  
  {error, not_found} -> % There is a session but unknown key  
end.  
  
m_server_storage:delete(<<"foo">>, Context).
```

Storage of secret server data

Sometimes the server code wants to attach data to the specific client that is not accessible to the client itself. An example is a secret during an OAuth handshake.

For this there is a special, unlimited, storage API, which is only accessible using the Erlang API:

```
m_server_storage:secure_store(<<"foo">>, #{ <<"my">> => <<"secret">> }, Context),  
  
case m_server_storage:secure_lookup(<<"foo">>, Context) of  
  {ok, Value} -> % Found the value  
  {error, no_session} -> % No session  
  {error, not_found} -> % There is a session but unknown key  
end.  
  
m_server_storage:secure_delete(<<"foo">>, Context).
```

The keys in the *secure* storage are separate from the keys in the normal storage. There can be a key with the same name and different values in both storages.

Storage process

If a value is stored in the server side storage then a process is started. This process holds all stored values, and times out after 900 seconds.

Any HTTP request with the proper `sid` will extend the lifetime of the storage process.

The storage process holds at most 100 KB of data, above that it will respond with a `full` error status.

Configuration keys

There are two configuration keys, both need a number:

- `mod_server_storage.storage_expire` The timeout in seconds of the storage process defaults to 900 seconds.
- `mod_server_storage.storage_maxsize` The maximum stored size in bytes, both the keys and the values are counted. Default is 100 KB.

Note that the storage is in-memory, so it is best to set the storage maxsize and expire as low as possible.

mod_signup

This module presents an interface for letting users register themselves.

Configuration

You can adjust this module's behaviour with the following *Module configuration* (page 60):

`mod_signup.request_confirm`

true (default) send a signup confirmation e-mail to new users

false disable the signup confirmation e-mail

`mod_signup.username_equals_email`

false (default) users have a username separate from their e-mail address and use that username for logging in

true the user's e-mail address is also the user's username, so users can log in with their e-mail address.

`mod_signup.member_category`

Name of the category that users created through sign up will be placed in.

Defaults to `person`.

`mod_signup.content_group`

Name of the content group that users created through sign up will be placed in.

Defaults to `default_content_group`.

`mod_signup.depiction_as_medium`

If set then any `depiction_url` is added as a medium record to the person who signed up. Normally the depiction is added as a separate *depending* image resource and connected from the person using a `depiction` predicate.

Config: Using the user's e-mail address as username

By setting a configuration value, it is possible to use the entered email address as the username.

Set the configuration value `mod_signup.username_equals_email` to `true`.

This makes the username equal to the email address, so that the user can log in using his email address instead of a separate user name. Note that when you allow a user to change his email, take care to update the `{username_pw, {Username, Password}}` identity as well, otherwise the username remains equal to the old email address.

Notifications

`signup_form_fields`

Fold for determining which signup fields to validate. This is an array of {Fieldname, Validate} tuples, defaulting to:

```
[
  {email, true},
  {name_first, true},
  {name_surname_prefix, false},
  {name_surname, true}
]
```

Observers can add / remove fields using the accumulator value that is passed into the notification.

`identify_verification{user_id=UserId, identity=Ident}`

Send verification requests to unverified identities.

`signup_check`

Fold for the signup preflight check. Allows to add extra user properties or abort the signup.

If no {ok, _Props1, SignupProps} is returned, but {error, Reason}, the signup is aborted.

`signup_done{id=Id, is_verified=IsVerified, props=Props, signup_props=SignupProps}`

Fired when a signup procedure is done and a user has been created.

`signup_confirm{id=UserId}`

Fired when a users have signed up and confirmed their identity (e.g. via e-mail).

`signup_confirm_redirect{id=UserId}`

Decide to which page a user gets redirected to after signup.

Todo

Add more documentation

mod_site_update

This module pulls updates of a site's code from a remote version control system. The supported version control systems are git and mercurial.

After enabling this module you will see a button **Update site** on the System -> Status page in the admin.

This button is only available to users with the *use.mod_site_update* right.

After a pull of new code, Zotonic will do everything that is needed. Including compilation of source code, reloading of dispatch rules, reloading of translations and reindexing the template directories.

This update of the system can take a while.

Webhook

In GitHub and other systems it is possible to set a *webhook* that is called when new updates are pushed to the repository.

In System > Modules there is a *Config* button next to the `mod_site_update` module. Click this and configure a secure token to be passed to the webhook.

The URL for GitHub et al is:

```
https://yoursite.test/api/model/site_update/post/webhook/<token>
```

Where `<token>` should be replaced with your configured token.

The token is saved in the config key `mod_site_update.webhook_token`.

Ensure that the external system performs a HTTP POST to the webhook, the posted body is dropped.

Zotonic status site

The `mod_site_update` module is also used by the zotonic status site.

Here the module adds buttons to update sites and a button to update the complete Zotonic system.

Site updates get their datamodel fixed by automatic restarts of modules, ensure that you increment the `-mod_schema(...)` version number.

Zotonic updates are a bit more complicated. Small updates will be fine. Large updates that also change the datamodel or add new Erlang dependencies need more compilation and a system restart. So be careful with using this update button.

mod_ssl_ca

See also:

mod_ssl_letsencrypt (page 397), *Port configurations* (page 630)

The `mod_ssl_ca` module adds support for using SSL certificates bought from a Certificate Authority.

A free alternative to CA provided tickets is Let's Encrypt, see *mod_ssl_letsencrypt* (page 397).

Certificate and key files

The certificate and key files are placed into the site sub-directory of the security directory. The subdirectory will be: `sitename/ca/`

Where *sitename* must be replaced with the name of your site.

The security directory can be found by inspecting the output of:

```
bin/zotonic config
```

The Zotonic *security* directory can be in one of the following directories:

- The environment variable `ZOTONIC_SECURITY_DIR`
- The `~/ .zotonic/security` directory
- The `/etc/zotonic/security` directory (only on Linux)
- The OS specific directory for application data files

The OS specific directories are:

- On Unix: `~/.config/zotonic/security/`
- On macOS: `~/Library/Application Support/zotonic/security/`

The default is the OS specific directory.

If there is a directory `priv/security/ca` inside your site's OTP application folder then that directory will be used.

The filenames are checked against their extension. When you copy your files to the `ca` directory then you need to ensure that they have the right extensions.

The following file extensions are expected:

- **.pem or .key** This holds the private key for the encryption. The key must be unlocked and in PKCS#1 format (see below).
- **.crt** This is the certificate. Usually it is supplied by the certificate authority where you bought it. It can also be a self signed certificate, see below.
- **.ca.crt, cabundle.crt or bundle.crt** This is the (optional) *CA bundle* that contains root and intermediate certificates for the certificate authority that issued the `.crt` certificate.

The certificate authority will supply these. All supplied certificates are concatenated, with the root certificate last.

The concatenation is a literal command, like:

```
cat intermediate.crt root.crt > cabundle.crt
```

Due to caching, it can take up to a minute before the new certificates are used.

Format of the private key

The Erlang SSL implementation accepts PKCS#1 and PKCS#8 format keys. OpenSSL generates (since 2010) PKCS#8 format keys.

A PKCS#1 key starts with:

```
-----BEGIN RSA PRIVATE KEY-----
```

A PKCS#8 key starts with:

```
-----BEGIN PRIVATE KEY-----
```

If there are problems then check if the `.key` or `.pem` file starts with one of the above strings.

Using SSL certificates

If you order a SSL certificate, the signing authority will ask you which kind of web server you are using and a CSR file. For the web server, select *other*. For the CSR, use the following command:

```
openssl req -out certificate.csr -new -newkey rsa:2048 -nodes -keyout certificate.  
↪key
```

When OpenSSL asks for the *Common Name* then fill in the site's hostname (e.g. `www.example.com`).

From the SSL certificate authority you will receive a signed `.crt` file and maybe a `cabundle.crt` file.

See the section *Certificate and key files* above for instructions how to use the `.crt` and `.key` files.

Generating a self signed certificate

There is no need to make your own self signed certificate as Zotonic will generate one for every site.

Nevertheless, if you want to use your own self signed certificate, then run the following commands:

```
openssl req -x509 -nodes -days 3650 -subj '/CN=www.example.com' -newkey rsa:2048 \
-keyout certificate.key -out certificate.crt
```

This generates a private key of 2048 bits and a certificate that is valid for 10 years.

mod_ssl_letsencrypt

See also:

[mod_ssl_ca](#) (page 395), [Port configurations](#) (page 630)

Request certificates from Let's Encrypt.

Let's Encrypt <<https://www.letsencrypt.com/>> provides free SSL certificates.

Zotonic can request these certificates automatically, easing deployment of https secured web sites.

Hostname & port requirements

There are some criteria that must for each site requesting a certificate.

1. Primary hostname(s) resolve using DNS
2. Resolved DNS address is not a LAN address
3. Site is reachable on the resolved address
4. Listening for the hostname on that address

Zotonic *must* listen on http port 80 and ssl port 443 for connections. If you use any other ports then requesting a certificate will fail.

See [Port configurations](#) (page 630) for more information about the configuring the correct port numbers and optional proxy settings.

Requesting a certificate

In the admin, go to System > Modules and ensure that `mod_ssl_letsencrypt` is enabled.

After `mod_ssl_letsencrypt` is enabled, go to System > SSL Certificates.

In the *Let's Encrypt* panel you can request a certificate. Check the alternative names you want to include in the certificates. (E.g. *example.com* and *www.example.com*).

The certificate request will run on the background and the status will be shown in the panel.

After a certificate was received, make sure that Let's Encrypt is the first module on the SSL Certificates list by disabling all modules above Lets's Encrypt.

Now go to your site using https, you should be see your site protected by a Let's Encrypt certificate.

Certificate and key files

The certificate and key files are placed into the site sub-directory of the security directory. The subdirectory will be: `sitename/letsencrypt/`

Where *sitename* must be replaced with the name of your site.

The security directory can be found by inspecting the output of:

```
bin/zotonic config
```

The Zotonic *security* directory can be in one of the following directories:

- The environment variable ZOTONIC_SECURITY_DIR
- The `~/.zotonic/security` directory
- The `/etc/zotonic/security` directory (only on Linux)
- The OS specific directory for application data files

The OS specific directories are:

- On Unix: `~/.config/zotonic/security/`
- On macOS: `~/Library/Application Support/zotonic/security/`

The default is the OS specific directory.

If there is a directory `priv/security/letsencrypt` inside your site's OTP application folder then that directory will be used.

mod_survey

Adds the concept of *survey resources*: user-definable forms which can be created in the admin interface and filled out by the website's visitors.

Survey question types

The following question types are defined in the survey.

likert Answer a question on a scale of 5 points, from “completely disagree” (1) to “completely agree” (5).

long answer An open question with a big text field.

matching Question type which allows you to match given answers to each other.

narrative Question type for specifying inline questions in a narrative fashion.

page break Breaks the survey into multiple pages.

short answer An open question with a single-lined text field. You have the option of specifying a validation like email, date, numeric.

thurstone A multiple choice field. Like multiple choice, but more powerful. The choices are translatable, and you have the possibility to select either a single answer, multiple answers or submit the form directly when choosing an answer.

true or false Answers a true or false question. You have the option to specify custom texts for both the options.

yes or no Like true or false, answers a true or false question. You have the option to specify custom texts for both the options.

multiple choice A simple multiple choice field that has the added option that the multiple choice can be a numeric value, in which case an overview of the total value will be shown in the printable list and beneath the survey pie chart. This is useful for creating forms which require you to enter an amount or quantity, e.g. for a reservation system. Multiple choice fields cannot currently be translated, use the “thurstone” question type in that case.

category Choose a single resource from a given category as the answer to this question.

subhead Renders a sub-heading between questions.

prompt Renders an extra prompt block.

text block Renders a text block between questions.

Intercepting survey submissions

When a survey is submitted, the survey module sends out a `#survey_submit{}` notification.

This notification has the following fields:

- `id` - The id of survey being submitted
- `handler` - A handler name (see below)
- `answers` - The answers that were filled in
- `missing` - answers that were missing
- `answers_raw` - Unprocessed answers, e.g. the raw submission

To intercept a survey submission you would observe this `survey_submit` notification, and return `ok`:

```
observe_survey_submit(#survey_submit{ id = SurveyId }, Context) ->
  ?DEBUG(SurveyId),
  ok.
```

Creating a custom survey handler

The survey edit page has a dropdown for so-called “survey handlers”. A survey handler is a property that is set on the resource that indicates the handler that needs to be taken. Handlers are collected using the `#survey_get_handlers{}` *fold* notification.

For instance, the following defines a handler called “email_me”:

```
observe_survey_get_handlers(#survey_get_handlers{}, All, Context) ->
[
  {<<"email_me">>, ?__(<<"E-mail me when survey is submitted">>, Context)}
  | All
].
```

Each handler will show up in the dropdown list and the editor can pick which handler he wants. The value chosen is passed along in the `handler` property of the survey submission, and as such can be used to intercept the survey submission:

```
observe_survey_submit(#survey_submit{ handler = <<"email_me">>, id = SurveyId },
  ↪Context) ->
  %% Do something here for surveys which have 'email_me' selected as handler
  ok;
observe_survey_submit(#survey_submit{}, _Context) ->
  %% Let other surveys use the default submission mechanism
  undefined.
```

Configurations keys

In the survey result editor it is possible to link an answer to a newly created person.

The category and content group for this person can be configured via the following two keys:

- `mod_survey.person_category`, default to `person`
- `mod_survey.person_content_group`, defaults to `default_content_group`

Todo

Add more documentation

mod_syslog

Todo

Not yet documented.

mod_tkvstore

See also:

[m_tkvstore](#) (page 581)

Simple (type,key)/value store. Stores data in the store with minimal latency and (local) serialization of get/put requests.

The store itself is implemented as a PostgreSQL table with columns *type*, *key* and *props*. *props* holds the value for the given type+key combination.

A model, [m_tkvstore](#) (page 581), is provided to give easy access to the type+key combinations from the templates, and to perform get/put operations from within Erlang.

mod_translation

This module provides support for dealing with multiple languages.

How content and static strings are translated is explained in full in [Translation](#) (page 53).

Language as part of the URL

By default, [mod_translation](#) (page 400) prefixes each URL (using [URL rewriting](#) (page 24)) in your website with the code of the current language. The idea behind this is that each language version of a [resource](#) gets its own URL, and is as such indexable for Google.

This behaviour is enabled by default, but can be switched off in the admin, by going to *Structure, Translation*. There is a checkbox labelled “Show the language in the URL”.

Alternatively you can set the config key `mod_translation.rewrite_url` to `false`.

Programmatically switching languages

In a template, you can use [mod_translation](#) (page 400)’s *postback* hook to switch between languages:

```
{% button text="Dutch" postback={set_language code="nl"} delegate=`mod_
↪translation` %}
```

Creates a button which switches to Dutch. And another one for english:

```
{% button text="English" postback={set_language code="en"} delegate=`mod_
↪translation` %}
```

Supporting right-to-left languages

For basic use you don't need to do anything. Zotonic base site adds a `lang` attribute to the `html` tag, and when a right-to-left language is selected (for instance Arabic), the browser will interpret `lang="ar"` and automatically adapt the content to right-to-left.

Custom right-to-left content

If you write your own templates, you can add the `lang` tag in the `html` or `body` tag, for instance:

```
<body {% include "_language_attrs.tpl" id=id %} >
```

This will generate the following, when Zotonic selected Arabic for the page with `id`:

```
<body xml:lang="ar" lang="ar" dir="rtl" class="rtl">
```

When you want to add an extra class added to the `rtl` or `ltr` class you can use:

```
<body {% include "_language_attrs.tpl" id=id class="my-body-class" %} >
```

To create individual right-to-left elements, you can use the same principle:

```
<div {% include "_language_attrs.tpl" %}></div>
```

And when you want to force a specific language:

```
<div {% include "_language_attrs.tpl" language='en' %} >This is English content</div>
```

mod_twitter

See also:

[mod_facebook](#) (page 370), [mod_linkedin](#) (page 377)

Import Twitter updates in your Zotonic site in realtime.

The Twitter module allows you to have Tweets 'mirrored' from Twitter into your Zotonic site. It does this by creating a background process which has a continuous HTTP connection to the Twitter Streaming API. This way, Tweets arrive instantly at your Zotonic site, no polling necessary!

Twitter login

First, create an app on Twitter. Point it's callback URL to your Zotonic site. Then in your site, [activate](#) (page 60) `mod_twitter`. Head to 'Auth' > 'External services' in the admin interface to enter your Twitter app's consumer key and secret. Enable Twitter login by checking the 'Use Twitter authentication' box. This will add a 'Log in with Twitter' button to the login form on your site.

Importing Tweets

The `mod_twitter` module can import Tweets into Zotonic using Twitter's [Streaming API](#).

[Activate](#) (page 60) `mod_twitter`. Then head to 'Auth' > 'External services' in the admin interface to enter your Twitter access token and secret.

The module will follow any user(s) in your site that you entered their Twitter ID for. You can so by going to a person's page in the admin. In the edit page's right sidebar, you will find a new item, 'Twitter ID'. Here, enter the [numeric Twitter ID](#) of the user. Only [public \(or unprotected\)](#) Tweets will be imported.

Domain model

The *domain model* for this module is the following: the module creates a *category* (page 7) ‘Tweet’ as a subcategory of ‘text’. Each time a Tweet is imported, a resource of this category is created.

From the Tweet there is an ‘author’ edge to the person that created the Tweet (i.e. the user who you set the Twitter ID on).

The body text of the Tweet resource is HTML will already be preprocessed with the *twitter* (page 485) filter, so URLs, hashtags and @-links are converted to HTML already when the Tweet is saved.

The original value of the Tweet (the JSON object received from Twitter) is stored inside the `tweet` property in the Tweet resource, and can be displayed like this:

```
{% print m.rsc[id].tweet %}
```

mod_video

See also:

mod_video_embed (page 403), *mod_oembed* (page 387), *mod_audio* (page 358), *media* (page 529)

Adds support for viewing and handling video medium items.

This module converts uploaded videos to h264 and adds a poster (preview) image of the movie.

Note: `mod_video` uses the command-line utilities `ffmpeg` and `ffprobe`. For `mod_video` to function correctly they must be present in the search path of Zotonic.

Uploading & conversion

The video module hooks into the media model to intercept any video upload. If a video is uploaded the following steps are done:

- The video is moved to the site’s `files/video_queue/` directory.
- A video conversion task is added to the pivot task queue, this task will restart a video conversion in case of any problems.
- A video conversion process is started, supervised by the video module.
- The uploaded medium is replaced by a static `lib/images/processing.png` image.

Only a single video conversion process is allowed to run at any time. This to prevent overloading the server.

After the video is converted the resource’s medium record is replaced with the converted video. The frame at 10 seconds (at 1 second for movies shorter than 30 seconds) is added as the preview image of the video.

If a video can’t be converted then the video is replaced with the error image, found in `lib/images/broken.png`.

Viewing

The video module extends the `{% media %}` tag for viewing `video/mp4` videos.

It uses the template `_video_viewer.tpl` for viewing. For the best viewing results, add `css/video.css` to your included css files.

mod_video_embed

See also:

[mod_oembed](#) (page 387), [mod_video](#) (page 402), [mod_audio](#) (page 358), [media](#) (page 529)

This module, if activated, checks the pasted URLs in the *create media / page* dialog of the admin. It will show an embed option for Youtube and Vimeo URLs. It will also cleanup pasted embed code for these and other services.

When used in the Zotonic site, the `{% media %}` tag then displays the embed code.

This module is accompanies [mod_oembed](#) (page 387) and can be used for integrating with services that do not have oEmbed support but do provide HTML embed-code functionality.

Todo

Add more documentation

mod_wires

Actions, tags (also known as screen components), and javascript for user interfaces using *wires*.

Used by [mod_admin](#) (page 346) and the other administrative modules.

Wires are actions that are directly coupled to user interface elements. These couplings are defined in the templates using the *wire* (page 551) tag.

mod_zotonic_site_management

Todo

Not yet documented.

6.1.2 Actions

An overview of all *wire actions* (page 56).

Actions

with_args

- Module: [mod_wires](#) (page 403)

Apply actions with arguments added.

This action takes a list of other actions. One or more arguments are added to the actions before the actions are executed. This action is mostly used in included templates or callbacks. An example can be seen with the *typeselect* (page 423) action.

Another example, assume we have a template “_list_action.tpl”:

```
{% for id in list %}
  <li><a id="{ { #list.id } }" href="#">{ { m.rsc[id].title } }</a></li>
  { % wire id=#list.id action={with_args action=my_action arg={id id}} % }
{% endfor %}
```

Then we can pass an action to this template:

```
{% include "_list_action.tpl" list=[1,2,3,4,5] my_action={redirect dispatch="admin_
↪edit_rsc"} %}
```

The result will be a list of titles for the pages with id 1..5. Every title will be a link to its admin page, as the argument *id* will be added to the *my_action*.

Admin

admin_tasks

- Module: *mod_admin* (page 346)

Action module which provides postback handlers for the “status” view of the admin:

- Rebuild search index
- Flush cache
- Renumber categories

Todo

Extend documentation

redirect_incat

- Module: *mod_admin* (page 346)

Todo

Not yet documented.

Modules (Admin)

module_rescan

- Module: *mod_admin_modules* (page 355)

Rescans all modules, to find all templates, lib files, dispatch rules, etc. again.

Todo

Extend documentation

module_toggle

- Module: *mod_admin_modules* (page 355)

Activate/deactivate a module in the module manager in the admin interface.

Todo

Extend documentation

Backup

backup_start

- Module: *mod_backup* (page 360)

Action which starts a manual backup.

Todo

Extend documentation

Config

config_delete

- Module: *mod_admin_config* (page 351)

Trigger the deletion of a configuration value. Used in the admin.

Todo

Extend documentation

config_toggle

- Module: *mod_admin_config* (page 351)

Toggle a configuration value. Used in the admin, for instance when displaying a “live” checkbox the state of which should reflect a config value.

Todo

Extend documentation

dialog_config_delete

- Module: *mod_admin_config* (page 351)

Open a dialog that asks confirmation to delete a configuration key/value pair.

Todo

Extend documentation

dialog_config_edit

- Module: *mod_admin_config* (page 351)

Open a dialog to edit a configuration key/value pair.

Todo

Extend documentation

dialog_config_new

- Module: *mod_admin_config* (page 351)

Open a dialog to create a new configuration key/value pair.

Todo

Extend documentation

Development**development_templates_stream**

- Module: *mod_development* (page 364)

Stream template updates to the user agent.

Todo

Extend documentation

Dialogs**dialog**

- Module: *mod_wires* (page 403)

See also:

actions *dialog_open* (page 407), *dialog_close* (page 407) and *overlay_open* (page 407).

Opens a dialog with a predefined HTML content and title.

Example:

```
{% button action={dialog title="Wisdom" text="<p>The world is a pancake.</p>"} %}
```

This opens a dialog with the title “Wisdom”. The dialog is empty except for the text “The world is a pancake”.

Normally, instead of this action, the action *dialog_open* (page 407) is used. The action *dialog_open* (page 407) shows a dialog that is rendered on the server.

Argument	Required	Description
title	required	Dialog header title
text	required	Dialog body text
width	optional	Dialog width in pixels. Use "large" for a wide dialog and "small" for a small dialog.
addclass	optional	classname will be appended to default dialog class
back-drop	optional	boolean (0, 1), or the string "static" for a modal dialog (does not close on backdrop click); default 1
center	optional	boolean (0, 1) default 1; set to 0 to align the dialog at the top

dialog_open

- Module: *mod_wires* (page 403)

See also:

actions *dialog_close* (page 407), *dialog* (page 406) and *overlay_open* (page 407).

Renders a template on the server and opens a dialog with the HTML output of the template.

Example:

```
{% button text="cancel" action={dialog_open title="Select a name" template="_select_name.tpl" arg=100} %}
```

The title of this new dialog will be “Select a name”, its contents are the output of rendering the template “_select_name.tpl”. All arguments are handed as arguments to the template. In this example the template “_select_name.tpl” is rendered with the arguments “title”, “template” and “arg”.

dialog_close

- Module: *mod_wires* (page 403)

See also:

actions *dialog_open* (page 407) and *dialog* (page 406).

Closes the currently open dialog. When there is no dialog open then nothing happens.

Example:

```
{% button text="cancel" action={dialog_close} %}
```

This button closes any open dialog when clicked.

overlay_open

- Module: *mod_wires* (page 403)

See also:

actions *overlay_close* (page 408), *dialog_open* (page 407) and *dialog* (page 406).

Renders a template on the server and opens a full screen overlay with the HTML output of the template.

Example:

```
{% button text="show story" action={overlay_open template="_story.tpl" id=1234} %}
```

This opens an overlay over the current content. The template `_story.tpl` will be rendered with the argument `id` (and possibly any other arguments). The rendered html will then be shown inside the overlay.

The overlay template is a `div` with the class `modal-overlay`. Extra classes can be added using the `class` argument:

```
{% wire action={overlay_open template="_splash.tpl" class="splash"} %}
```

overlay_close

- Module: *mod_wires* (page 403)

See also:

actions *overlay_open* (page 407), *dialog_open* (page 407) and *dialog* (page 406).

Closes the currently open overlay. When there is no overlay open then nothing happens.

Example:

```
{% button text="cancel" action={overlay_close} %}
```

This button closes any open overlay when clicked.

DOM Elements

add_class

- Module: *mod_wires* (page 403)

See also:

actions *remove_class* (page 413) and *toggle_class* (page 415).

Add a css class to an html element.

Example:

```
{% button action={add_class target="myid" class="newclass"} %}
```

Adds the CSS class “newclass” to the element with HTML id “myid”.

animate

- Module: *mod_wires* (page 403)

Add a `$(...).animate` jQuery call to the target element.

Arguments:

- speed
- easing
- options

Todo

Extend documentation

buttonize

- Module: *mod_wires* (page 403)

Todo

Not yet documented.

effect

- Module: *mod_wires* (page 403)

Add a `$(...).effect` jQuery call to the target element.

Todo

Extend documentation

fade_in

- Module: *mod_wires* (page 403)

See also:

actions *toggle* (page 415), *show* (page 414), *hide* (page 409), *fade_out* (page 409), *slide_down* (page 414), *slide_up* (page 415), *slide_fade_in* (page 414) and *slide_fade_out* (page 414).

Show an element by animating the opacity.

Example:

```
{% button action={fade_in target="myid"} %}
```

Shows the element with id “myid” when the button is clicked.

fade_out

- Module: *mod_wires* (page 403)

See also:

actions *toggle* (page 415), *show* (page 414), *hide* (page 409), *fade_in* (page 409), *slide_down* (page 414), *slide_up* (page 415), *slide_fade_in* (page 414) and *slide_fade_out* (page 414).

Hide an element by animating the opacity.

Example:

```
{% button action={fade_out target="myid"} %}
```

Hides the element with id “myid” when the button is clicked.

hide

- Module: *mod_wires* (page 403)

See also:

actions [toggle](#) (page 415), [show](#) (page 414), [fade_in](#) (page 409), [fade_out](#) (page 409), [slide_down](#) (page 414), [slide_up](#) (page 415), [slide_fade_in](#) (page 414) and [slide_fade_out](#) (page 414).

Hide an element without any animation.

Example:

```
{% button action={hide target="myid"} %}
```

Hides the element with id “myid” when the button is clicked.

insert_after

- Module: [mod_wires](#) (page 403)

Insert the result of a render action after of an HTML element.

Todo

Extend documentation

insert_before

- Module: [mod_wires](#) (page 403)

Insert the result of a render action before an HTML element.

Todo

Extend documentation

insert_bottom

- Module: [mod_wires](#) (page 403)

See also:

actions [insert_after](#) (page 410), [insert_before](#) (page 410), [insert_top](#) (page 411) and [update](#) (page 416).

Inserts HTML after the contents of an HTML element.

Adds a template or a literal HTML text after the existing content.

Example:

```
<div id="mydiv"><p>Bye Bye.</p></div>
{% button text="hello" action={insert_bottom target="mydiv" text="<p>Hello World!</p>" %}
```

After the button is clicked, the contents of the div will be `<p>Bye Bye.</p><p>Hello World!</p>`.

Another example, now rendering a template:

```
<ul id="mylist"><li>Some item</li></ul>
{% button text="hello" action={insert_bottom target="mylist" template="_list_item.
  ↳tpl" id=42} %}
```

This insert the output of the template `_list_item.tpl` below the existing ``. All arguments to the update action are also arguments to the template.

insert_top

- Module: *mod_wires* (page 403)

See also:

actions *insert_after* (page 410), *insert_before* (page 410), *insert_bottom* (page 410) and *update* (page 416).

Inserts HTML before the contents of an HTML element.

Adds a template or a literal HTML text before the existing content.

Example:

```
<div id="mydiv"><p>Bye Bye.</p></div>
{% button text="hello" action={insert_top target="mydiv" text="<p>Hello World!</p>"} %}
```

After the button is clicked, the contents of the div will be `<p>Hello World!</p><p>Bye Bye.</p>`.

Another example, now rendering a template:

```
<ul id="mylist"><li>Some item</li></li>
{% button text="hello" action={insert_top target="mylist" template="_list_item.tpl" id=42} %}
```

This insert the output of the template `_list_item.tpl` above the existing ``. All arguments to the update action are also arguments to the template.

jquery_effect

- Module: *mod_wires* (page 403)

Trigger various jQuery effects on the target element. Mostly, each of these effects have their own action as a shortcut, for example *show* (page 414), *hide* (page 409).

Arguments:

- type - one of *show*, *hide*, *remove*, *slide_toggle*, *toggle*, *set_class*, *add_class*, *remove_class*, *fade_in*, *fade_out*, *slide_up*, *slide_fade_out*, *slide_fade_in*, *disable*, *enable*, *effect*, *animate*.
- speed
- class
- easing
- effect
- options

Todo

Extend documentation

mask

- Module: *mod_wires* (page 403)

See also:

action [unmask](#) (page 416), [mask_progress](#) (page 412)

Places a mask over an element, useful for blocking user interaction during lengthy postbacks.

Example:

```
<a id="{#fb_logon}" href="#facebook">
  
</a>
{% wire id=#fb_logon
  action={mask target="logon_outer" message="Waiting for Facebook ."}
  action={redirect dispatch="facebook_authorize"} %}
```

In this example the *logon_outer* div will be masked while the browser is being redirected to Facebook.

Form postbacks are automatically masked.

Note that you need to include a css and a js file to have working masks:

```
{% lib "css/jquery.loadmask.css" %}
{% lib "js/modules/jquery.loadmask.js" %}
```

This action takes three possible arguments:

Argument	Description	Example
body	Mask the whole page	body
target	The id of the element to be masked.	target="search-form"
message	Message to show next to the spinner image.	message="Searching..."
delay	Delay (in milliseconds) before the mask is shown. Only shows the mask during lengthy actions.	delay=200

mask_progress

- Module: [mod_wires](#) (page 403)

Sets the progress bar of a [mask](#) (page 411).

The progress bar can be set to a percentage in the range 0...100. The target of the `mask_progress` must be the same as the target of an earlier `mask` action.

Example:

```
{% wire action={mask_progress target="logon_outer" percent=50} %}
```

In this example the *logon_outer* progress bar will show as halfway (50%).

move

- Module: [mod_wires](#) (page 403)

Move an element to another place, appending it to the target. The element is given by id with the `element` argument, or with the `element_sel` argument for a CSS selector.

Todo

Extend documentation

remove

- Module: *mod_wires* (page 403)

See also:

Actions (page 403), *button* (page 533)

Remove an element from the page.

For example, the following removes the *foo* div from the page:

```
<div id="foo">I am the foo div</div>
{% button text="Remove foo" action={remove target="foo"} %}
```

Without target, the action removes its triggering element:

```
{% button text="Click me to remove me" action={remove} %}
```

remove_class

- Module: *mod_wires* (page 403)

See also:

actions *add_class* (page 408) and *toggle_class* (page 415).

Remove a CSS class from an HTML element.

Example:

```
{% button action={remove_class target="myid" class="newclass"} %}
```

Removes the CSS class “newclass” from the element with HTML id “myid”.

replace

- Module: *mod_wires* (page 403)

Replace the target HTML element by new one.

Todo

Extend documentation

set_class

- Module: *mod_wires* (page 403)

Set the class of an element.

Example:

```
<div id="x" class="not-init"></div>
{% button text="Init" action={set_class target="x" class="init"} %}
```

This uses the jQuery *attr('class', class_name)* method to set the new class.

show

- Module: *mod_wires* (page 403)

See also:

actions *toggle* (page 415), *hide* (page 409), *fade_in* (page 409), *fade_out* (page 409), *slide_down* (page 414), *slide_up* (page 415), *slide_fade_in* (page 414) and *slide_fade_out* (page 414).

Show an element without any animation.

Example:

```
{% button action={show target="myid"} %}
```

Shows the element with id *myid* when the button is clicked.

slide_down

- Module: *mod_wires* (page 403)

See also:

actions *toggle* (page 415), *show* (page 414), *hide* (page 409), *fade_in* (page 409), *fade_out* (page 409), *slide_up* (page 415), *slide_fade_in* (page 414) and *slide_fade_out* (page 414).

Show an element by animating the height.

Example:

```
{% button action={slide_down target="myid"} %}
```

Shows the element with id *myid* when the button is clicked.

slide_fade_in

- Module: *mod_wires* (page 403)

See also:

actions *toggle* (page 415), *show* (page 414), *hide* (page 409), *fade_in* (page 409), *fade_out* (page 409), *slide_down* (page 414), *slide_up* (page 415) and *slide_fade_out* (page 414).

Show an element by animating the height and opacity.

Example:

```
{% button action={slide_fade_in target="myid"} %}
```

Shows the element with id *myid* when the button is clicked.

slide_fade_out

- Module: *mod_wires* (page 403)

See also:

actions *toggle* (page 415), *show* (page 414), *hide* (page 409), *fade_in* (page 409), *fade_out* (page 409), *slide_down* (page 414), *slide_up* (page 415) and *slide_fade_in* (page 414).

Hide an element by animating the height and opacity.

Example:

```
{% button action={slide_fade_out target="myid"} %}
```

Hides the element with id *myid* when the button is clicked.

slide_toggle

- Module: *mod_wires* (page 403)

Toggle an element by sliding it up and down.

Todo

Extend documentation

slide_up

- Module: *mod_wires* (page 403)

See also:

actions *toggle* (page 415), *show* (page 414), *hide* (page 409), *fade_in* (page 409), *fade_out* (page 409), *slide_down* (page 414), *slide_fade_in* (page 414) and *slide_fade_out* (page 414).

Hide an element by animating the height.

Example:

```
{% button action={slide_up target="myid"} %}
```

Hides the element with id *myid* when the button is clicked.

toggle

- Module: *mod_wires* (page 403)

See also:

actions *show* (page 414), *hide* (page 409), *fade_in* (page 409), *fade_out* (page 409), *slide_down* (page 414), *slide_up* (page 415), *slide_fade_in* (page 414) and *slide_fade_out* (page 414).

Toggle the visibility of an element.

Example:

```
{% button action={toggle target="myid"} %}
```

Shows the element with id *myid* if it was hidden, otherwise hide it.

toggle_class

- Module: *mod_wires* (page 403)

See also:

actions *add_class* (page 408) and *remove_class* (page 413).

Toggle a CSS class from an HTML element.

Example:

```
{% button action={toggle_class target="myid" class="newclass"} %}
```

When the HTML element with id “myid” has the CSS class “newclass” then it is removed, otherwise it is added.

unmask

- Module: *mod_wires* (page 403)

See also:

action *mask* (page 411).

Removes a mask that was placed over an element using the *mask* (page 411) action.

Example:

```
{% wire action={unmask target="logon_outer"} %}
```

In this example the mask over the *logon_outer* div will be removed.

update

- Module: *mod_wires* (page 403)

See also:

actions *update_iframe* (page 417), *insert_top* (page 411) and *insert_bottom* (page 410).

Updates the content of an HTML element with a template or a literal HTML text.

Example:

```
<div id="mydiv"><p>Bye Bye.</p></div>
{% button text="hello" action={update target="mydiv" text="<p>Hello World!</p>"} %}
```

When clicked, the contents of the div will be set to the HTML fragment *<p>Hello World!</p>*. This replaces any content present.

Note: Use the *update_iframe* (page 417) action for updating the contents of an *iframe* element.

Another example, now rendering a template:

```
<ul id="mylist"><li>Some item</li></li>
{% button text="hello" action={update target="mylist" template="_list_item.tpl"
↪id=42} %}
```

This updates the ** with the output of the template *_list_item.tpl*. All arguments to the update action are also arguments to the template.

Argument	Description	Example
target	The id of the element receiving the rendered HTML.	<code>target="my-view"</code>
text	Literal HTML text to be inserted, no escaping will be done.	<code>text="Hello World"</code>
template	Name of the template to be rendered.	<code>template="_list_view.tpl"</code>
include_all	Add this argument to include all templates with the same name. If not added then the best template will be used.	<code>include_all</code>
catinclude	Add this argument to use a catinclude (page 515) instead of a normal include of the template. The <i>id</i> argument <i>must</i> be present for a catinclude to work.	<code>catinclude id=1</code>

All other arguments are passed as-is to the included template(s).

update_iframe

- Module: [mod_wires](#) (page 403)

See also:

action [update](#) (page 416).

Note: This action is only used to update an `iframe` element. Use the [update](#) (page 416) action for updating the contents of a normal HTML element.

Updates the content of an `iframe` with a template or a literal HTML text.

Example:

```
<iframe id="preview"></iframe>
{% button text="Show Email" action={update_iframe target="preview" template="email.
→tpl" recipient_id=m.acl.user} %}
```

When clicked, the contents of the `iframe` will be set to the rendered `email.tpl` template. This replaces any content present.

Argument	Description	Example
target	The id of the <code>iframe</code> receiving the rendered HTML document.	<code>target="my-view"</code>
text	Literal HTML doc to be inserted, no escaping will be done.	<code>text="<html>...</html>"</code>
template	Name of the template to be rendered.	<code>template="page.tpl"</code>
catinclude	Add this argument to use a catinclude (page 515) instead of a normal include of the template. The <i>id</i> argument <i>must</i> be present for a catinclude to work.	<code>catinclude id=1</code>

All other arguments are passed as-is to the included template(s).

Editor

editor_add

- Module: [mod_wires](#) (page 403)

Add WYSIWYG editor controls to all `textarea`'s with the `z_editor` class in the target.

Todo

Extend documentation

editor_remove

- Module: *mod_wires* (page 403)

Remove any WYSIWYG editor controls from all textarea's with the `z_editor` class in the target.

Todo

Extend documentation

zlink

- Module: *mod_admin* (page 346)

Used for inserting an internal link in the TinyMCE editor in the admin.

Todo

Extend documentation

zmedia

- Module: *mod_admin* (page 346)

Used for triggering the insertion of a media item in the TinyMCE editor in the admin.

Todo

Extend documentation

zmedia_choose

- Module: *mod_admin* (page 346)

Used after a media item is selected in the media chooser for the TinyMCE editor.

Todo

Extend documentation

zmedia_has_chosen

- Module: *mod_admin* (page 346)

Used by the admin as a callback when a media file has been selected for insertion into the rich-text editor.

Todo

Extend documentation

Events

postback

- Module: *mod_wires* (page 403)

This action sends a message to the event handler on the server.

Example:

```
{% button text="Go" action={postback postback="go" action={growl text="sent message"
↪ "}} %}
```

Note: The *button* (page 533) scomp can also take a postback argument directly.

After clicking the button the event *go* will be sent to the *controller* module on the server and a *growl* (page 427) message will be displayed.

The *event/2* function in the controller module will be called as:

```
event(#postback{message=go, trigger=TriggerId, target=TargetId}, Context)
```

This action can have the following arguments:

Argument	Description	Example
post-back	The message that will be send to the server module.	postback={ my_message arg="hello" }
de- e- gate	The name of the Erlang module that will be called. Defaults to the controller module generating the page.	delegate="my_module"
ac- tion	Any other actions that will be executed when the postback is done. This parameter can be repeated.	action={ show target="wait" }
in- ject_args	If set to <i>true</i> , and postback is a tuple (as in the <i>my_message</i> example in this table), any values from the args in the postback will replace the arg value in the postback argument. This is useful when the arg is coming from an outer action and not set explicitly in the template code (as is done in the example for illustration). The value of <i>some_arg</i> in the postback handler will be <i>123</i> .	{ postback postback={ my_event some_arg } inject_args some_arg=123 }
qarg	Post the value of an input or select with the postback. The value of the <i>qarg</i> argument is the id of the element to be posted. Multiple <i>qarg</i> arguments can be given. On the server the value will be available as a normal query argument using <i>z_context:get_q/2</i>	qarg="my-input-id"

New in version 0.9.0: Added *inject_args* option.

trigger_event

- Module: *mod_wires* (page 403)

Trigger a named { % wire % } with an action. All args will be args to the named wire. The trigger's name argument is the name of the wire.

Todo

Extend documentation

publish

- Module: *mod_mqtt* (page 381)

Publishes a message on the topic tree of the current page.

Example:

```
{% button text="Hello" action={publish topic="some/topic" js_msg="Hello World!"} %}
```

When clicked, the message set to the message "Hello World! " is published.

Another example, now publishing a more complex message:

```
{% button text="hello" action={publish topic="some/topic" foo="bar" spam="eggs"} %}
```

When clicked, the message set to the message {foo: "bar", spam: "eggs"} is published.

Prefix the topic with `bridge/origin/` to relay it to the topic tree on the server.

Argument	Description	Example
topic	The topic of the message.	<i>topic="test"</i>
js_msg	Literal javascript message to publish. Will be escaped.	<i>js_msg="Hello World"</i>
<key>	A literal javascript value, will be escaped and added to object	<i>spam="eggs"</i>

Forms

disable

- Module: *mod_wires* (page 403)

See also:

action *enable* (page 420).

Sets the “disabled” attribute of a HTML tag and adds the CSS class “disabled”.

Example:

```
<input id="myid" type="text" value="hello" />
{% button text="disable" action={disable target="myid"} %}
```

After clicking the button the input will be:

```
<input id="myid" disabled="disabled" class="disabled" type="text" value="hello" />
```

enable

- Module: *mod_wires* (page 403)

See also:

action [disable](#) (page 420).

Resets the “disabled” attribute of a HTML tag and removes the CSS class “disabled”.

Example:

```
<input id="myid" disabled="disabled" class="disabled" type="text" value="hello" />
{% button text="enable" action={enable target="myid"} %}
```

After clicking the button the input will be:

```
<input id="myid" class="" type="text" value="hello" />
```

event

- Module: [mod_wires](#) (page 403)

Bind actions to a jQuery event or submit a form.

This action is the base action for the [wire](#) (page 551) scomp. Normally this event is not used directly.

focus

- Module: [mod_wires](#) (page 403)

Add a `$(...).focus()` jQuery call to the target element to give it input focus.

form_reset

- Module: [mod_wires](#) (page 403)

Resets the target form to its initial state.

Todo

Extend documentation

reset

- Module: [mod_wires](#) (page 403)

Resets the enclosing form, a specifically targeted form or the closest form to an element.

Example:

```
<form method="get" action="/search">
  <input type="text" name="q" value="" />
  {% button text="search" action={submit} %}
  {% button text="cancel" action={reset} %}
</form>
```

Another example:

```
<form id="search-form" method="get" action="/search">
  <input type="text" id="q" name="q" value="" />
</form>
{% button text="search" action={submit closest="q"}
{% button text="cancel" action={reset closest="q"} %}
```

Clicking on the button will reset the form *search-form* as it is the closest form to the element with id *q*.

The reset form action is mostly used in the result of event handlers.

This action takes two possible arguments, when neither is defined then the form enclosing the trigger element will be reset.

Argument	Description	Example
target	The id of the form to be reset.	target="search-form"
closest	The id of an element that is close to the form to be reset. When no argument value is supplied then it defaults to the id of the trigger element (for example the button the action is coupled to).	closest

set_value

- Module: *mod_wires* (page 403)

Set the value of a form field.

Example:

```
<input type="text" id="x" name="xyz" value="" />
{% button text="fill" action={set_value target="x" value="etaoinstrdlu"} %}
```

Clicking on the button will set the value of the input element to the most interesting string *etaoinstrdlu*.

This action can set the value of any input element, select or text area. It uses the jQuery *val()* method to set the value.

submit

- Module: *mod_wires* (page 403)

Submits the enclosing form, a specifically targeted form or the closest form to an element.

Example:

```
<form method="get" action="/search">
  <input type="text" name="q" value="" />
  {% button text="search" action={submit} %}
</form>
```

Another example:

```
<form id="search-form" method="get" action="/search">
  <input type="text" id="q" name="q" value="" />
</form>
{% button text="search" action={submit closest="q"} %}
```

Clicking on the button will submit the form *search-form* as it is the closest form to the element with id *q*.

The submit action is mostly used in the result of event handlers.

This action takes two possible arguments, when neither is defined then the form enclosing the trigger element will be submitted.

Argument	Description	Example
target	The id of the form to be submitted.	target="search-form"
closest	The id of an element that is close to the form to be submitted. When no argument value is supplied then it defaults to the id of the trigger element (for example the button the action is coupled to).	closest

typeselect

- Module: *mod_wires* (page 403)

Show possible selections whilst typing.

Performs a search for the typed text whilst typing in an input field. Shows possible matching pages in a selectable list.

Example:

```
<form method="get" action="/search">
  <input type="search" id="person" name="person" value="" />
  <ul id="suggestions"></ul>
  <input type="hidden" id="person_id" value="" />
  {% wire id="person" type="input"
    action={typeselect cat="person"
                    target="suggestions"
                    action_with_id={with_args action={set_value target=
↪ "person_id"} arg={value select_id}}
                    action={submit}}
  %}
</form>
```

This is a rather complicated example. It connects the typeahead action to the input element. The list of suggestions will be shown in the `` with id *suggestions*. Only pages in the category *person* will be found.

The listed suggestions will have two actions attached. One action will set the value of the hidden input element *person_id* to the id of the selected suggestion (which is a *page*). The other action will submit the form.

The *action_with_id* arguments are always performed before the *action* arguments.

The *typeselect* action accepts the following arguments:

Argument	Description	Example
target	The id of element that will show the list of suggestions.	target="mylist"
cat	The category for the searched pages. This argument can be repeated.	cat="text"
template	Template used to show the list of possible pages. This defaults to the template <code>"_action_typeselect_result.tpl"</code> . The template gets the following arguments: result (list of ids), action_with_id and action.	template="_show_suggestions.tpl"
action_with_id	Actions executed when a suggestion is selected. The id of the selected page will be added as the <i>id</i> parameter. This argument can be repeated.	action_with_id={postback post-back="page_select"}
action	Actions executed when a suggestion is selected. This list is executed after the <i>action_with_id</i> actions. This argument can be repeated.	action={slide_up target="form-id"}

validation_error

- Module: *mod_wires* (page 403)

Render a validation error on the target. Text is given in the *text* argument.

Todo

Extend documentation

JavaScript

script

- Module: *mod_wires* (page 403)

This action executes JavaScript directly. It can be used to interface with non-Zotonic JavaScript libraries and functions.

Example:

```
{% button text="hello" action={script script="alert('hello world')"} %}
```

Clicking on the button will show a JavaScript alert with the text *hello world* in it.

Using template variables:

```
{% with "world" as recipient %}
  {% button
    text="hello"
    action={
      script
      script="alert('hello " ++ recipient ++ "')"
    }
  %}
{% endwith %}
```

Mailing list

dialog_mail_page

- Module: *mod_mailinglist* (page 379)

Shows the dialog to mail the current page (*resource*) to a single e-mail address. This is used in the frontend of a site to “share” the current page over e-mail.

Todo

Extend documentation

dialog_mailing_page

- Module: *mod_mailinglist* (page 379)

Shows the dialog to mail the current page (*resource*) to a mailing list. This is used in the admin “mailing status” interface.

Todo

Extend documentation

mailing_page_test

- Module: *mod_mailinglist* (page 379)

Post a message to the test mailing list, given with the `id` argument.

The `on_success` argument decides which actions are triggered after the page has been sent.

Todo

Extend documentation

mailinglist_confirm

- Module: *mod_mailinglist* (page 379)

Confirm a mailinglist subscription. Required argument is the `confirm_key`.

Other arguments:

- `on_success` - actions which get executed when the subscription is confirmed.
- `on_error` - actions which get executed when the subscription fails (e.g. wrong confirm key).

Todo

Extend documentation

mailinglist_unsubscribe

- Module: *mod_mailinglist* (page 379)

Cancel a mailing list subscription. The recipient id is given with the `id` argument.

The `on_success` argument decides which actions are triggered after unsubscribe is successful; `on_error` actions are triggered when unsubscribe fails.

Todo

Extend documentation

Notifications**alert**

- Module: *mod_wires* (page 403)

See also:

actions [growl](#) (page 427) and [confirm](#) (page 426).

Show an alert dialog.

Example:

```
{% button action={alert text="hello world"} %}
```

Shows an alert dialog with the text “hello world”.

Alert accepts the following arguments:

Argument	Description	Example
title	Title of the alert.	title=”Alert”
text	The text to be displayed.	text=”Hola!”
button	Text for the button. Defaults to “OK”	button=”Bummer”
only_text	Set this to not show the “OK” button.	only_text
action	Action to be done when the user clicks on the OK button. There can be multiple actions.	
backdrop	Show backdrop: true, false, or the string “static”	backdrop=false

The alert dialog is rendered using the `_action_dialog_alert.tpl` template. Override this template to change the contents of the alert dialog.

confirm

- Module: [mod_wires](#) (page 403)

See also:

actions [alert](#) (page 425) and [growl](#) (page 427).

Show a JavaScript confirm message and on confirmation triggers one or more actions and/or sends a postback to the server.

Example:

```
{% button action={confirm
    text="Format hard disk?"
    ok="Go Ahead!"
    action={growl text="Better not"}
    is_danger}
%}
```

Shows a JavaScript dialog with the question “Format hard disk?”. If this dialog is confirmed then the growl message “Better not” will appear. If the dialog is denied or canceled then nothing happens.

If there is a postback defined then the event handler for the postback will be called like:

```
event(#postback{message=Message, trigger=TriggerId, target=TargetId}, Context).
```

Confirm accepts the following arguments:

Argument	Description	Example
text	The text to be displayed.	text=_ "The answer to life and the rest?"
title	Title above the alert, defaults to _ "Confirm"	title=_ "Rescue the world, with an answer"
ok	The text of the ok button, defaults to _ "OK"	text="42"
cancel	The text of the cancel button, defaults to _ "Cancel"	text="No, thanks for the fish"
text_template	Template used to render the text, all action arguments are passed to the template.	text_template=_ "fancy_confirm.tpl"
action	One or more actions to be executed on confirmation. This argument can be repeated.	action={alert text="you said ok"}
on_cancel	One or more actions to be executed on cancelation	on_cancel={alert text="you said cancel"}
postback	Event to be sent back to the server if the ok button is clicked.	postback="clicked_confirm"
delegate	Erlang module handling the postback. Defaults to the controller generating the page.	delegate="my_event_module"
is_danger	If the 'ok' button should be flagged as dangerous.	is_danger

growl

- Module: *mod_wires* (page 403)

See also:

actions *alert* (page 425) and *confirm* (page 426); and *Enabling Growl Notifications* (page 277).

Show a message in the upper right corner of the browser window. The message will automatically disappear after some time.

Example:

```
{% button action={growl text="hello world"} %}
```

Shows a message with the text "hello world".

Growl accepts the following arguments:

Argument	Description	Example
text	The text to be displayed.	text="Hola!"
stay	When true then the message does not disappear automatically	stay
type	Type of the message, one of "notice" or "error". Default is "notice".	type="error"

Page handling

redirect

- Module: *mod_wires* (page 403)

This action redirects the browser to another page or back to the previous page.

Example:

```
{% button text="home" action={redirect location="/" } %}
```

Redirects back to the home page when the button is clicked.

Back in history example:

```
{% button text="Back" action={redirect back} %}
```

After clicking the button the browser will go back to the last page using the JavaScript history.

Example of using dispatch rules for the redirect location:

```
{% button text="edit" action={redirect dispatch="admin_edit_rsc" id=my_id} %}
```

When clicked the browser is redirected to the admin edit page for the *resource* with the id of *my_id*.

This action can have the following arguments:

Argument	Description	Example
back	When given then the browser is directed to the previous page.	back
dispatch	The name of a dispatch rule. All other parameters are assumed to be parameters for the dispatch rule.	dispatch="admin"
id	When back and dispatch are not defined then the redirect uri will be the page_url of the resource.	id=42
location	The http address to redirect to. Can be an url with or without host name.	location="http://example.com"

reload

- Module: *mod_wires* (page 403)

Reload the current page.

Example:

```
{% button text="refresh" action={reload} %}
```

Clicking on the button will reload the page.

Predicates and Connections

dialog_predicate_new

- Module: *mod_admin_predicate* (page 356)

Show a dialog for creating a new *predicate*.

Todo

Extend documentation

link

- Module: *mod_admin* (page 346)

See also:

unlink (page 429)

Add an *edge* between two *resources*. Used in the admin.

The edge is selected with either:

- the argument *edge_id*
- the arguments *subject_id*, *predicate*, *object_id*

For instance:


```
{% button
  text="Add"
  class="btn"
  action={
    link
    subject_id=id
    predicate="contains"
    object_id=other_id
    action={
      reload
    }
  }
%}
```

Other arguments:

- `element_id`
- `edge_template`
- `action` - actions executed after linking

Todo

Extend documentation

unlink

- Module: *mod_admin* (page 346)

See also:

link (page 428)

Remove an *edge* between two *resources*. Used in the admin.

The edge is selected with either:

- the argument `edge_id`
- the arguments `subject_id`, `predicate`, `object_id`

For instance:

```
{% button
  text="Remove"
  class="btn"
  action={
    unlink
    subject_id=id
    predicate="contains"
    object_id=other_id
    action={
      reload
    }
  }
%}
```

Other arguments:

- `hide` - selector to fade out after unlink
- `edge_template` - passed on to the undo action template

- action - actions executed after unlink
- undo_action - passed on to the undo action template
- undo_message_id - defaults to *unlink-undo-message*

After update, an undo message is rendered in the *undo_message_id* target, with the template *_action_unlink_undo.tpl*.

Todo

Extend documentation

Resources

delete_media

- Module: *mod_admin* (page 346)

Delete a media file from a *resource*, without confirmation.

Todo

Extend documentation

delete_rsc

- Module: *mod_admin* (page 346)

Delete a *resource*, without confirmation.

Todo

Extend documentation

dialog_delete_rsc

- Module: *mod_admin* (page 346)

Open a dialog to confirm the deletion of a *resource*.

Todo

Extend documentation

dialog_duplicate_rsc

- Module: *mod_admin* (page 346)

Open a dialog to duplicate the current *resource* with a new id and title.

Todo

Extend documentation

dialog_edit_basics

- Module: *mod_admin* (page 346)

Open a dialog to edit the “basic” information of a *resource*.

The basic information usually comprises of the title, the summary and the category, but what exactly is displayed as “basic” info is dependent on the *category* of the resource and can be changed per category by making a category specific template named `_admin_edit_basics_form.tpl` which is included using a *catinclude* (page 515).

For instance, to create a special “basics” dialog for the category *news*, you would create a template called `_admin_edit_basics_form.news.tpl`

Todo

Extend documentation

dialog_media_upload

- Module: *mod_admin* (page 346)

Shows the admin dialog for uploading a media item. See *Media* (page 32).

Todo

Extend documentation

dialog_new_rsc

- Module: *mod_admin* (page 346)

Show the admin dialog for creating a new *resource*.

When the resource is created, the user is redirected to the admin edit page. This action exports a postback called `new_page` which is used to create the page:

```
{% wire id=#form type="submit"
  postback={new_page subject_id=subject_id predicate=predicate_
↪redirect=redirect
              actions=actions callback=callback}
  delegate=`action_admin_dialog_new_rsc`
%}
```

This postback has the following arguments:

- `subject_id + predicate`: Create an edge from the given subject to this new page, using the given predicate.
- `redirect`: Boolean flag whether or not to redirect to the edit page. Defaults to `true`.
- `actions`: Any actions to perform after the resource is created.
- `callback`: JavaScript function to call when the subject edge has been created.
- `objects`: A list of `[object, predicate]` pairs which are created as outgoing edges from the new page to the given objects. The object can be a resource ID or a resource name. Example:

```
objects=[ [m.acl.user, "author"] ]
```

creates an “author” edge from the new page to the currently logged in user.

Todo

Extend documentation

Search

moreresults

- Module: *mod_wires* (page 403)

Show more results of the current search query inline on the page.

The `moreresults` action is an alternative to using a next/previous pager to paginate through search results. Instead, `moreresults` lets you load more results from the current search, directly onto the same page. This feature is similar to Twitter’s *more* button, Slashdot’s *many more* button, and others.

Using it is quite simple. The only special thing you need is that every result item should go into its own template. The minimal example is something like the following:

```
{% with m.search[{query cat="media" pagelen=10 }] as result %}
<div id="results">
  {% for id in result %}
    {% include "_item.tpl" %}
  {% endfor %}
</div>

{% button text="more..." action={moreresults result=result
                                target="results"
                                template="_item.tpl"}

%}
{% endwith %}
```

The number of items that get added is equal to the `pagelen` setting of the search.

When there are no more items, the `moreresults` button will get disabled automatically.

Arguments

result Points to search result that you want to show more of.

target Container that the new items get appended to.

template Contains template that will be appended.

catinclude Render `template` through a *catinclude* (page 515).

is_result_render Normally the template is called for every row in the search result. This is useful for lists. Sometimes all results must be rendered together, for example when special grouping is needed. Add `is_result_render` to render all results together in one template.

Example:

```
{% with m.search[{query cat="media" pagelen=16 }] as result %}
<div id="results">
  {% include "_items.tpl" %}
</div>
```

```
{% lazy action={moreresults result=result
                target="results"
                template="_items.tpl"
                is_result_render
                visible}

%}
{% endwith %}
```

Note that here we use the `lazy scomp`, which will perform the action if it is scrolled into view. Because we are using the `lazy scomp` we have to add the `visible` argument so that the re-loaded `moreresults` action will be wired for visibility and not on for a click. In this way the page loads automatically more results if the user is scrolls down.

Where `_items.tpl` displays the found pages in rows of four elements:

```
{% for ids in result/chunk:4 %}
<div class="row-fluid">
  {% for id in ids %}
    <div class="span4">
      <h3><a href="{{ id.page_url }}">{{ id.title }}</a></h3>
      <p>{{ id.summary }}</p>
    </div>
  {% endfor %}
</div>
{% empty %}
<div class="row-fluid"></div>
{% endfor %}
```

Templates

template

- Module: *mod_wires* (page 403)

Render a template. When used in a postback action, the result will be sent back with the response data for the postback.

This is useful when you want to send the output from a template back as response in a postback or submit event handler.

Example:

```
z_render:wire({template, [{template, "my_response.tpl"}, {data,
Response}]}, Context).
```

Template accepts the following arguments:

Argument	Description	Example
template	Name of template to render.	template="my_template.tpl"
•	Any other arguments will be passed on to the template being rendered.	id=123

User

auth_disconnect

- Module: *mod_authentication* (page 359)

Todo

Not yet documented.

delete_username

- Module: *mod_admin_identity* (page 353)

Delete the username from a user, no confirmation.

Todo

Extend documentation

dialog_delete_username

- Module: *mod_admin_identity* (page 353)

Open a dialog to confirm the deletion of the username of a user.

Todo

Extend documentation

dialog_set_username_password

- Module: *mod_admin_identity* (page 353)

Show a dialog for setting a username / password on the given *resource* (which is usually a person).

Todo

Extend documentation

dialog_user_add

- Module: *mod_admin_identity* (page 353)

Show a dialog for adding a user. This creates a *person resource* and adds a username / password to it.

Todo

Extend documentation

logoff

- Module: *mod_wires* (page 403)

This action logs off the current user and reloads the current page as the anonymous visitor.

Example:

```
{% button text="Log off" action={logout} %}
```

After clicking the button the page will reload and the current user will be signed out.

6.1.3 Controllers

This is the full list of *controllers* that are available in Zotonic. For more general information about controllers, see the *Controllers* (page 20) manual.

controller_admin

- Module: *mod_admin* (page 346)

The admin controller is the main controller behind which admin pages are served. Its main purpose is that it does an authentication check (Is current user allowed to use the module *mod_admin*).

The *template* parameter decides which admin template gets served, and defaults to *admin.tpl*.

Todo

Extend documentation

controller_admin_acl_rules_export

- Module: *mod_acl_user_groups* (page 341)

Todo

Not yet documented.

controller_admin_backup

- Module: *mod_backup* (page 360)

Shows the admin backup screen where you can download nightly backups that were made by *mod_backup* (page 360).

Todo

Extend documentation

controller_admin_backup_revision

- Module: *mod_backup* (page 360)

Shows the admin backup revisions screen where you can see older version for a *resource*.

Todo

Extend documentation

controller_admin_category_sorter

- Module: *mod_admin_category* (page 350)

Shows the admin category screen where you can edit the *category* tree, rearranging the categories, adding new categories, or removing existing ones.

Todo

Extend documentation

controller_admin_comments

- Module: *mod_comment* (page 361)

Shows an admin screen with an overview of most recently created comments. The screen offers the option to moderate the comments or delete them entirely.

Todo

Extend documentation

controller_admin_comments_settings

- Module: *mod_comment* (page 361)

Shows an admin settings screen where you can edit settings related to *mod_comment* (page 361).

Todo

Extend documentation

controller_admin_config

- Module: *mod_admin_config* (page 351)

Shows the admin config editor. Here you can edit the key/value pairs of *m_config* (page 560).

System Configuration

Make a new config setting				
Module	Key	Value	Modified	
site	sign_key_simple	tfUQW2dEQu	20 Nov 2012, 08:28	Delete Edit
site	sign_key	kodyyC7RJDNrDUYw31TH40XQ7bCWU3x39jww2cGNUuyxnraYp7	12 Oct 2012, 18:44	Delete Edit
zotonic	version	0.9-dev	19 Nov 2012, 18:15	Delete Edit

Todo

Extend documentation

controller_admin_edit

- Module: *mod_admin* (page 346)

The main admin edit controller. This controller serves the edit page where *resources* can be edited.

The screenshot shows the Zotonic admin interface for editing a resource. The top navigation bar includes links for Dashboard, Content, Structure, Modules, Auth, and System, along with a search bar and a language selector set to EN. The main content area is titled 'Previously Unpublished' and shows the resource's metadata and editing options.

Resource Metadata:

- Title (en):** Previously Unpublished
- Summary (en):** (Empty text area)
- Short title (en):** (Empty text field)
- Address:** (Empty text field)
- Geodata:** (Empty text field)

Editing Options:

- Publish this page:** Includes buttons for Save, Cancel, and checkboxes for Published (checked), Featured, and Protect (checked). There are also buttons for Delete and Duplicate.
- Access control:** (Expandable section)
- Publication period:** (Expandable section)
- Date range:** (Expandable section)
- Translations:** Shows the current language as English (checked) and lists other available languages: Eesti, Español, Français, Deutsch, Nederlands, Türkçe, and Polski.
- Page connections:** (Expandable section)

Rich Text Editor:

- Includes a toolbar with various formatting options (bold, italic, list, link, etc.).
- The main text area is currently empty.
- The **Path** is set to `p`.

Buttons:

- + add block:** A button at the bottom right of the main content area.

Todo

Extend documentation

controller_admin_mailing_preview

- Module: *mod_mailinglist* (page 379)

This controller shows a preview of what a resource that is being mailed would look like, in a popup window.

Todo

Extend documentation

controller_admin_mailing_status

- Module: *mod_mailinglist* (page 379)

This controller shows the mailing status of a *resource*. It lists each mailing list available in the system, and shows whether or not the current resource has already been sent to the list.

Per mailinglist, it offers the options to send the resource right now, or schedule it for later delivery.

There are also buttons for sending the resource to a test mailing list or to a single email address.

Todo

Extend documentation

controller_admin_mailinglist

- Module: *mod_mailinglist* (page 379)

This controller shows the mailing lists that are available in the system.

For each list, it shows the number of recipients and the title. Clicking a list shows the *recipients* (page 438) of the mailing list.

Todo

Extend documentation

controller_admin_mailinglist_recipients

- Module: *mod_mailinglist* (page 379)

Shows the recipients of the current mailing list. The recipients are listed in three columns, and have a checkbox next to them to deactivate them.

Clicking a recipient shows a popup with information about the recipient, where you can edit the e-mail address and the recipient's name details.

The page also offers buttons for importing and exporting lists of email addresses.

Todo

Extend documentation

controller_admin_media_preview

- Module: *mod_admin* (page 346)

A controller for rendering preview thumbnails of any media embedded in a richtext-editor component of a *resource* on the *admin edit controller* (page 437) page.

Todo

Extend documentation

controller_admin_module_manager

- Module: *mod_admin_modules* (page 355)

Shows the list of Zotonic modules currently known to the system.

The list is sorted based on the module's status: active modules are listed first, non-active modules next.

Each module has a button which let you toggle the active status of the module.

Todo

Extend documentation

controller_admin_referrers

- Module: *mod_admin* (page 346)

Shows the list of pages (*resources*) which refer to this *resource* through an *edge*.

Todo

Extend documentation

controller_admin_seo

- Module: *mod_seo* (page 389)

Shows a form with settings related to Search Engine Optimization.

Todo

Extend documentation

controller_admin_statistics

- Module: *mod_admin_statistics* (page 357)

Todo

Not yet documented.

controller_api

- Module: *mod_base* (page 360)

See also:

Dispatch rules (page 22) and *Controllers* (page 20).

`controller_api` processes authorized REST API requests: It provides an easy way to create API calls to allow computer programs to perform functions on your Zotonic site.

`controller_api` by default intercepts all URLs according to the patterns `/api/:topic`.

The topic can refer to a model. In this case the topic pattern is one of:

- `/api/mymodel/get/foo/bar` maps to `m_mymodel:m_get ([<<"foo">>, <<"bar">>], Msg, Context)`

- `/api/mymodel/post/foo/bar` maps to `m_mymodel:m_delete([<<"foo">>, <<"bar">>], Msg, Context)`
- `/api/mymodel/delete/foo/bar` maps to `m_mymodel:m_delete([<<"foo">>, <<"bar">>], Msg, Context)`

In all case the `Msg` is an MQTT message map, with the `payload` set to the body of the received request. In case of a GET or DELETE the payload is set to a map of the query arguments.

Note that for a POST the payload might not be the complete message as additional query arguments are passed via `z_context:get_q(<<"argument_name">>, Context)`. If the model function is called using MQTT, then all arguments are contained in the payload.

The API controller will publish a message to the topic, and wait for max 60 seconds for a response on the response topic.

If there is a query argument `response_topic` then the API controller will only publish the message and immediately return.

controller_authentication

- Module: *mod_authentication* (page 359)

This controller manages all authentication cookies. It is called by the `auth` model in the browser.

Todo

Not yet documented.

controller_export

- Module: *mod_export* (page 370)

Todo

Not yet documented.

controller_export_resource

- Module: *mod_export* (page 370)

Todo

Not yet documented.

controller_file

- Module: *mod_base* (page 360)

See also:

controller_file_id (page 442), *lib* (page 528), *image* (page 520), *image_url* (page 526)

Serve an uploaded-, resized- or library file.

This controller is used to serve files and images. It is able to manipulate an image according to the parameters supplied.

Image manipulation parameters are signed to prevent random image manipulations on the request of visitors, which might result in a denial of service due to processing- or disk space limitations.

This controller serves all files with a very long client side caching time and handles if-modified-since checks. Text files are served with gzip compression if the user-agent supports it.

Multiple files can be served in a single request; the controller concatenates them into a single file. See the [lib](#) (page 528) tag for more information. The creators of the files have to ensure that they can be properly concatenated.

Dispatch rules and options

Example dispatch rules:

```
{image, ["image", '*'], controller_file, []},
{lib, ["lib", '*'], controller_file, [{root, [lib]}]}
```

controller_file has the following dispatch options:

Option	Description	Example
root	List of root directories where files are located. Use 'lib' for the library files. This defaults to the site's "files/archive" directory.	{root, [lib]}
path	Default file to be served. Used for files like "robots.txt" and "favicon.ico".	{path,"misc/robots.txt"}
content_disposition	If the file should be viewed in the browser or downloaded. Possible values are inline and attachment. Defaults to the browser's defaults by not setting the "Content-Disposition" response header.	{content_disposition, inline}
acl	Extra authorization checks to be performed.	See ACL options (page 441).
max_age	Max age, used for Cache and Expires. Value is an integer, number of secs.	{max_age,3600}

ACL options

[Authorization](#) (page 59) checks to perform, in addition to the `acl_action` dispatch option, can be given in the `acl` dispatch option, and accepts the following options:

ACL option	Description	Example
is_auth	Disable anonymous access to this resource.	{acl, is_auth}
logoff	Log out user before processing the request.	{acl, logoff}
{Action, Resource}	Check if user is allowed to perform Action on Resource. The example is equivalent to the options {acl_action, edit}, {id, my_named_page}.	{acl, {edit, my_named_page}}
[{Action, Resource}]	A list of checks to be performed, as above.	{acl, [{view, secret_page}, {update, 345}]}
ignore	Don't perform any access control checks. Be careful to add your own checks in the rendered template and all its included templates.	{acl, ignore}

More about the search root

The search root can be a list with one or more of the following:

- The atom *lib* for finding library files in the *lib* directory of modules.
- The atom *template* for finding files in the *template* directory of modules.
- A directory name (binary or string). This directory name must be absolute or relative to the *files* directory of the site.
- A tuple *{module, ModuleName}* to refer to a module. The module must implement the functions *file_exists/2* and *file_forbidden/2*.
- A tuple *{id, RscId}* where *RscId* is the unique name or id of a resource. The resource must have a medium record containing an uploaded file.

Note that located files are aggressively cached. Changes to the lookup routines or files will take a while before they are visible for new requests.

CSS and JavaScript templates

Note: `controller_file` replaces `controller_file_readonly` and `controller_lib`

If a file with a *lib* or *template* root is not found, then the same filename with the addition of *.tpl* is checked. For example *styles.css.tpl*. If found then the template will be rendered against an empty site context. This means that, with the current implementation, the template will not receive the current language, user etc. This behavior may change in the future.

New in version 0.11.

controller_file_id

- Module: *mod_base* (page 360)

Redirect to the controller `controller_file`.

This controller maps a resource id to the filename of the medium associated with the resource.

For the redirect it uses the dispatch rule defined in the dispatch options.

Examples from `mod_base`:

```
{media_inline, [{"media", "inline", "id", id}, controller_file_id, [ {dispatch, media_
↳ inline}, {ssl, any} ]}],
{media_inline, [{"media", "inline", '*'}, controller_file, [ {content_disposition, _
↳ inline}, {ssl, any} ]}],
```

The first dispatch rule will redirect to the second. If no associated file was found, then a 404 is returned.

See also:

controller_file (page 440)

controller_fileuploader

- Module: *mod_fileuploader* (page 375)

The `fileuploader` controller is used to upload parts of a file.

For uploading data it accepts a POST with two arguments:

- `name` is provided by the `model/fileuploader/post/new` method and uniquely identifies the file being uploaded.
- `offset` provides the start point of the uploaded segment in the file. The size of the segment is derived from the uploaded data.

The body should be of `application/octet-stream` type with binary data.

The result is a JSON, describing the current status of the uploaded file.

Use a GET for fetching the status of an upload, only the `name` argument is needed.

```
{
  "result": {
    "filename": "test.jpg",
    "is_complete": false,
    "missing": [
      {
        "size": 10,
        "start": 0
      }
    ],
    "name": "WZkhXoaMwrK2StUHmdpp",
    "received": 0,
    "size": 10,
    "upload_url": "https://zotonic.test:8443/fileuploader/upload/
↪WZkhXoaMwrK2StUHmdpp"
  },
  "status": "ok"
}
```

Status `"error"` is returned if the `name` is unknown or any error occurred.

controller_hello_world

- Module: *mod_development* (page 364)

See also:

controller_ping (page 449)

Simple controller that always serves the string `Hello, World!`

This controller can be used for a dispatch rule to check if the server is responding.

controller_http_error

- Module: *mod_base* (page 360)

This controller is called for serving http error pages.

The controller will try to match the content type of the response with the expected content type for the request.

For html responses it will render a `page_error.code.tpl` where *code* is the error code for the response, for example 404 or 500.

In the template the following variables are available:

error_code The http response error code, an integer like 500.

error_erlang (Optional) In case of an Erlang error, the textual version of the error. Could be a short word or a longer descriptive error message.

error_table (Optional) If an error stack was available then this is a list of table rows:

```
[ IsTemplate, Module, Fun, Args, {File, Line} ]
```

Where `Module` is the template name if `IsTemplate` is `true`.

error_dump (Optional) Some raw internal error information. This is given if there is an error but the error could not be translated into a erlang error and table.

For JSON returns a simple error is returned, for example:

```
{ "code": 404, "status": "Not Found" }
```

For image results a transparent 1 pixel gif is served.

For Javascript and css a text file with a comment is served.

For plain text a simple error like `Not Found` is served.

controller_id

- Module: *mod_base* (page 360)

Handle different content representations of a page.

Redirects to different representations of a page, depending on the requested content type. The redirect is done using a “303 See Other” status. A “404 Not Found” or “410 Gone” is returned if the requested page never existed or has been deleted.

When no content types are requested then `text/html` is selected.

This controller is also used for a page’s short url representation.

Example dispatch rule (from *mod_base*):

```
{id, ["id", id], controller_id, []}
```

This controller does not have any dispatch options.

This controller handles the following query argument:

Option	Description	Example URL
<code>id</code>	Id of the requested <i>resource</i> .	<code>/id/1234</code>

The list of provided content types is collected with a *foldr* notification (see *Notifications* (page 69)) of the type `content_types_dispatch`. Modules should add their provided content types in front of the accumulator. The added entries are tuples: `{MimeType, DispatchRuleName}`.

Example of adding a content type handler adding a text/plain handler with the dispatch rule `rsc_text`:

```
observe_content_types_dispatch(#content_types_dispatch{}, Acc, _Context) ->
  [{"text/plain", rsc_text} | Acc].
```

controller_keyserver_key

- Module: *mod_base* (page 360)

Todo

Not yet documented.

controller_language_set

- Module: *mod_translation* (page 400)

Controller which sets the language as given in the `code` argument, and redirects the user back to the page given in the `p` argument.

Todo

Extend documentation

controller_letsencrypt_challenge

- Module: *mod_ssl_letsencrypt* (page 397)

Todo

Not yet documented.

controller_letsencrypt_ping

- Module: *mod_ssl_letsencrypt* (page 397)

Todo

Not yet documented.

controller_log_client

- Module: *mod_logging* (page 378)

Todo

Not yet documented.

controller_logoff

- Module: *mod_authentication* (page 359)

Controller that logs off a user, destroying the session. It also removes any “remember me” cookies the user has, so that auto-logon is disabled.

Todo

Extend documentation

See also:

controller_authentication (page 440), *Authentication* (page 58).

controller_logon_done

- Module: *mod_authentication* (page 359)

See also:

logon_ready_page (page 607), *controller_authentication* (page 440)

This controller is used as a jumping stone after a log on from the `/logon` page. The `p` argument is passed from the `/logon` page.

The controller will notify observers of `#logon_ready_page{ request_page = P }` to see where to redirect next.

The notification is a *first* (page 70), so the first module responding with something else than `undefined` will determine the redirect.

If no redirection is returned, and the `p` argument is empty, then the user is redirected to the home page `/`.

controller_mailinglist_export

- Module: *mod_mailinglist* (page 379)

Controller which downloads the given mailinglist id as a CSV file.

Todo

Extend documentation

controller_mqtt_transport

- Module: *mod_base* (page 360)

Controller for transport of MQTT data between the browser (client) and server.

This controller accepts Websocket connections and out-of-band HTTP POSTs.

The HTTP POSTs must have a valid ticket. See *m_mqtt_ticket* (page 569) on how to obtain such a ticket and more information about tickets.

The Websocket connection is a normal MQTT transport over Websocket.

For the authentication the Websocket accepts two methods:

- Cookie authentication with `z.auth` cookie, see *controller_authentication* (page 440).
- Username and password in the MQTT connect

controller_nocontent

- Module: *mod_base* (page 360)

Todo

Not yet documented.

controller_oauth2_access_token

- Module: *mod_oauth2* (page 386)

Todo

Not yet documented.

controller_oauth2_service_authorize

- Module: *mod_oauth2* (page 386)

Todo

Not yet documented.

controller_oauth2_service_redirect

- Module: *mod_oauth2* (page 386)

Todo

Not yet documented.

controller_page

- Module: *mod_base* (page 360)

Show a rsc as a HTML page.

This controller is used to show the HTML page of a *resource*. A “404 Not Found” or “410 Gone” page is shown if the requested page never existed or has been deleted.

The user will be redirected to the `logon` URL when the current user is not allowed to view the page.

This controller also adds a `noindex` response header when the page’s *seo_noindex* (page 389) flag is set.

Example dispatch rule:

```
{page, ["page", id], controller_page, []}
```

Dispatch arguments

`controller_page` recognizes the following arguments inside the dispatch pattern:

Argument	Description	Example URL
id	The id of the page (rsc) to be shown. This can be the numerical id or the unique name of a page.	/page/12345

Dispatch options

The following options can be given to the dispatch rule:

Option	Description	Example
id	Id or unique name of the resource to be shown. This overrules any id in the query arguments. Use <code>user_id</code> for the id of the current user.	{id, page_about}
template	Name of the template to be rendered. Defaults to “page.tpl” Can also be a tuple of the following form: {cat, Name}. See also: cat-include (page 515).	{template, “about.tpl”} {template, {cat, “home. tpl”}}
cat	The category the resource that is requested has to be. If a page of a different category is requested, a 404 is shown.	{cat, text}
acl_action	What ACL action will be checked. Defaults to ‘view’; but can also be ‘edit’ if users need edit permission on the rsc to be able to access the resource.	{acl_action, edit}
acl	Extra authorization checks to be performed.	See ACL options (page 448).
is_canonical	Whether this URL should be considered the canonical URL of the page. If so, the controller will redirect to the sc’s page path if set. Defaults to true.	{is_canonical, false}
seo_noindex	Ask crawlers to not index this page.	seo_noindex
nocache	Prevent browser caching this page.	nocache

ACL options

[Authorization](#) (page 59) checks to perform, in addition to the `acl_action` dispatch option, can be given in the `acl` dispatch option, and accepts the following options:

ACL option	Description	Example
is_auth	Disable anonymous access to this resource.	<code>{acl, is_auth}</code>
logoff	Log out user before processing the request.	<code>{acl, logoff}</code>
<code>{Action, Resource}</code>	Check if user is allowed to perform Action on Resource. The example is equivalent to the options <code>{acl_action, edit}</code> , <code>{id, my_named_page}</code> .	<code>{acl, {edit, my_named_page}}</code>
<code>[{Action, Resource}]</code>	A list of checks to be performed, as above.	<code>{acl, [{view, secret_page}, {update, 345}]}</code>
ignore	Don't perform any access control checks. Be careful to add your own checks in the rendered template and all its included templates.	<code>{acl, ignore}</code>

See also:

[controller_template](#) (page 452).

controller_ping

- Module: [mod_base](#) (page 360)

Simple controller for connection tests, used on the `/test/connection` page.

It always responds with the four character string pong. The usual path is: `/.zotonic/ping`

controller_redirect

- Module: [mod_base](#) (page 360)

Redirect to another url.

This controller redirects a request to another URL. The URL can be a fixed URL, the location of a fixed page id or the name of a dispatch rule.

Example dispatch rule using the redirect controller:

```
{redir, ["plop"], controller_redirect, [{url, "/newplop"}, {is_permanent, true}]}
```

This redirects any requests of `"/plop"` permanently to `"/newplop"`.

It has the following dispatch options:

Option	Description	Example
url	The url of the new location the browser is sent to.	{url, “/example”}
dispatch	Name of a dispatch rule to use for the location url. All arguments (except dispatch and is_permanent) are used as parameters for the dispatch rule.	{dispatch, admin}
id	Id of the page to redirect to. The controller will redirect to the page_url of this id. The id can be an integer or the name of the page (use an atom or a binary) or the atom <i>user_id</i> for the id of the current user.	{id, 123}
qargs	A list with querystring arguments to use in the new dispatch rule. Specifies what query (or dispatch) arguments to use from this dispatch rule into the dispatch rule that is being redirected to.	{qargs, [id, slug]}
is_permanent	Use a permanent (301) or temporary redirect (307). Defaults to false.	{is_permanent, false}
acl	Perform access control check before redirect. Defaults to no check.	{acl, is_auth}

This controller does only handle request arguments that are specifically noted in the “qargs” list (and then only when the “dispatch” argument is set).

ACL options

[Authorization](#) (page 59) checks to perform, in addition to the `acl_action` dispatch option, can be given in the `acl` dispatch option, and accepts the following options:

ACL option	Description	Example
is_auth	Disable anonymous access to this resource.	{acl, is_auth}
logoff	Log out user before processing the request.	{acl, logoff}
{Action, Resource}	Check if user is allowed to perform Action on Resource. The example is equivalent to the options {acl_action, edit}, {id, my_named_page}.	{acl, {edit, my_named_page}}
[{Action, Resource}]	A list of checks to be performed, as above.	{acl, [{view, secret_page}, {update, 345}]}
ignore	Don't perform any access control checks. Be careful to add your own checks in the rendered template and all its included templates.	{acl, ignore}

Example

A dispatch rule that always redirects /foo/12312/slug to /bar/12312/slug:

```
{bar, ["bar", id, slug], controller_page, [{template, "bar.tpl"}]},
{bar_redirect, ["foo", id, slug], controller_redirect, [{dispatch, bar}, {qargs,
↪ [id, slug]}]}
```

controller_signup

- Module: *mod_signup* (page 393)

Controller which displays a form to sign up (rendered from `signup.tpl`).

It also implements the necessary postbacks to perform the signup and log a user in.

Todo

Extend documentation

controller_signup_confirm

- Module: *mod_signup* (page 393)

Controller which displays the confirmation page where the user can confirm his signup.

The template used is `signup_confirm.tpl`.

Todo

Extend documentation

controller_static_pages

- Module: *mod_base* (page 360)

Serve a static page or pages.

With this controller it is possible to add a folder with static files as a sub-site to your Zotonic site. Add the folder and all files to a directory in your template directory or your site's directory and define the directory in a dispatch rule.

Example dispatch rule:

```
{oldsite, ["old", '*'], controller_static_pages, [{root, "priv/old_site"}]}
```

When a file `a.txt` is requested this resource will check for `a.txt` and `a.txt.tpl`. When it finds a `.tpl` file then that file be handled as a template. All dispatch configuration variables are available in the template.

Directories will be redirected to the directory name with a `/` appended. The resource serves the file `index.html` or `index.html.tpl` for the directory contents. If these are not found, it will give a 404 page, unless the `allow_directory_index` option is set; in which case a directory listing is displayed.

It has the following dispatch options:

Option	Description	Example
root	Name of the directory in the site directory containing the static files. The root is a path name relative to the current site's base directory or <code>{files, "some/path"}</code> for a path relative to a site's files directory.	<code>{root, "priv/oldsite"}</code>
use_cache	Whether or not served files are cached in memory for an hour. Defaults to false. Use this for high-volume traffic when the files themselves do not change often.	<code>{use_cache, true}</code>
allow_directory_index	Whether or not to serve a directory listing when no index file is found. Defaults to false. The directory index is rendered using <code>template-directory_index</code> . New in version 0.9.	<code>{allow_directory_index, true}</code>

This resource does not handle any request arguments.

controller_template

- Module: *mod_base* (page 360)

Show a template.

This controller renders the template configured in the dispatch rules.

Example dispatch rule:

```
{home, [], controller_template, [{template, "home.tpl"}]}
```

This will render the `home.tpl` template at the url `/`.

Dispatch arguments

`controller_template` recognizes the following arguments inside the dispatch pattern:

Argument	Description	Example URL
id	A resource id to be used in the template. This can be the numerical id or the unique name of a page. More commonly the id is given as a dispatch option.	<code>/page/12345</code>

Dispatch options

The following options can be given to the dispatch rule:

Option	Description	Example
template	Name of the template to be rendered. Can also be a tuple of the following form: <i>{cat, Name}</i> . See also: <i>catinclude</i> (page 515).	<code>{template, "home.tpl"}</code> <code>{template, {cat, "home.tpl"}}</code>
anonymous	Render the template always as the anonymous user, even when an user is logged on. Defaults to false.	<code>{anonymous, true}</code>
content_type	The content type provided by the dispatch rule. Defaults to "text/html".	<code>{content_type, "application/json"}</code>
max_age	The number of seconds of how long to cache this file in the browser. Sets the response header: <i>Cache-control: public; max-age=X</i> .	<code>{max_age, 3600}</code>
acl_action	What ACL action will be checked. Defaults to 'view'; but can also be 'edit' if users need edit permission on the rsc to be able to access the resource.	<code>{acl_action, edit}</code>
acl	Extra authorization checks to be performed.	See <i>ACL options</i> (page 453).
id	Id or unique name of a resource to be referenced in the rendered template. This overrules and id from the query arguments.	<code>{id, page_about}</code>
seo_noindex	Ask crawlers to not index this page.	<code>seo_noindex</code>
no-cache	Prevent browser caching this page.	<code>nocache</code>
http_status	The HTTP status code to return. This defaults to 200.	<code>{http_status, 418}</code>

ACL options

Authorization (page 59) checks to perform, in addition to the `acl_action` dispatch option, can be given in the `acl` dispatch option, and accepts the following options:

ACL option	Description	Example
<code>is_auth</code>	Disable anonymous access to this resource.	<code>{acl, is_auth}</code>
<code>logoff</code>	Log out user before processing the request.	<code>{acl, logoff}</code>
<code>{Action, Resource}</code>	Check if user is allowed to perform Action on Resource. The example is equivalent to the options <code>{acl_action, edit}</code> , <code>{id, my_named_page}</code> .	<code>{acl, {edit, my_named_page}}</code>
<code>[{Action, Resource}]</code>	A list of checks to be performed, as above.	<code>{acl, [{view, secret_page}, {update, 345}]}</code>
<code>ignore</code>	Don't perform any access control checks. Be careful to add your own checks in the rendered template and all its included templates.	<code>{acl, ignore}</code>

See also:

controller_page (page 447).

controller_website_redirect

- Module: *mod_base* (page 360)

This controller does a redirect to the `website` property of the given *resource*.

Todo

Extend documentation

6.1.4 Filters

Filters (page 29) transform template variables before they are rendered.

Binaries

first

- Module: *mod_base* (page 360)

See also:

tail (page 481), *last* (page 454)

Returns the first character or element.

Returns the first byte of a binary or the first element of a list. An empty binary is returned when the input is empty.

For example:

```
{{ value|first }}
```

If the value is `hello` then the output is `h`.

Note: This function is safe to use with multibyte character values, if the input is a binary.

For a regular list:

```
{{ [1,2,3]|first }}
```

The filtered value is `1`.

It is also possible to fetch the first *N* elements or characters:

```
{{ [1,2,3]|first:2 }}
```

The filtered value is `[1, 2]`.

Or, with a string:

```
{{ "hello"|first:2 }}
```

The filtered value is `"he"`.

last

- Module: *mod_base* (page 360)

See also:

first (page 454)

Returns the last character or element.

Returns the last element of the value. When the value is a list then the last element of the list is returned, when the value is a binary then the last *byte* of the binary is returned.

For example:

```
{{ value|last }}
```

When value is the list `hello` then the output will be `o`.

Note: This function is not safe to use with multibyte character values, use with care.

length

- Module: *mod_base* (page 360)

Returns the length of the value.

The length of a list is the number of elements in the list, the length of a binary is the number of bytes in the binary.

For example:

```
{{ value|length }}
```

When value is the list `"hello"` then the output will be `"5"`.

Note: With multi-byte values this function does not return the number of characters, it returns the number of bytes. This may change in a future release.

to_binary

- Module: *mod_base* (page 360)

See also:

stringify (page 500)

Convert the input to a binary value.

Example:

```
{{ 42|to_binary }}
```

Results in the binary value `<<"42">>`.

This filter uses the `z_convert:to_binary/1` function.

Booleans

yesno

- Module: *mod_base* (page 360)

Show a boolean value as a text.

Given a string mapping values for `true`, `false` and (optionally) `undefined`, returns one of those strings according to the value.

Non-empty values are converted to their boolean value first using `z_convert:to_boolean/1`.

Example:

```
{{ 1|yesno:"ja,nee" }}
```

Will output `"ja"`; this:

```
{{ 0|yesno:"ja,nee" }}
```

Will output “nee”.

yesno accepts these values:

Value	Argument	Output
true	“yeah,no,maybe”	“yeah”
false	“yeah,no,maybe”	“no”
undefined	“yeah,no,maybe”	“maybe”
undefined	“yeah,no”	“no”

Dates

add_hour

- Module: *mod_base* (page 360)

See also:

sub_hour (page 461), *add_day* (page 456), *add_week* (page 457), *add_month* (page 456), *add_year* (page 457)

Adds an hour to a date. The value must be of the form `{{Y,M,D},{H,I,S}}`.

For example:

```
{{ value|add_hour }}
```

When the value is `{{2008,12,10},{15,30,0}}`, the output is `{{2008,12,10},{16,30,0}}`.

The filter has an optional argument which defines the number of hours to add:

```
{{ value|add_hour:3 }}
```

When the value is `{{2008,12,10},{15,30,0}}`, the output is `{{2008,12,10},{18,30,0}}`.

add_day

- Module: *mod_base* (page 360)

See also:

sub_day (page 462), *add_week* (page 457), *add_month* (page 456), *add_year* (page 457)

Adds a day to a date. The value must be of the form `{{Y,M,D},{H,I,S}}`.

For example:

```
{{ value|add_day }}
```

When the value is `{{2008,12,10},{15,30,0}}`, the output is `{{2008,12,11},{15,30,0}}`.

The filter has an optional argument which defines the number of days to add:

```
{{ value|add_day:3 }}
```

When the value is `{{2008,12,10},{15,30,0}}`, the output is `{{2008,12,13},{15,30,0}}`.

add_month

- Module: *mod_base* (page 360)

Adds a month to a date. The value must be of the form `{{Y,M,D},{H,I,S}}`.

For example:

```
{{ value|add_month }}
```

When the value is `{{2008,12,10},{15,30,0}}`, the output is `{{2009,1,10},{15,30,0}}`.

The filter has an optional argument which defines the number of months to add:

```
{{ value|add_month:3 }}
```

When the value is `{{2008,12,10},{15,30,0}}`, the output is `{{2009,3,10},{15,30,0}}`.

add_week

- Module: [mod_base](#) (page 360)

Adds a week to a date. The value must be of the form `{{Y,M,D},{H,I,S}}`.

For example:

```
{{ value|add_week }}
```

When the value is `{{2008,12,10},{15,30,0}}`, the output is `{{2008,12,17},{15,30,0}}`.

The filter has an optional argument which defines the number of weeks to add:

```
{{ value|add_week:4 }}
```

When the value is `{{2008,12,10},{15,30,0}}`, the output is `{{2009,1,7},{15,30,0}}`.

add_year

- Module: [mod_base](#) (page 360)

Adds a year to a date. The value must be of the form `{{Y,M,D},{H,I,S}}`.

For example:

```
{{ value|add_year }}
```

When the value is `{{2008,12,10},{15,30,0}}`, the output is `{{2009,12,10},{15,30,0}}`.

The filter has an optional argument which defines the number of years to add:

```
{{ value|add_year:3 }}
```

When the value is `{{2008,12,10},{15,30,0}}`, the output is `{{2011,12,10},{15,30,0}}`.

date

- Module: [mod_base](#) (page 360)

See also:

[date_range](#) (page 459), [datediff](#) (page 460), [timesince](#) (page 463), [now](#) (page 529)

Formats a date or datetime according to the format specified in the argument.

The date should be a tuple `{Y,M,D}` and the datetime should be a tuple `{{Y,M,D},{H,I,S}}`. Dates and datetimes are always assumed to be in *local time*.

An example:

```
{{ mydate|date:"Y-m-d" }}
```

If mydate is {2009, 6, 1} this returns 2009-06-01 as output.

To show the year of the current date:

```
{{ now|date:"Y" }}
```

See also the [timesince](#) (page 463) filter to display a human readable *relative* time like *10 hours ago*.

Timezones

Dates in Zotonic are stored in UTC. If a date is displayed then it is converted to the timezone of the current request context. This timezone can be one of the following, in order of preference:

- Preferred timezone set by the user
- Timezone of the user-agent
- Default timezone of the site
- Default timezone of the Zotonic server
- UTC

A specific timezone can be enforced by adding a second parameter to the date-filter. For example, to display a date in UTC:

```
{{ mydate|date:"Y-m-d H:i T":"UTC" }}
```

Timezone and *all day* date ranges

If a resource's date range is set with the *date_is_all_day* flag then the dates are not converted to or from UTC but stored as-is. This needs to be taken into account when displaying those dates, otherwise a conversion from (assumed) UTC to the current timezone is performed and the wrong date might be displayed.

The timezone conversion can be prevented by adding the *date_is_all_day* flag to the date-filter as the timezone. Example, for displaying the start date of a resource:

```
{{ id.date_start|date:"Y-m-d":id.date_is_all_day }}
```

Date formatting characters

Date uses the same format as PHP's date function with some extensions and some omissions.

All supported formatting characters are listed below:

Character	Description
a	"a.m." or "p.m." (note that this follows Associated Press style and adds periods).
A	Uppercase "AM" or "PM".
b	Month, textual, in three lowercase characters.
c	ISO-8601 date format.
d	Day of the month in two digits with leading zero, i.e. "01" to "31".
D	Day of the week, textual, three letters of which the first one uppercase.
e	Show era when date is BCE (Before Common Era, so before the year 1).
E	Always show era.
f	If minutes is zero then show only the hour, otherwise the hour and the minutes. Hours are shown using the "g" form

Table 6.2 – continued from

Character	Description
F	Month, textual, full english name with first character in uppercase.
g	12 Hour format without leading zero, i.e. “1” to “12”.
G	24 Hour format without leading zero, i.e. “0” to “23”.
h	12 Hour format with leading zero, i.e. “01” to “12”.
H	24 Hour format with leading zero, i.e. “00” to “23”.
i	Minutes with leading zero, i.e. “00” to “59”.
i	Daylight saving time flag. “1” if DST is in effect, “0” if no DST.
j	Day of the month without leading zero, i.e. “1” to “31”.
l	(lowercase L) Day of the week, textual, full english name with first character in uppercase.
L	Boolean for whether the year is a leap year. Returns the string “True” or “False”.
m	Month with leading zero, i.e. “01” to “12”.
M	Month, textual, in three characters, first character in uppercase.
n	Month without leading zero, i.e. “1” to “12”.
N	Month abbreviation in Associated Press style. March, April, June and July are shown in full. September as “Sept.”
O	Difference to Greenwich Mean Time (GMT).
P	Time in 12 hour format with minutes and “a.m.” or “p.m.” appended. Minutes are left off if they are zero, and the st
r	RFC 2822 formatted date.
s	Seconds with leading zero.
S	English ordinal suffix for the day of the month, 2 characters; i.e. “st”, “nd”, “rd” or “th”.
t	Number of days in the given month, i.e. “28” to “31”.
T	Timezone used for displaying the date.
U	Seconds since the Unix epoch of January 1, 00:00:00 GMT.
w	Day of the week, numeric. 0 For sunday to 6 for saturday.
W	ISO-8601 week number of the year, starting on mondays.
x	Year in (at least) four digits, in accordance with ISO 8601.
y	Year in two digits.
Y	Full year. BCE years are shown as a positive number. Use e or E to add the era.
z	Day of the year, i.e. 1 to 366.

To construct a date in a template, the filter also accepts Erlang lists as input, so the following will work:

```
{{ [1990,10,10]|date:"j F Y" }}
```

Will output *10 October 1990*. This also works with datetimes:

```
{{ [[1990,10,10],[10,11,12]]|date:"j F Y - H:i:s" }}
```

Will output *10 October 1990 - 10:11:12*.

date_range

- Module: *mod_base* (page 360)

See also:

date (page 457)

Show a date range.

Filter to simplify displaying datetime ranges. When displaying a datetime range, the display of the dates and times often depends if the date parts of the datetimes are equal or not.

Take the following code:

```
{{ [fromdate, todate]|date_range:[format_ne, sep, format_eq] }}
```

If the `dates` of `fromdate` and `today` are equal then the output will be as if the following were written:

```
{{ fromdate|date:format_ne }}{{ sep }}{{ todate|date:format_eq }}
```

However, if the dates are not equal then the output will be as if the following were written:

```
{{ fromdate|date:format_ne }}{{ sep }}{{ todate|date:format_ne }}
```

datediff

- Module: *mod_base* (page 360)

See also:

date (page 457)

Calculate the difference between two dates, returning a single part of that difference.

The filter takes a list with 2 parts *[start, end]* as date range argument.

The filter argument the “part” that will be extracted, and is one of *Y, M, D, H, I, S*.

Example, where *start* = 2012-02-02, *end* = 2012-03-01:

```
{{ [end, start]|datediff:"M" }}
```

Returns *1*, since the difference in months between those 2 dates is 1.

eq_day

- Module: *mod_base* (page 360)

See also:

ne_day (page 461)

Tests if the value is a date and equal to the argument. The value and the argument must be a tuple of the format *{Y, M, D}* or *{{Y, M, D}, {H, I, S}}*.

For example:

```
{% if value|eq_day:othervalue %}same day{% endif %}
```

This outputs “same day” if *value* and *othervalue* are dates and on the same day.

This is useful for conditions, in combination with for example the *if* tag.

in_future

- Module: *mod_base* (page 360)

See also:

in_past (page 461)

Tests if a date is in the future.

Tests if the value is a date and in the future. The value must be a tuple of the format *{Y, M, D}* or *{{Y, M, D}, {H, I, S}}*. When the value is not a date, or *datetime*, the result will be undefined.

For example:

```
{% if value|in_future %}That day has yet to come.{% endif %}
```

This outputs “That day has yet to come.” if the value is a date and in the future.

in_past

- Module: *mod_base* (page 360)

See also:

in_future (page 460)

Tests if a date is in the past.

Tests if the value is a date and in the past. The value must be a tuple of the format {Y, M, D} or {{Y, M, D}, {H, I, S}}. When the value is not a date or datetime, the result is undefined.

For example:

```
{% if value|in_past %}Those days have gone.{% endif %}
```

This outputs “Those days have gone.” if the value is a date and in the past.

ne_day

- Module: *mod_base* (page 360)

See also:

eq_day (page 460)

Tests if two dates are not equal.

Tests if the value is a date and not equal to the argument. The value and the argument must be a tuple of the format {Y, M, D} or {{Y, M, D}, {H, I, S}}.

For example:

```
{% if value|ne_day:othervalue %}different days{% endif %}
```

This outputs “different days” if value and othervalue are dates and different.

This is useful in combination with for example the if tag.

sub_hour

- Module: *mod_base* (page 360)

See also:

add_hour (page 456), *sub_day* (page 462), *sub_week* (page 462), *sub_month* (page 462), *sub_year* (page 463)

Subtracts an hour from a date. The value must be of the form {{Y, M, D}, {H, I, S}}.

For example:

```
{{ value|sub_hour }}
```

When the value is {{2008, 12, 10}, {15, 30, 0}} then the output is {{2008, 12, 10}, {14, 30, 0}}.

The filter has an optional argument which defines the number of hours to subtract:

For example:

```
{{ value|sub_hour:3 }}
```

When the value is {{2008, 12, 10}, {15, 30, 0}} then the output is {{2008, 12, 10}, {12, 30, 0}}.

sub_day

- Module: *mod_base* (page 360)

See also:

add_day (page 456), *sub_week* (page 462), *sub_month* (page 462), *sub_year* (page 463)

Subtracts a day from a date. The value must be of the form `{{Y,M,D},{H,I,S}}`.

For example:

```
{{ value|sub_day }}
```

When the value is `{{2008,12,10},{15,30,0}}` then the output is `{{2008,12,9},{15,30,0}}`.

The filter has an optional argument which defines the number of days to subtract:

For example:

```
{{ value|sub_day:3 }}
```

When the value is `{{2008,12,10},{15,30,0}}` then the output is `{{2008,12,7},{15,30,0}}`.

sub_month

- Module: *mod_base* (page 360)

See also:

sub_day (page 462), *sub_week* (page 462), *add_month* (page 456), *sub_year* (page 463)

Subtracts a month from a date. The value must be of the form `{{Y,M,D},{H,I,S}}`.

For example:

```
{{ value|sub_month }}
```

When the value is `{{2008,12,10},{15,30,0}}` then the output is `{{2008,11,10},{15,30,0}}`.

The filter has an optional argument which defines the number of months to subtract:

For example:

```
{{ value|sub_month:3 }}
```

When the value is `{{2008,12,10},{15,30,0}}` then the output is `{{2008,12,7},{15,30,0}}`.

sub_week

- Module: *mod_base* (page 360)

See also:

sub_day (page 462), *add_week* (page 457), *sub_month* (page 462), *sub_year* (page 463)

Subtracts a week from a date. The value must be of the form `{{Y,M,D},{H,I,S}}`.

For example:

```
{{ value|sub_week }}
```

When the value is `{{2008,12,10},{15,30,0}}` then the output is `{{2008,12,3},{15,30,0}}`.

The filter has an optional argument which defines the number of weeks to subtract:

For example:

```
{{ value|sub_week:3 }}
```

When the value is `{{2008,12,10},{15,30,0}}` then the output is `{{2008,11,19},{15,30,0}}`.

sub_year

- Module: *mod_base* (page 360)

See also:

sub_day (page 462), *sub_week* (page 462), *sub_month* (page 462), *add_year* (page 457)

Subtracts a year from a date. The value must be of the form `{{Y,M,D},{H,I,S}}`.

For example:

```
{{ value|sub_year }}
```

When the value is `{{2008,12,10},{15,30,0}}` then the output is `{{2007,12,10},{15,30,0}}`.

The filter has an optional argument which defines the number of years to subtract:

For example:

```
{{ value|sub_year:3 }}
```

When the value is `{{2008,12,10},{15,30,0}}` then the output is `{{2005,12,10},{15,30,0}}`.

timesince

- Module: *mod_base* (page 360)

See also:

date (page 457), *now* (page 529)

Show a readable version of a date/time difference.

Translates the difference between two dates into a simple readable string like “2 minutes, 10 seconds ago”.

Optionally takes an argument with the date to compare against, which is by default the current local date/time.

Example:

```
{{ my_date|timesince }}
```

When “my_date” is `{{2008,12,10},{15,30,0}}` and the current date/time is `{{2009,11,4},{13,50,0}}` then this outputs “10 months, 24 days ago”. When the time value is in the future then it outputs a string like “in X minutes”.

This function does not take daylight saving changes into account.

Extra arguments

The `timesince` filter can take several extra arguments, in the order of arguments:

- Base date to use. Useful to show the difference between two dates, defaults to `now`
- Text to use for the relative time designations, defaults to “ago, now, in”

- Format for the printout. Now two options 1 and 2, for the number of components shown. For example 2 will show *2 minutes, 10 seconds ago* where 1 will show *2 minutes ago*

Example

Show the time between creation and modification of a resource:

```
{{ id.created|timesince:id.modified:"":1 }}
```

This might display something like:

```
10 days
```

utc

- Module: *mod_base* (page 360)

See also:

date (page 457)

Translates a datetime from local time to UTC.

For example:

```
{{ id.modified|utc|date:"Ymd:His\Z" }}
```

Displays the modification date and time of a resource in Universal Time.

Encryption

md5

- Module: *mod_base* (page 360)

Translates a string to a **md5** hex value.

For example:

```
{{ "The quick brown fox jumps over the lazy dog"|md5 }}
```

Creates:

```
9E107D9D372BB6826BD81D3542A419D6
```

Note that MD5 is not considered to be a very safe encryption algorithm.

sha1

- Module: *mod_base* (page 360)

Translate a string to a sha1 hex value.

This filter creates a SHA-1 checksum of the input string. It is output as a hex digest:

```
{{ "Hello world"|sha1 }}
```

Outputs the value: “7B502C3A1F48C8609AE212CDFB639DEE39673F5E”

Character escaping

brlinebreaks

- Module: *mod_base* (page 360)

See also:

linebreaksbr (page 468)

Translate HTML `
` elements into ASCII newlines (`\n`).

The following string:

```
{{ "foo<br/>bar"|brlinebreaks }}
```

will evaluate to `foo\nbar`.

Note: Non-closing line breaks (`
`) are currently not converted.

escape

- Module: *mod_base* (page 360)

See also:

force_escape (page 468), *escape_check* (page 465)

HTML escape a text. Escapes all reserved HTML characters in the value. Escaped strings are safe to be displayed in a HTML page. When you echo a query string argument or path variable then you must escape the value before displaying it on a HTML page.

The following characters are replaced:

Character	Replacement
>	<code>&gt;</code>
<	<code>&lt;</code>
"	<code>&quot;</code>
'	<code>&#039;</code>
&	<code>&amp;</code>

The escaping is only applied if the filter is not within an `{% autoescape on %}` block. If you always want escaping to be applied, use the *force_escape* (page 468) filter.

For example:

```
{{ value|escape }}
```

When the value is `<hel&lo>` then the output is `<hel&lo>`.

Note: this filter is not part of a module, it is built into the template compiler.

escape_check

- Module: *mod_base* (page 360)

See also:

force_escape (page 468), *escape* (page 465)

Ensures that an HTML escaped value is properly escaped.

Checks for all reserved HTML characters if they are properly escaped.

Escaped strings are safe to be displayed in a HTML page. When you echo a query string argument or path variable then you must escape the value before displaying it on a HTML page.

The following characters are replaced:

Character	Replacement
>	>
<	<
"	"
'	'
&	&

If you always want escaping to be applied, use the *force_escape* (page 468) filter.

For example:

```
{{ value|escape_check }}
```

When the value is <hel&lo> then the output is <hel&lo>.

escape_ical

- Module: *mod_base* (page 360)

See also:

escape (page 465)

Escape the value according to the RFC2445 rules.

A double quote becomes \"; a comma becomes \,; a colon becomes " : "; a semicolon becomes \; ; a backslash becomes \\ and a newline becomes \n.

It also ensures that any single line is maximum 70 characters long by splitting the lines with newline/space combinations.

For example:

```
{{ value|escape_ical }}
```

When the value is abc:d;e then the output is abc" : "d\;e.

escape_link

- Module: *mod_base* (page 360)

See also:

urlize (page 473)

Convert any URLs in a plaintext into HTML links, with adding the rel="nofollow" attribute.

Example:

```
{{ "http://foo.bar/"|escape_link }}
```

Outputs:

```
<a href="http://foo.bar/" rel="nofollow">http://foo.bar/</a>
```

This filter is very useful when displaying user-generated plaintexts, like comments.

escapejs

- Module: *mod_base* (page 360)

See also:

escape (page 465), *escapejson* (page 467)

Escapes the value for insertion in JavaScript output.

For example:

```
{{ value|escapejs }}
```

When the value is `he' llo` then the output is `he\x27llo`.

Internally, this calls `z_utils:js_escape/1` to perform the escaping.

Note: when generating JSON output, be sure to use *escapejson* (page 467), as JSON escaping is subtly different from JS escaping.

escapejson

- Module: *mod_base* (page 360)

See also:

escape (page 465), *escapejs* (page 467)

Escapes the value for insertion in JSON output.

For example:

```
{{ value|escapejson }}
```

When the value is `he' llo` then the output is `he\x27llo`.

Internally, this calls `z_utils:json_escape/1` to perform the escaping.

escapexml

- Module: *mod_base* (page 360)

Escape the value for insertion in xml output.

For example:

```
{{ value|escapexml }}
```

When the value is `<he' llo>` then the output is `<he l'l o>.`

fix_ampersands

- Module: *mod_base* (page 360)

See also:

escape (page 465)

Replaces ampersands in the value with “&” entities.

For example:

```
{{ value|fix_ampersands }}
```

If the value is `hel&lo` then the output is `hel&lo`.

force_escape

- Module: *mod_base* (page 360)

See also:

escape (page 465), *escape_check* (page 465)

HTML escapes a text.

Applies HTML escaping to a string (see the *escape* (page 465) filter for details). In contrary to the *escape* filter, the *force_escape* filter is applied *immediately* and returns a new, escaped string. This is useful in the rare cases where you need multiple escaping or want to apply other filters to the escaped results. Normally, you want to use the *escape* (page 465) filter.

For example:

```
{{ value|force_escape }}
```

If the value is `hel&lo` then the output is `hel&lo`.

linebreaksbr

- Module: *mod_base* (page 360)

See also:

brlinebreaks (page 465)

Translate ASCII newlines (`\n`) into HTML `
` elements.

The following string:

```
{{ "foo\nbar"|linebreaksbr }}
```

will evaluate to `foo
bar`.

slugify

- Module: *mod_base* (page 360)

See also:

stringify (page 500)

Converts a text into a slug.

Makes the value safe for use as a part in an url. Mostly used for adding titles or descriptions to an url.

For example:

```
{{ value|slugify }}
```

When value is “Nichts ist unmöglich!” then the output will be “nichts-ist-unmoglich”.

unescape

- Module: *mod_base* (page 360)

See also:

escape (page 465), *force_escape* (page 468)

Removes HTML escaping from a text.

Expands the entities added by the *escape* (page 465) filter or *force_escape* (page 468) filter. This is useful when you want to display a field from the database in a text-only format medium.

For example:

```
Title: {{ m.rsc[id].title|unescape }}
```

Be careful that you only use this filter when you are absolutely sure that the output is not used in HTML or XML.

urlencode

- Module: *mod_base* (page 360)

See also:

sanitize_url (page 472), *is_site_url* (page 507), *url_abs* (page 508), *url* (page 507), *urldecode* (page 508)

Make a text safe for URLs.

Translates all url unsafe characters in the value to their percent encoding.

For example:

```
{{ value|urlencode }}
```

When value is “msg=Hello&World” then the output is “msg%3DHello%26World”.

Forms

pickle

- Module: *mod_base* (page 360)

Pickle an Erlang value so that it can be safely submitted with a form.

The Erlang value is encoded using *erlang:term_to_binary/1* and signed with the site’s secret key. In Erlang the value can be unpacked using *z_utils:depickle/2*

Usage:

```
<input type="hidden" name="confirm" value="{{ 'Hello world' | pickle }}" />
```

This will generate something like:

```
<input type="hidden" name="confirm" value="duZTXcXaxuruD3dhpt-
↳rxWokrhuDbQAAAAAtIZWxsbyB3b3JsZA" />
```

HTML

show_media

- Module: *mod_base* (page 360)

See also:

[*embedded_media*](#) (page 473), [*without_embedded_media*](#) (page 473)

Convert the image markers in HTML from the Rich Text editor into image tags.

When you add images in the Rich Text editor of the Zotonic admin, the HTML body text does not store `` tags, but instead, special markers containing the picture id plus size and alignment hints.

The `show_media` tag converts these special markers (which are in fact HTML comments) back into image tags. For this, it uses the template called `_body_media.tpl`. This template is located in `mod_base` and can be overruled with your own.

Images that are inlined in the body text can have these parameters:

align Either `block`, `left` or `right`. Default is `block`.

size Choose between `small`, `medium`, `large`. These are used for selecting the mediaclass of the image. Which will be like `body-media-large`. The default size is `large`, displayed as a block (see align above).

crop Checkbox if you want to force cropping of the image to the bounding box of the mediaclass. If checked then the crop in the mediaclass is overruled. If not checked then the crop in the mediaclass is used.


link Checkbox if you want to let the image include a link to its own page.


link_url If `link` is checked then this can be used to set the URL for the link, the link url defaults to the page-url of the shown medium item.

caption Caption that will be displayed below the media item. Per default the summary of the media item is used for the caption. If the caption is empty, then no caption is added.

These parameters can be set in the editor dialog 'Media Properties'. This dialog is shown after clicking on an image in the wysiwyg editor.

Media Properties





Size


☐ Small
 ☐ Medium
 ☒ Large

Crop

☐ Crop image

Link

☒ Link to media



Caption

Planting trees is a good way to celebrate birthdays.

Save

Cancel

Template arguments

The `_body_media.tpl` is included using a *catinclude* for the media item. In this way you can switch templates depending on the item category being displayed.

You can add your own `_body_media.tpl` templates. It will be supplied with the following data:

Argument	Description
id	The id of the medium item to be displayed.
size	The size, <code>small</code> , <code>medium</code> or <code>large</code>
mediaclass	Mediaclass, for example <code>body-media-large</code>
align	Alignment, <code>block</code> , <code>left</code> or <code>right</code>
crop	If crop is forced, <code>true</code> or <code>undefined</code>
link	If the image should link to the medium page
link_url	Link to use, defaults to the medium page
caption	The caption from the editor

Besides the above all context variables are passed, this gives the Erlang code the possibility to change the behavior of the media rendering.

6.1. Reference

471

striptags

- Module: *mod_base* (page 360)

Removes all HTML tags from the value.

Useful as part of filtering input from external sources.

For example:

```
{{ value|striptags }}
```

When value is “Helloworld” then the output will be “Helloworld”.

truncate_html

- Module: *mod_base* (page 360)

See also:

truncate (page 502)

Truncate a HTML text to a maximum length.

The HTML text is truncated to the maximum length specified with the argument. The text will be truncated at the maximum length, if any text remains then the ellipsis character ... is added and all open HTML tags are closed.

For example:

```
{{ value|truncate_html:8 }}
```

If the value is hello world then the output is hello wo....

Entities like “&” are counted as a single character.

Self closing entities like and
 are not counted as characters.

Truncating character

An optional second argument defines which text will be added if the text is truncated:

```
{{ value|truncate_html:8:" (more)" }}
```

If the value is hello world then the output is hello wo (more).

sanitize_html

- Module: *mod_base* (page 360)

Sanitize a HTML code. Removes elements and attributes that might be dangerous, like <script> elements.

This filter parses the complete HTML string and ensures that the output is safe and valid HTML.

Do not use (without caching) on busy pages, as the sanitization process is cpu and memory intensive.

sanitize_url

- Module: *mod_base* (page 360)

Sanitize an URL. Removes URLs that might be dangerous, like javascript: URLs.

Ensure that the input to this filter is HTML unescaped. As the filter is not unescaping any HTML entities.

urlize

- Module: *mod_base* (page 360)

See also:

escape_link (page 466)

Find urls in the given input and make them clickable.

Example:

```
{{ "http://foo.bar/"|urlize }}
```

Outputs:

```
<a href="http://foo.bar/">http://foo.bar/</a>
```

This filter is very similar to the *escape_link* (page 466) filter.

embedded_media

- Module: *mod_base* (page 360)

See also:

show_media (page 469), *without_embedded_media* (page 473)

Fetch media ids that are embedded in the *body*, *body_extra* and *text* blocks of your page.

This filter lets you loop over every image that is embedded in the texts of the given page:

```
{% for media_id in id|embedded_media %}
  {% media media_id width=315 extent %}
{% endfor %}
```

There is an optional (boolean) argument to only fetch media ids from the *body* and *body_extra* properties:

```
{% for media_id in id|embedded_media:0 %}
  {% media media_id width=315 extent %}
{% endfor %}
```

You can also fetch all media ids embedded in a text:

```
{% for media_id in id.body|embedded_media %}
  {% media media_id width=315 extent %}
{% endfor %}
```

without_embedded_media

- Module: *mod_base* (page 360)

See also:

show_media (page 469), *embedded_media* (page 473)

Filter out media ids that are embedded in the *body*, *body_extra* and *text* blocks of your page.

This filter lets you loop over every image that is not included in the embedded *body* texts of the given page. This makes it easy to only show images that have not been shown already:

```
{% for media_id in m.rsc[id].media|without_embedded_media:id %}
  {% media media_id width=315 extent %}
{% endfor %}
```

The only argument to the filter is the *id* of the page that you want to consider for filtering from the body text. There is an optional second argument to only consider media ids in the *body* and *body_extra* properties:

```
{% for media_id in m.rsc[id].media|without_embedded_media:id:0 %}
  {% media media_id width=315 extent %}
{% endfor %}
```

Lists

after

- Module: *mod_base* (page 360)

Return the first element after another element in a list. For example:

```
{{ [1,2,3]|after:2 }}
```

Evaluates to the value 3.

If the element is not part of the list, or is the last element in the list, the returned value is *undefined*.

before

- Module: *mod_base* (page 360)

Return the first element before another element in a list. For example:

```
{{ [1,2,3]|before:2 }}
```

Evaluates to the value 1.

When the element is not part of the list, or is the first element in the list, the returned value is *undefined*.

chunk

- Module: *mod_base* (page 360)

See also:

split_in (page 481), *vsplit_in* (page 482)

This filter splits a list in shorter lists. It splits an array in sub-arrays of at most a given length. This is useful when displaying a list of items in columns or rows.

For example:

```
{% for s in [1,2,3,4,5]|chunk:2 %}
  {% for n in s %}{{ n|format_number }} {% endfor %} *
{% endfor %}
```

This displays 1 2 * 3 4 * 5 *, as the array is split in three chunks. The last chunk is not filled to the maximum length. Then number of chunks depends on the length of the input list, this in contrary to the *split_in* filters where the number of splits is fixed and the length per split is variable.

exclude

- Module: *mod_base* (page 360)

See also:

is_visible (page 495), *is_a* (page 494), *filter* (page 475)

Filters a list on the value of a property, either on absence or inequality.

This is the inverse of *filter* (page 475).

Testing presence

To filter a list of values:

```
{% print somelist|exclude:`p` %}
```

Results in a list where all elements **do not have** the property `p` defined and where the property (after conversion to boolean) is `false`.

This can be used to filter a list of resource ids on the absence of a property. For example, to see all unpublished elements in a list of resource ids:

```
{% print [1,2,3,4,5,6]|exclude:`is_published` %}
```

To find all pages from page connection `hasdocument` that **do not have** an image:

```
{% print id.o.hasdocument|exclude:`depiction` %}
```

Testing equality

A second argument can be added to test on inequality:

```
{% print somelist|exclude:`title`: "Untitled" %}
```

Shows all elements whose `title` property **is not** “Untitled”.

Below is another example of inversely filtering a list:

```
{% with m.search[{latest cat='gallery'}] as result %}
  {% if result.total > 0 %}
    {% with result|exclude:`name`: "page_home_gallery"|random as gallery_rsc_id %}
      {% include "_gallery_widget.tpl" id=gallery_rsc_id %}
    {% endwith %}
  {% endif %}
{% endwith %}
```

The example above filters against a search result and returns only elements whose name **is not** “page_home_gallery”.

filter

- Module: *mod_base* (page 360)

See also:

is_visible (page 495), *is_a* (page 494), *exclude* (page 475)

Filters a list on the value of a property, either on presence or equality.

Testing presence

To filter a list of values:

```
{% print somelist|filter:`p` %}
```

Results in a list where all elements have the property `p` defined and where the property (after conversion to boolean) is `true`.

This can be used to filter a list of resource ids on the presence of a property. For example, to see all published elements in a list of resource ids:

```
{% print [1,2,3,4,5,6]|filter:`is_published` %}
```

To find all pages from page connection `hasdocument` that have an image:

```
{% print id.o.hasdocument|filter:`depiction` %}
```

Testing equality

A second argument can be added to test on equality:

```
{% print somelist|filter:`title`:"Untitled" %}
```

Shows all elements whose `title` property is “Untitled”.

flatten_value

- Module: *mod_base* (page 360)

Flatten a list to a comma separated string.

Example:

```
{{ [ "a", 100, "c" ]|flatten_value }}
```

Gives:

```
a,100,c
```

As list of only integers in the range 32..255 is assumed to be a string.

group_by

- Module: *mod_base* (page 360)

Groups items of a list by a property.

When the item is an integer then it is assumed to be the id of a resource. This is especially useful for grouping items in for-loops.

For example:

```
{% for grp in value|group_by:"a" %} ... loop over grp ... {% endfor %}
```

When `value` is the three element list:


```
[
  [{a,1}, {b,1}],
  [{a,1}, {b,2}],
  [{a,2}, {b,3}]
]
```

then the output of `group_by` “a” will be the two element list:

```
[
  [[{a,1}, {b,1}], [{a,1}, {b,2}]],
  [[{a,2}, {b,3}]]
].
```

index_of

- Module: *mod_base* (page 360)

See also:

element (page 507)

Returns the index of the first occurrence of the item in the given list.

For example:

```
{{ [44,11,2,443,2] | index_of:11 }}
```

Returns 2.

Note: Erlang list indices are always 1-based.

is_list

- Module: *mod_base* (page 360)

Test if a value is a list:

```
{% if [1,2,3] | is_list %}Yes, this is a list {% endif %}
```

join

- Module: *mod_base* (page 360)

See also:

element (page 507), *tail* (page 481), *split* (page 481)

Joins the elements of a list. Joins the elements of the input list together, separated by the argument.

For example:

```
{{ value | join:", " }}
```

If the value is the list `["hello", "world"]` then the output will be `"hello, world"`.

It is possible to use a special separator between the last two elements of the list, for the list `["Jan", "Piet", "Klaas"]` the following example:

```
{{ list | join:", ":"_or" }} }}
```

Gives as result:

```
Jan, Piet or Klaas
```

The spaces around the last separator are added by the filter.

make_list

- Module: *mod_base* (page 360)

Forces the value to a list.

For example:

```
{% print value|make_list %}
```

When value is the tuple {"a", "b"} then the output is the list ["a", "b"].

This filter is especially useful for loops using the {% for %} tag:

```
{% for v in value|make_list %}
  {{ v|escape }}
{% endfor %}
```

member

- Module: *mod_base* (page 360)

Finds a value in a list.

Checks if the value is part of the argument. The argument must be a list. Returns a boolean value.

For example:

```
{% if value|member:[1,2,3] %}
  One of the first three
{% endif %}
```

When value is the integer 2 then the output is “One of the first three”.

nthtail

- Module: *mod_base* (page 360)

See also:

first (page 454), *tail* (page 481)

Fetch the nth tail of a list.

Useful when you want to skip the first N elements of a list when looping.

For example:

```
{% for a in value|nthtail:2 %}{{ a|format_number }}{% endfor %}
```

When value is the list [1, 2, 3] then the output is 3.

random

- Module: *mod_base* (page 360)

See also:

randomize (page 479), *rand* (page 490)

Returns a random value from a list of values. When the input is an empty list or not a list then the result is undefined.

For example:

```
{{ ["a", "b", "c"] | random }}
```

The output of this is one of “a”, “b” or “c”.

randomize

- Module: *mod_base* (page 360)

See also:

rand (page 490), *random* (page 479)

Shuffle a list of values.

For example:

```
{{ ["a", "b", "c"] | randomize }}
```

The output of this is the same list, but the order of the elements randomized. So for instance: [”c”, “a”, “b”].

range

- Module: *mod_base* (page 360)

Generate a list of integers, with an optional step.

For example:

```
{{ 1 | range:4 }}
```

Generates the list [1, 2, 3, 4].

The second filter argument is the step size:

```
{{ 1 | range:10:2 }}
```

Generates the list [1, 3, 5, 7, 9].

reversed

- Module: *mod_base* (page 360)

Reverse a list.

For example:

```
{{ value | reversed }}
```

When value is ["hello", "world"] then the output is "worldhello".

The main use for this filter is to reverse lists of values or search results. There is no support for multi-byte unicode characters, this is only a problem when applying the filter directly to a string value.

New in version 0.6.

slice

- Module: *mod_base* (page 360)

Perform array-slice operations on a list or string.

If the argument is a binary then it is handles as an UTF-8 encoded string.

Given a list [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]

Get all elements from element M to element N:

```
{{ list|slice:[3,7] }} -> [3,4,5,6,7]
{{ list|slice:[3,-3] }} -> [3,4,5,6,7]
{{ list|slice:[-7,-3] }} -> [4,5,6,7]
{{ list|slice:[-7,7] }} -> [4,5,6,7]
```

Get all elements except the first N:

```
{{ list|slice:[3,] }} -> [3,4,5,6,7,8,9,0]
{{ list|slice:[-7,] }} -> [4,5,6,7,8,9,0]
```

Get all elements up to element N:

```
{{ list|slice:[,3] }} -> [1,2,3]
{{ list|slice:[3] }} -> [1,2,3]
```

Get all elements except the last N:

```
{{ list|slice:[,-3] }} -> [1,2,3,4,5,6,7]
{{ list|slice:[-3] }} -> [1,2,3,4,5,6,7]
```

If the slice start is after the slice end then [] is returned:

```
{{ list|slice:[2,1] }} -> []
```

An empty slice returns the whole input:

```
{{ list|slice:[,] }} -> [1,2,3,4,5,6,7,8,9,0]
```

sort

- Module: *mod_base* (page 360)

The *sort* filter takes a list of items to sort. Items can be a ordinary list of terms, or a list of resources to be filtered based on their properties. Sort order and properties to sort on are given as arguments to the filter.

By default it sorts the list in *ascending* order, and resource lists are sorted on their *id* if no property is specified.

Example:

```
{{ [4, 6, 2, 3, 5]|sort }}
```

Sorts the list of numbers in *ascending* order.

Example:

```
{{ [4, 6, 2, 3, 5]|sort:'desc' }}
```

Sorts the list of numbers in *descending* order.

Example:

```
{% for r in id.o.author|sort:['title', 'desc', 'modified'] %}
  do something with `r`...
{% endfor %}
```

This will sort on *title* in *ascending* order first, then on *modified* in *descending* order. Any number of properties may be added, each one can have it's own sort order, or use the current one.

See [m_rsc](#) (page 571) for a list of properties available to sort on.

Sort order may be either *ascending* or *descending* (may be abbreviated as *asc*, +, *desc*, - or as string version of those).

split

- Module: [mod_base](#) (page 360)

See also:

[join](#) (page 477)

Splits the filter value into a list of values.

The input value is split by the filter argument, for example:

```
{{ "foo bar baz"|split:" " }}
```

Will create the list ["foo", "bar", "baz"].

split_in

- Module: [mod_base](#) (page 360)

See also:

[chunk](#) (page 474), [vsplit_in](#) (page 482)

This filter split a list in shorter lists. It splits an array in N sub-arrays of more or less equal length. This is useful when displaying a list of items in columns.

For example:

```
{% with [1,2,3,4,5,6]|split_in:3 as a,b,c %}
  {% for n in a %}{{ n|format_number }} {% endfor %}
{% endwith %}
```

This displays 1 4. The variable b will be [2, 5] and the variable c will be [3, 6].

tail

- Module: [mod_base](#) (page 360)

See also:

[first](#) (page 454), [nthtail](#) (page 478)

Fetch the tail of a list.

Returns the tail of a list. Useful when you want to skip the first element of a list when looping.

For example:

```
{% for a in value|tail %}{{ a|format_number }}{% endfor %}
```

When value is the list [1, 2, 3] then the output is 23.

vsplit_in

- Module: *mod_base* (page 360)

See also:

chunk (page 474), *split_in* (page 481)

This filter splits a list in shorter lists. It splits an array in N sub-arrays of more or less equal length. This is useful when displaying a list of items in columns.

Note that it splits the array in a different way than *split_in* (page 481) does: The filter *split_in* takes alternating elements from the array, where *vsplit_in* takes complete runs at a time. See the example below.

For example:

```
{% with [1,2,3,4,5,6]|vsplit_in:3 as a,b,c %}
  {% for n in a %}{{ n|format_number }} {% endfor %}
{% endwith %}
```

This displays 1 2. The variable b will be [3, 4] and the variable c will be [5, 6].

without

- Module: *mod_base* (page 360)

Remove the items given in the argument from the filter value.

For example:

```
{% print [1,2,3]|without:[2] %}
```

prints:

```
[1,3]
```

This filter also works on list-like values like resource edges:

```
{% for id in id.o.tags|without:some_id.o.tags %}
```

Iterates of all *tags* edges of the given *id*, for each *id* that is not also an edge of *some_id*.

Mailing list

inject_recipientdetails

- Module: *mod_mailinglist* (page 379)

Adds recipient query string details to hyperlinks.

This filter is meant for use inside e-mail templates. It replaces each occurrence of ## with the details of the subscriber that the mail is sent to, encoded as query string arguments.

Each occurrence of ## will be transformed to recipient details in the following form:

```
?email=foo@bar.com&name_first=John&name_last=Doe
```

Its use case is when sending a mailing with a link in it which arrives at a webpage where the user has to enter his e-mail address and name details. Using this filter, those parameters can be conveniently be pre-filled.

Menu

menu_flat

- Module: *mod_menu* (page 381)

See also:

menu_subtree (page 484), *menu_trail* (page 484)

Flattens the rsc menu structure for use in a template loop.

Example:

```
{% for item in m.rsc[id].menu|menu_flat %}
...
{% endif %}
```

menu_is_visible

- Module: *mod_menu* (page 381)

Filters a list of menu items on visibility and existence. Only top-level menu items that are both visible and exist are kept in the list. Note that sub-menus are not filtered, they need to be filtered separately.

The *is_visible* (page 495) filter can't be used due to the structure of a menu item list.

Example:

```
{% with m.rsc.main_menu.menu|menu_is_visible as menu %}
  {% if menu %}
    <ul>
      {% for item in menu %}
        <li>{{ item.id.title }}</li>
      {% endfor %}
    </ul>
  {% endif %}
{% endwith %}
```

menu_expand

- Module: *mod_menu* (page 381)

Takes a menu, or a menu resource id, and adds all haspart objects connected to the menu ids.

With this filter it is possible to add a collection to the menu and have the pages in the collection automatically added to the menu.

Example:

```
{% with m.rsc.main_menu.menu|menu_expand as menu %}
  {% if menu %}
    {% with id|menu_trail:menu as trail %}
      <ul>
        {% for item in menu %}
```

```
<li {% if item.id|member:trail %}class="active"{% endif %}>
  <a href="{ { item.id.page_url } }">{ { item.id.title } }</a>
</li>
{% endfor %}
</ul>
{% endwith %}
{% endif %}
{% endwith %}
```

menu_rsc

- Module: *mod_menu* (page 381)

Todo

Not yet documented.

menu_subtree

- Module: *mod_menu* (page 381)

See also:

menu_trail (page 484), *menu_flat* (page 483)

Get the subtree of an id in a menu (if any).

Returns the subtree of the filter value. Useful for showing a part of a menu when browsing sub-pages.

If the given id is not found inside the menu, it returns undefined.

If no argument is given, it takes menu from the resource with the name `main_menu`.

menu_trail

- Module: *mod_menu* (page 381)

See also:

menu_subtree (page 484), *menu_flat* (page 483)

Return a breadcrumb navigation trail for the given id.

This filter locates the filter value which represents the current page in the main menu or in the menu saved in the resource of the given id.

For example:

```
{% print id|menu_trail:55 %}
```

Could print the list `[13, 33]` if ids 13 and 33 are the parents of the *id* argument in the menu resource 55.

If no argument is given, it takes menu from the resource with the name `main_menu`.

Instead of a single id it is possible to give a list of ids. The trail for the first id that returns a trail in the menu is returned.

Showing Menu Trail only for submenu items

It is sometimes useful to suppress the menu trail on top level items. Here is how to do it.

The *menu_trail* (page 484) includes the whole path through the menu to the current page if it is reachable that way. Sometimes it may seem pointless to show the menu trail if we are on the first level of the menu. If we want to avoid this we need to avoid rendering a trail when it is less than two items long.

One simple condition change to *_article_chapeau.tpl* from the *blog* skeleton makes this work:

```
{% with id|menu_trail as parents %}
  {% if parents|length > 1 %}
    <h5 class="chapeau">
      {% for p in parents %}
        <a href="{{ m.rsc[p].page_url }}">{{ m.rsc[p].title }}</a>
        {% if not forloop.last %}&raquo;{% endif %}
      {% endfor %}</h5>{% endif %}{% endwith %}
```

The key here is `{% if parents|length > 1 %}` in place of just `{% if parents %}`.

The *if* (page 518) tag is now rendering the *menu_trail* only if there are two or more items in it which - as I mentioned before - happens when you are at least two levels deep in the menu.

Miscellaneous

gravatar_code

- Module: *mod_comment* (page 361)

Calculate the gravatar code for an e-mail address:

```
{{ "arjan@scherpenisse.net"|gravatar_code }}
```

Will output:

```
3046ecab06c4f9cdb49963a96636e5ef
```

This hash can then be used for displaying Gravatar images.

twitter

- Module: *mod_twitter* (page 401)

Format a plain text Tweet into HTML.

This filter creates hyperlinks out of the embedded URLs, @username-references and #tag-hashtags in a Tweet, like Twitter does.

For example:

```
{{ "@acscherp"|twitter }}
```

Converts into:

```
<a href="https://twitter.com/acscherp">@acscherp</a>
```

ip2country

- Module: *mod_geoip* (page 376)

Maps an IP address to a country using the MaxMind GeoIP2 database.

The module `mod_geoip` must be enabled to use this filter.

Example, print the country code of the current visitor:

```
{% print m.req.peer|ip2country %}
```

Might print `nl`.

If the IP address could not be mapped then `undefined` is returned.

ip2geo

- Module: *mod_geoip* (page 376)

Maps an IP address to information about that country.

The module `mod_geoip` must be enabled to use this filter.

Example, print the information for the current visitor:

```
{% print m.req.peer|ip2geo %}
```

Might print (depending on the visitor's IP address):

```
#city =>
  {trans, [{zh, <<230, 160, 188, 231, 189, 151, 229, 174, 129, 230,
    160, 185>>},
    {ru, <<208, 147, 209, 128, 208, 190, 208, 189, 208, 184, 208, 189,
    208, 179, 208, 181, 208, 189>>},
    {pt, <<"Groningen">>},
    {ja, <<227, 131, 149, 227, 131, 173, 227, 131, 188, 227, 131, 139,
    227, 131, 179, 227, 130, 178, ...>>},
    {fr, <<"Groningue">>},
    {es, <<"Groninga">>},
    {en, <<"Groningen">>},
    {de, <<"Groningen">>}}]},
continent =>
  #{code => <<"EU">>,
    name =>
      {trans, [{zh, <<230, 172, 167, 230, 180, 178>>},
        {ru, <<208, 149, 208, 178, 209, 128, 208, 190, 208, 191, 208, 176>>},
        {pt, <<"Europa">>},
        {ja, <<227, 131, 168, 227, 131, 188, 227, 131, 173, 227, 131, 131,
        227, 131, 145>>},
        {fr, <<"Europe">>},
        {es, <<"Europa">>},
        {en, <<"Europe">>},
        {de, <<"Europa">>}}]}},
country =>
  #{is_eu => true, iso => <<"nl">>,
    name =>
      {trans, [{zh, <<232, 141, 183, 229, 133, 176>>},
        {ru, <<208, 157, 208, 184, 208, 180, 208, 181, 209, 128, 208, 187,
        208, 176, 208, ...>>},
        {pt, <<"Holanda">>},
        {ja, <<227, 130, 170, 227, 131, 169, 227, 131, 179, 227, 131, 128,
        231, ...>>},
        {fr, <<"Pays-Bas">>},
        {es, <<"Holanda">>},
        {en, <<"Netherlands">>},
        {de, <<"Niederlande">>}}]}},
location =>
```

```
#{accuracy_radius => 100, latitude => 53.2124,
  longitude => 6.5538, postcode => <<"9726">>,
  timezone => <<"Europe/Amsterdam">>},
subdivisions =>
  [{code => <<"GR">>,
    name =>
      {trans, [{ru, <<208,147,209,128,208,190,208,189,208,184,
                    208,189,208,179,...>>},
                {pt, <<"Groninga">>},
                {fr, <<"Groningue">>},
                {es, <<"Groninga">>},
                {en, <<"Groningen">>},
                {de, <<"Groningen">>}]}}]}
```

The `trans` records can be shown directly, Zotonic will select the correct language:

```
{% with m.req.peer|ip2geo as info %}
  {{ info.country.name }} {% if info.city %} - {{ info.city }}{% endif %}
{% endwith %}
```

This might print (depening on the peer):

```
Netherlands - Groningen
```

If the IP address could not be mapped then `undefined` is returned.

is_letsencrypt_valid_hostname

- Module: *mod_ssl_letsencrypt* (page 397)

See also:

mod_ssl_letsencrypt (page 397)

Test if a hostname can be used for a Let's Encrypt certificate.

Criteria are:

1. Does resolve using DNS
2. The resolved address is not a LAN address
3. The address is reachable
4. And the current site is listening for the hostname on that address

The site must listen on port 80 for connections.

For example, check if the current site is reachable as *example.com*:

```
{% if "example.com"|is_letsencrypt_valid_hostname %}
  Wow, this site is example.com!?!
{% endif %}
```

Numbers

filesizeformat

- Module: *mod_base* (page 360)

This filter formats a numeric value as KB, MB etc. This filter can be used to display a number of bytes in a human readable format of kilo- or megabytes.

For example:

```
{{ 532671|filesizeformat }}
```

Will output 520.2 KB.

format_duration

- Module: *mod_base* (page 360)

See also:

format_number (page 489), *format_integer* (page 488), *format_price* (page 489)

Show a duration in hours, minutes and seconds.

Takes as input a number representing a number of seconds. Outputs a human readable form.

For example:

```
{{ 123|format_duration }}
```

Will output 2m3s.

And:

```
{{ 3601|format_duration }}
```

Will output 1h0m1s.

format_integer

- Module: *mod_base* (page 360)

See also:

to_integer (page 491), *format_number* (page 489), *format_price* (page 489), *format_duration* (page 488)

Show an integer value.

Formats an integer value as a list, assuming a radix of ten.

For example:

```
{{ value|format_integer }}
```

When the value is the integer 123 then the output is the list 123.

The *format_integer* filter has an optional argument to pad the returned string with zeros to a fixed length. For example:

```
{{ 123|format_integer:5 }}
```

Will output 00123. And when the number does not fit:

```
{{ 123|format_integer:2 }}
```

Will output “**”.

Note: This option only works for positive integers.

format_number

- Module: *mod_base* (page 360)

See also:

format_integer (page 488), *format_price* (page 489), *format_duration* (page 488),

Show an integer or float.

Formats integer and float values as a list, assuming a radix of ten.

For example:

```
{{ value|format_number }}
```

When the value is the float 12.0 then the output is the list 12.0.

format_price

- Module: *mod_base* (page 360)

See also:

format_number (page 489), *format_integer* (page 488), *format_duration* (page 488)

Show a price with decimals.

Formats integer and float values as a number with two decimals.

For example:

```
{{ value|format_price }}
```

When the value is the float 12.1 then the output is the list 12.10.

An undefined price will have the output “-”.

Note: the decimal separator is currently always a dot, independent of the user’s language.

is_even

- Module: *mod_base* (page 360)

Test if an integer value is even:

```
{% if 2|is_even %}Yes, the number 2 is even{% endif %}
```

is_number

- Module: *mod_base* (page 360)

Test if a value is a number (integer or floating point):

```
{% if 1|is_number %}Yes, this is a number{% endif %}
```

max

- Module: *mod_base* (page 360)

See also:

[min](#) (page 490), [minmax](#) (page 490)

Take the maximum of the filter value and its first argument.

The following:

```
{% print 102|to_integer|max:103 %}
```

Prints 103.

min

- Module: [mod_base](#) (page 360)

See also:

[max](#) (page 489)

Take the minimum of the filter value and its first argument.

The following:

```
{% print 102|to_integer|min:103 %}
```

Prints 102.

minmax

- Module: [mod_base](#) (page 360)

See also:

[max](#) (page 489), [min](#) (page 490)

Force the given value in the given range.

This clamps the filter value between the two filter arguments.

Example:

```
{% print 3|to_integer|minmax:10:20 %}
```

This will print 10, since that is the minimum value allowed.

Passing in undefined will not clamp the value but return undefined.

rand

- Module: [mod_base](#) (page 360)

See also:

[randomize](#) (page 479), [random](#) (page 479)

Generates a random number. The number is from, and including, 1 up to, and including, the input value.

Example:

```
{{ 100|rand|format_integer }}
```

Might output “42”.

The rand filter can also generate a floating point value by given it a floating point number or a string representing a floating point number as input. It will generate a number from, but not including, 0 to, and including, the input value:

```
{{ "4.0"|rand|format_number }}
```

Might output “3.1415926536”

round

- Module: *mod_base* (page 360)

Round a floating point value to the nearest integer.

Example:

```
{{ 3.5|round }}
```

Results in the integer value 4.

round_significant

- Module: *mod_base* (page 360)

Round a number value to a number of significant digits. The significance defaults to two digits.

Example:

```
{{ 1256|round_significant }}
{{ 1256|round_significant:1 }}
{{ 1256|round_significant:3 }}
```

Results in:

```
1300
1000
1260
```

Floating point values are also rounded to a number of digits. For example if n is set to 1256.78 then:

```
{{ n|round_significant }}
{{ n|round_significant:3 }}
{{ n|round_significant:5 }}
```

Results in:

```
1300.0
1260.0
1256.8
```

Input values that are not a number are converted to a floating point value. If the conversion fails then undefined is returned.

to_integer

- Module: *mod_base* (page 360)

See also:

[to_binary](#) (page 455), [format_number](#) (page 489), [format_integer](#) (page 488)

Convert the input to an integer value.

Example:

```
{{ "123"|to_integer }}
```

Results in the integer value 123.

This filter uses the `z_convert:to_integer/1` function.

Regular Expressions

match

- Module: [mod_base](#) (page 360)

Match a value with a regular expression.

Returns true if the value matches the regular expression. This is handy for checking if a string starts or ends with a particular value.

Usage:

```
{% if value|match:".foo$" %}
```

Checks if the value ends with “foo”.

replace

- Module: [mod_base](#) (page 360)

Regular expression replacement of a pattern with a string.

Replaces the sub strings matching a regular expression with a new string value.

For example:

```
{{ "abcba"|replace:["b","x"] }}
```

The output will be the string “axcxa”.

If you do not specify the replacement value, the replacement value is assumed to be the empty string:

```
{{ "abcba"|replace:"b" }}
```

The output will be the string “aca”.

Note: This filter is inefficient, as it will compile and match a regular expression while serving the template. Try to do string replacements when you *save* your content, and not when you serve the content.

Resource lists

group_firstchar

- Module: [mod_base](#) (page 360)

See also:

[group_title_firstchar](#) (page 493)

Group a list of *sorted resource* (page 571) ids on their first letter of the title or another rsc property. After grouping, it splits this list in a number of more-or-less even columns.

This is useful for displaying multiple columns of alphabetically sorted pages, in which the pages are grouped by the first letter, for instance a search result.

For instance, this piece of template code:

```
<table>
{% for cols in m.search[{query cat="country" sort="pivot_title"}]|group_firstchar:
  ↳ "title":3 %}
  <td>
    {% for group in cols %}
    <b>{{ group.first }}</b>
    {% for id in group.result %}
    <li>
      {{ id.title }}
    </li>
    {% endfor %}
  {% endfor %}
</td>
{% endfor %}
</table>
```

Groups the list of ids by title in three columns. It then uses nested for-loops to render a table similar to this:

A	D	G
Afghanistan	Denmark	Georgia
Azerbeidjan		Germany
	E	Grenada
B	Ecuador	Guadeloupe
Belgium	Egypt	Greenland
Belarus	Eritrea	Guinea
Brazil	Estonia	
		H
C	F	Haiti
Canada	Fiji	
	Finland	

As you can see, all three columns have approximately the same size, although the size of the individual groups differs.

When no nr. of columns is given, the groups are returned in a single column.

group_title_firstchar

- Module: [mod_base](#) (page 360)

See also:

[group_firstchar](#) (page 492)

Similar to [group_firstchar](#) (page 492), but always uses the `title` column from the rsc table.

This is merely a shortcut, simplifying the template syntax:

```
<table>
{% for cols in m.search[...]|group_title_firstchar:4 %}
  <td>
```

```
{% for group in cols %}
<b>{{ group.first }}</b>
{% for id in group.result %}
<li>
    {{ id.title }}
</li>
{% endfor %}
{% endfor %}
</td>
{% endfor %}
</table>
```

Groups alphabetically on title, in four columns.

is_a

- Module: *mod_base* (page 360)

See also:

is_not_a (page 495), *is_visible* (page 495), *filter* (page 475)

Filter a list of resource ids on category, or test if a single resource id belongs to a category.

This filter can be applied to a list of resource ids or a single resource id.

When it is applied to a list then it will filter the list of ids. Only those resource ids that belong to a certain category remain. Optionally the filter only returns the first n matches.

When applied to a single integer (resource id), then it will return a boolean. True when the id belongs to the parameter's category, false otherwise.

Apply to a single resource id

Example:

```
{{ 1|is_a:"person"|yesno }}
```

Will output “yes”, because the resource with id 1 is a person (the System Administrator).

Apply to a list of resource ids

When applied to a list of ids:

```
{% for part_id in m.rsc[id].o.haspart|is_a:"person" %}
    {{ m.rsc[part_id].title }}
{% endfor %}
```

This will list all collection members that are a person. While:

```
{% for part_id in m.rsc[id].o.haspart|is_a:"person":3 %}
    {{ m.rsc[part_id].title }}
{% endfor %}
```

Lists only the first three collection members that are a person.

is_not_a

- Module: *mod_base* (page 360)

See also:

is_a (page 494), *is_visible* (page 495), *filter* (page 475)

is_not_a mirrors *is_a* (page 494). It is particularly useful when iterating over a category and excluding members of a sub-category (iterating over all images associated with a page except images in the thumbnail category).

Example for looping over all media in a rsc but excluding the thumbnail resources:

```
{% for m in m.rsc[id].media|is_not_a:"thumbnail" %}
...
{% endfor %}
```

is_visible

- Module: *mod_base* (page 360)

See also:

is_a (page 494), *is_not_a* (page 495), *filter* (page 475)

Filter a list of resource ids so that only the visible ids remain.

This filter can be applied to a list of resource ids or a single resource id.

This filter can be applied to a list of resource ids. Only those resource ids that are visible for the current user remain. Optionally the filter only returns the first n matches.

An example:

```
<ul>
{% for part_id in m.rsc[id].o.haspart|is_visible %}
  <li>{{ m.rsc[part_id].title }}</li>
{% endfor %}
</ul>
```

This will list all collection members that are visible, preventing empty list items.

Whilst:

```
{% for part_id in m.rsc[id].o.haspart|is_visible:3 %}
  {{ m.rsc[part_id].title }}
{% endfor %}
```

Lists only the first three collection members that are visible.

Resources

admin_merge_diff

- Module: *mod_admin_merge* (page 355)

Todo

Not yet documented.

content_type_label

- Module: *mod_base* (page 360)

Todo

Not yet documented.

content_type_urls

- Module: *mod_base* (page 360)

Todo

Not yet documented.

summary

- Module: *mod_base* (page 360)

Extract a summary from a resource.

The value of the filter is the resource's id.

This uses the *summary* field if available, but if it is empty, uses the first paragraph of the body text to show a summary.

Useful for displaying excerpts of texts that do not always have a separate summary provided.

Example:

```
{{ id|summary }}
```

It is also possible to give a maximum length to which the text will be truncated:

```
{{ id|summary:80 }}
```

temporary_rsc

- Module: *mod_admin* (page 346)

Creates a temporary resource if its input value is not defined.

The created resource receives the properties of the second parameter.

After the resource is created, every hour a check is made if the resource has been edited and still registered in the Server Storage.

Note: *mod_server_storage* must be enabled for this filter to work.

If the resource is abandoned and not changed since its creation, then it is automatically deleted.

Only a single temporary resource can exist per category per session. Requesting a second resource of the same category for a session will return the earlier created resource. If the earlier resource has been updated, then a new resource is created.

This filter is used for situations where an edit form is needed but an intermediate dialog requesting for the title of the resource is unwanted.

Example:

```
{% with id|temporary_rsc:{props title=_ "New" category=`article`} as id %}
  {% wire id=#form type="submit" postback=`rscform` delegate=`controller_admin_
↪edit` %}
  <form id="{{ #form }}" action="postback">
    <input type="hidden" name="id" value="{{ id }}" />
    <input type="hidden" name="is_published" value="1" />
    <input type="text" name="title" value="{{ id.title }}" />
    ...
  </form>
{% endwith %}
```

See [mod_admin](#) (page 346), [mod_server_storage](#) (page 391)

Strings

append

- Module: [mod_base](#) (page 360)

See also:

[insert](#) (page 498)

Appends the argument to the value.

For example:

```
{{ value|append:" world" }}
```

When value is `hello` then the output will be `hello world`.

capfirst

- Module: [mod_base](#) (page 360)

See also:

[upper](#) (page 502)

Converts the first character of the value to uppercase.

For example:

```
{{ value|capfirst }}
```

When value is `hello world` then the output is `Hello world`.

At the moment this only works for the characters a through z. Accented characters (like ü) are not yet supported.

center

- Module: [mod_base](#) (page 360)

See also:

[rjust](#) (page 500), [ljust](#) (page 498)

Centers the value in a field of a certain width using spaces.

For example:

```
{{ value|center:7 }}
```

When value is `hello` then the output is `_hello_` (with spaces).

Centering only works for single byte character values. At this moment there is no support for multi-byte unicode characters.

insert

- Module: *mod_base* (page 360)

See also:

append (page 497)

Prepends the argument in front of the value.

For example:

```
{{ value|insert:"world " }}
```

When value is “hello” then the output will be “world hello”.

is_valid_email

- Module: *mod_email_status* (page 370)

Todo

Not yet documented.

ljust

- Module: *mod_base* (page 360)

See also:

rjust (page 500), *center* (page 497)

Justifies the value in a field of a certain width to the left, with spaces.

For example:

```
{{ value|ljust:7 }}
```

When value is `hello` then the output is `hello__` (with spaces).

Justifying only works for single byte character values. At this moment there is no support for multi-byte unicode characters.

log_format_stack

- Module: *mod_logging* (page 378)

Escapes and formats a Javascript string with a stack trace to readable HTML.

Used in the logging module to show the User-Interface errors with their stack traces.

lower

- Module: *mod_base* (page 360)

See also:

upper (page 502)

Translates the value to lower case.

For example:

```
{{ value|lower }}
```

When value is “Hello World” then the output is “hello world”.

Note: There is partial support for multi-byte unicode characters.

normalize_email

- Module: *mod_admin_identity* (page 353)

Normalize an email address, used in the identity management.

The email address is lowercased and trimmed.

For example:

```
{{ "ME@Example.Com " |normalize_email }}
```

Evaluates to the value `me@example.com`.

replace_args

- Module: *mod_base* (page 360)

Replace $\$N$ placeholders in string from a list of replacement values.

The input string is searched for any $\$N$ tags, which is replaced with the N ’th value from the list of arguments supplied to the filter.

Example usage:

```
{{ "Replace $1 and give it some $2." |replace_args:["item", "meaning"] }}
```

Will result in: “Replace item and give it some meaning.”

The $\$N$ tag may be escaped by `\` to avoid replacement.

N may be in the range [1..9]. If N is out of range of the provided args, the $\$N$ tag is left as-is.

The $\$N$ tags may come in any order, any number of times.

When replacing a single tag, the replacement argument may be given directly:

```
{{ "John is $1" |replace_args:"the one" }}
```

Outputs: “John is the one”

New in version 0.8.

rjust

- Module: *mod_base* (page 360)

See also:

ljust (page 498), *center* (page 497)

Justifies the value in a field of a certain width to the right, using spaces.

For example:

```
{{ value|rjust:7 }}
```

When value is `hello` then the output is `__hello` (with spaces).

Justifying only works for single byte character values. At this moment there is no support for multi-byte unicode characters.

stringify

- Module: *mod_base* (page 360)

See also:

slugify (page 468), *to_binary* (page 455)

Translates atoms, integers and floats to strings. The undefined value is translated to the empty string. Does not translate tuples.

For example:

```
{{ value|stringify }}
```

When *value* is undefined then the output will be `""`.

to_name

- Module: *mod_base* (page 360)

Map a string to a *name*. That is a lowercased string with only `[a-z0-9_]` characters.

Example:

```
{{ "Hello World!"|to_name }}
```

Results in the string value `hello_world`.

toc

- Module: *mod_base* (page 360)

Filter to derive a Table Of Contents from a HTML body.

This filter extracts a nested table of contents from the `h2..h6` elements in a HTML text.

The headers may not have any attributes (ie. only `<h2>`).

All sections are wrapped in `<div>` elements, this to make it possible to make the headers sticky without having them overlap.

Example usage:


```
{% with id.body|toc as toc, body %}
  {% include "page-parts/_toc.tpl" toc=toc %}
  {{ body|show_media }}
{% endwith %}
```

The returned ToC tree is a nested tree of 3-tuples:

```
[ {<<"toc1">>, <<"Text of header">>, [ ... ]}, ... ]
```

The third element is a list of sub-headers.

To generate a nested table of contents from the above structure:

```
{% include "page-parts/_toc.tpl" toc=toc %}
```

And then the page-parts/_toc.tpl as:

```
{% with level|default:1 as level %}
{% if toc %}
  <ol class="toc-level-{{ level }}">
    {% for p, text, sub in toc %}
      <li>
        <a href="#{{ p }}">{{ text }}</a>
        {% include "page-parts/_toc.tpl" toc=sub level=level+1 %}
      </li>
    {% endfor %}
  </ol>
{% endif %}
{% endwith %}
```

tokens

- Module: *mod_base* (page 360)

Returns a list of tokens from input string, separated by the characters in the filter argument.

This filter maps directly onto the `string:tokens/2` function in `stdlib` from the Erlang/OTP distribution.

Example:

```
{{ "abc defxxghix jkl"|tokens:"x " }}
```

Will give the list of tokens: ["abc", "def", "ghi", "jkl"].

translate

- Module: *mod_translation* (page 400)

See also:

translation (page 506)

Translates a (English) value to the current language or the given language.

If the input is a `#trans{}` record then it is extended with the translations from the `.po` files before the language lookup is done. For this the `#trans{}` record *must* have the English translation.

Example with the default language lookup (accessible via the template variable `z_language`):

```
{{ "Cancel"|translate }}
```

If the current language is `de` then the output is: "Abbrechen".

An example with a specific language:

```
{{ "Cancel"|translate:"nl" }}
```

The output would be "Annuleer".

trim

- Module: *mod_base* (page 360)

Removes whitespace at the start and end of a string.

For example:

```
{{ value|trim }}
```

When the value is " hello " then the output is "hello".

Internally, this calls `z_string:trim/1` to perform the trimming.

truncate

- Module: *mod_base* (page 360)

See also:

truncate_html (page 472)

Truncate a text to a maximum length.

The text is truncated to the maximum length specified with the argument. The text is always truncated at a word boundary. If the truncation is not after punctuation then the unicode ellipsis ... character is appended.

For example:

```
{{ value|truncate:8 }}
```

If the value is `hello world.` then the output is `hello...`

Entities like `&` are counted as a single character.

This filter is multibyte aware: Multi-byte UTF-8 characters are assumed to be non-breaking characters.

Truncating character

An optional second argument defines which text will be added if the text is truncated:

```
{{ value|truncate:8:" (more)" }}
```

If the value is `hello world.` then the output is `hello (more).`

upper

- Module: *mod_base* (page 360)

See also:

capfirst (page 497), *lower* (page 499)

Translates the value to upper case.

For example:

```
{{ value|upper }}
```

When value is “Hello World” then the output is “HELLO WORLD”.

Note: There is partial support for multi-byte unicode characters.

Survey

survey_answer_split

- Module: *mod_survey* (page 398)

Todo

Not yet documented.

survey_any_correct_answer

- Module: *mod_survey* (page 398)

Determine if any answer of a question was correct. Used for showing results of thurstone test questions.

survey_any_wrong_answer

- Module: *mod_survey* (page 398)

Determine if any answer of a question was wrong. Used for showing results of thurstone test questions.

survey_as_pages

- Module: *mod_survey* (page 398)

Split the page blocks into pages, prepare them for easy display in the survey question editor.

See *mod_survey* (page 398)

Todo

Not yet documented.

survey_is_stop

- Module: *mod_survey* (page 398)

Check if there is a ‘stop’ question in list of (survey) blocks

See *mod_survey* (page 398)

Todo

Not yet documented.

survey_is_submit

- Module: *mod_survey* (page 398)

Todo

Not yet documented.

survey_prepare_matching

- Module: *mod_survey* (page 398)

Todo

Not yet documented.

survey_prepare_narrative

- Module: *mod_survey* (page 398)

Todo

Not yet documented.

survey_prepare_thurstone

- Module: *mod_survey* (page 398)

Todo

Not yet documented.

survey_test_max_points

- Module: *mod_survey* (page 398)

Counts the total of all points that can be received for all *test* questions. Non *test* questions are not counted.

Usage:

```
{{ survey_id|survey_test_max_points }}
```

Return the number of points if all questions are corectly answered.

See *mod_survey* (page 398)

Translation

is_rtl

- Module: *mod_translation* (page 400)

See also:

[language_dir](#) (page 505), [language](#) (page 505)

Check if the given language is a rtl or ltr language.

Example:

```
{% if z_language|is_rtl %}
    You are browsing in an RTL language
{% endif %}
```

It currently returns `true` only for Arabic (`ar`), Farsi (`fa`) and Hebrew (`he`).

language

- Module: [mod_translation](#) (page 400)

See also:

[language_dir](#) (page 505), [is_rtl](#) (page 504)

Return the language the resource (or translated text) will be displayed in.

Example:

```
{{ id|language }}
```

The languages of the resource will be fetched and using the currently selected interface language (variable `z_language`) the language for the resource to be displayed in will be returned.

language_dir

- Module: [mod_translation](#) (page 400)

Return `rtl` or `ltr` depening on the direction of the language.

Example:

```
<div dir="{{ z_language|language_dir }}">
    { _ Text _ }
</div>
```

Input can also be an resource (page) id. In that case the language that the resource is shown in will be used to determine the direction.

Example:

```
<div lang="{{ id|language }}" dir="{{ id|language_dir }}">
    {{ id.body }}
</div>
```

It currently returns `rtl` for Arabic (`ar`), Farsi (`fa`) and Hebrew (`he`).

See also:

[language](#) (page 505), [is_rtl](#) (page 504)

language_sort

- Module: [mod_translation](#) (page 400)

Sort a list of language codes or map with languages on their name. Return a list of {Code, LanguageProps} pairs.

LanguageProps is a map:

```
#{
  fallback => [],
  language => <<"nl">>,
  language_atom => nl,
  name => <<"Nederlands">>,
  name_en => <<"Dutch">>,
  sort_key => <<"dutch">>,
  sublanguages => [
  ]
}
```

trans_filter_filled

- Module: *mod_base* (page 360)

Filters all empty translations from a property.

This is used if it is important to show a text, but not all translations are filled in.

The filter takes as input a resource or other variable and as argument the property to be shown.

Example usage:

```
{{ id|trans_filter_filled:`body` }}
```

If the resource `id` has the `body` property:

```
{trans, [{en, <<>>}, {nl, <<"Hallo">>}]}
```

Then this will show `Hallo`, even if the language is set to `en`.

translation

- Module: *mod_base* (page 360)

See also:

translate (page 501)

Lookup a specific translation in a translated text. If the text is not translated then the text is returned as-is.

The filter takes as input a value or other variable and as argument the language to be shown.

Example usage:

```
{{ text|translation:"en" }}
```

If the variable `text` has the value:

```
{trans, [{en, <<"Hello">>}, {nl, <<"Hallo">>}]}
```

Then this will show `Hello`, even if the current template language is set to `nl`.

This filter is especially useful for filling in forms with language specific strings to be edited.

Tuples

element

- Module: *mod_base* (page 360)

See also:

before (page 474), *after* (page 474)

Select an element from a tuple or list of tuples.

For example:

```
{{ value|element:1 }}
```

When value is a list of tuples `[[312, 0.34], [200, 0.81]]` then the output is the list `[312, 200]`.

When value is just a tuple, `{123, 22, 11}`, the output of `|element:1` is `123`.

URLs and links

is_site_url

- Module: *mod_base* (page 360)

See also:

sanitize_url (page 472), *url_abs* (page 508), *url* (page 507), *urlencode* (page 469)

Test if the given URL is a url for the current site.

If the current site handles requests for the hostname `example.com`, then all of the following expressions will echo `true`:

```
{{ "https://example.com"|is_site_url }}
{{ "#foo"|is_site_url }}
{{ "/page/path"|is_site_url }}
{{ "//example.com"|is_site_url }}
```

The following will echo `false`:

```
{{ "https://foo.test"|is_site_url }}
{{ "example.com"|is_site_url }}
{{ "//foo.test"|is_site_url }}
```

url

- Module: *mod_base* (page 360)

See also:

url_abs (page 508), *url* (page 532), *sanitize_url* (page 472), *is_site_url* (page 507), *urlencode* (page 469)

Generates the relative URL for the given dispatch information.

An *relative URL* is an URL that excludes the protocol and hostname. For example `/foo/bar`.

For example, generate a url for the dispatch rule `home` with an extra argument `hello`:

```
{{ {home hello="world"}|url }}
```

This is similar to:

```
{% url home hello="world" %}
```

Difference between the tag and the filter is that the filter can be used in expressions or with passed values.

url_abs

- Module: *mod_base* (page 360)

See also:

url (page 507), *url* (page 532), *sanitize_url* (page 472), *is_site_url* (page 507), *urlencode* (page 469)

Generates an absolute URL for the given dispatch information.

An *absolute URL* is an URL that includes the protocol and hostname. For example `https://example.com/foo/bar`.

For example, generate a url for the dispatch rule `home` with an extra argument `hello`:

```
{{ {home hello="world"}|url_abs }}
```

This is similar to:

```
{% url_abs home hello="world" %}
```

Difference between the tag and the filter is that the filter can be used in expressions or with passed values.

urlencode

- Module: *mod_base* (page 360)

See also:

sanitize_url (page 472), *is_site_url* (page 507), *url_abs* (page 508), *url* (page 507), *urlencode* (page 469)

Decode a text where characters are encoded as URL-safe characters.

Translates all percent encoded characters back to their original encoding.

For example:

```
{{ value|urlencode }}
```

When value is “`msg%3DHello%26World`” then the output is “`msg=Hello&World`”.

parse_url

- Module: *mod_base* (page 360)

Parses an URL (URI) using `uri_string:parse/1`.

Example:

```
{% print "https://example.com:8443/foo?a=b#test"|parse_url %}
```

Gives:

```
{#fragment => <<"test">>, host => <<"example.com">>,  
  path => <<"/foo">>, port => 8443, query => <<"a=b">>,  
  scheme => <<"https">>}
```


Variables

as_atom

- Module: *mod_base* (page 360)

Convert a value to an Erlang atom.

Atoms are represented in the template language as strings between backticks, ``like this``. Some template function require their arguments to be atoms, and this filter helps in converting any value to such an atom:

```
{{ "foobar"|as_atom }}
```

evaluates to the atom `foobar`.

default

- Module: *mod_base* (page 360)

See also:

if (page 509), *is_defined* (page 510), *is_undefined* (page 510), *if_undefined* (page 510)

Provide an alternative value in case a value has a falsy value (0, false, undefined or empty string).

For example:

```
{{ value|default:1 }}
```

if

- Module: *mod_base* (page 360)

See also:

if (page 518), *if_undefined* (page 510)

Selects an argument depending on a condition.

For example:

```
{{ value|if:"yes":"no" }}
```

This is a shortcut for using the *if* (page 518) tag. The same can be expressed as follows:

```
{% if value %}yes{% else %}no{% endif %}
```

Note that falsy values (0, false, undefined or empty string) evaluate to false.

Elaborate examples

```
{% with is_i18n|if:r.translation[lang_code].body:r.body as body %}
```

So if `is_i18n` evaluates to true, `body` is assigned to `r.translation[lang_code].body`, else to `r.body`.

```
{% include "_language_attrs.tpl" id=pid class=(pid==id)|if:"active":"" %}
```

Add parameter `class` to the included template; when `pid` equals `id`, `class` is `"active"`, otherwise an empty string.

if_undefined

- Module: *mod_base* (page 360)

See also:

is_defined (page 510), *is_undefined* (page 510), *if* (page 509)

Tests whether a value is undefined, returning the given argument.

Whereas the *!default* filter also falls back to the default value when a value is an empty string or `false`, this filter *only* falls back to its value when the input value is the Erlang `undefined` atom.

This can be used for setting values which default to true if they are never set.

For example:

```
{% if value|if_undefined:true %}The value is true or undefined{% endif %}
```

If the value is undefined, the output will be “The value is true or undefined”.

is_defined

- Module: *mod_base* (page 360)

See also:

is_undefined (page 510), *if_undefined* (page 510), *if* (page 509)

Tests if a value is defined.

Checks if the value is not empty and outputs a boolean true or false. This is useful in combination with the *if* (page 518) tag.

For example:

```
{% if value|is_defined %}The value was defined{% endif %}
```

When the value is “foo” then the output “The value was defined”.

is_undefined

- Module: *mod_base* (page 360)

See also:

is_defined (page 510), *if_undefined* (page 510), *if* (page 509)

Tests if a value is undefined.

Checks if the value is empty and outputs a boolean true or false. This is useful in combination with the *if* (page 518) tag.

For example:

```
{% if value[1]|is_undefined %}The first elemeent of value was undefined{% endif %}
```

If the value is `[]` then the output is The first elemeent of value was undefined.

make_value

- Module: *mod_base* (page 360)

Todo

Not yet documented.

pprint

- Module: *mod_base* (page 360)

See also:

print (page 530)

Pretty print a zotonic value in a template.

Pretty printing a zotonic value in a template is handy during development. It outputs the value of an erlang variable in Html.

Usage:

```
{{ value | pprint }}
```

This output is similar to the *print* (page 530) tag, only are the values of the pprint filter not wrapped in <pre> tag.

to_json

- Module: *mod_base* (page 360)

Display any value as in JSON (JavaScript Object Notation).

For example:

```
{{ [1,2,3] | to_json }}
```

Converts this list to a valid JSON UTF-8 encoded string which can be directly used in JavaScript calls.

Another example for a JSON object:

```
{{ %{ a: 1, b: 2 } | to_json }}
```

6.1.5 Tags

With *tags* (page 29) you add logic and flexibility to your templates.

Built-in tags

The *tags* (page 29) that are included with Zotonic.

all catinclude**See also:**

tags *catinclude* (page 515) and *all include* (page 512).

Include a template for all a resource's categories from all modules.

This is an extension on the *catinclude* (page 515) tag. It will include all templates with the given name, instead of the first one found. Templates are defined in modules, because of that multiple modules can define a template with the same name.

This `catinclude` extension will include all available templates in the same order as defined in [catinclude](#) (page 515). Where the templates per category will be rendered in the order of their module's defined priority. This is the order in which they are listed in the module admin page.

Examples of this mechanism can be found in [mod_admin](#) (page 346), for example the main menu and the category specific editing fields on the edit page.

An example usage:

```
{% all catinclude "hello.tpl" id arg="val" %}
```

Includes all templates with the base name *hello.tpl* for the `id`'s category hierarchy.

For example, in the case of a *news* article:

- all templates with the name *.hello.news.tpl*.
- all templates with the name *.hello.article.tpl*.
- all templates with the name *.hello.text.tpl*.
- all templates with the name *.hello.tpl*.

all include

See also:

tag [include](#) (page 526).

Call all modules to include a certain template.

- `{% all include "foobar.tpl" %}` will include all templates named *foobar.tpl*.
- `{% include "foobar.tpl" %}` only includes the highest priority *foobar.tpl*.

Exactly the same [module priority](#) (page 63) is also valid for all files in the `lib/` directory of modules.

This allows any module to change the static css, javascript, images, *favicon.ico*, *robots.txt* and other static files with its own version.

This is an extension on the [include](#) (page 526) tag. It will include all templates with the given name, instead of the first one found. Templates are defined in modules, because of that multiple modules can define a template with the same name.

For example when you have two modules (*mod_a* and *mod_b*), both with the template *_name.tpl*. When the template in *mod_a* is defined as:

```
this is mod_a's {{ hello }}
```

and in *mod_b* as:

```
this is mod_b's {{ hello }}
```

then the tag:

```
{% all include "_name.tpl" hello="world" %}
```

Will output:

```
this is mod_a's world  
this is mod_b's world
```

The modules will be called in the order of their defined priority. This is the order in which they are listed in the module admin page.

Examples of this mechanism can be found in [mod_admin](#) (page 346), for example the main menu and the category specific editing fields on the edit page.

Another example is the `_html_head.tpl` template which is included from the template-base template and allows all modules to add HTML to the head of a generated HTML page.

autoescape

Automatically apply HTML escaping to values.

The `autoescape` tag controls the current auto-escaping behavior. This tag takes either *on* or *off* as an argument and that determines whether auto-escaping is in effect inside the block.

When auto-escaping is in effect, all variable content has HTML escaping applied to it before placing the result into the output (but after any filters have been applied). This is equivalent to manually applying the escape filter to each variable.

Example:

```
{{ value }}
{% autoescape on %}
  {{ value }}
{% endautoescape %}
```

When the variable value contains `Hello` then this will output:

```
<b>Hello</b>
&lt;b&gt;Hello&lt;/b&gt;
```

block

See also:

[extends](#) (page 516) and [overrides](#) (page 530).

Define a block in a template and overrules a block from an inherited template.

The `block` tag is used for replacing blocks in inherited templates.

For example, when we have a template `base.tpl`, in which we define a block called *name*:

```
Hello {% block name %}my{% endblock %} world.
```

And we define a second template, `page.tpl`, which [extends](#) (page 516) the first template:

```
{% extends "base.tpl" %}
{% block name %}Peter's{% endblock %}
```

Then the result of rendering `page.tpl` will be:

```
Hello Peter's world.
```

If we do not include the block definition, so `page.tpl` just contains the [extends](#) (page 516) tag:

```
{% extends "base.tpl" %}
```

then the output will be:

```
Hello my world.
```

The name of a block must be a valid identifier, consisting of alphanumeric characters (a-z, 0-9) and the underscore character.

cache

Cache a frequently used block for later reuse.

Cache the output of the enclosed block. The block can be named. Cache blocks with the same name will use each others cached entries, when the cache entry is still valid.

Example:

```
{% cache 3600 now %}Local time is {% now "Y-m-d H:i:s" %}{% endcache %}
```

This caches the output of the block for an hour. The name for the cache is “now”.

The cache duration and name are optional. The default cache duration is 0 (zero) seconds, which gives parallel rendering protection (see below) though will not store the result in the cache.

The cache tag protects against the situation where parallel requests need to render the same block at the same time. Instead of all the processes rendering the block in parallel, one process will render the block and share the result with the other—waiting—processes.

Example, prevent parallel rendering (slam dunk) by non logged on visitors:

```
{% cache if_anonymous %} insert complicated page here {% endcache %}
```

Besides the duration and the cache name the `{% cache %}` tag also accepts the following arguments:

Argument	Description	Example
vary	This argument can be used multiple times. Cache blocks with different vary arguments have different cache keys. The arguments are assumed to be keys in the cache. When one of the vary keys is updated or invalidated then the cached block will be invalidated.	vary=myid
cat	Category the cached block depends on. This argument can be used multiple times for specifying multiple categories. The categories are not added to the cache key, only added as cache dependencies.	cat="news"
if	Only cache the block when the argument evaluates to true.	if=can_cache
if_anonymous	Only cache the block when the current visitor is not logged on (i.e. an anonymous visitor)	if_anonymous
visible_for	Sets the access control user for rendering the block. With this you can force to only show public items for logged on users. Valid values are “user”, 3, “group”, 2, “community”, 1, “world”, “public”, 0	visible_for="world"

The cache tag can be disabled by setting the config key `mod_development.nocache`. This can be done on the `/admin/development` page.

call

Call an Erlang function.

The `{% call %}` tag is used to call the `render/3` function of the module specified by the argument.

For example:

```
{% call mymodule a=1 b=2 %}
```

Will call `mymodule:render([a,1],[b,2], TemplateVariables, Context)`. Where *TemplateVariables* is the map with all template variables and *Context* is the current Zotonic request *context*.

The `render/2` function must return either `{ok, IoList}` or `{error, Reason}`.

If `{error, Reason}` is returned then the Reason is rendered as `io_lib:format("~p", [Reason])`

For compatibility with Django it is possible to pass a single value instead an argument list:

```
{% call mymodule with value %}
```

This will call `mymodule:render(Value, TemplateVariables, Context)`.

catinclude

See also:

all catinclude (page 511), which is useful to include multiple templates.

Include another template based on the category of a resource. The include tag is replaced with the contents of the included template file. You can give arguments to the included template, they will be assigned as variables in the context of the included template.

Example:

```
{% catinclude "hello.tpl" id %}
```

Assuming that the resource whose id is the value of the template variable *id*, and that the unique name property of the resource is *my_page_name* is a news article then *catinclude* will consider the following templates:

```
hello.name.my_page_name.tpl
hello.news.tpl
hello.article.tpl
hello.text.tpl
hello.tpl
```

This because *news* is a subcategory of *article*, which is a subcategory of *text*. When one of the previous templates is not found then the base template *hello.tpl* is tried. The *catinclude* tag will only include the first file it finds, and stops after having found a file to include.

If the resource has a unique name (the *name* property is set), this property is also considered for the *catinclude* lookup, before the category-based template names. If the resource has *name* set to *foobar*, it will first look for *hello.name.foobar.tpl*, then for *hello.news.tpl*, etc.

The tag accepts extra arguments, which will be passed as template variables to the included template. The inclusion is always done at runtime, because the selected template depends on the category of the referenced resource.

The resource id will be available in the included template as the variable *id*.

Instead of passing an id, you can also pass in a list of category names which are to be search for. These names need to be atoms or strings, for example:

```
{% catinclude "hello.tpl" [ `text`, `article` ] %}
```

This will search for the following templates, in order:

```
hello.article.tpl
hello.text.tpl
hello.tpl
```

Note the search order is reversed from the list order, you should add the *most specific selector last*!

See the *include* (page 526) for caching options and argument handling.

comment

Ignore part of a template.

Everything inside a `{% comment %}` block is not output.

Example:

```
This will show.  
{% comment %} And this will not show {% endcomment %}
```

This will output:

```
This will show.
```

An alternative to the `{% comment %}` tag is to use the `{# ... #}` construct:

```
This will show.  
{# And this will not show #}
```

The big advantage of this notation is that the contents of the `{# ... #}` construct don't need to be grammatically correct, as they will not be parsed. The contents of a `{% comment %}` block must be correct as they will be parsed by the template compiler.

cycle

Rotate through a list of values.

Rotates through a list of values and outputs them. `{% cycle %}` Is used within a `{% for %}` loop.

Note: You can not apply filters to the cycle values.

Example:

```
{% for a in [1,2,3,4] %}{% cycle "bleu" "blanc" "rouge" %} {% endfor %}
```

Will output “bleu blanc rouge bleu”.

The list values can be either a string literal, a number literal, a variable or an automatic id (`#name`).

extends

See also:

[block](#) (page 513), [inherit](#) (page 527) and [overrides](#) (page 530).

Inherit markup from another template.

Note: A template that extends another template contains only the extends tag and [block](#) (page 513) tags.

Signal that this template extends another template. The extends tag must be the first tag in a template that inherits from another template.

Example:

```
{% extends "base.tpl" %}
```

All named blocks in this template will replace the similar named blocks in the template *base.tpl*.

Unlike Django the template name must be a string literal, variables are not allowed.

filter

Filter the contents of a block through variable filters.

Filters can also be piped through to each other.

Example:

```
{% filter escape | lower %}
The text will be lowered and escaped. So you can use <, > and & without any_
↳problems.
{% endfilter %}
```

New in version 0.8.

firstof

Not implemented, but exists in Zotonic for forward compatibility with future ErlyDTL and Django versions.

for

Loop over multiple values in a list or search result.

`{% for %}` loops over a list of values. For example to loop over a list of colors:

```
{% for color in ["bleu", "blanc", "rouge"] %}{{ color }}{% endfor %}
```

This will output “bleublancrouge”.

Or to show the latest ten news article titles:

```
{% for id in m.search[{{latest cat="news" pagelen=10}}] %}
  {{ m.rsc[id].title }}
{% endfor %}
```

The argument list can also contain tuples or sublists, in that case you can assign all parts of the list items to different variables at once:

```
{% for nr,color in [ [1,"bleu"], [2,"blanc"], [3,"rouge"] ] %}
  {{ nr }}: {{ color }}
{% endfor %}
```

This will output:

```
1: bleu
2: blanc
3: rouge
```

The for loop sets a number of variables available in the loop:

Variable	Description
forloop.counter	The current iteration of the loop (1-indexed)
forloop.counter0	The current iteration of the loop (0-indexed)
forloop.revcounter	The number of iterations from the end of the loop (1-indexed)
forloop.revcounter0	The number of iterations from the end of the loop (0-indexed)
forloop.first	True if this is the first time through the loop
forloop.last	True if this is the last time through the loop
forloop.parentloop	For nested loops, this is the loop “above” the current one

For ... empty

`{% for %}` can take an optional `{% empty %}` clause that will be displayed if the given list is empty.

For example:

```
{% for id in m.search[latest cat="news" pagelen=10] %}
  {{ m.rsc[id].title }}
{% empty %}
  Sorry, there is no news.
{% endfor %}
```

This will output “Sorry, there is no news.” when the “latest” search did not find any pages within the category “news”.

if

See also:

ifequal (page 519) and *ifnotequal* (page 520).

Show a block if the condition is true.

The `{% if %}` tag evaluates an expression and if the result is true (boolean true, number unequal to zero, non empty string or a non empty list) then the contents of the if-block are output.

Note: Besides the `{% elif %}` tag we also support the alias `{% elseif %}`.

If the if-test fails then the optional `{% elif %}` blocks are evaluated. If both the if-test and all elif-tests fail, then the `{% else %}` block contents are output.

Example:

```
{% if genre == "pop" %}
  Popular music.
{% elif genre == "classical" %}
  Classical music.
{% elif genre == "jazz" %}
  Jazz
{% else %}
  The genre isn't pop, classical or jazz.
{% endif %}
```

An `{% if %}` and `{% elif %}` tag can have an “and” or “or” expression as argument:

```
{% if person_list and show_persons and full_moon %}
  There are persons that we can show during full moon.
{% endif %}
```

Or for example:

```
{% if new_moon or daytime %} Guess you can't see the moon. {% endif %}
```

It is also possible to mix “and” and “or” in one expression, so this is a valid:

```
{% if full_moon or daytime and cloudy %}
```

The evaluation order is: and, xor, and then or. Left to right associative. So the above is equivalent to:

```
{% if full_moon or (daytime and cloudy) %}
```

The “not” operator can be used to negate a boolean value:

```
{% if full_moon or daytime or not clearsky %}
```

if-with

The `if` is often combined with the `with` tag. For example:

```
{% with m.search[{{latest cat='news' pagelen=10}}] as result %}
  {% if result %}
    <h3>{_ Latest news _}</h3>
    <ul>
      {% for id in result %}
        <li><a href="{{ id.page_url }}">{{ id.title }}</a></li>
      {% endfor %}
    </ul>
  {% endif %}
{% endwith %}
```

To make this easier it is possible to combine the `if` and `with` tags in a single expression:

```
{% if m.search[{{latest cat='news' pagelen=10}}] as result %}
  <h3>{_ Latest news _}</h3>
  <ul>
    {% for id in result %}
      <li><a href="{{ id.page_url }}">{{ id.title }}</a></li>
    {% endfor %}
  </ul>
{% endif %}
```

The `as` can also be used in the `elif` expressions:

```
{% if expression1 as x %}
  ...
{% elif expression2 as y %}
  ...
{% else %}
  ...
{% endif %}
```

ifchanged

Not implemented, but exists in Zotonic for forward compatibility with future ErlyDTL and Django versions.

ifequal

See also:

if (page 518) and *ifnotequal* (page 520).

Show a block when two values are equal.

The `{% ifequal %}` tag tests if its two arguments are equal. If so then the contents of the `{% ifequal %}` block are output, otherwise the contents of the optional `{% else %}` block are output.

For example:

```
{% ifequal value 5 %}
  Value is equal to 5.
{% else %}
  Value is {{ value }} which is not 5.
{% endifequal %}
```

It is only possible to compare arguments that are variables (with optional filters) or constants. Examples of constants are numbers, strings or lists.

ifnotequal

See also:

if (page 518) and *ifequal* (page 519).

Show a block when two values are not equal.

The `{% ifnotequal %}` tag tests if its two arguments are unequal. If so then the contents of the `{% ifnotequal %}` block are output, otherwise the contents of the optional `{% else %}` block are output.

For example:

```
{% ifnotequal value 5 %}
  Value is {{ value }} which is not 5.
{% else %}
  Value is 5.
{% endifnotequal %}
```

It is only possible to compare arguments that are variables (with optional filters) or constants. Examples of constants are numbers, strings or lists.

image

See also:

Media (page 32) developer guide.

See also:

Media classes (page 33) for some options that are only available in *mediaclass* files.

See also:

image_url (page 526), *image_data_url* (page 525) and *media* (page 529) tags.

Show a still image using an `` element. The image will be automatically resized to the desired size and filters. For video, use the *media* (page 529) tag instead.

For example:

```
{% image "cow.jpg" width=200 height=200 crop %}
```

This will generate an image tag for the image “cow.jpg” (in the files/archive directory) of 200x200 pixels. The image will be resized and cropped to the requested size. The image tag will be something like (the checksum will vary per sign key):

```

```

The file argument can be one of the following:

- filename relative to the archive folder (“cow.jpg” is always present)
- resource id of a resource with attached file (mostly of the category “media”)
- property list of a resource’s medium record.

So, to render a resource’s depiction:

```
{% image id width=200 height=200 %}
```

Please note that even if you supply no arguments, the image will be processed to be scaled, based on the *quality* (page 524) argument’s default value.

Arguments

width

The maximum width of the image. Example:

```
width=200
```

height

The maximum height of the image. Example:

```
height=200
```

nowh

Do not generate the width and height attributes. Per default the image width and height are calculated and added to the generated `img` tag. This option prevents the addition of width and height attributes.

mediaclass

The *media class* (page 33) of the image. Example:

```
mediaclass="thumb"
```

The mediaclass is added as a CSS class to the generated image, prefixed with `mediaclass-`. For example:

```
{% img mediaclass="large" %}
```

Might become:

```

```

background

The background color for transparent image parts, specified as ImageMagick colors. Example:

```
background="white"
```

removebg

Removes the image background. Accepts an optional fuzziness parameter (from 0 to 100). Examples:

```
removebg
removebg=50
```

Optionally a background color can be given as well:

```
removebg="black,50"
```

blur

Blur the image, making it less sharp. See ImageMagick blur for valid argument values. Example:

```
blur="20x8"
```

rotate3d

Rotate the image in three dimensions: roll, tilt and pan. The size of the original image and canvas is maintained. This results in clipping at the edges.

rotate3d is useful for straightening photos of flag objects that are taken under an angle.

The rotate3d accepts three arguments:

```
rotate3d=[ 1, 2, -3 ]
```

This rotates the image:

- Roll 1 degree clockwise around the Z axis (like css rotateZ).
- Tilt 2 degrees clockwise around the X axis (like css rotateX)
- Pan 3 degrees counter clockwise around the Y axis (like css rotateY)

The center of the rotation is at the center of the image.

rotate3d, rotate and cropp are applied before other operations.

rotate

Rotate the image by a multiple of 90 degrees. The image size is adjusted according to the rotation:

```
rotate=90
```

This will rotate the image 90 degrees clockwise. Use a negative number to rotate counter clockwise.

Acceptable values are: 0, 90, 180, 270, -90, -180, and -270

rotate3d, rotate and cropp are applied before other operations.

cropp

Crop percentages from the sides of an image. The image is cropped by a percentage of the width and/or height. The crop's argument is a list of numbers, in the order: left, right, top, bottom:

```
cropp=[ 10, 15, 20, 30 ]
```

The example above crops 10% from the left side, 15% from the right, 20% from the top, and 30% from the bottom. The resulting image will be 65% of the original width and 50% of the original height.

rotate3d, rotate and cropp are applied before other operations.

crop

Crop the image. The resulting image will have the exact width and height as described in the width and height arguments (see above).

The `crop` argument determines the cropping center. It either has the form `+x+y` (a set of coordinates in the image) or one of `north`, `north_east`, `east`, `south_east`, `south`, `south_west`, `west`, `north_west` and `center` (the default). To define the cropping in your template:

```
crop="south"
crop="+100+100"
crop=[100, 100]
```

The cropping center can also be determined by editors on the media item's admin page (using [mod_image_edit](#) (page 376)). Without any argument, the image will be cropped around the user-defined cropping center:

```
crop
```

The coordinate of the cropping center is relative to the original image, before rotate and crop operations.

If [mod_media_exif](#) (page 380) and [mod_image_edit](#) (page 376) are enabled then the focal point information of the image is taken as the cropping center for automatic cropping.

extent

Add whitespace around the image until it fits the requested dimensions. Resize the image so that it fits inside the width/height box, then extend the image with a white background.

upscale

Forces the image to scale up to the requested dimensions.

flip

Mirror left and right sides of the image.

flop

Mirror the top and bottom of the image.

grey

Make the image greyscale.

brightness

Change the brightness of an image. A percentage in the range of -100 .. 100(%). Negative values darken the image, positive brighten the image. This applies a linear multiplier to the input image, similar to the css brighten filter. Defaults to 0, no change.

contrast

Change the contrast of an image. A percentage in the range of -100 .. 100(%). Negative values decrease the contrast of the image, positive values increase the contrast. A value of -100 results in a gray image. Defaults to 0, no change.

lossless

Controls whether resized image should become JPEG (`lossless=false`) or PNG/GIF images (`lossless=true`). When set to `auto`, PNG and GIF images will stay PNG/GIF images after resizing.

This protects PNG/GIF clip art and logos from being encoded as JPEGs and becoming blurry.

Defaults to `false`. Examples:

```
lossless=true
lossless=`auto`
lossless=false
```

interlace

Make a progressive JPEG or interlaced PNG/GIF image.

Use `line` or `plane` to create an interlaced PNG or GIF or progressive JPEG image.

`line` uses scanline interlacing (RRR...GGG...BBB...RRR...GGG...BBB...), and `plane` uses plane interlacing (RRRRRR...GGGGGG...BBBBBB...).

For JPEG images this is per default set to `plane`. To disable set to `none`.

Example:

```
interlace=plane
```

mono

Make the image black and white.

quality

Set the quality of the resulting JPEG. An integer between 0 and 100, where 100 is best quality. The default quality is inversely proportional to the output image resolution: higher-resolution images still look good even with a limited quality. Note that images smaller than 400x400 are sharpened before JPEG compression.

Example:

```
quality=70
```

link

Add a `<a>` tag around the generated `` tag. The destination depends on the value given.

Possible values:

- `none`; links to the image page itself
- an integer: to the page with that id
- any other value: assumed to be a URL.

alt

The text for the `alt="..."` attribute of the ``. Example:

```
alt="A nice image"
```

class

The text for the `class="..."` attribute of the ``. Example:

```
class="figure"
```

format

Use `webp` to produce an image in WEBP format. This reduces the size of images about about 30%. When used with `lossless` set to `auto` it uses lossless compression when the input image is in gif or png format. Note: `webp` is currently the only possible option. Example:

```
format=`webp`
```

absolute_url

Ensure that the generated URL contains the *hostname and port* (page 533).

image_data_url

See also:

Media (page 32) developer guide.

See also:

Media classes (page 33) for some options that are only available in *mediaclass* files.

See also:

image (page 520), *image_url* (page 526) and *media* (page 529) tags.

Generate a `data:` url of a still image.

The `{% image_data_url %}` tag is used generate a data url with the image data.

`{% image_data_url %}` accepts all parameters of the `{% image %}` tag but only outputs a data url and not the `` element to display it.

Example:

```
{% image_data_url "lib/images/trans.gif" %}
```

Generates:

```
data:image/gif;base64,R0lGODlhAQABAJAAAAAAAAACH5BAEUAAALAAAAAAAAABAEAAACRAEAOw==
```

The `image_data_url` tag can be used in image tags or in css:

```

```

image_url

See also:

Media (page 32) developer guide.

See also:

Media classes (page 33) for some options that are only available in *mediaclass* files.

See also:

image (page 520), *image_data_url* (page 525) and *media* (page 529) tags.

Generate the url of a still image.

The `{% image_url %}` tag is used generate the url of an image. `{% image_url %}` accepts all parameters of the `{% image %}` tag but only outputs the url and not the `` element to display it.

include

See also:

all include (page 512) and *catinclude* (page 515).

Include another template. The include tag is replaced with the contents of the included template file. You can give arguments to the included template, they will be assigned as variables in the context of the included template.

Note: For compatibility with DTL we accept the optional *with* keyword between the template name and the arguments:

```
{% include "_hello.tpl" with name="Peter" %}
```

Example:

```
{% include "_hello.tpl" name="Peter" %} world.
```

If *_hello.tpl* contains the text:

```
Hello {{ name }}'s
```

Then this will output the text Hello Peter's world..

If the template name is a string literal then the template will be inlined. When it is an expression then the template will be included during runtime.

Optional

If the included template is not required, a *optional* keyword may be used:

```
{% optional include "might-not-exist.tpl" %}
```

Unique ids

Automatically generated ids (`{{ #name }}`) are unique within an included template and do not clash with similarly named ids in the including template.

Caching of the included template

See also:

[cache](#) (page 514)

The output of the included template can be cached. This is useful when rendering the template takes considerable time, for example when the template shows a list of recent news items, which comprises a query, fetching and rendering a list of news items. To cache such a list:

```
{% include "recent_news_items.tpl" max_age=3600 %}
```

Caching is enabled by defining one of the caching arguments:

Argument	Description	Example
max_age	The maximum time the output can be cached, in seconds. Specifying 0 for the maximum age does not cache the output but does protect against slamunks, multiple requests rendering the same template at the same time will share the output of the rendering.	max_age=3600
vary	Dependency keys for the cached output. If a cache key with the same name is flushed or invalidated then the cached output of this template is also invalidated. You can use category names here.	vary="news"
sudo	If supplied then access control is disabled whilst rendering the included template. This will show any content not visible for the current user. Use with care.	sudo
anondo	Include the template as an anonymous user. Any <i>ACL checks</i> (page 555) in the included template will be executed as if the visitor was not logged in.	anondo
runtime	If supplied then the included template is not inlined but included during evaluation of the calling template. Only the supplied arguments are available as variables in the included template.	runtime

inherit

See also:

[block](#) (page 513), [extends](#) (page 516) and [overrides](#) (page 530).

Include the markup of an extended template into the extending template.

Say you have a template `hello.tpl` containing:

Listing 6.2: templates/hello.tpl

```
{% block test %}
This is content from hello.tpl
{% endblock %}
```

And in your site you have a `world.tpl` template, defined as:

Listing 6.3: templates/world.tpl

```
{% extends "hello.tpl" %}
{% block test %}
    First line
    {% inherit %}
    This is more content from world.tpl
{% endblock %}
```

Then, the result of rendering the template `world.tpl` will be:

```
First line
This is content from hello.tpl
This is more content from world.tpl
```

javascript

Adds javascript that will be run after jQuery has been initialized. In dynamic content it will run after the DOM has been updated with the template where the javascript was defined.

Example:

```
{% javascript %}
...
{% endjavascript %}
```

lib

See also:

[*mod_development*](#) (page 364).

Combine css and javascript includes in a single request.

Generates a `<link />` or `<script />` element for including the given libraries. This combines all css and javascript files in one request, saving multiple roundtrips to the server.

Example:

```
{% lib
    "css/zp-compressed.css"
    "css/zp-admin.css"
    "css/zp-wysiwyg.css"
    "css/zp-dialog.css"
    "css/zp-formreplace.css"
    "css/zp-finder.css"
    "css/zp-growl.css"
%}
```

Will output:

```
<link href="/lib/css/zp-compressed~zp-admin~zp-wysiwyg~zp-dialog~zp-formreplace~zp-
finder~zp-growl~63417066183.css" type="text/css" media="all" rel="stylesheet" />
```

The number at the end is the Unix modification time of the most recently changed file.

The *lib* tag supports optional arguments to control the resulting html tag. The arguments are supplied after the list of files:

```
{% lib
    "css/not-so-important-styles.css"
    minify
    async
%}
```

Accepted arguments are:

Option	Default	Description
absolute_url	false	If true, prefix the generated URL with “https://{hostname}/”.
title	<empty>	Specify a value for the title attribute of the link tag.
media	“all”	Specify value for the media attribute of the link tag.
rel	“stylesheet”	Specify value for the rel attribute of the link tag.
minify		Force minification use as { <code>% lib ... minify %</code> }
async		Load css or javascript asynchronously, use as { <code>% lib ... async %</code> }

load

See also:

[Module tags](#) (page 533)

Loads the given custom tags for use in the templates. Normally not needed, as custom tags are automatically loaded through the module system.

media

See also:

[Media](#) (page 32) developer guide.

See also:

[Media classes](#) (page 33) for some options that are only available in *mediaclass* files.

See also:

[image](#) (page 520), [image_url](#) (page 526) and [image_data_url](#) (page 525) tags.

Show embed, video or audio media.

The {`% media %`} tag is similar to the [image](#) (page 520) tag. It accepts the same arguments but where {`% image %`} is guaranteed to give an still image, {`% media %`} can also generate embed, video or audio elements.

The {`% media %`} tag is not implemented in the core of Zotonic. It depends on modules implementing the tag which arguments and media formats are accepted.

An example of a module using {`% media %`} is [mod_video_embed](#) (page 403) which enables the use of embed code from sites as youtube and vimeo. *Mod_video_embed* will echo the embed code for a {`% media %`} tag and output a still image for an {`% image %`} tag.

now

See also:

the [date](#) (page 457) filter for the possible format characters.

Show the current date and time.

Displays the current local date and time, formatted according to the given date format string.

Example:

```
{% now "Y-m-d" %}
```

Did output “2008-12-10” on december 10, 2008.

There is also a variable called `now`, which holds the current date:

```
{{ now | date: "Y-m-d" }}
```

Is equivalent to using the `{% now %}` tag.

overrides

See also:

[block](#) (page 513), [inherit](#) (page 527) and [extends](#) (page 516).

Inherit markup from like named template in another module.

Signal that this template extends a template with the same name in a module with lower priority.

Note: A template that overrides another template contains only the [overrides](#) (page 530) tag and [block](#) (page 513) tags.

The *overrides* tag must be the first tag in the template.

Example, say a template “page.tpl” contains the following:

```
{% overrides %}  
{% block title %} My new title {% endblock %}
```

All named blocks in this template will replace the similar named blocks in the template *page.tpl* that is “next in line” to be used.

This is useful if you want to use a template from a module, and the template is mentioned in (for example) a dispatch rule. Now you can override and extend that template in your own modules without changing the dispatch rules or the original module.

Make sure your module has a higher priority (lower number) than the module containing the overruled template.

print

See also:

[debug](#) (page 536)

Show the contents of a value expression.

The `{% print %}` tag is used to dump the contents of a variable in a HTML safe way. It is very useful for debugging and inspecting variables during template development.

For example:

```
{% print value %}
```

When value is "`Hello world!`" then the example above returns the output:

```
<pre>  
&lt;b>Hello&lt;/b> world!  
</pre>
```

It can also print complex values like nested lists and tuples, for which it uses Erlang's `io_lib:format/2` function.

raw

Make a literal section which does not interpret tags.

The `{% raw %}` tag takes everything between the `{% raw %}` and `{% endraw %}` without interpretation. It is useful for surrounding javascript or pieces of code with (for example) `{%` in it.

Example:

```
This echos: {{ a }}
{% raw %}
This does not echo {{ a }}
{% endraw %}
{{ a }}
```

Will output:

```
This echos: hello
This does not echo {{ a }}
hello
```

regroup

Not implemented tag, for forward compatibility with future ErlyDTL and Django versions.

spaceless

Removes whitespace between HTML tags.

Note: spaceless does not remove non breaking spaces and other whitespace.

Example:

```
{% spaceless %}
<div>
  <p>Test test test</p>
</div>
{% endspaceless %}
```

After rendering:

```
<div><p>Test test test</p></div>
```

New in version 0.8.

templatetag

Not implemented, but exists in Zotonic for forward compatibility with future ErlyDTL and Django versions.

translate

See also:

[*trans \(variable substitution\)*](#) (page 532).

Translate a text value using gettext.

Translate the text contained in the tag into the currently selected language.

Example:

```
{_ translate me _}
```

If the active language is “nl” then this will output “vertaal mij”. Of course depending on the available translations.

If a translation is not available then the text is output as-is without any translation.

trans (variable substitution)

See also:

[*translate*](#) (page 531).

Translate a text value using gettext and substitute variables.

Example:

```
{% trans "Hello {foo} World" foo=1234 %}
```

This translates the text using the available po files. The variables in the translated text will be substituted with the value of the arguments.

If the active language is “en” then the example above will output “Hello 1234 World”, in “nl” it will be “Hallo 1234 wereld”. Of course depending on the available translations.

If a translation is not available then the text is output as-is without any translation.

If a “{” is needed in the string, then repeat it:

```
{% trans "Hello {{foo}}, and this is {foo}." foo=1234 %}
```

Will echo “Hello {foo}, and this is 1234.”

url

Generate the URL for a named dispatch rule. In this way it is possible to automatically change the generated URLs when the dispatch rules are modified.

For example to generate the URL to the admin for editing a *page*, use:

```
{% url admin_edit_rsc id=myid %}
```

Assuming myid is 42 then this will generate (on most Zotonic sites) the URL “/admin/edit/42”. The name “admin_edit_rsc” can be found in the dispatch-mod_admin-dispatch rules of *mod_admin* (page 346). Which *dispatch rules* (page 22) are available depends on which *Modules* (page 341) are enabled.

When the dispatch rule named in the first argument is unknown then an empty string is returned. There is no error message. This is to prevent breaking the web site when modules are enabled or disabled.

Arguments not named in the path of the dispatch rule are added to the query string of the returned URL:

```
{% url admin_edit_rsc id=42 foo="bar" %}
```


Returns the URL “/admin/edit/42?foo=bar”.

Please note that the best way to generate the URL of a page (resource) is to use:

```
{{ m.rsc[myid].page_url }}
```

Generate absolute URLs

By default, the `{% url %}` tag generates relative URLs. Add the argument `absolute_url` to generate absolute URLs that include the scheme and hostname:

```
{% url admin_edit_rsc id=42 foo="bar" absolute_url %}
```

will return a URL like “http://example.com/admin/edit/42?foo=bar”.

Finding out the current dispatch rule

The name of the current dispatch rule is always available in a template under the name `zotonic_dispatch`.

Check *Global template variables* (page 554) for a full overview of variables that are always available in the templates.

with

Assign a complex value to a variable.

The `{% with %}` tag assigns the result of a variable expression to a new variable. This is useful when accessing an “expensive” method (e.g., one that hits the database) multiple times. The `{% with %}` tag is often used in conjunction with the search model *m.search* (page 578).

For example:

```
{% with m.search[{{latest cat="news"}}] as latest_news %}
  The latest {% latest_news|length %} news articles:
  {% for id in latest_news %}
    {{ m.rsc[id].title }}
  {% endfor %}
{% endwith %}
```

This outputs the number of latest news articles and also the titles of the news articles. The search is only done once when the `{% with %}` tag assigns the variable “latest_news”.

Multiple assignments

It is also possible to assign multiple variables with a single *with* statement, like this:

```
{% with "value1", "value2" as foo, bar %}
  {{ foo }}
  {{ bar }}
{% endwith %}
```

This will output `value1 value2`.

Module tags

Module tags are provided by modules, so their availability depends on *which modules are activated* (page 60) in your Zotonic instance.

button

- Module: *mod_wires* (page 403)

Makes a button with an action attached.

This is an easy way to make a button and attach one or more actions or a postback.

For example:

```
{% button text="Click me" action={alert text="Hello Word!"} %}
```

This show a button with the text “Click me”. When clicked it will trigger the *alert* (page 425) action, showing an alert message with the text “Hello World!”.

Another example:

```
{% button text="Postback" postback={my_postback arg1=1 arg2=2} %}
```

When clicked it will call the `event/2` function in the controller that served the page. The function will be called as:

```
event(#postback{message={mypostback, [{arg1,1}, {arg2,2}]},
      trigger=TriggerId, target=TargetId}, Context)
```

Where `TriggerId` and `TargetId` are both the HTML id of the button.

`button` accepts the following arguments:

Argument	Description	Example
text	The text on the button, defaults to “Submit”	text=”click me”
post-back	An event sent to the delegate or the resource serving the page.	post-back=”hello”
tag	The type of HTML tag that will be created. Defaults to “button”.	tag=”a”
dele-gate	The name of the erlang module to be called for handling the postback.	dele-gate=”myresource”
action	The action to be triggered when the button is clicked. There can be more than one action argument.	action={ show target=”msg” }
id	Id of the button.	id=#submit
class	The css class of the button. This argument can be repeated to add multiple classes.	class=”submit”
style	The css style of the button.	style=”color: #fc0”
tabindex	The value for the <i>tabindex</i> property.	tabindex=1
type	The type attribute of the button.	type=”submit”
title	The title attribute of the button.	title=”click to submit”
dis-abled	The disabled attribute of the button, set to true or false. When the button is disabled then the class “disabled” id added to the class list.	disabled=true

chart_pie

- Module: *mod_base* (page 360)

Show a pie chart.

This is an utility tag providing a simplified interface to the pie chart feature of the *{% google_chart %}* (page 539) tag. It has an easier way to define the data.

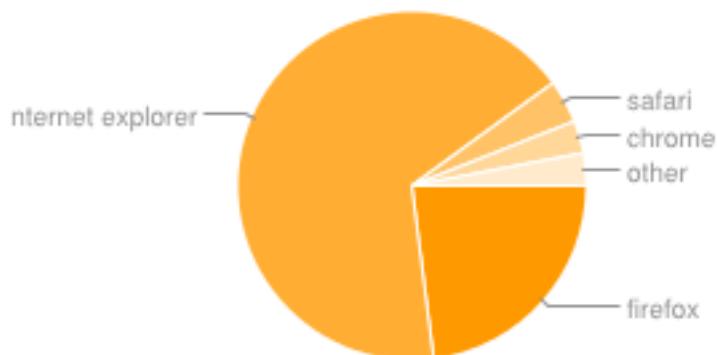
Example of simple pie chart:

```
{% chart_pie data=[["firefox",23],
                    ["internet explorer", 67],
                    ["safari",4],
                    ["chrome",3],
                    ["other", 3]]
%}
```

This generates the following image tag:

```
<img class='google_chart' alt='google chart'
      src='http://chart.apis.google.com/chart?cht=p&chts=909090,10&chs=300x150&chg=0,
      ↪0,1,5&chf=bg,s,ffffff|c,s,ffffff&chdlp=b&chbh=-3,3,7&chxt=x&
      ↪chxl=0:|firefox|internet%20explorer|safari|chrome|other&chxs=0,909090,10&chco=&
      ↪chds=0,100&chd=t:23,67,4,3,3&chls=1,1,0' width='300' height='150' />
```

Or, as an image:



The tag `chart_pie` accepts the following arguments:

Argument	Description	Example
<code>data</code>	The data for the pie chart. A list of pairs of {label, value} or [label, value].	<code>[{"nl",300},{uk,"200"}]</code>
<code>colors</code>	The colors for the pies. A list of colors, when there are more data points than colors then the colors are interpolated. Colors are specified in hexadecimal. Defaults to Google default colors.	<code>colors=["ffcc00","ccff00","00ffcc"]</code>
<code>threed</code>	Set to true to have a 3D effect on the pie chart. Defaults to false.	<code>threed=true</code>
<code>width</code>	The width of the generated pie chart, in pixels. Defaults to 300.	<code>width=450</code>
<code>height</code>	The height of the generated pie chart, in pixels. Defaults to 150.	<code>height=200</code>

Other arguments can be found at the [google_chart](#) (page 539) tag.

See also:

[google_chart](#) (page 539) and [chart_pie3d](#) (page 535).

chart_pie3d

- Module: [mod_base](#) (page 360)

Show a pie chart with 3D effect.

This scomp is just a convenient interface to the `{% chart_pie %}` (page 534) scomp with the `threed` argument set to true.

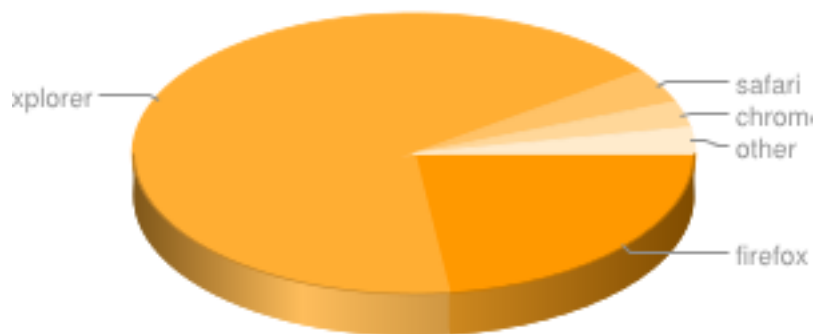
For example:

```
{% chart_pie3d data=[["firefox",23],
                    ["internet explorer", 67],
                    ["safari",4],
                    ["chrome",3],
                    ["other", 3]]
%}
```

Gives:

```
<img class='google_chart' alt='google chart' src='http://chart.apis.google.com/
↪chart?&cht=p3&chts=909090,10&chs=300x150&chg=0,0,1,5&chf=bg,s,ffffff|c,s,ffffff&
↪chdlp=b&chbh=-3,3,7&chxt=x&chxl=0:|firefox|internet
↪%20explorer|safari|chrome|other&chxs=0,909090,10&chco=&chds=0,100&chd=t:23,67,4,
↪3,3&chls=1,1,0' width='300' height='150' />
```

Or, as an image:



See also:

[google_chart](#) (page 539) and [chart_pie](#) (page 534).

cotonic_pathname_search

- Module: [mod_base](#) (page 360)

Todo

Not yet documented.

debug

- Module: [mod_base](#) (page 360)

Shows which variables are assigned for use in the current template's scope:

```
{% debug %}
```

This displays a HTML table with columns for the variable's name and its value, and displays one variable per row:

```
<table>
<tbody>
  <tr>
    <td>session_id</td>
    <td>
      <tt>"qUHoeKodUHXbpwrc16Vj"</tt>
    </td>
  </tr>
</tbody>
</table>
```

```

        </td>
    </tr>

    <tr>
        <td>q</td>
        <td>
            <tt>[{"zotonic_host","examplesite"},"zotonic_dispatch","home"]</tt>
        </td>
    </tr>

    <tr>
        <td>template</td>
        <td>
            <tt>"home.tpl"</tt>
        </td>
    </tr>

    <tr>
        <td>zotonic_dispatch</td>
        <td>
            <tt>home</tt>
        </td>
    </tr>
</tbody>
</table>

```

See also:

[print](#) (page 530)

draggable

- Module: [mod_wires](#) (page 403)

Mark a html element as draggable.

The draggable tag is used in conjunction with the [{% droppable %}](#) (page 538) tag to implement drag & drop. Elements that are marked as draggable can be dropped on elements marked as droppable. Drag & drop generates dragdrop events that are sent to the [controller](#) or the [delegate](#).

For example:

```

<div id="drag1">Drag me</div>
{% draggable id="drag1" tag="drag-one" %}

```

Now the div with id “drag1” can be dragged. When it is dropped then `event/2` of the controller or delegate Erlang module is called signaling the drag (and also the drop to the module receiving the droppable events):

```

event({drag, Drag, Drop}, Context).

```

Where both Drag and Drop are #dragdrop records:

```

-record(dragdrop, {tag, delegate, id}).

```

The draggable tag accepts the following arguments:

Argument	Description	Example
id	The id of the element that becomes draggable.	id="drag1"
tag	The tag of the draggable that is sent as part of the drag and drop events. This can be any value, including a tuple.	tag={subject_list id=42 changed=false}
clone	Clone the element when dragging or drag the element itself. Defaults to false.	clone=true
revert	When the element has to revert to its starting position. Defaults to "invalid", i.e. when the drop was above an invalid position. Other options are true, false and "valid".	revert=false
axis	Constrain the drag movement to either the x or y direction. Normally the drag is not constrained. Acceptable values are "x" or "y" axis="x"	
handle	The css selector that is the handle to drag with. Defaults to the whole element.	handle="handleclass"
group	The name of this drag group, for use in the droppable element's "accept" argument. Multiple groups are allowed.	group="edges"
opacity	Change the opacity while dragging. Defaults to "0.8".	opacity="0.5"
delegate	The Erlang module that will receive the drag event after a successful drop.	

See also:

the [draggable](#) (page 538) tag.

draggable

- Module: [mod_wires](#) (page 403)

Mark an element as valid drag destination.

The droppable tag is used in conjunction with the [{% draggable %}](#) (page 537) tag to implement drag & drop. Elements that are marked as droppable can receive drops of draggable elements. Drag & drop generates dragdrop events that are sent to the [controller](#) or the [delegate](#).

For example:

```
<div id="dropzone">Drop your stuff here</div>
{% droppable id="dropzone" tag="drop-tag" %}
```

Now draggable elements can be dropped onto the div with id "dropzone". When a draggable is dropped then `event/2` of the controller or delegate Erlang module is called signaling the drop (and also the drag to the module receiving the draggable events):

```
event({drop, Drag, Drop}, Context).
```

Where both Drag and Drop are #dragdrop records:

```
-record(dragdrop, {tag, delegate, id}).
```

The droppable tag accepts the following arguments:

Argument	Description	Example
id	The id of the element that will accept drops of draggables.	id="dropzone"
tag	The tag of the droppable that is sent as part of the drag and drop events. This can be any value, including a tuple.	tag={subject id=123}
active	The droppable will have this CSS class added when there is an acceptable draggable being dragged.	active="draghere"
hover	The droppable will have this CSS class added when there is an acceptable draggable hovering over it.	active="dropnow"
accept	The group the droppable accepts. See the group argument of the draggable. A droppable can accept multiple groups, just repeat the accept argument.	accept="edges"
delegate	The Erlang module that will receive the drop event after a successful drop.	

See also:

the [draggable](#) (page 537) tag.

google_chart

- Module: [mod_base](#) (page 360)

Make charts with Google.

This tag interfaces to the [Google chart API](#) to dynamically generate charts.

For example:

```
{% google_chart type="line" axis={axis position="bottom" labels=["jan", "feb", "mar",
↪ ""]}
  axis={axis position="left" labels=[0,25,50,75,100]} data=[{data values=[20,80,
↪ 33]}] %}
```

Will generate the following image tag:

```
<img class='google_chart' alt='google chart'
  src='http://chart.apis.google.com/chart?&cht=lc&chts=909090,10&
↪ chs=300x150&chg=0,0,1,5&chf=bg,s,ffffff|c,s,ffffff&chdlp=b&chbh=-
↪ 3,3,7&chxt=x,y&chxl=0:|jan|feb|mar|1:|0|25|50|75|100&chxs=0,909090,
↪ 10|1,909090,10&chco=&chds=0,100&chd=t:20,80,33&chls=1,1,0'
  width='300' height='150' />
```

[View the chart.](#)

Google chart accepts the following arguments:

Argument	Description	Example
type	Kind of chart is generated. One of “line”, “sparkline”, “stacked_horizontal_bar”, “stacked_vertical_bar”, “grouped_horizontal_bar”, “grouped_vertical_bar”, “pie” or “pie3d”. Defaults to “line”.	type=”sparkline”
id	The id of the generated image tag.	id=”mychart”
class	CSS class of the generated image tag. The class “google_chart” is always added.	class=”chart”
style	CSS style attribute for the generated image tag.	style=”border: 1px solid #fcc”
title	Title shown on the chart.	title=”Browser shares”
color	Color for the title. Must be in hexadecimal, defaults to “909090”.	color=”ffcc00”
font_size	Font size in pixels for the title. Defaults to 10.	font_size=12
width	Width of the generated chart. Defaults to 300.	width=450
height	Height of the generated chart.	height=200
grid_x	X axis grid step size.	grid_x=10
grid_y	Y axis grid step size.	grid_y=10
grid_line_length	Length of line segment for the grid lines, defaults to 1.	grid_line_length=1
grid_blank_length	Length of gaps in the grid lines, defaults to 5.	grid_blank_length=5
background_color	Background color for the complete chart. in hexadecimal, defaults to “ffffff”.	background_color=”331133”
chart_color	Background color for the chart area. In hexadecimal, defaults to “ffffff”.	chart_color=”113311”
legend_location	Where the legend is placed. One of “top”, “left”, “bottom” or “right”. Defaults to “bottom”.	legend_location=”right”
axis	Description of an axis. You can given more than one axis argument. See the section Axis styles and labels (page 540) below.	
data	The data to be shown. You can give more than one data argument. See Data (page 541) definition below.	
bar_space	Space in pixels between the bar of a bar chart. Defaults to 3.	bar_space=5
bar_group_space	Space in pixels between the groups of bars of a bar chart. Defaults to 7.	bar_group_space=10

Axis styles and labels

Axis styles and labels are available for line charts and bar charts.

An example for an axis argument is:

```
axis={axis font_size=10 color="909090" position="top" labels=["jan", "feb", "mar"]}
```

This defines an axis that will be displayed above the chart. It has three labels and will be displayed in a 10 pixel font with color “909090”. You can give a multiple axis arguments and also a list of axes for each argument.

Valid arguments for an axis are:

Argument	Description	Example
font_size	Size of the labels in pixels. Defaults to 10.	font_size=12
color	Color for the label texts, in hexadecimal RGB. Defaults to “909090”.	color=”cc0000”
position	Which axis is defined. One of “top”, “right”, “bottom” and “left”. Defaults to “top”. You can have multiple definitions for a position.	position=”bottom”
labels	A list with labels displayed. The labels will be evenly distributed over the axis.	labels=[2006,2007,2008]

Data

All data definitions to be displayed. Each data definition is a record with arguments. You can supply all data sets at once or with multiple data arguments.

Example with one data set:

```
data=[{data line_width=1 min_value=1 max_value=100 values=[10,20,42,34,68,73,80]}]
```

Valid arguments for a data record are:

Argument	Description	Example
line_width	The width of the line in pixels. Defaults to 1.	line_width=2
line_length	Length of line segment in pixels. Defaults to 1.	line_length=1
blank_length	Length of blank segment in pixels. Defaults to 0.	blank_length=1
min_value	The minimum value for the data set, used for the axis. Defaults to 0.	min_value=-100
max_value	The maximum value for the data set, used for the axis. Defaults to 100.	max_value=100
color	The color used for this data set. Hexadecimal RGB value.	color="ffcc00"
legend	Label for the dataset as shown in the legend.	legend="monthly sales"
values	The values for drawing the chart. Must be a list.	values=[0,10,5,8]

See also:

tags [chart_pie](#) (page 534) and [chart_pie3d](#) (page 535).

inplace_textbox

- Module: [mod_base](#) (page 360)

Render a JS-aided inplace textbox.

Example:

```
{% inplace_textbox value="def.val." delegate="my_resource" hint="edit" %}
```

Todo

Improve documentation

lazy

- Module: [mod_base](#) (page 360)

Custom tag which adds a 'loader' image to the page and performs a one-time action when loader comes into view.

`mod_geomap` uses this to load the map JavaScript once the admin widget has been opened by the user.

Example:

```
<div id="{{ #lazy }}">
    {% lazy action={update target=#lazy id=id template="_geomap_admin_location_map.
    ↳tpl"} %}
</div>
```

`lazy` accepts the following arguments:

Argument	Description	Example
image	The source of the image tag. Defaults to <i>/lib/images/spinner.gif</i> .	image="/lib/images/loading.gif"
class	The css class of the image. Defaults to <i>z-lazy</i> .	class="loader"

live

- Module: *mod_mqtt* (page 381)

Live updating templates connected to *MQTT topics* (page 381).

Note: The live tag is provided by *mod_mqtt* (page 381), which must be enabled.

This tag renders templates that are automatically re-rendered after a publication to an MQTT topic.

Example

An example of a template showing the newest content of a resource:

```
{% live template="_detail.tpl" topic=id id=id %}
```

This renders the template `_detail.tpl`. If the resource with id `id` is updated then the template will be replaced with a freshly rendered template.

The tag can subscribe to multiple topics at once.

Add the argument `catinclude` to do a *catinclude* (page 515) instead of a normal *include* (page 526). For a `catinclude` the argument `id` must be present:

```
{% live template="_detail.tpl" topic=id catinclude id=id %}
```

Live topics

Any MQTT topic can be used. The topics are interpreted as local to the page. There are three special topics:

- Use any integer to map to the resource's update topic. For example if `id` is 1234 then the topic will be `bridge/origin/model/rsc/event/1234`
- Use the tuple `{object id=...}` to listen to changes of outgoing connections from a page. An example of a mapped topic is `bridge/origin/model/edge/event/1234/o/+`. Use the tuple `{object id=... predicate=...}` to listen to changes of a specific predicate of a page. An example of a mapped topic is `bridge/origin/model/edge/event/1234/o/author`
- Use the tuple `{subject id=...}` to listen to changes of incoming connections to a page. An example of a mapped topic is `bridge/origin/model/edge/event/1234/s/author`

Note that the topics refer to *client side topics*, that is why the bridge is used to subscribe to server side model events.

It is possible to subscribe to client topics like `"my/local/topic"` and have the actions triggered by publish to `cotonic.broker.publish("my/local/topic", {})`; (with any payload).

Live actions

It is possible to wire actions or postbacks to a MQTT topic.

Use the *wire* tag (page 551) with argument `type={mqtt topic=... topic=...}` to connect to one or more MQTT topics:

```
{% wire type={mqtt topic="bridge/origin/public/hello"} action={growl text="hello"}
→ %}
```

And in Erlang this will trigger the above *growl*:

```
z_mqtt:publish(<<"public/hello">>, <<>>, z:c(mysite)).
```

loremipsum

- Module: *mod_base* (page 360)

Inserts a piece of “lorem ipsum” text into the page.

The loremipsum tag inserts the following fake-latin text:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam lobortis, lacus id molestie suscipit, enim leo pharetra velit, ac cursus felis mauris at velit. Cras ultrices, massa pharetra faucibus cursus, neque leo bibendum mauris, eu vulputate leo magna vitae lacus. Etiam pharetra gravida elementum. Maecenas lacinia, sem sed eleifend euismod, risus velit interdum nulla, convallis sagittis orci dui a metus. Aliquam tristique orci a ipsum dapibus viverra. Donec vestibulum varius ante, vitae placerat magna ultrices ac. Donec nec ante magna. Donec porttitor, arcu a condimentum aliquam, lectus nunc rhoncus quam, id ornare neque ligula quis dui. Pellentesque sit amet lectus augue, ut ullamcorper nisi. Donec et rutrum sem. In porta ultricies nibh, in sagittis lacus euismod at. Etiam consectetur tristique tellus, quis tristique dui vulputate vel. Curabitur sagittis gravida dui, vel sagittis lectus suscipit ac.

The loremipsum tag accepts the following arguments:

Argument	Description	Example
words	The number of words to be displayed from the text above.	words=10

New in version 0.5.

Changed in version 0.6: Added `words` argument.

mailinglist_subscribe

- Module: *mod_mailinglist* (page 379)

Show the mailinglist subscription form to subscribe to a certain mailinglist id.

Parameters:

id Required; the id of the mailinglist *resource* that is being subscribed to.

template Which form template to render. Defaults to the template `_scomp_mailinglist_subscribe.tpl`.

All other parameters are passed in to the template which is being rendered.

The form is at least supposed to have an *email* input field. Besides *email*, it can have *name_first*, *name_surname_prefix* and *name_surname* fields, which will be stored in the recipient table.

menu

- Module: *mod_menu* (page 381)

Show a page menu.

This tag is part of the module `mod_menu`. The `{% menu %}` tag is used to generate the HTML for the menu defined in the admin.

You can define multiple menus in your site. By default there is one menu, called “main_menu”. If you want another one, create a page of type “page menu” (under “Categorization”) and start editing your menu. You can use the “menu_id” argument to select which menu you want to display.

Example:

```
{% menu id=id %}
```

Generates something like:

```
<ul id="navigation" class="nav">
  <li>
    <a href="/" class="welcome">Home</a>
  </li>
  <li>
    <a href="/features" class="page_features">Features</a>
  </li>
  <li>
    <a href="/documentation" class="documentation active">Documentation</a>
  </li>
  <li>
    <a href="/documentation/628/installation" class="page_install">Install</a>
  </li>
</ul>
```

The menu has the following features:

- The menu is a unordered list.
- The id of the menu is `navigation` and can be prepended with param `id_prefix`.
- The class of the menu is set with param `class` (default `nav`).
- Menu items are a `` with a single `<a>`
- The link of the menu item referring to the current page has the class `active`.
- Every link also gets the unique name of the target as a class.
- Every menu item can have single level submenus. A submenu has the same properties as the menu.

Argument	Description	Example
<code>id</code>	Set this to the id of the current shown page and it wil highlight its page path.	
<code>menu_id</code>	The id of the menu that you want to display. If left empty, the main menu is shown.	
<code>id_prefix</code>	String prepended to menu id.	
<code>class</code>	HTML class for the list; default “nav”.	
<code>maxdepth</code>	Maximum depth of the menu; default 999.	
<code>template</code>	Template to render the menu; default “_menu.tpl”	

pager

- Module: *mod_base* (page 360)

Show a pager for search results.

This generates a pager as seen on the search results pages. It is used in conjunction with a paged search result.

For example, a fulltext search where the search parameters come from the query string:

```
{% with m.search.paged[{fulltext cat=q.qcat text=q.qs page=q.page pagelen=20}] as result %}
<ul>
  {% pager result=result dispatch="admin_overview_rsc" qargs %}
  {% for id,score in result %}
    <li><a href="">{{ m.rsc[id].title }}</a></li>
  {% empty %}
    <li>Nothing found</li>
  {% endfor %}
</ul>
{% endwith %}
```

This will show a list of titles and above that the links to the next, previous and other pages.

Note: that we are using `m.search.paged` here and not `m.search` (page 578). The pager only works with results from `m.search.paged`.

The generated pager code will look something like (when searching for the text “filter”):

```
<ul class="pager block">
<li><a href="#">< prev</a>
<li class="current"><a href="/admin/overview?qs=filter&page=1">1</a></li>
<li><a href="/admin/overview?qs=filter&page=2">2</a></li>
<li><a href="/admin/overview?qs=filter&page=3">3</a></li>
<li><a href="/admin/overview?qs=filter&page=4">4</a></li>
<li class="pager-sep">...</li>
<li><a href="/admin/overview?qs=filter&page=5">5</a></li>
<li><a href="/admin/overview?qs=filter&page=2">next></a></li>
</ul>
```

The pager tag accepts the following arguments:

Argument	Description	Example
result	The result from a search. This must be a <code>#search_result</code> or a list. Note that the records should be the result of a <code>m.search.paged</code> and not of a <code>m.search</code> call.	<code>result=mysearchresult</code>
dispatch	Name of the dispatch rule to be used for the page urls. Defaults to the dispatch rule of the current page.	<code>dispatch="searchresult"</code>
qargs	Append all the arguments in the HTTP request's query string whose name starts with a 'q' as an argument to the dispatch rule.	<code>qargs</code>
hide_single_page	When this argument is true, do not show the pager when the result fits on one page (e.g. the pager will be useless).	<code>hide_single_page=1</code>
template	Name of the template for rendering the pager. Defaults to <code>_pager.tpl</code> . See below for specific arguments passed.	<code>template="_pager.tpl"</code>
page	Current page number, the first page is page number 1. Fetched from the search result, this argument, or <code>q.page</code> .	<code>page=2</code>
pagelen	Number of items per page, fetch from the search result or <code>q.pagelen</code> . Defaults to 20.	<code>pagelen=10</code>
*	Any other argument is used as an argument for the dispatch rule.	

Pager template

The pager is rendered using a template. The default template for the pager is `_pager.tpl`.

The pager template receives the following variables:

- `prev_url` The url to the previous page, undefined if at first page.

- `next_url` The url to the next page, undefined if at last page.
- `pages` A list of tuples. Either `{PageNumber, Url}` or `{undefined, sep}` (*sep* is an atom).
- `page` The current page number
- All other arguments passed to the `scomp` (**attention:** these are also used as dispatch arguments)

The default `_pager.tpl` displays an ellipsis for the `sep` entry.

If the result set is empty then the template is not rendered. The template is also not rendered if there is a single page *and* `hide_single_page` is set.

Result argument

It is also possible to pass a list, a `#rsc_list{ list=Ids }` record, or a list of lists (pages) for the resut. In this case you need to perform the pagination for displaying the results yourself. You can use the [chunk](#) (page 474) for this. The pager `scomp` will fetch the `page` and `pagelen` from the pager arguments, or from the query arguments (if any). If the list is pre-chunked then the pages does not need the `pagelen` argument.

poll

- Module: [mod_survey](#) (page 398)

Show a given survey (with the `id` parameter) as a “poll”. This presents a simpler interface, in which the user is directly asked to enter some information, e.g. make a choice between certain things:

```
{% poll id=123 %}
```

script

- Module: [mod_wires](#) (page 403)

This tag is the placeholder where all generated JavaScript scripts will be output on the page.

Zotonic generates JavaScript for the actions and other template logic. This script needs to be added to the page. The `{% script %}` `scomp` designates the place where the `<script>` element with all the generated javascript can be placed.

Normally the `{% script %}` `scomp` is placed at the end of the page, just above the `</body>`.

Note that all JavaScripts generated after the `{% script %}` `scomp` will not be included in the generated page. Only a single `{% script %}` `scomp` is allowed on any page.

Example:

```
{# at the bottom of the template ... #}
{% script %}
</body>
</html>
```

This will generate something similar to:

```
<script type='text/javascript'>
$(function() {
  z_pageid="MouKz4PgRCROM5efU8iL";
  z_postback_loop();

  $('#vtretq').bind('click', function(event) { window.location = "/admin/edit/647";
  ↪return z_opt_cancel(this); } );
  z_init_postback_forms();
  z_default_form_postback = "bVcYIST9JOG/AQZkP9SOZmc//
  ↪GqDaAVrAAZzdWJtaXRkAA11bmRlZmluZWRRkAA11bmRlZmluZWRRkZAANcmVzb3VyY2VfcGFncQ==";
```

```
});
</script>
</body>
</html>
```

Note that the contents of this block will be completely different per page.

The script scomp can have the following arguments:

Argument	Description	Example
no-startup	Exclude the page initialization code from the script, only includes the scripts from actions etc. Default is to include the page initialization code.	no-startup
no-stream	Do not start the bi-directional communication layer (over WebSockets or comet).	no-stream
format	Select a different format than the <code><script/></code> tag. For now this accepts "html" (for the <code><script/></code> tag), "escapejs" for an escaped javascript string, and "js" for a normal javascript string. Default is "html".	format="html"

WebSockets / Comet communication

Unless `no-stream` is added as a parameter, this tag also causes the WebSockets or Comet communication layer to be initiated.

When available, a WebSocket connections is opened, otherwise a long polling Comet connection is started. The WebSockets connection will also be used for sending Ajax requests to the server.

See also [Transport](#) (page 73) for details on the bi-directional communication.

sortable

- Module: [mod_wires](#) (page 403)

Mark an element as sortable.

Sortables are part of a sorter. Sortables in a sorter can be reordered by drag & drop.

Example:

```
{% sorter id="sorter" tag="mysorter" %}
{% sortable id="sort1" tag=1 %}
{% sortable id="sort2" tag=2 %}
<ul id="sorter">
  <li id="sort1">Sortable 1</li>
  <li id="sort2">Sortable 2</li>
</ul>
```

This creates a list where the order of the list items can be changed by dragging and dropping them.

When the order of the sortables is changed, an event is sent to the sorter's page [controller](#) or [delegate](#). The event contains the ordered list of sortable tags.

The event handler function is called like:

```
event({sort, Sortables, Sorter}, Context).
```

Where "Sortables" is the list of sortables and "Sorter" is the sorter. Both are "#dragdrop" records:

```
-record(dragdrop, {tag, delegate, id}).
```

Where “tag” is the tag of the sortable or sorter, “delegate” is the module that handles the event and “id” is the HTML id of the sortable or sorter.

Note: if the tag is a string then the #dragdrop tag will be an atom.

The sortable can have the following arguments:

Argument	Description	Example
id	The HTML id of the sortable element.	id="mysortable1"
tag	Tag that identifies the sortable to the event handler. Can be any value. A string will be converted to an atom.	tag="my_atom_value"
delegate	The delegate of the sortable, currently unused will be passed in the sortables's #dragdrop record.	delegate="mymodule"
class	A CSS class that will be added to the sortable. The class “sortable” will always be added.	class="dragitem"

See also:

the [sorter](#) (page 548) tag.

sorter

- Module: [mod_wires](#) (page 403)

A sorter is a container for sortables.

A sorter contains sortables and handles the events when the order of the sortables is changed. Sortables in a sorter can be reordered by drag & drop.

Example:

```
{% sorter id="sorter" tag="mysorter" %}
{% sortable id="sort1" tag=1 %}
{% sortable id="sort2" tag=2 %}
<ul id="sorter">
  <li id="sort1">Sortable 1</li>
  <li id="sort2">Sortable 2</li>
</ul>
```

This creates a list where the order of the list items can be changed by dragging and dropping them.

When the order of the sortables is changed, an event is send to the sorter's page [controller](#) or [delegate](#). The event contains the ordered list of sortable tags.

The event handler function is called like:

```
event({sort, Sortables, Sorter}, Context).
```

Where “Sortables” is the list of sortables and “Sorter” is the sorter. Both are #dragdrop records:

```
-record(dragdrop, {tag, delegate, id}).
```

Where “tag” is the tag of the sortable or sorter, “delegate” is the module that handles the event and “id” is the HTML id of the sortable or sorter.

Note: If the tag is a string then the #dragdrop tag will be an atom.

The sorter can have the following arguments:

Argument	Description	Example
id	The HTML id of the sortable element.	id="mysorter"
tag	Tag that identifies the sortable to the event handler. Can be any value. A string will be converted to an atom.	tag="my_atom_value"
delegate	The delegate of the sorter. The sort event will be send to the delegate module. Defaults to the resource that handled the page request.	delegate="mymodule"
class	A CSS class that will be added to the sorter. The class "sorter" will always be added.	class="bunny-sorter"
handle	jQuery selector for the handles of the sortables. When not defined then the whole sortable can be clicked on for dragging.	handle=".sortable-handle"
connect_group	Name of the group this sorter connects with. Sortables from this sorter can then be dragged to sorters with that group name. This argument can be repeated to connect with multiple groups. Special values are "all" and "none" to either connect to all other sorters or to no other sorter.	connect_group="bunnies"
group	The group of this sorter. Used in connection with the "connect_group" argument. Sortables can be dragged between sorters of the same group.	group="cows"
axis	If defined the items can only be dragged horizontally or vertically. Possible values are "x" and "y".	axis="y"
containment	Constrains dragging of the sortables to within the bounds of the specified element. Possible values are "parent", "document", "window", or a jQuery selector.	containment="parent"
opacity	Opacity a sortable is set to when being dragged. Defaults to "1.0".	opacity="0.8"
placeholder	Class that gets applied to the otherwise white space that will show between sortables as the new place of the sortable.	class="drophere"

See also:

the [sortable](#) (page 547) tag.

spinner

- Module: [mod_base](#) (page 360)

Add an AJAX activity indicator.

Whenever an AJAX call is made the HTML element with the id #spinner will be shown during the call.

You can add a spinner element to your page yourself or use this tag to place it somewhere on your page.

Example:

```
{% spinner %}
```

Outputs the HTML code:

```
<div id="spinner" class="spinner" style="display: none">
  
</div>
```

The spinner tag accepts a single argument "image". The "image" argument must contain the URL for the image displayed. It defaults to "/lib/images/spinner.gif".

tabs

- Module: [mod_base](#) (page 360)

Make a HTML element into a tab set.

Note: There is no default styling for jQuery UI elements in the zotonic CSS files. See these two threads in the zotonic users mailinglist: [does tabs scomp still work?](#), and [playing with tabs scomp](#). See also the [jQuery UI documentation](#).

This is a simple interface to the jQuery UI tabs functionality. It will show tabs to switch between different panes.

The only argument is “id” which is the id of the container of the tabs.

The following example will show an interface with three tabs:

```
{% tabs id="tabs" %}
<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p>Proin elit arcu, rutrum commodo, vehicula tempus.</p>
  </div>
  <div id="tabs-2">
    <p>Morbi tincidunt, dui sit amet facilisis feugiat.</p>
  </div>
  <div id="tabs-3">
    <p>Mauris eleifend est et turpis.</p>
  </div>
</div>
```

validate

- Module: [mod_wires](#) (page 403)

The validator tag accepts the following arguments:

Argument	Description	Example
id	The id of the input element to be validated.	id="password_field"
type	The validator for the input element. Can be specified multiple times to add multiple validators. The value depends on the validator chosen but is always a record {validatorname arg=value ...}	type={ acceptance }
trigger	Id of the element that triggers the validation. Defaults to the value of “id”. The validator is triggered by a change of this. Almost never used.	trigger="title"
target	Target of validator, defaults to the trigger. Almost never used.	
name	Use this when the name of the input element is unequal to the id. The name is used by the server side code to validate the received input. Defaults to the target argument (which defaults to id).	name="password_field"
valid_message	Message to show when the field passes validation. Defaults to the empty string.	valid_message="ok!"
failure_message	Argument passed to the type argument. See individual validators.	
message_after	The id of the element after which the failure message should be shown. Defaults to the id argument.	message_after="signup_tos_agree"
only_on_blur	Normally validates on change, unless only_on_blur is set.	only_on_blur
wait	Time in msec to wait for validation after the last keystroke. Default: 0.	wait=100
only_on_submit	Whether the validation should be done when entering data or only on submit of the form. Set this to suppress validation when entering data.	only_on_submit

See also:

- the list of [Validators](#) (page 582)

- *Forms and validation* (page 37) in the Developer Guide

wire

- Module: *mod_wires* (page 403)

Connect actions and events to a HTML element.

The wire tag is the basis for most Ajax interaction on web pages. It allows to connect actions to HTML elements. Examples of *Actions* (page 403) are *show* (page 414) / *hide* (page 409) elements or *postback* (page 419) to the server.

Wire actions to an element

The primary use of wire is to connect an action to a HTML element or javascript event.

Example:

```
{% wire id="show" action={show target="message"} %}
<a id="show" href="#">Click to show a message</a>
<p style="display: none" id="message">Hello World!</p>
```

A click on the link will show the message “Hello World!”.

Wire postbacks to the server

Wire can post an event back to the server. Use for this the “postback” argument:

```
{% wire id="mybutton" postback={hello world="round"} %}
<button id="mybutton">Post event to server</button>
```

When the button is clicked the event function of the page controller will be called as:

```
event(#postback{message=Postback, trigger=TriggerId, target=TargetId}, Context).
```

Where “Postback” will be {hello, [{world, <<"round">>}]} and both “TriggerId” and “TargetId” will be <<"mybutton">>.

Wire form submit events

Wire is also used to connect a form to the event routine handling the form.

Example:

```
{% wire id="myform" type="submit" postback="some_tag" action={toggle target=
  ↳ "message"} %}
<form id="myform" method="post" action="postback">
  <input type="text" name="title" value="" />
  <button id="mybutton" type="submit">Submit</button>
</form>
```

The wire tag redirects the submit of the form to the event routine of the controller. A submit will also toggle the visibility of the “message” element on the page. Note that the action is “postback”, this is obligatory.

The event routine will be called as:

```
event(#submit{message=Tag, form=FormId, target=TargetId}, Context).
```

Where Tag will be the postback set by the wire tag (in the example the atom `some_tag`) and FormId and TargetId are both the HTML id of the submitted form. The posted input fields can be fetched using `z_context:get_q/2`, `z_context:get_q_all/1` or `z_context:get_q_validated/2`.

```
Title = z_context:get_q(<<"title">>, Context),
AllArgs = z_context:get_q_all(Context);
```

There are some extra arguments added to every form post:

Post argument	Description	Example
<code>z_trigger_id</code>	Id of the HTML element that triggered the submit.	<code><<"mybutton">></code>
<code>z_pageid</code>	Id of the page in the browser, used to connect comet and other communication between the browser and the server.	<code><<"1uTs-bzIsWqmPpF32">></code>
<code>postback</code>	Signed postback set by the wire tag. Handled internally.	

Wire to the page load or unload

A `{% wire %}` without an id will bind the actions and/or postback to the window instead of an element. Omitting a type as well will execute all actions and/or postback on page load.

Example:

```
{% wire action={alert text="Welcome to this page."} %}
```

The type “unload” will execute all actions and/or postback when leaving the page:

```
{% wire type="unload" action={alert text="Bye."} %}
```

Call a wire action from JavaScript

Use `{% wire name="myname" %}` to define a named action and trigger it from JavaScript with `z_event("myname")`. See: [Wires](#) (page 72).

Wire an action to a MQTT topic

Note: `mod_mqtt` (page 381) must be enabled before wiring to a topic

Use `{% wire type={mqtt topic=... topic=...} %}` to connect to one or more MQTT topics.

Example:

```
{% wire type={mqtt topic=~site/public/hello} action={growl text="hello"} %}
```

And in Erlang this will trigger the above *growl*:

```
z_mqtt:publish(<<"~site/public/hello">>, <<>>, z_acl:sudo(z:c(mysite))).
```

See also *live* (page 542)

Arguments

The wire tag accepts the following arguments:

Argument	Description	Example
id	HTML id of the element the action gets connected to. When the id is not given then the event is bound to the window.	id="mybutton"
type	The type of the event triggering the action. Defaults to "click". Other types are: "enterkey", "interval", "continuation", "submit" or one of the jQuery events "blur", "focus", "load", "resize", "scroll", "unload", "beforeunload", "click", "dblclick", "mousedown", "mouseup", "mousemove", "mouseover", "mouseout", "mouseenter", "mouseleave", "change", "select", "keydown", "keypress", "keyup" or "error". The types can be extended by modules using the <code>#action_event_type</code> notification. The type must be a tuple, an example is the <code>{mqtt topic=...}</code> type provided by mod_mqtt (page 381)	type="submit"
propagate	Specify this when you don't want the event to be canceled after handling the wire. Useful for event types like focus, click etc. .. versionadded:: 0.6.1	propagate
target	Possible target for the action. The meaning of this argument depends on the action, defaults to id.	
action	Action wired to the element. This parameter can be repeated to wire more than one action at a time. The value is a single or a list of action records.	action={toggle target="message"}
postback	Postback that will be sent to the event handler of the controller or the delegate. Either a string, which will be send as an atom, or a tagged property list. The example will be in Erlang <code>{myevent, [{foo, 1}, {bar, 2}]}</code> .	postback="ajaxevent" postback={myevent foo=1 bar=2}
delegate	Name of the Erlang module that will receive the postback. Defaults to the controller that handled the page request.	delegate="event_handler"

See also:

the tag [wire_args](#) (page 553) and the list of predefined [Actions](#) (page 403).

wire_args

- Module: [mod_wires](#) (page 403)

Add extra arguments to wired actions.

The tag `wire_args` is used to append extra arguments to actions before wiring those actions. This is useful when an action is handed down as an argument to a template but still needs the id of a local element.

For example:

```
{% with {my_action some_arg=some_value} as my_action %}
  {% for n in [1,2,3,4] %}
    <a id="{{#id.n}}" href="#">click {{n}}</a>
    {% wire_args action=my_action the_link_id=#id.n %}
  {% endfor %}
{% endwith %}
```

This wires the action `{my_action some_arg=some_value the_link_id=...}` to the links.

The following arguments are part of the wire tag and can't be used for argument appending: "id", "type", "target", "action", "postback" and "delegate".

See also:

the *wire* (page 551) tag.

worker

- Module: *mod_base* (page 360)

Todo

Not yet documented.

6.1.6 Global template variables

These variables are always available for *rendering in templates* (page 29).

zotonic_version

The version of Zotonic, for example "1.0-dev".

zotonic_dispatch

An atom with the name of the dispatch rule that was applied to render the current page.

zotonic_dispatch_path

A list containing the request path used as initial input for the dispatcher. The path is split on / and after an optional rewrite. This means that the list doesn't contain the language prefix. For example, the path `/en/foo/bar?a=b` will give the list `[<<"foo">>, <<"bar">>]`.

zotonic_dispatch_path_rewrite

Same as `zotonic_dispatch_path`, but set to the path after an optional internal request rewrite inside the dispatcher. For example if a resource has its *page_path* set to `/foo` and the requested path is `/en/foo` then the `zotonic_dispatch_path` will be set to `[<<"foo">>]` and the `zotonic_dispatch_path_rewrite` could be set to something like `[<<"page">>, <<"1234">>, <<"foo-slug">>]`.

z_language

The currently selected language. This an atom, for example: `en`.

q

A key/value list containing the current request's query variables. For GET requests, these are the arguments passed from the query string (e.g. `?foo=bar`); for POST requests, these are also the values posted in the POST form. For more access to the raw request data, look at the `m_req` (page 570) model.

now

The UTC date and time in Erlang tuple notation, for instance `{{2014,4,17},{13,50,2}}`.

m

`m` is not really a value, but it's an indicator to trigger a lookup in one of Zotonic's *Models* (page 555). For instance the `m_rsc` (page 571) model is always exposed and can be used like this `{{ m.rsc[123].title }}`.

true, false, and undefined

The values `true`, `false` and `undefined` are predefined.

6.1.7 Models

Template *models* (page 30) provide data to templates.

m_acl

- Module: `core`

The `m_acl` model gives access the id of the currently logged in user, and provides a mechanism to do basic access control checks.

The following `m_acl` model properties are available in templates:

Property	Description
<code>user</code>	Returns the current user id. If not logged in, this returns <code>undefined</code> .
<code>is_admin</code>	Check if the current user is allowed to access the admin. Internally, this checks the <code>use, mod_admin_config ACL</code> .
<code>use, admin, view, delete, update, insert, link</code>	These properties are shortcuts to check if the current user is allowed to do some action.
<code>is_allowed</code>	Perform custom ACL checks which are different from the ones mentioned.
<code>authenticated</code>	Used before the other ACL checks to check if a <i>typical</i> user is allowed to perform some actions. Example: <code>m_acl.authenticated.insert.article</code> If an user is logged on the that user's permissions are used.

This example prints a greeting to the currently logged in user, if logged in:

```
{% if m_acl.user %}
  Hello, {{ m_rsc[m_acl.user].title }}!
{% else %}
  Not logged in yet
{% endif %}
```

This example checks if the user can access the admin pages:

```
{% if m.acl.is_admin %} You are an admin {% endif %}
```

This example performs a custom check:

```
{% if m.acl.is_allowed.use.mod_admin_config %}  
    User has rights to edit the admin config  
{% endif %}
```

And to check if a resource is editable:

```
{% if m.acl.is_allowed.update[id] %}  
    User can edit the resource with id {{ id }}  
{% endif %}
```

A short hand for the above is (assuming *id* is an integer):

```
{% if id.is_editable %}  
    User can edit the resource with id {{ id }}  
{% endif %}
```

m_acl_rule

- Module: *mod_acl_user_groups* (page 341)

Not yet documented.

m_acl_user_group

- Module: *mod_acl_user_groups* (page 341)

Not yet documented.

m_admin

- Module: *mod_admin* (page 346)

This model exposes some meta-information for the use in `mod_admin` templates.

There are two properties available:

1. Pivot queue count

Retrieve the count of the pivot queue, the queue which decides which resources need re-indexing.

To view the number of items in the pivot queue, do the following:

```
{% print m.admin.pivot_queue_count %}
```

2. Default published setting for the new-resource dialog

```
{% if m.admin.rsc_dialog_is_published %} ... {% endif %}
```

m_admin_blocks

- Module: *mod_admin* (page 346)

Not yet documented.

m_admin_config

- Module: *mod_admin_config* (page 351)

Not yet documented.

m_admin_identity

- Module: *mod_admin_identity* (page 353)

Not yet documented.

m_admin_menu

- Module: *mod_admin* (page 346)

This model holds the admin menu, which is built up by calling each module to add items to the menu.

You can extend the admin menu by observing the *admin_menu* (page 611) notification.

m_admin_note

- Module: *mod_admin* (page 346)

Add an editorial note to any resource.

The note is entered on the admin edit page. Only people with edit permission on the resource *and* access to the admin are allowed to see and edit notes.

m_admin_status

- Module: *mod_admin* (page 346)

Not yet documented.

m_auth2fa

- Module: *mod_auth2fa* (page 359)

Not yet documented.

m_authentication

- Module: *mod_authentication* (page 359)

Not yet documented.

m_backup

- Module: *mod_backup* (page 360)

Not yet documented.

m_backup_revision

- Module: *mod_backup* (page 360)

Not yet documented.

m_category

- Module: core

This model can retrieve information about the resource category hierarchy in different ways.

Categories are the principal categorization (typing) system of resources. Every page is assigned to exactly one category. Categories themselves are organized in a tree-like hierarchy.

A category is a resource with a special `category` record attached to it to store metadata related to the category hierarchy. The `m_category` model provides accessors to this category tree and individual category information.

An example of a category tree, as returned by `{% print m.category.tree %}`:

```
[[{id,101},
  {parent_id,undefined},
  {level,1},
  {children,[]},
  {path,"e"},
  {left,1000000},
  {right,1000000}},
[{id,104},
  {parent_id,undefined},
  {level,1},
  {children,[[{id,106},
    {parent_id,104},
    {level,2},
    {children,[[{id,109},
      {parent_id,106},
      {level,3},
      {children,[]},
      {path,"hjm"},
      {left,4000000},
      {right,4000000}]}]},
    {path,"hj"},
    {left,3000000},
    {right,4000000}]}]},
  {path,"h"},
  {left,2000000},
  {right,4000000}},
...]
```

About the complete category tree

The following `m_category` model properties are available in templates:

Property	Description	Example value
<code>tree</code>	Return the complete forest of category trees as nested property lists.	See above.
<code>tree2</code>	Return the forest of category trees as nested property lists. Up to the children of the children.	See above.
<code>tree_flat</code>	Return a list of tuples for the category tree. This list is intended for select lists. There is a special field for the indentation. The returned list consists of proplists. The list does not contain the “meta” category, which contains the categories “predicate”, “category” etc.	See above entries.
<code>tree_flat meta</code>	Same as <code>tree_flat</code> but with the categories in the <i>meta</i> category.	See above entries.

About a single category

The `m_category` has some special properties defined when fetching a category, they are accessed by id or category name. For example:

```
{{ m.category[104].tree }}
{{ m.category.text.tree }}
```

Property	Description	Example value
tree	The category tree below and including the indexing category.	See above.
tree1	The list of direct children below the indexing category.	See above.
tree2	The category tree below and including the indexing category, up to the children of the children.	See above.
tree_flat	The category tree below and including the indexing category, up to the children of the children. As a flattened list.	See above.
path	List of parent category ids from the root till the category, excluding the indexing category.	[104, 106]
is_a	List of the parent category names from the root till the category, including the current category.	[text, article, news]
image	A random depiction for this category. The returned image filename comes from one of the pages within this category.	<<"2009/10/20/flat-world-proof.jpg">>
parent_id	The page id of the parent category. Returns an integer or, for a root category, undefined.	104
nr	The category nr. Used for building the tree, will change when categories are added or removed. An integer.	2
level	The depth of the category. Level 1 is the root, 2 and more are below the root.	1
left	The lowest value of the nr range of this category, including its sub categories.	2
right	The highest value of the nr range of this category, including its sub categories.	8
name	The unique page name of this category. A binary.	<<"text">>
path	The path through the hierarchy of categories to this category.	[104, 106]

m_client_local_storage

- Module: [mod_mqtt](#) (page 381)

Model to access the `localStorage` on the client (browser).

The client-id and routing topic in the *context* must be set when calling the functions in this module. This is the case for all MQTT topic calls from the client.

The `localStorage` on the client is accessed via publishing to the topic `~client/model/localStorage/...`. In the client the topic is `model/localStorage/...`.

m_client_session_storage

- Module: [mod_mqtt](#) (page 381)

Model to access the `sessionStorage` on the client (browser).

The client-id and routing topic in the *context* must be set when calling the functions in this module. This is the case for all MQTT topic calls from the client.

The `sessionStorage` on the client is accessed via publishing to the topic `~client/model/sessionStorage/...`. In the client the topic is `model/sessionStorage/...`.

m_comment

- Module: *mod_comment* (page 361)

Accesses comments on a page.

Comments are stored in the `comment` table. Comments are no separate rsc records because that will add many extra records and also because of access control restrictions.

When a page is not visible to a certain user then its comments shouldn't be visible as well. To simplify this check the comments are placed separate and made part of the rsc record.

This separate comment table also helps with cleaning up comments when the rsc record is deleted.

Todo

Finish `m_comment`

m_config

- Module: `core`

Zotonic has two places where a site's configuration is kept:

- the site's *config file* (page 626) (accessible through *m_site* (page 579))
- the site's `config` database table. Entries in the `config` table overrule any module settings from the config file.

Note: Configuration keys are only accessible from templates using `{{ m.config }}` for users with administrator rights. To access other configs, use or add specific models.

All `m_config` keys can be thought of as tuples `{Module, Key, Value}`, where `Value` is a complex value that can have a text value but also any other properties. Only configuration with text values can be edited in admin.

The `config` model table has different access methods. Below are examples of what is possible and how to access the configuration.

Fetch the value of a config key

Example, fetching the config key value of `mod_emailer.from_email`:

```
{{ m.config.mod_emailer.email_from.value }}
```

Or, from Erlang:

```
m_config:get_value(mod_emailer, email_from, Context).
```

Where `m.config.mod_emailer.email_from` returns a property list which is much like:

```
[
  {id, 5},
  {module, <<"mod_emailer">>},
  {key, <<"email_from">>},
  {value, <<"no-reply@example.com">>},
  {props, undefined},
  {created, {{2009, 11, 7}, {20, 48, 27}}},
  {modified, {{2009, 11, 7}, {20, 48, 27}}}
]
```

If the database does not contain a `mod_email.email_from` configuration parameter, Zotonic falls back to the *site config file* (page 626) and tries to find the parameter there.

When the config key would have any extra value (besides the value property) then they would be visible as extra properties and the property “props” would not have been present.

When the configuration comes from the site config then the id property is not present.

Fetching all configurations of a module

This only returns configurations from the config table; configuration keys from the site config are not mixed in. This might change in the future:

```
{% for key, value in m.config.mod_emailer %}
...
{% endfor %}
```

Fetching all configurations

This only returns configurations from the config table. Configurations from the site config are not mixed in. This might change in the future:

```
{% for mod, keys in m.config %}
  {% for key, value in keys %}
    ...
  {% endfor %}
{% endfor %}
```

Listening for config changes

`m_config` uses `z_notifier` to broadcast events when values are changed or deleted:

```
#m_config_update{module=Module, key=Key, value=Value}
```

for “complex” property value updates/inserts a slightly different notification is used:

```
#m_config_update_prop{module=Module, key=Key, prop=Prop, value=Value}
```

For config key deletes, the `#m_config_update{}` record is broadcast with undefined for the value:

```
#m_config_update{module=Module, key=Key, value=undefined}
```

m_content_group

- Module: *mod_content_groups* (page 362)

Not yet documented.

m_custom_redirect

- Module: *mod_custom_redirect* (page 364)

Not yet documented.

m_development

- Module: *mod_development* (page 364)

Not yet documented.

m_edge

- Module: core

See also:

m_rsc (page 571), *m_media* (page 568)

Access information about page connections.

Edges represent the connections between resources. They are implemented as tuples {EdgeId, SubjectId, PredicateId, ObjectId, OrderNr}. The edge id is a unique id representing the edge, it can be used with edit actions. The OrderNr defines the order of the edges *with respect to the subject*.

Most edge information is accessed using the *m_rsc* (page 571) model, but some information can only be accessed with the *m_edge* model.

This model implements two template accessible options. They are mainly used to obtain the edge's id for edit pages.

The following *m_edge* model properties are available in templates:

Property	Description	Example value
o	Returns a function that accepts a page id and a predicate. The end result is a list of tuples {PageId, EdgeId} which are objects of the page. Example usage: <code>m.edge.o[id].author</code>	<code>[[{204,13},{510,14},{508,15}]</code>
s	Identical to the “o” property, except that this function returns the subject edges.	
o_props	Similar to <code>m.edge.o[id].author</code> above, but returns a property list for the edges instead of the 2-tuple.	<code>[{id, 86062}, {subject_id, 10635}, {predicate_id, 304}, {object_id, 57577}, {seq, 1}, {creator_id, 1}, {created, { {2015,11,17}, {11,23,32} }}]</code>
s_props	Similar to <code>m.edge.s[id].author</code> above, but returns a property list for the edges instead of the 2-tuple.	
edges	Returns a function that accepts a page id. The end result is a list of edges per predicate where the predicate is an atom and the edges are property lists. Example usage: <code>m.edge[10635]</code>	See example below.
id	Look up an edge id by a subject/predicate/object triple. Example usage: <code>m.edge.id[subject_id]. ↳relation[object_id]</code> or: <code>m.edge.id[subject_ ↳id][predicate_ ↳name][object_id]</code> Returns undefined if the edge does not exist; otherwise returns an integer.	213

Example return value for `{% print m.edge[10635] %}`:

```
[{about, [[{id, 86060},
  {subject_id, 10635},
  {predicate_id, 300},
  {name, <<"about">>},
  {object_id, 17433},
  {seq, 1},
  {created, {{2015, 11, 17}, {11, 22, 11}}},
  {creator_id, 1}]]},
{author, [[{id, 6},
```

```
{subject_id,10635},
{predicate_id,301},
{name,<<"author">>},
{object_id,10634},
{seq,1000000},
{created,{{2015,2,3},{16,23,20}}},
{creator_id,1}}]}
```

m_editor_tinymce

- Module: *mod_editor_tinymce* (page 367)

Not yet documented.

m_email_dkim

- Module: *mod_email_dkim* (page 368)

DomainKeys Identified Mail Signatures (RFC 6376) is a method to add a signature to outgoing emails. This enables recipients to check who sent an email and if the email was not changed in transit.

This module generates an RSA keypair which will be used when signing outgoing emails.

The keys are stored in the security directory of Zotonic.

If this module is activated then a panel with information is added to the System > Email configuration menu. The panel includes information about the DKIM key that needs to be added to the DNS.

m_email_receive_recipient

- Module: *mod_email_receive* (page 369)

Not yet documented.

m_email_status

- Module: *mod_email_status* (page 370)

Tracks the send/bounce/error status of all outgoing emails.

m_facebook

- Module: *mod_facebook* (page 370)

Not yet documented.

m_filestore

- Module: *mod_filestore* (page 371)

The filestore uses two tables for its administration.

Main administration table

The `filestore` table administrates which file is stored at what service. It also contains flags for flagging files that need to be deleted or moved from the remote service to the local file system.

The definition is as follows:

```
create table filestore (
  id serial not null,
  is_deleted boolean not null default false,
  is_move_to_local boolean not null default false,
  error character varying(32),
  path character varying(255) not null,
  service character varying(16) not null,
  location character varying(400) not null,
  size int not null default 0,
  modified timestamp with time zone not null default now(),
  created timestamp with time zone not null default now(),

  constraint filestore_pkey primary key (id),
  constraint filestore_path_key unique (path),
  constraint filestore_location_key unique (location)
)
```

The *path* is the local path, relative to the site's `files` directory. For example: `archive/2008/12/10/tycho.jpg`

The *service* describes what kind of service or protocol is used for the remote storage. Currently the service is always set to `s3`.

The *location* is the full url describing the location of the file on the remote service. For example: `https://s.greencloud.com/mworrell-zotonic-test/archive/2008/12/10/tycho.jpg`

Upload queue

The upload queue holds file paths of newly added files that need to be uploaded. These file paths are relative to the `files` directory.

Periodically the system polls this table to see if any files need uploading. Any entry older than 10 minutes will be handled and an upload process will be started.

m_fileuploader

- Module: *mod_fileuploader* (page 375)

Model to start uploads, upload a block and delete uploads.

Topics

`model/fileuploader/post/new` starts a new fileupload.

The posted message must include the filename and the file size:

```
{
  filename: "test.jpg",
  size: 1000
}
```

The upload URL and current status is returned:

```
{
  "result": {
    "filename": "test.jpg",
    "is_complete": false,
    "missing": [
      {
        "size": 1000,
        "start": 0
      }
    ],
    "name": "WZkhXoaMwrK2StUHmdpp",
    "received": 0,
    "size": 10,
    "upload_url": "https://zotonic.test:8443/fileuploader/upload/
↪WZkhXoaMwrK2StUHmdpp"
  },
  "status": "ok"
}
```

Status "error" is returned if the name is unknown or any error occurred.

model/fileuploader/post/delete/+name delete a fileupload.

model/fileuploader/post/upload/+name/+offset upload data to a fileupload.

The offset must be an integer and the message payload must be binary data.

model/fileuploader/get/status/+name fetch the current fileupload status.

m_hierarchy

- Module: core

The category hierarchy tables have been replaced by *m_hierarchy*. This model defines named hierarchies of resources (pages).

If the categories are changed then the system needs to update the *pivot_category_nr* field of all resources. With the introduction of *m_hierarchy* this renumbering is much more efficient and will only affect a minimal number of resources.

The *m_hierarchy* module is also used for the content- and user group hierarchies, as used by the new *mod_acl_user_groups* module.

m_identity

- Module: core

See also:

Access control (page 58), *mod_authentication* (page 359), *mod_twitter* (page 401) or *mod_facebook* (page 370).

The *m_identity* model manages usernames and other user identities. *mod_authentication* (page 359) uses it to store and check salted passwords, but also provides a safe storage for user tokens of any kind, as used by *mod_facebook* (page 370) and *mod_twitter* (page 401).

Note that a *user* does not have to be of the *person* category per se, in Zotonic anything can have identities attached to it.

The following *m_identity* model properties are available in templates:

Property	Description	Example value
is_user	Check if a page id is an user. Return a bool. Usage: m.identity[page_id].is_user	true
user-name	Fetch the username, if any, of an user. Returns a binary or undefined. Usage: m.identity[page_id].username	<<"admin">>

m_image_edit

- Module: *mod_image_edit* (page 376)

See also:

image (page 520), *Arguments* (page 521)

Manages the medium_edit_settings property for non destructive image editing.

m_import_csv_data

- Module: *mod_import_csv* (page 377)

Not yet documented.

m_l10n

- Module: *mod_l10n* (page 377)

Not yet documented.

m_linkedin

- Module: *mod_linkedin* (page 377)

Not yet documented.

m_log

- Module: *mod_logging* (page 378)

Not yet documented.

m_log_email

- Module: *mod_logging* (page 378)

Not yet documented.

m_log_ui

- Module: *mod_logging* (page 378)

Not yet documented.

m_mailinglist

- Module: *mod_mailinglist* (page 379)

Not yet documented.

m_media

- Module: core

Access to data about uploaded files and other media.

The `medium` (singular form of `media`) table stores all information of uploaded files or other media. Every resource can contain a single medium. A resource with a medium is most often of the category image, audio, video or document.

In template the `m_media` model is used to fetch the medium record by the resource id: `m.media[id]`. This is the same function as with `m.rsc[id].medium` except, that the `m_rsc` model does access control checks and the `m_media` does not.

The `m_media` model implements all functions to handle media files and is used by other Erlang modules.

Properties of a medium record

A medium record has minimally the following properties, other properties can be added by modules.

Property	Description	Example value
id	Id of the medium record, equal to the page id.	512
filename	Filename and path of the uploaded file, relative to the archive directory.	<<"2009/10/20/zotonic-datamodel.jpg">>
rootname	Root name of the filename.	<<"zotonic-datamodel">>
original_filename	Filename as suggested by the user agent when uploading the file. Can contain illegal characters.	<<"Zotonic-datamodel.jpg">>
mime	Mime type of the medium.	<<"image/jpeg">>
width	Width in pixels.	536
height	Height in pixels.	737
orientation	Exif orientation of the image.	1
sha1	Optional sha1 checksum of uploaded file. Undefined when not present.	
size	Size in bytes of the uploaded file.	71585
pre-view_filename	Optional filename for a generated file preview.	
pre-view_width	Optional. Width of the generated preview.	
pre-view_height	Optional. Height of the generated preview.	
is_deletable_file	Whether the file should be deleted when the medium record is deleted. A boolean.	true
is_deletable_preview	Whether the optionally generated preview file should be deleted when the medium record is deleted. A boolean.	false
created	Timestamp when the medium record is created.	{{2009,10,20},{13,47,27}}

m_microsoft

- Module: *mod_microsoft* (page 381)

Check with `useauth` if the authentication using the Microsoft identity platform is enabled and configured.

m_modules

- Module: core

Access information about which *modules* (page 59) are installed and which ones are active.

To test if a module is activated, for instance `mod_signup`:

```
{% if m.modules.active.mod_signup %}
    {# Do things that depend on mod_signup #}
{% endif %}
```

To print a list of all active modules:

```
{{ m.modules.all|pprint }}
```

m_mqtt_ticket

- Module: *mod_base* (page 360)

A ticketing system for out of band MQTT posts via HTTP.

The controller *controller_mqtt_transport* (page 446) can accept HTTP posts. These posts MUST include a ticket and a topic for the payload to be sent.

The ticket is obtained via `model/mqtt_ticket/post/new`. This can only be accessed via MQTT, as the routines will check for a valid `client_id`.

If a ticket is created then the current request context is saved for max 30 seconds. If a request comes in then the ticket is used to fetch the stored context and the following information is copied over from the stored context to the HTTP request context:

- MQTT client-id
- MQTT client-topic
- MQTT routing-id
- ACL user id
- ACL auth_options
- ACL read-only flag
- Timezone
- Language

A ticket can only be used once.

m_oauth2

- Module: *mod_oauth2* (page 386)

Not yet documented.

m_oauth2_consumer

- Module: *mod_oauth2* (page 386)

Not yet documented.

m_oauth2_service

- Module: *mod_oauth2* (page 386)

Not yet documented.

m_predicate

- Module: core

Retrieve information about predicates. Predicates are the *labels* on edges (connections between resources) that give meaning to an edge. An example is the predicate “author” which refers to the authors of an article. Predicates form together with the referring and the referred page a triple {subject, predicate, object}.

Each predicate has a list of valid subject categories and valid object categories. This is used to filter the list of predicates in the admin edit page, and also to filter the list of found potential objects when making a connection.

A full predicate definition can be fetched by name or id with:

```
{{ m.predicate.author }}
{{ m.predicate[104] }}
```

Which both return a property list with information about the predicate. The property list contains all page properties and the properties: “pred” which is the atomic predicate name, “subject” which is a list of valid subject categories and “object” with is a list of valid object categories.

The following m_predicate model properties are available in templates:

Property	Description	Example value
all	Return a property list of all predicates. Keys are the atomic predicate name, values are property lists with information about the predicate. The property list contains all page properties and the properties: “pred” which is the atomic predicate name, “subject” which is a list of valid subject categories and “object” with is a list of valid object categories.	[{about, [{pred,,about}, {subject,[104]}, {object,[], {id,300}, ... }]
object_category	Used to derive the list of valid object categories for a predicate. Example usage: m.predicate.object_category.author Note: Each id is a 1-tuple.	[{104}, ...]
subject_category	Used to derive the list of valid subject categories for a predicate. Example usage: m.predicate.subject_category.author Note: Each id is a 1-tuple.	[{674}, ...]

m_ratelimit

- Module: [mod_ratelimit](#) (page 388)

Not yet documented.

m_req

- Module: core

This model gives access to the request variables from within a template.

Sometimes you need direct access to request variables in your template. The m_req model is meant for this. It exposes some values from the Cowmachine request record.

Fetching a single value

You can fetch individual values by key, for example:

```
{{ m.req.host|escape }}
```

Viewing all request variables

Use the *print* (page 530) tag to get a complete overview of all request variables:

```
{% print m.req|make_list %}
```

This will show something like:

```
[{method,<<"GET">>},
 {version,{1,1}},
 {peer,<<"127.0.0.1">>},
 {is_ssl,false},
 {host,<<"mysite.test">>},
 {raw_path,<<"/en/page/1234?foo=bar">>},
 {path,<<"/en/page/1234">>},
 {qs,[{<<"foo">>,<<"bar">>}]},
 {referrer,<<"http://mysite.test:8000/">>},
 {user_agent,<<"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/601.4.
↪4 (KHTML, like Gecko) Version/9.0.3 Safari/601.4.4">>},
 {is_crawler,false},
 {req_id,525158920},
 {headers,[{<<"accept">>,
  <<"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8">>},
  {<<"accept-encoding">>,<<"gzip, deflate">>},
  {<<"accept-language">>,<<"en-us">>},
  {<<"cache-control">>,<<"max-age=0">>},
  {<<"connection">>,<<"keep-alive">>},
  {<<"cookie">>,
    "z_logon=; z_sid=LopWHBmHXC94virnboZhBHLKV6m1Cga; z_ua=c%3Ddesktop%26u
↪%3D1%26t%3D1%26w%3D1920%26h%3D1200"},
  {<<"dnt">>,<<"1">>},
  {<<"host">>,<<"mysite.test:8000">>},
  {<<"referer">>,<<"http://mysite.test:8000/">>},
  {<<"user-agent">>,
    <<"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/601.4.4
↪(KHTML, like Gecko) Version/9.0.3 Safari/601.4.4">>}]},
 {timezone,<<"UTC">>},
 {language,en}]
```

Please note that all values are raw and not escaped, take care to escape the values before you use them in your templates, using the *escape* (page 465) filter.

The *make_list* (page 478) filter is used to force the evaluation of the model; otherwise it would just print {m, req, undefined}.

m_rsc

- Module: core

See also:

Resources (page 26), *The Zotonic data model* (page 5), *m_edge* (page 562), *m_media* (page 568), *m_rsc_gone* (page 577).

The main resource model, which is the central part of the *Zotonic data model* (page 5). This model provides an interface to all resource (“page”) information. It also provides an easy way to fetch edges from pages without needing to use the *m_edge* (page 562) model.

Properties of the resource model

A resource has the following properties accessible from the templates:

Property	Description	Example value
id	Id of the page, an integer.	42
title	Title of the page. Returns a binary.	<<"Breaking News">>
short_title	Short title of the page. Used in menus. Returns a binary.	<<"News!">>
summary	Summary of the page. Returns a binary or undefined.	<<"Page summary.">>
body	The HTML body of a page. Returns a binary or undefined.	<<"<p>Hello</p>">>
date_start	Start date when the page has a period. Examples are events or the birth date of a person. Returns a datetime tuple or undefined.	{{2008,12,10},{15,30,00}}
date_end	End date when the page has a period. Returns a datetime tuple or undefined. When there is a start date then there is also an end date.	{{2009,12,5},{23,59,59}}
name	Unique name of the page. Returns a binary or undefined. Valid characters are a-z, 0-9 and _	<<"page_home">>
page_path	Unique path of the page, used for url generation. Returns a binary or undefined. Valid characters are a-z, 0-9, / and -	<<"/">>
is_page_path_multiple	Allow the page to be served on multiple URLs	false
page_url	The url of the page. Derived using the page's category, the page id and its slug. Returns a non flattened list. Returns the binary page_path when it is set. The additional parameter <code>with</code> can be used to pass extra (optional) query arguments to the url, for instance: <pre>{{ id.page_url with t=now date:"U" }} {{ id.page_url with t="new" u=m.acl.user.id }}</pre>	<<"/blog/42">>
page_url_abs	The absolute url of the page. Same as <code>page_url</code> but then with added protocol, hostname and port.	<<"http://example.org/blog/42">>
default_page_url	The page without considering its <code>page_path</code> setting.	<<"/page/42/my-slug">>
is_authoritative	Whether this page originated on this site or is imported and maintained on another site. Return a boolean.	true

Continued on next page

Table 6.3 – continued from previous page

Property	Description	Example value
uri	The absolute unique uri of this resource. Refers to the “id” dispatch rule for authoritative (local) resources. Returns a binary.	<<“http://example.com/id/42”>>
category_id	Id of the category the page belongs to. Returns an integer.	102
category	Category record the page belongs to. Returns a property list.	[{id,102},{parent_id,undefined}, ... ,{name, <<“person”>>}]
seo_noindex	Whether to let search engines index this page. Returns a boolean or undefined.	false
slug	Slug used for url generation, appended to page urls. Binary or undefined. Valid characters are a-z, 0-9 and -	<<“the-world-is-flat”>>
seo_desc	Page description for search engines. Returns a binary or undefined.	<<“The truth about the world’s shape”>>
is_me	Check if this page is the current user’s person page. Returns a boolean.	false
is_visible	Check if this page is visible for the current user. Returns a boolean.	true
is_editable	Check if this page is editable by the current user. Returns a boolean.	false
is_linkable	Check if this page can be connected to another page. Returns a boolean.	false
isingroup	Check if the current user is a member of the group the page belongs to. Returns a boolean.	true
exists	Check if the page exists. Useful when checking if a named page is present or not. Returns a boolean.	true
is_a	Returns a list of the category hierarchy the page belongs to. The list is suitable for indexing with category atoms. Example usage: <pre>{{ m.rsc[id].is_a.article }}</pre>	[{text,true}, {article,true}]
is_cat	Direct check if a page is a certain category. More efficient than is_a. Example usage: <pre>{{ m.rsc[id].is_cat.person }}</pre>	true
is_featured	If featured checked or not. Returns a boolean	false

Continued on next page

Table 6.3 – continued from previous page

Property	Description	Example value
is_protected	If this page is protected from deletion. Returns a boolean. Resources are protected by a simple table called <code>protect</code> that prevents accidental deletions of <code>rsc</code> records. It does this by having a foreign key constraint that prohibits the deletion of the referred <code>rsc</code> record.	false
is_dependent	If set to <i>true</i> then this page should only exist if there are incoming edges to this page. This flag is checked when an edge is deleted.	true
is_published	If this page has been published. Returns a boolean	true
publication_start	Start date of the publication period. Returns a datetime tuple.	{{2009,12,24},{9,0,0}}
publication_end	End date of the publication period. Returns a datetime tuple.	{{9999,8,17},{12,0,0}}
is_published_date	If this page is published and the current date/time is within the set <code>publication_start/end</code> range. Note that no ACL checks are performed, use <i>is_visible</i> to check if a resource is visible for the current user.	true
visible_for	Visibility level. Returns an integer. The actual meaning depends on the active ACL module.	0
content_group_id	Content group this resource belongs to. Defaults to the id of the content group named <i>default_content_group</i> or <i>system_content_group</i> for resources within the <i>meta</i> category. See mod_content_groups (page 362)	31415
o	Used to access the objects of page: the pages this page refers to. Returns a function which should be indexed with the edge's predicate name (atom). When indexed the function will return a list of integers. Example usage: <pre>{{ m.rsc[id].o.author[1].title }}</pre> This returns the first author that is linked from this page.	fun(Predicate,Context)

Continued on next page

Table 6.3 – continued from previous page

Property	Description	Example value
s	Access the subjects of a page: the pages that are referring to this page. Returns a function which should be indexed with the edge's predicate name (atom). When indexed the function will return a list of integers. Example usage: <pre>{{ m.rsc[id].s.author[1].title }}</pre> This returns the first article that links to me with a author connection.	fun(Predicate,Context)
op	Returns a list of all predicates on edges from this page. The predicates are atoms.	[about, related]
sp	Returns a list of all predicates on edges to this page. The predicates are atoms.	[author]
predicates_edit	Returns a list of all allowed predicates from this page. Used for editing the page. Returns a list of predicate ids (in contrast with the atoms of op and sp).	[308,300,304,303,302,300]
media	Return a list of all media ids connected to the page. The media are connected with the predicate “depiction”.	[842,3078]
medium	Return a property list describing the file or medium attached to the page. A medium record is present for pages that are an image, video etc. Returns undefined when there is no medium defined. See the model m_media for more information.	[{id,512}, {filename, <<“2009/1...”>>, ... }]
depiction	Return the medium record that can be used for the image of a page. Either returns a medium page attached to the page or the medium record of the page itself. When no medium is found then undefined is returned.	[{id,512}, {filename, <<“2009/1...”>>, ... }]
image_url	An url of the depiction, using the mediaclass image defined in mod_base	<<“/image/...”>>
image_url_abs	The absolute image_url, that is including https:	<<“http://example.com/...”>>
thumbnail_url	An url of the depiction, using the mediaclass thumbnail defined in mod_base	<<“/image/...”>>

Continued on next page

Table 6.3 – continued from previous page

Property	Description	Example value
thumbnail_url_abs	The absolute <code>thumbnail_url</code> , including https:	<<"http://example.com/...">>
email	E-mail address. Returns a binary or undefined.	<<"me@example.com">>
website	URL of a website. Returns a binary or undefined.	<<"http://zotonic.com">>
is_website_redirect	Tell <i>controller_page</i> to redirect to the website URL instead of showing the HTML page.	false
phone	Phone number. Returns a binary or undefined.	<<"+31201234567">>
phone_alt	Alternative phone number. Returns a binary or undefined.	undefined
phone_emergency	Phone number to call in emergencies.	<<"112">>
address_street_1	Address line 1. Returns a binary or undefined.	
address_street_2	Address line 2. Returns a binary or undefined.	
address_city	City part of address. Returns a binary or undefined.	
address_postcode	Postcode part of address. Returns a binary or undefined.	
address_state	State part of address. Returns a binary or undefined.	
address_country	Country part of address. Returns a binary or undefined.	
mail_street_1	Mailing address line 1. Returns a binary or undefined.	
mail_street_2	Mailing address line 2. Returns a binary or undefined.	
mail_city	City part of mailing address. Returns a binary or undefined.	
mail_postcode	Postcode part of mailing address. Returns a binary or undefined.	
mail_state	State part of mailing address. Returns a binary or undefined.	
mail_country	Country part of mailing address. Returns a binary or undefined.	
name_first	First name of person. Returns a binary or undefined.	
name_middle	Middle name of person. Returns a binary or undefined.	
name_surname_prefix	Prefix for the surname of a person. Returns a binary or undefined.	<<"van der">, <<"von">>
name_surname	Surname or family name of person. Returns a binary or undefined.	

Escaping

All strings that are stored inside resources are automatically *HTML escaped*. This means that these texts do not require any processing when they are being displayed in the browser, which causes major performance gains.

There are some fields in the resource that are exceptions to these rule, namely, the `body` field and any fields whose name ends in `_html`. These fields are assumed to contain HTML text and are sanitized on save instead of

escaped.

Dates

Dates are stored as a standard Erlang date time tuple, for example `{{2008,12,10},{15,30,00}}`. Dates are stored and retrieved in UTC (universal time). When displaying a date, (e.g. with the [date](#) (page 457) filter), the date is automatically converted into the time zone of the site or that of the user.

Printing all properties of a resource

In your templates, you can loop over the properties of a resource like this:

```
{% for k,v in m.rsc[id] %}
  {{ k }} - {{ v }} <br/>
{% endfor %}
```

And also using the [print](#) (page 530) tag:

```
{% print m.rsc[id] %}
```

m_rsc_gone

- Module: core

See also:

[The Zotonic data model](#) (page 5), [m_rsc](#) (page 571)

This model tracks deleted resources (see [m_rsc](#) (page 571)). Its primary goal is to be able to determine if a resource never existed, has been deleted or has been replaced by another resource.

Information kept

Only very basic information of the deleted resource is kept in the `rsc_gone` table. It is enough for referring to a new location, giving correct errors or to determine who deleted a resource.

It is not enough to undelete a resource. The module [mod_backup](#) (page 360) retains enough information about past versions to be able to undelete a resource. Currently there is no support for an undelete.

Properties

Whenever a [m_rsc](#) (page 571) record is deleted some information from that resource is copied to the `rsc_gone` table.

The following properties are saved:

Property	Description	Example value
id	Id of the resource, an integer.	42
new_id	If the resource is replaced by another resource then this is the id of that other resource.	341
new_uri	If the resource is moved to another place on the web then this is the uri of the new location.	<<"http://example.com/hello">>
name	The name (if any) of the deleted resource.	<<"page_hello">>
uri	The uri of the authoritative source for the resource.	<<"http://foo.bar/hello">>
page_path	The page path (if any) of the deleted resource.	<<"/hello">>
is_authoritative	Whether the resource originated on this site or was imported and maintained on another site. Return a boolean.	true
creator_id	The id of the creator of the deleted resource.	1
created	The date the deleted resource was created.	{{2008,12,10},{15,30,00}}
modifier_id	The id of the user that deleted the resource.	2718
modified	The date the resource was deleted.	{{2012,12,5},{23,59,59}}

m_search

- Module: core

See also:

Search (page 44) , *pager* (page 544) tag , *mod_search* (page 388) module , *Custom search* (page 323)

The m_search model provides access to different kinds of search queries for searching through models.

Most searches in Zotonic are implemented in the *mod_search* (page 388) module, searching through the `rsc` table in different kinds of ways.

Though, any module can implement a search by observing the `search_query` notification.

The search module is used inside templates. For example, the following snippet fetches the latest 10 modified pages in the “text” category:

```
{% for id in m.search[{{latest cat="text" pagelen=10}} %}
  {{ m.rsc[id].title }}
{% endfor %}
```

Another example, searching for a text and requesting the second page with 20 results at a time:

```
{% for id, rank in m.search.paged[{{fulltext text=query_string page=2 pagelen=20}}
↔ %}
  {{ m.rsc[id].title }}
{% endfor %}
```

m_seo

- Module: *mod_seo* (page 389)

Not yet documented.

m_seo_sitemap

- Module: *mod_seo_sitemap* (page 390)

Not yet documented.

m_server_storage

- Module: *mod_server_storage* (page 391)

Model to access the server side storage of data.

See the module documentation of *mod_server_storage* (page 391) for information.

m_signup

- Module: *mod_signup* (page 393)

Exported APIs:

- **model/signup/get/confirm_redirect** return the URL for redirecting the current user after a signup. This is done by calling (first) the `#signup_confirm_redirect{ id = UserId }` notification. If `undefined` is returned then the URL defaults to the personal page of the user. If no user is logged on then the URL of the home page (dispatch rule `home`) is returned.

m_site

- Module: `core`

Retrieve information that is stored in the *site configuration* (page 626). If you want to query values from the config table instead, you should use *m_config* (page 560).

Note: In general the site configurarion is only accessible via the `m.site` template model for users with administrator rights. Exceptions are `{{ m.site.title }}`, hostname configurations and the *paglen*.

Fetch a site configuration key

Example, fetching the site configuration key “hostname”:

```
{{ m.site.hostname }}
```

Fetching all configurations

It is easy to loop over all site configurations:

```
{% for key, value in m.site %}
  {{ key }} -- {{ value }} <br>
{% endfor %}
```

Overriding config values

Zotonic has two places where a site’s configuration is kept. One is in the site’s config files, the other in the config table. The config table (accessible through *m_config* (page 560)) overrules any module settings from the config file, for rows where the *module* key of the config value is set to *site*.

Within the site configuration, you can override module-specific configuration: Module configurations are defined with a property key equal to the module name, with a property list as value. For example:

```
{mod_foo, [ {hostname, "localhost"} ]}
```

will put the default “hostname” setting of the imaginary `mod_foo` module to `localhost`.

Default config values

Sites have the following default config settings:

Property	Description	Example value
environ-ment	Set the DTAP status of the site. Can be one of production, development, test, acceptance, education or backup. Default: production	develop-ment
hostname	The hostname of the site	exam-ple.com
title	The title of the site.	“My Awesome Blog”
protocol	The main protocol of the site. Used to construct urls. Default “http”.	“https”
docu-ment_domain	The document domain used for cross domain iframe javascripts. Default is the same as the cookie_domain.	www.example.com
cookie_domain	The domain to use on cookies. This defaults to undefined, which will equal the domain of the current request.	.exam-ple.com
ses-sion_expire_1	The initial timemout after setting up a process and waiting for the first ping from the page. Default: 40 (seconds)	10
ses-sion_expire_n	Session inactivity timeout after receicing some communication from the page. Default: 3600 (1 hour)	120 (2 minutes)
ses-sion_expire_ina-ctive	User inactivity timeout after seeing that the user has not been active. Default: 14400 (4 hours)	3600 (1 hour)
site	The name of the site, an atom.	wwwzo-tonic.
hsts	Indicate if the site should use Strict Transport Security. When set, the browser will no longer use insecure http to access the site. Warning: Be sure your site is accessible via https before enabeling this feature. Default: false	true
hsts_maxage	The time, in seconds which browsers are allowed to remember the HSTS setting of the site. Default: 17280000 (200 days)	
hsts_include_subdomains	When set, the browser also does not use http for subdomains. Default: false	
hsts_preload	When set, the site’s HSTS setting will be stored by the browser vender. This means that browsers only use https. Use with care, because it is hard to revert wrong settings. Default: false	

m_site_update

- Module: *mod_site_update* (page 394)

Model for checking if a site has version control enabled.

Also implements the webhook to force a pull of new code from the version control system.

The URL for the webhook is:

```
https://yoursite.test/api/model/site_update/post/webhook/<token>
```

Where <token> should be replaced with your configured token.

The token can be configured in the admin on System > Modules and then the config of the mod_site_update module.

The token is saved in the config key `mod_site_update.webhook_token`.

m_ssl_letsencrypt

- Module: *mod_ssl_letsencrypt* (page 397)

Not yet documented.

m_survey

- Module: *mod_survey* (page 398)

Not yet documented.

m_sysconfig

- Module: core

Note: System configurations are only accessible from templates, using `m.sysconfig`, for users with administrator rights.

Gives access to the Zotonic system configuration from the `zotonic.config` file(s).

m_template

- Module: core

Todo

Not yet documented.

m_tkvstore

- Module: *mod_tkvstore* (page 400)

See also:

mod_tkvstore (page 400)

Simple read-only interface to the typed key-value store of *mod_tkvstore* (page 400). To get a value from the store: use `m.tkvstore.type.key`, like this:

```
{% print m.tkvstore.keytype.examplekey %}
```

When the key is not a literal value but a variable or a number, use the following notation:

```
{% print m.tkvstore.keytype[keyvar] %}
```

To store data in a key, use `m_tkvstore:put/4`, as follows:

```
m_tkvstore:put(KeyType, KeyVar, Value, Context).
```

For instance, from within Erlang, let's store *edge data* for a given edge id, 3434:

```
Edgeid = 3434,
Value = <<"Hello">>,
m_tkvstore:put(edge_data, EdgeId, Value, Context).
```

Now, to retrieve it in the template, do the following:

```
{{ m.tkvstore.edge_data[3434] }}
```

This will output the string `Hello`.

Note that the value can be any type: not only a simple string but also a list, tuple, or any other Erlang composite type.

m_translation

- Module: *mod_translation* (page 400)

The `m_translation` model gives easy access to language and translation related information.

The following `m_translation` model properties are available in templates:

Property	Description
<code>language</code>	The current language.
<code>language_list</code>	The list of all configured languages.
<code>language_list_enabled</code>	The list of all enabled languages.

This is an example of the languages returned by `m_translation.language_list`:

```
[{en, [{is_enabled,true}, {language,<<"English">>}]},  
{fr, [{is_enabled,false}, {language,<<"Français">>}]},  
{nl, [{is_enabled,true}, {language,<<"Nederlands">>}]},  
{tr, [{is_enabled,true}, {language,<<"Türkçe">>}]}
```

For example to list all enabled languages in a select box:

```
<select>  
{% for code,props in m_translation.language_list_enabled %}  
  <option value="{{ code }}" {% if m_translation.language == code %}selected{%_  
→endif %}>{{ props.language }}</option>  
{% endfor %}  
</select>
```

m_twitter

- Module: *mod_twitter* (page 401)

Not yet documented.

6.1.8 Validators

See also:

The *validate* (page 550) tag

See also:

Forms and validation (page 37) in the Developer Guide

Validator types for the tag `{% validate %}`.

6.1.9 Notifications

This is a list of all built-in *notifications* (page 69) that Zotonic sends. *Observe* (page 70) these notifications in your code to add custom functionality.

ACL notifications

acl_context_authenticated

Set the context to a typical authenticated user. Used by `m_acl.erl`

Type: *first* (page 70)

Return: authenticated `#context{}` or undefined

```
#acl_context_authenticated{} properties: none
```

acl_is_allowed

Check if a user is authorized to perform an operation on a an object (some resource or module). Observe this notification to do complex or more fine-grained authorization checks than you can do through the ACL rules admin interface. Defaults to `false`.

Type: *first* (page 70)

Return: `true` to allow the operation, `false` to deny it or `undefined` to let the next observer decide

#acl_is_allowed{} properties:

- `action`: `view|update|delete|insert|use|atom`
- `object`: `term`

Example

Deny anyone from viewing unpublished resource except those who have update rights on the resource (usually the creator and the administrator):

```
observe_acl_is_allowed(#acl_is_allowed{action = view, object = Id}, Context) ->
  case m_rsc:p_no_acl(Id, is_published_date, Context) of
    undefined ->
      %% Let next observer decide
      undefined;
    true ->
      %% Resource is published: let next observer decide
      undefined;
    false ->
      %% Resource is unpublished
      case z_acl:is_allowed(update, Id, Context) of
        true ->
          %% User has update rights, so let next observer decide
          undefined;
        false ->
          %% Deny viewing rights on unpublished resource
          false
      end
  end;
observe_acl_is_allowed(#acl_is_allowed{}, _Context) ->
  %% Fall through
  undefined.
```

In this observer, we return `undefined` in those cases where we do not want to deny access. We don't grant the access right away but give the next observer the change to decide whether viewing is allowed (for instance, based on the resource's category and content group and the user's group).

acl_is_allowed_prop

Check if a user is authorized to perform an action on a property. Defaults to `true`.

Type: *first* (page 70)

Return: `true` to grant access, `false` to deny it, `undefined` to let the next observer decide

#acl_is_allowed_prop{} properties:

- `action`: `view|update|delete|insert|atom`

- object: term
- prop: binary

acl_is_owner

Check if a user is the owner of a resource. `id` is the resource id.

Type: *first* (page 70)

Return: `true`, `false` or `undefined` to let the next observer decide

#acl_is_owner{ } properties:

- `id`: `m_rsc:resource_id()`
- `creator_id`: `m_rsc:resource_id()`
- `user_id`: `m_rsc:resource_id()`

acl_logoff

Clear the associated access policy for the context.

Type: *first* (page 70)

Return: `updated z:context()` or `undefined`

#acl_logoff{ } properties: none

acl_logon

Initialize context with the access policy for the user.

Type: *first* (page 70)

Return: `updated z:context()` or `undefined`

#acl_logon{ } properties:

- `id`: `m_rsc:resource_id()`
- `options`: map

acl_mqtt

MQTT acl check, called via the normal acl notifications. Actions for these checks: subscribe, publish

Type: *first* (page 70)

Return:

#acl_mqtt{ } properties:

- `topic`: list
- `is_wildcard`: boolean
- `packet`: `mqtt_packet_map:mqtt_packet()`

acl_user_groups

Return the groups for the current user.

Type: *first* (page 70)

Return: [m_rsc:resource_id()] or undefined

#acl_user_groups{} properties: none

acl_user_groups_modify

Modify the list of user groups of an user. Called internally by the ACL modules when fetching the list of user groups an user is member of.

Type: *foldl* (page 70)

Return: [m_rsc:resource_id()]

#acl_user_groups_modify{} properties:

- id: m_rsc:resource_id() | undefined
- groups: list

Authentication notifications

See also:

user_is_enabled (page 610)

auth_checked

Fold over the context after logon of user with username, communicates valid or invalid password

Type: *first* (page 70)

Return:

#auth_checked{} properties:

- id: undefined|m_rsc:resource_id()
- username: binary
- is_accepted: boolean

auth_confirm

Confirm a user id.

Type: *foldl* (page 70)

Return: z:context()

#auth_confirm{} properties:

- id: m_rsc:resource_id()

auth_confirm_done

A user id has been confirmed.

Type: *notify* (page 69)

Return:

#auth_confirm_done{} properties:

- `id: m_rsc:resource_id()`

auth_client_logon_user

Send a request to the client to login an user. The `zotonic.auth.worker.js` will send a request to `controller_authentication` to exchange the one time token with a `z.auth` cookie for the given user. The client will redirect to the Url.

Type: *first* (page 70)

Return: `ok` | `{error, term() }`

#auth_client_logon_user{} properties:

- `user_id: m_rsc:resource_id()`
- `url: union`

auth_client_switch_user

Send a request to the client to switch users. The `zotonic.auth.worker.js` will send a request to `controller_authentication` to perform the switch.

Type: *first* (page 70)

Return: `ok` | `{error, term() }`

#auth_client_switch_user{} properties:

- `user_id: m_rsc:resource_id()`

auth_logon

User logs on. Add user-related properties to the logon request context.

Type: *foldl* (page 70)

Return: `z:context()`

#auth_logon{} properties:

- `id: m_rsc:resource_id()`

auth_logoff

User is about to log off. Modify (if needed) the logoff request context.

Type: *foldl* (page 70)

Return: `z:context()`

#auth_logoff{} properties:

- `id: m_rsc:resource_id() | undefined`

auth_options_update

Update the given (accumulator) authentication options with the request options. Note that the request options are from the client and are unsafe.

Type: *foldl* (page 70)

Return: map ()

#auth_options_update{} properties:

- request_options: map

auth_postcheck

First for logon of user with username, called after successful password check.

Type: *first* (page 70)

Return: 'undefined' | ok | {error, Reason}

#auth_postcheck{} properties:

- id: m_rsc:resource_id()
- query_args: map

auth_precheck

First for logon of user with username, check for ratelimit, blocks etc.

Type: *first* (page 70)

Return: 'undefined' | ok | {error, Reason}

#auth_precheck{} properties:

- username: binary

auth_reset

First to check for password reset forms, return undefined, ok, or {error, Reason}.

Type: *first* (page 70)

Return:

#auth_reset{} properties:

- username: undefined|binary

auth_validate

First to validate a password. Return {ok, RscId} or {error, Reason}.

Type: *first* (page 70)

Return:

#auth_validate{} properties:

- username: undefined|binary
- password: undefined|binary

auth_validated

Authentication against some (external or internal) service was validated

Type: *first* (page 70)

Return:

#auth_validated{ } properties:

- service: atom
- service_uid: binary
- service_props: map
- unknown: unknown
- identities: list
- is_connect: boolean
- is_signup_confirm: boolean

Dispatch notifications

content_types_dispatch

Get available content types and their dispatch rules Example: {"text/html", page} A special dispatch rule is 'page_url', which refers to the page_url property of the resource.

Type: *foldr* (page 70)

Return: [{ContentType, DispatchRule}]

#content_types_dispatch{ } properties:

- id: m_rsc:resource()

dispatch

Final try for dispatch, try to match the request. Called when the site is known, but no match is found for the path

Type: *first* (page 70)

Return: {ok, RscId::integer()}, {ok, #dispatch_match{}}, {ok, #dispatch_redirect{}} or undefined

#dispatch{ } properties:

- host: binary
- path: binary
- method: binary
- protocol: union
- tracer_pid: union

dispatch_host

Try to find the site for the request Called when the request Host doesn't match any active site.

Type: *first* (page 70)

Return: {ok, #dispatch_redirect{}} or undefined

#dispatch_host{} properties:

- host: binary
- path: binary
- method: binary
- protocol: union

dispatch_rewrite

Rewrite a URL before it will be dispatched using the `z_sites_dispatcher`

Type: *foldl* (page 70)

Return:

#dispatch_rewrite{} properties:

- is_dir: boolean
- path: binary
- host: unknown

page_url

Fetch the url of a resource's html representation

Type: *first* (page 70)

Return: {ok, Url} or undefined

#page_url{} properties:

- id: `m_rsc:resource_id()`
- is_a: list

url_abs

Make a generated URL absolute, optionally called after `url_rewrite` by `z_dispatcher`

Type: *first* (page 70)

Return:

#url_abs{} properties:

- url: unknown
- dispatch: unknown
- dispatch_options: unknown

url_rewrite

Rewrite a URL after it has been generated using the `z_dispatcher`

Type: *foldl* (page 70)

Return:

#url_rewrite{} properties:

- dispatch: atom

- args: list

Edge notifications

edge_delete

An edge has been deleted. Note that the Context for this notification does not have the user who deleted the edge.

Type: *notify* (page 69)

Return: return value is ignored

#edge_delete{} properties:

- subject_id: m_rsc:resource()
- predicate: atom
- object_id: m_rsc:resource()
- edge_id: pos_integer

Example

Perform some action when an edge is deleted:

```
-include_lib("zotonic_core/include/zotonic.hrl").
-export([observe_edge_delete/2]).

observe_edge_delete(#edge_delete{edge_id = Id}, Context) ->
    %% Consult the edge_log table to get the late edge's details
    Edge = z_db:assoc_row("select * from edge_log where edge_id = $1", [Id], Context),

    ?DEBUG(Edge),
    %% logged is when the deletion was logged; created is when the edge was
    %% originally created
    %% [{id,11},{op,<<"DELETE">>},{edge_id,25},{subject_id,341},{predicate_id,300},
    <<{predicate,<<"about">>},{object_id,338},{seq,1000000},{logged,{2016,10,13},{10,
    <<23,21}}},{created,{2016,10,13},{10,23,13}}}]

    %% Do something...

    ok.
```

edge_insert

An edge has been inserted. Note that the Context for this notification does not have the user who created the edge.

Type: *notify* (page 69)

Return: return value is ignored

#edge_insert{} properties:

- subject_id: m_rsc:resource()
- predicate: atom
- object_id: m_rsc:resource()
- edge_id: pos_integer

edge_update

An edge has been updated Note that the Context for this notification does not have the user who updated the edge.

Type: *notify* (page 69)

Return: return value is ignored

#edge_update{ } properties:

- subject_id: m_rsc:resource()
- predicate: atom
- object_id: m_rsc:resource()
- edge_id: pos_integer

E-mail notifications

email_add_handler

Add a handler for receiving e-mail notifications

Type: *first* (page 70)

Return: {ok, LocalFrom}, the unique localpart of an e-mail address on this server.

#email_add_handler{ } properties:

- notification: unknown
- user_id: unknown
- resource_id: unknown

email_bounced

Bounced e-mail notification. The recipient is the e-mail that is bouncing. When the the message_nr is unknown the it is set to 'undefined'. This can happen if it is a "late bounce". If the recipient is defined then the Context is the depickled z_email:send/2 context.

Type: *notify* (page 69)

Return:

#email_bounced{ } properties:

- message_nr: binary|undefined
- recipient: binary|undefined

email_dkim_options

Return the options for the DKIM signature on outgoing emails. Called during email encoding.

Type: *first* (page 70)

Return: list() options for the DKIM signature

#email_dkim_options{ } properties: none

email_drop_handler

Drop an e-mail handler for a user/resource id. (notify). The notification, user and resource should be the same as when the handler was registered.

Type: *first* (page 70)

Return:

#email_drop_handler{ } properties:

- notification: unknown
- user_id: unknown
- resource_id: unknown

email_ensure_handler

Add a handler for receiving e-mail notifications

Type: *first* (page 70)

Return: {ok, LocalFrom}, the unique localpart of an e-mail address on this server.

#email_ensure_handler{ } properties:

- notification: unknown
- user_id: unknown
- resource_id: unknown

email_failed

Notify that we could NOT send an e-mail (there might be a bounce later...) The Context is the depickled z_email:send/2 context.

Type: *notify* (page 69)

Return:

#email_failed{ } properties:

- message_nr: binary
- recipient: binary
- is_final: boolean
- reason: bounce|retry|illegal_address|smtp_host|sender_disabled|error
- retry_ct: non_neg_integer|undefined
- status: binary|undefined

email_is_blocked

Check if an email address is blocked

Type: *first* (page 70)

Return:

#email_is_blocked{ } properties:

- recipient: binary

email_received

Notification sent to a site when e-mail for that site is received

Type: *first* (page 70)

Return:

#email_received{ } properties:

- to: unknown
- from: unknown
- localpart: unknown
- localtags: unknown
- domain: unknown
- reference: unknown
- email: unknown
- headers: unknown
- is_bulk: boolean
- is_auto: boolean
- decoded: unknown
- raw: unknown

email_send_encoded

Add a handler for receiving e-mail notifications

Type: *first* (page 70)

Return: {ok, LocalFrom}, the unique localpart of an e-mail address on this server.

#email_send_encoded{ } properties:

- message_nr: binary
- from: binary
- to: binary
- encoded: binary
- options: gen_smtp_client:options()

email_sent

Notify that we could NOT send an e-mail (there might be a bounce later...) The Context is the depickled z_email:send/2 context.

Type: *notify* (page 69)

Return:

#email_sent{ } properties:

- message_nr: binary
- recipient: binary
- is_final: boolean

email_status

Email status notification, sent when the validity of an email recipient changes

Type: *notify* (page 69)

Return:

#email_status{ } properties:

- recipient: binary
- is_valid: boolean
- is_final: boolean
- is_manual: boolean

Import/export notifications

import_csv_definition

Find an import definition for a CSV file by checking the filename of the to be imported file.

Type: *first* (page 70)

Return: #import_csv_definition{ } or undefined (in which case the column headers are used as property names)

#import_csv_definition{ } properties:

- basename: binary
- filename: file:filename_all()

import_resource

An external feed delivered a resource. First handler can import it. Return:: {ok, m_rsc:resource_id()}, {error, Reason}, or undefined

Type: *first* (page 70)

Return:

#import_resource{ } properties:

- source: atom|binary
- source_id: integer|binary
- source_url: binary
- source_user_id: binary|integer
- user_id: integer
- name: binary
- props: m_rsc:props_all()
- urls: list
- media_urls: list
- data: any

export_resource_content_disposition

mod_export - return the {ok, Disposition} for the content disposition.

Type: *first* (page 70)

Return: {ok, <<"inline">>} or {ok, <<"attachment">>}

#export_resource_content_disposition{} properties:

- dispatch: atom
- id: m_rsc:resource_id() | undefined
- content_type: binary

export_resource_content_type

mod_export -

Type: *first* (page 70)

Return: {ok, "text/csv"}) for the dispatch rule/id export.

#export_resource_content_type{} properties:

- dispatch: atom
- id: m_rsc:resource_id() | undefined

export_resource_data

mod_export - fetch a row for the export, can return a list of rows, a binary, and optionally a continuation state. Where Values is [term()], i.e. a list of opaque values, to be formatted with #export_resource_format. Return the empty list of values to signify the end of the data stream.

Type: *first* (page 70)

Return: {ok, Values|binary()}, {ok, Values|binary(), ContinuationState} or {error, Reason}

#export_resource_data{} properties:

- dispatch: atom
- id: m_rsc:resource_id() | undefined
- content_type: binary
- state: term

export_resource_encode

mod_export - Encode a single data element.

Type: *first* (page 70)

Return: {ok, binary()}, {ok, binary(), ContinuationState} or {error, Reason}

#export_resource_encode{} properties:

- dispatch: atom
- id: m_rsc:resource_id() | undefined
- content_type: binary

- data: term
- state: term

export_resource_filename

mod_export - return the {ok, Filename} for the content disposition.

Type: *first* (page 70)

Return: {ok, Filename} or undefined

#export_resource_filename{ } properties:

- dispatch: atom
- id: m_rsc:resource_id() | undefined
- content_type: binary

export_resource_footer

mod_export - Fetch the footer for the export. Should cleanup the continuation state, if needed.

Type: *first* (page 70)

Return: {ok, binary()} or {error, Reason}

#export_resource_footer{ } properties:

- dispatch: atom
- id: m_rsc:resource_id() | undefined
- content_type: binary
- state: term

export_resource_header

mod_export - Fetch the header for the export.

Type: *first* (page 70)

Return: {ok, list() | binary()}, {ok, list() | binary(), ContinuationState} or {error, Reason}

#export_resource_header{ } properties:

- dispatch: atom
- id: m_rsc:resource_id() | undefined
- content_type: binary

export_resource_visible

mod_export - Check if the resource or dispatch is visible for export.

Type: *first* (page 70)

Return: true or false

#export_resource_visible{ } properties:

- dispatch: atom

- `id: m_rsc:resource_id() | undefined`

Media notifications

media_identify_extension

Try to find a filename extension for a mime type (example: ".jpg")

Type: *first* (page 70)

Return: Extension (for example <<" .png">>) or undefined

#media_identify_extension{} properties:

- `mime: binary`
- `preferred: undefined|binary`

media_identify_file

Try to identify a file, returning a map with file properties.

Type: *first* (page 70)

Return: map with binary keys, especially <<"mime">>, <<"width">>, <<"height">>, <<"orientation">>

#media_identify_file{} properties:

- `filename: file:filename_all()`
- `original_filename: binary`
- `extension: binary`

media_import

Notification to translate or map a file after upload, before insertion into the database Used in `mod_video` to queue movies for conversion to mp4. You can set the `post_insert_fun` to something like `fun(Id, Medium, Context)` to receive the medium record as it is inserted.

Type: *first* (page 70)

Return: modified `#media_upload_preprocess{}`

#media_import{} properties:

- `url: binary`
- `host_rev: list`
- `mime: binary`
- `metadata: tuple`

media_import_medium

Notification to import a medium record from external source. This is called for non-file medium records, for example embedded video. If the medium record is not recognized then it will not be imported. The handling module is responsible for sanitizing and inserting the medium record.

Type: *first* (page 70)

Return: `ok | {error, term()}.`

#media_import_medium{} properties:

- id: m_rsc:resource_id()
- medium: map

media_preview_options

Modify the options for an image preview url or tag. This is called for every image url generation, except if the 'original' image option is passed. The mediaclass in the options is not yet expanded.

Type: *foldl* (page 70)

Return: modified property list of image options

#media_preview_options{} properties:

- id: m_rsc:resource_id() | undefined
- width: non_neg_integer
- height: non_neg_integer
- options: proplists:proplist()

media_import_props

Notification to translate or map a file after upload, before insertion into the database Used in mod_video to queue movies for conversion to mp4. You can set the post_insert_fun to something like fun(Id, Medium, Context) to receive the medium record as it is inserted.

Type: *first* (page 70)

Return: modified #media_upload_preprocess{}

#media_import_props{} properties:

- prio: pos_integer
- category: atom
- module: atom
- description: binary|z:trans()
- rsc_props: map
- medium_props: z_media_identify:media_info()
- medium_url: binary
- preview_url: binary|undefined
- importer: atom

media_replace_file

Notification that a medium file has been changed (notify) The id is the resource id, medium contains the medium's property list.

Type: *notify* (page 69)

Return: return value is ignored

#media_replace_file{} properties:

- id: unknown

- medium: unknown

media_stillimage

See if there is a 'still' image preview of a media item. (eg posterframe of a movie) Return:: {ok, ResourceId} or undefined

Type: *first* (page 70)

Return:

#media_stillimage{} properties:

- id: m_rsc:resource_id() | undefined
- props: z_media_identify:media_info()

media_update_done

Media update done notification. action is 'insert', 'update' or 'delete'

Type: *notify* (page 69)

Return:

#media_update_done{} properties:

- action: insert | update | delete
- id: m_rsc:resource_id()
- pre_is_a: list
- post_is_a: list
- pre_props: map | undefined
- post_props: map | undefined

media_upload_preprocess

Notification to translate or map a file after upload, before insertion into the database Used in mod_video to queue movies for conversion to mp4. You can set the post_insert_fun to something like fun(Id, Medium, Context) to receive the medium record as it is inserted.

Type: *first* (page 70)

Return: modified #media_upload_preprocess{}

#media_upload_preprocess{} properties:

- id: union
- mime: binary
- file: file:filename_all() | undefined
- original_filename: file:filename_all() | undefined
- medium: z_media_identify:media_info()
- post_insert_fun: function | undefined

media_upload_props

Notification that a medium file has been uploaded. This is the moment to change properties, modify the file etc. The folded accumulator is the map with updated medium properties.

Type: *foldl* (page 70)

Return: modified medium properties map

#media_upload_props{} properties:

- id: m_rsc:resource_id() | insert_rsc
- mime: binary
- archive_file: file:filename_all() | undefined
- options: list

media_upload_rsc_props

Notification that a medium file has been uploaded. This is the moment to change resource properties, modify the file etc. The folded accumulator is the map with updated resource properties.

Type: *foldl* (page 70)

Return: modified resource properties map

#media_upload_rsc_props{} properties:

- id: m_rsc:resource_id() | insert_rsc
- mime: binary
- archive_file: unknown
- options: list
- medium: z_media_identify:media_info()

media_viewer

Request to generate a HTML media viewer for a resource

Type: *first* (page 70)

Return: {ok, Html} or undefined

#media_viewer{} properties:

- id: unknown
- props: z_media_identify:media_info()
- filename: union
- options: list

media_viewer_consent

Optionally wrap HTML with external content so that it adheres to the cookie/privacy settings of the current site visitor. Typically called with a 'first' by the code that generated the media viewer HTML, as that code has the knowledge if viewing the generated code has any privacy or cookie implications. Return {ok, HTML} or undefined

Type: *first* (page 70)

Return:

#media_viewer_consent{ } properties:

- `id: m_rsc:resource_id()` | undefined
- `consent: union`
- `html: iodata`
- `viewer_props: z_media_identify:media_info()`
- `viewer_options: list`

Pivot notifications**See also:**

rsc_pivot_done (page 603)

pivot_fields

Foldr to change or add pivot fields for the main pivot table. The rsc contains all rsc properties for this resource, including pivot properties. Fold with a map containing the pivot fields.

Type: *foldl* (page 70)

Return:

#pivot_fields{ } properties:

- `id: m_rsc:resource_id()`
- `raw_props: m_rsc:props()`

pivot_rsc_data

Fold over the resource props map to extend/remove data to be pivoted

Type: *foldl* (page 70)

Return:

#pivot_rsc_data{ } properties:

- `id: m_rsc:resource_id()`

pivot_update

Pivot just before a `m_rsc_update` update. Used to pivot fields before the pivot itself.

Type: *foldr* (page 70)

Return:

#pivot_update{ } properties:

- `id: m_rsc:resource_id()`
- `raw_props: m_rsc:props()`

custom_pivot

Add custom pivot fields to a resource's search index (map) Result is a list of {module, props} pairs. This will update a table "pivot_<module>". You must ensure that the table exists.

Type: *map* (page 70)

Return:

#custom_pivot{} properties:

- id: m_rsc:resource_id()

Resource notifications

rsc_delete

Resource will be deleted. This notification is part of the delete transaction, it's purpose is to clean up associated data.

Type: *notify* (page 69)

Return:

#rsc_delete{} properties:

- id: m_rsc:resource_id()
- is_a: list

rsc_get

Resource is read, opportunity to add computed fields Used in a foldr with the read properties as accumulator.

Type: *foldr* (page 70)

Return:

#rsc_get{} properties:

- id: m_rsc:resource_id()

rsc_import_fetch

Fetch the data for an import of a resource. Returns data in the format used by m_rsc_export and m_rsc_import.

Type: *first* (page 70)

Return: {ok, map()} | {error, term()} | undefined

#rsc_import_fetch{} properties:

- uri: binary

rsc_insert

Foldr for an resource insert, these are the initial properties and will overrule the properties in the insert request. Use with care. The props are the properties of the later insert, after escaping/filtering but before the #rsc_update{} notification below.

Type: *foldr* (page 70)

Return: proplist accumulator

#rsc_insert{ } properties:

- props: m_rsc:props()

rsc_merge

Map to signal merging two resources. Move any information from the loser to the winner. The loser will be deleted.

Type: *map* (page 70)

Return:

#rsc_merge{ } properties:

- winner_id: m_rsc:resource_id()
- loser_id: m_rsc:resource_id()
- is_merge_trans: boolean

rsc_pivot_done

Signal that a resource pivot has been done.

Type: *notify* (page 69)

Return:

#rsc_pivot_done{ } properties:

- id: m_rsc:resource_id()
- is_a: list

rsc_query_item

Send a notification that the resource 'id' is added to the query query_id.

Type: *notify* (page 69)

Return: return value is ignored

#rsc_query_item{ } properties:

- query_id: m_rsc:resource_id()
- match_id: m_rsc:resource_id()

rsc_update

An updated resource is about to be persisted. Observe this notification to change the resource properties before they are persisted.

The props are the resource's props *_before_* the update, but *_after_* filtering and sanitization. The folded value is {ok, UpdateProps} for the update itself.

Type: *foldr* (page 70)

Return: {ok, UpdateProps} or {error, term() }

#rsc_update{ } properties:

- action: insert|update
- id: m_rsc:resource_id()

- `props: m_rsc:props()`

An updated resource is about to be persisted. Observe this notification to change the resource properties before they are persisted.

Arguments

`#rsc_update`

action Either insert or update.

id Id of the resource.

props Map with resource properties.

`{ok, UpdateProps} | {error, Reason}` And/remove resource properties before the update is persisted.

Context Site context

Example

Add a property before the resource is persisted:

```
observe_rsc_update(#rsc_update{action = insert, id = Id}, {ok, Props}, Context) ->
    %% Set an extra property
    {ok, Props#{ <<"extra_property">> => <<"special value!">> }}.
observe_rsc_update(#rsc_update{action = insert, id = Id}, {error, _} = Error, _
    => Context) ->
    Error.
```

`rsc_update_done`

An updated resource has just been persisted. Observe this notification to execute follow-up actions for a resource update.

Type: *notify* (page 69)

Return: return value is ignored

`#rsc_update_done{}` properties:

- `action`: insert|update|delete
- `id`: `m_rsc:resource_id()`
- `pre_is_a`: list
- `post_is_a`: list
- `pre_props`: `m_rsc:props()`
- `post_props`: `m_rsc:props()`

pre_is_a List of resource categories before the update.

post_is_a List of resource categories after the update.

pre_props Map of properties before the update.

post_props Map of properties after the update.

Example

Add some default edges when a resource is created:

```
observe_rsc_update_done(#rsc_update_done{action = insert, id = Id, post_is_a = PostIsA, post_props = Props}, Context) ->
  case lists:member(activity, PostIsA) of
    false ->
      ok;
    true ->
      m_my_rsc:create_default_edges(Id, Context),
      ok
  end;
observe_rsc_update_done(#rsc_update_done{}, _Context) ->
  %% Fall through
  ok.
```

rsc_upload

Upload and replace the resource with the given data. The data is in the given format. Return {ok, Id} or {error, Reason}, return {error, badarg} when the data is corrupt.

Type: *first* (page 70)

Return:

#rsc_upload{} properties:

- id: unknown
- format: json|bert
- data: unknown

Survey notifications

survey_get_handlers

Fetch list of handlers for survey submits.

Type: *foldr* (page 70)

Return: list with tuples: [{handler_name, TitleForDisplay}, ...]

#survey_get_handlers{} properties: none

survey_is_allowed_results_download

Check if the current user is allowed to download a survey.

Type: *first* (page 70)

Return: true, false or undefined

#survey_is_allowed_results_download{} properties:

- id: unknown

survey_is_submit

Check if a question is a submitting question.

Type: *first* (page 70)

Return: true, false or undefined

#survey_is_submit{ } properties:

- unknown: unknown

survey_submit

A survey has been filled in and submitted.

Type: *first* (page 70)

Return:

#survey_submit{ } properties:

- id: unknown
- handler: unknown
- answers: unknown
- missing: unknown
- answers_raw: unknown

User notifications

identity_password_match

Notification that a user's identity has been verified.

Type: *notify* (page 69)

Return:

#identity_password_match{ } properties:

- rsc_id: unknown
- password: unknown
- hash: unknown

identity_verification

Request to send a verification to the user. Return ok or an error Identity may be undefined, or is a identity used for the verification.

Type: *first* (page 70)

Return:

#identity_verification{ } properties:

- user_id: unknown
- identity: unknown

identity_verified

Notification that a user's identity has been verified.

Type: *notify* (page 69)

Return:

#identity_verified{} properties:

- user_id: unknown
- type: unknown
- key: unknown

logon_options

Check for logon options, called if logon_submit returns *undefined*. This is used to fetch external (or local) authentication links for an username. Return:: map ()

Type: *foldl* (page 70)

Return:

#logon_options{} properties:

- payload: map

logon_ready_page

Check where to go after a user logs on.

Type: *first* (page 70)

Return: a URL or *undefined*

#logon_ready_page{} properties:

- request_page: union

logon_submit

Handle a user logon. The posted query args are included. Return:: {ok, UserId} or {error, Reason}

Type: *first* (page 70)

Return:

#logon_submit{} properties:

- payload: map

set_user_language

Set the language of the context to a user's preferred language

Type: *first* (page 70)

Return:

#set_user_language{} properties:

- id: m_rsc:resource_id()

request_context

Refresh the context or request process for the given request or action Called for every request that is not anonymous and before every MQTT relay from the client. Example: mod_development uses this to set flags in the process dictionary.

Type: *foldl* (page 70)

Return: #context{ }

#request_context{ } properties:

- phase: union
- document: map

session_context

Refresh the context or request process for the given request or action Called for every request that is not anonymous and before every MQTT relay from the client. Example: mod_development uses this to set flags in the process dictionary.

Type: *foldl* (page 70)

Return: #context{ }

#session_context{ } properties:

- request_type: http|mqtt
- payload: union

signup

Request a signup of a new or existing user. Arguments are similar to #signup_url{ } Returns {ok, UserId} or {error, Reason}

Type: *first* (page 70)

Return:

#signup{ } properties:

- id: m_rsc:resource_id() | undefined
- props: map
- signup_props: list
- request_confirm: boolean

signup_check

signup_check Check if the signup can be handled, a fold over all modules. Fold argument/result is {ok, Props, SignupProps} or {error, Reason}

Type: *foldl* (page 70)

Return: {ok, Props, SignupProps} or {error, Reason}

#signup_check{ } properties:

- props: map
- signup_props: list

signup_confirm

Signal that a user has been confirmed. (map, result is ignored)

Type: *notify* (page 69)

Return:

#signup_confirm{} properties:

- `id: m_rsc:resource()`

signup_confirm_redirect

Fetch the page a user is redirected to after signing up with a confirmed identity

Type: *first* (page 70)

Return: a URL or undefined

#signup_confirm_redirect{} properties:

- `id: m_rsc:resource()`

signup_done

Signal that a user has been signed up (map, result is ignored)

Type: *map* (page 70)

Return:

#signup_done{} properties:

- `id: m_rsc:resource()`
- `is_verified: boolean`
- `props: map`
- `signup_props: list`

signup_failed_url

Signup failed, give the error page URL. Return {ok, Url} or undefined. Reason is returned by the signup handler for the particular signup method (username, facebook etc)

Type: *first* (page 70)

Return:

#signup_failed_url{} properties:

- `reason: unknown`

signup_url

Handle a signup of a user, return the follow on page for after the signup. Return {ok, Url} 'props' is a map with properties for the person resource (email, name, etc) 'signup_props' is a proplist with 'identity' definitions and optional follow on url 'ready_page' An identity definition is {Kind, Identifier, IsUnique, IsVerified}

Type: *first* (page 70)

Return:

#signup_url{} properties:

- props: map
- signup_props: list

user_context

Set #context fields depending on the user and/or the preferences of the user.

Type: *foldl* (page 70)

Return:

#user_context{} properties:

- id: m_rsc:resource_id()

user_is_enabled

Check if a user is enabled. Enabled users are allowed to log in. Return true, false or undefined. If undefined is returned, the user is considered enabled if the user resource is published.

Type: *first* (page 70)

Return:

#user_is_enabled{} properties:

- id: m_rsc:resource_id()

Other notifications**action_event_type**

Render the javascript for a custom action event type. The custom event type must be a tuple, for example: `{% wire type={live id=myid} action={...} %}</code> Must return {ok, Javascript, Context}`

Type: *first* (page 70)

Return:

#action_event_type{} properties:

- event: tuple
- trigger_id: string
- trigger: string
- postback_js: iolist
- postback_pickled: string|binary
- action_js: iolist

activity

An activity in Zotonic. When this is handled as a notification then return a list of patterns matching this activity. These patterns are then used to find interested subscribers.

Type: *map* (page 70)

Return:

#activity{ } properties:

- version: pos_integer
- posted_time: unknown
- actor: unknown
- verb: atom
- object: unknown
- target: unknown

activity_send

Push a list of activities via a ‘channel’ (eg ‘email’) to a recipient. The activities are a list of #activity{ } records.

Type: *foldl* (page 70)

Return:

#activity_send{ } properties:

- recipient_id: unknown
- channel: unknown
- queue: unknown
- activities: list

admin_edit_blocks

Used in the admin to fetch the possible blocks for display

Type: *foldl* (page 70)

Return:

#admin_edit_blocks{ } properties:

- id: m_rsc:resource_id()

admin_menu

Used for fetching the menu in the admin.

Type: *foldl* (page 70)

Return: list of admin menu items

#admin_menu{ } properties: none

Example

By observing this notification, you can add your own menu items to the admin menu. The `admin_menu` notification is a fold which build up the menu, allowing each callback to add and remove menu items as they wish.

For example, this add a menu separator and an “edit homepage” button to the “content” submenu:

```
-include_lib("zotonic_core/include/zotonic.hrl").
-include_lib("zotonic_mod_admin/include/admin_menu.hrl").

observe_admin_menu(#admin_menu{}, Acc, _Context) ->
[
    #menu_separator{parent=admin_content},
    #menu_item{
        id=admin_edit_homepage,
        parent=admin_content,
        label="Edit homepage",
        url={admin_edit_rsc, [{id, page_home}]}
    } | Acc
].
```

The default submenu names are *admin_content*, *admin_structure*, *admin_modules*, *admin_auth* and *admin_system*, but you are free to add your own submenus.

admin_rscform

Used in the admin to process a submitted resource form

Type: *first* (page 70)

Return:

#admin_rscform{ } properties:

- id: m_rsc:resource_id()
- is_a: list

category_hierarchy_save

Save (and update) the complete category hierarchy

Type: *notify* (page 69)

Return:

#category_hierarchy_save{ } properties:

- tree: unknown

comment_insert

Notification to signal an inserted comment. 'comment_id' is the id of the inserted comment, 'id' is the id of the resource commented on.

Type: *notify* (page 69)

Return:

#comment_insert{ } properties:

- comment_id: integer
- id: m_rsc:resource_id()

cors_headers

Set CORS headers on the HTTP response.

Type: *first* (page 70)

Return:

#cors_headers{ } properties:

- headers: list

debug

Push some information to the debug page in the user-agent. Will be displayed with `io_lib:format("~p: ~p~n", [What, Arg])`, be careful with escaping information!

Type: *first* (page 70)

Return:

#debug{ } properties:

- what: unknown
- unknown: unknown

debug_stream

Internal message of `mod_development`. Start a stream with debug information to the user agent. 'target' is the id of the HTML element where the information is inserted. 'what' is the kind of debug information being streamed to the user-agent.

Type: *first* (page 70)

Return:

#debug_stream{ } properties:

- target: unknown
- unknown: unknown

dropbox_file

Handle a new file received in the 'files/dropbox' folder of a site. Unhandled files are deleted after a hour.

Type: *first* (page 70)

Return:

#dropbox_file{ } properties:

- filename: unknown

filewatcher

Broadcast some file changed, used for livereload by `mod_development`

Type: *notify* (page 69)

Return: return value is ignored

#filewatcher{ } properties:

- verb: modify|create|delete
- file: binary
- basename: binary
- extension: binary

hierarchy_updated

Signal that the hierarchy underneath a resource has been changed by mod_menu

Type: *notify* (page 69)

Return:

#hierarchy_updated{ } properties:

- root_id: binary|integer
- predicate: atom
- inserted_ids: list
- deleted_ids: list

http_log_access

Access log event for http. Called from the z_stats.

Type: *notify_sync* (page 70)

Return:

#http_log_access{ } properties:

- timestamp: erlang:timestamp()
- status: undefined|non_neg_integer
- status_category: xxx|1xx|2xx|3xx|4xx|5xx
- method: binary
- metrics: map

language

Notify that the session's language has been changed

Type: *notify* (page 69)

Return:

#language{ } properties:

- language: atom

m_config_update

Site configuration parameter was changed

Type: *notify* (page 69)

Return: return value is ignored

#m_config_update{ } properties:

- module: atom
- key: term
- value: term

m_config_update_prop

Site configuration parameter was changed

Type: *notify* (page 69)

Return: return value is ignored

#m_config_update_prop{ } properties:

- module: unknown
- key: unknown
- prop: unknown
- value: unknown

mailinglist_mailing

Send a page to a mailinglist (notify) Use {single_test_address, Email} when sending to a specific e-mail address.

Type: *first* (page 70)

Return:

#mailinglist_mailing{ } properties:

- list_id: unknown
- page_id: unknown

mailinglist_message

Send a welcome or goodbye message to the given recipient. The recipient is either an e-mail address or a resource id. 'what' is send_welcome, send_confirm, send_goobye or silent.

Type: *notify* (page 69)

Return:

#mailinglist_message{ } properties:

- what: unknown
- list_id: unknown
- recipient: unknown

menu_rsc

Fetch the menu id belonging to a certain resource

Type: *first* (page 70)

Return:

#menu_rsc{ } properties:

- id: m_rsc:resource()

menu_save

Save the menu tree of a menu resource

Type: *notify* (page 69)

Return:

#menu_save{ } properties:

- id: unknown
- tree: unknown

middleware

Delegates the request processing.

Type: *foldl* (page 70)

Return: updated `z:context()`

#middleware{ } properties:

- on: request|wellformed|handled

module_activate

A module has been activated and started.

Type: *notify* (page 69)

Return:

#module_activate{ } properties:

- module: atom
- pid: pid

module_deactivate

A module has been stopped and deactivated.

Type: *notify* (page 69)

Return:

#module_deactivate{ } properties:

- module: atom

multiupload

Handle an uploaded file which is part of a multiple file upload from a user-agent. The upload is a #upload record or a filename on the server.

Type: *first* (page 70)

Return: `#context{ }` with the result or undefined

#multiupload{ } properties:

- upload: term|string

- `query_args`: `list`

output_html

Fold for mapping non-iolist output to iolist values. Used when outputting a rendered HTML tree. Folded accumulator is: `{ MixedHtml, Context }`

Type: *foldl* (page 70)

Return:

#output_html { } properties:

- `html`: `term`

postback_event

Message sent by a user-agent on a postback event. Encapsulates the encoded postback and any additional data. This is handled by `z_transport.erl`, which will call the correct event/2 functions.

Type: *first* (page 70)

Return:

#postback_event { } properties:

- `postback`: `unknown`
- `trigger`: `unknown`
- `target`: `unknown`
- `triggervalue`: `unknown`
- `data`: `unknown`

postback_notify

Handle a javascript notification from the postback handler. The `message` is the the request, `trigger` the id of the element which triggered the postback, and `target` the id of the element which should receive possible updates. `#postback_notify` is also used as an event.

Type: *first* (page 70)

Return: `undefined` or `#context { }` with the result of the postback

#postback_notify { } properties:

- `message`: `unknown`
- `trigger`: `unknown`
- `target`: `unknown`
- `data`: `unknown`

resource_headers

Let all modules add resource specific response headers to the request. The accumulator is the list of headers to be set.

Type: *foldl* (page 70)

Return: `list ({binary(), binary()})`

#resource_headers{ } properties:

- id: m_rsc:resource_id() | undefined

sanitize_element

Sanitize an HTML element.

Type: *foldl* (page 70)

Return:

#sanitize_element{ } properties:

- element: tuple
- stack: list

sanitize_embed_url

Sanitize an embed url. The hostpart is of the format: <<"youtube.com/v...">>.

Type: *first* (page 70)

Return: undefined, false or a binary with a acceptable hostpath

#sanitize_embed_url{ } properties:

- hostpath: binary

This notification is used to sanitize *embed urls* passed with the media import routines.

Example usage in a module, where URLs from some public broadcasters are allowed:

```
-export ([
    observe_sanitize_embed_url/2
]).

observe_sanitize_embed_url(#sanitize_embed_url{hostpath= <<"media.vara.nl/", _/
    ↪binary>> = Url}, _Context) ->
    Url;
observe_sanitize_embed_url(#sanitize_embed_url{hostpath= <<"biografie.vara.nl/", _/
    ↪binary>> = Url}, _Context) ->
    Url;
observe_sanitize_embed_url(#sanitize_embed_url{hostpath= <<"js.vpro.nl/", _/binary>
    ↪> = Url}, _Context) ->
    Url;
observe_sanitize_embed_url(#sanitize_embed_url{hostpath= <<"embed.vpro.nl/", _/
    ↪binary>> = Url}, _Context) ->
    Url;
observe_sanitize_embed_url(_, _Context) ->
    undefined.
```

Note the undefined returned if no other patterns match. This allows other modules to check for different patterns.

scomp_script_render

Add extra javascript with the { % script % } tag. (map) Used to let modules inject extra javascript depending on the arguments of the { % script % } tag. Must return an iolist()

Type: *map* (page 70)

Return:

#scomp_script_render{ } properties:

- is_nostartup: boolean
- args: list

search_query

Map a custom search term to a #search_sql_term{ } record.

Type: *first* (page 70)

Return: #search_sql_term{ }, [], or undefined

#search_query{ } properties:

- name: union
- args: union
- offsetlimit: tuple
- search: union

See also:

Custom search (page 323), *Search* (page 44)

search_query_term

Map a custom search term to a #search_sql_term{ } record.

Type: *first* (page 70)

Return: #search_sql_term{ }, [], or undefined

#search_query_term{ } properties:

- term: binary
- arg: any

security_headers

Check and possibly modify the http response security headers All headers are in lowercase.

Type: *first* (page 70)

Return:

#security_headers{ } properties:

- headers: list

This is called when the *security headers* are set for the request, which is done at the start of the request handling, before any controller callback is called.

That is done so early to ensure that any returned payload has all required security headers.

The default security header list is:

```
[
  % Content-Security-Policy is not added by default
  % {<<"content-security-policy">>, <<"script-src 'self' 'nonce-'>>}
  {<<"x-xss-protection">>, <<"1">>},
  {<<"x-content-type-options">>, <<"nosniff">>},
  {<<"x-permitted-cross-domain-policies">>, <<"none">>},
```

```
{<<"referrer-policy">>, <<"origin-when-cross-origin">>},  
{<<"x-frame-options">>, <<"sameorigin">>}  
]
```

If the controller option `allow_frame` is set to *true* then the `x-frame-options` header is not added.

The `security_headers` notification does a *first* to fetch the security headers. The default headers are passed in the `headers` field of the notification.

If the notification returns a list with `<<"content-security-policy">>` then in the value of the Content-Security-Policy header the string `'nonce- '` is replaced with the unique nonce for the request.

The nonce can be added to script tags:

```
<script type="text/javascript" nonce="{{ m.req.csp_nonce }}">  
    // Inline javascript here  
</script>
```

It can be requested in Erlang code using:

```
CSPNonce = z_context:csp_nonce(Context).
```

Or via the `m_req` model:

```
CSPNonce = m_req:get(csp_nonce, Context).
```

Note that the nonce is only set iff the `Context` is a HTTP request context. It is *not* set for MQTT contexts.

service_authorize

Request API logon

Type: *first* (page 70)

Return:

#service_authorize{} properties:

- `service_module`: unknown

ssl_options

Request the SSL certificates for this site. The `server_name` property contains the hostname used by the client. (first) Returns either 'undefined' or a list of ssl options (type `ssl:ssl_option()`)

Type: *first* (page 70)

Return:

#ssl_options{} properties:

- `server_name`: binary

tkvstore_delete

Delete a value from the typed key/value store

Type: *notify* (page 69)

Return: return value is ignored

#tkvstore_delete{} properties:

- type: unknown
- key: unknown

tkvstore_get

Get a value from the typed key/value store

Type: *first* (page 70)

Return:

#tkvstore_get {} properties:

- type: unknown
- key: unknown

tkvstore_put

Put a value into the typed key/value store

Type: *notify* (page 69)

Return:

#tkvstore_put {} properties:

- type: unknown
- key: unknown
- value: unknown

url_fetch_options

Determine the URL fetch options for fetching the content of an URL. Used by z_fetch.erl.

Type: *first* (page 70)

Return: z_url_fetch:options()

#url_fetch_options {} properties:

- method: get|post|put|delete
- host: binary
- url: binary
- options: z_url_fetch:options()

validate_query_args

Called just before validation of all query arguments by z_validation. This is the moment to filter any illegal arguments or change query arguments.

Type: *foldl* (page 70)

Return: {ok, list({binary(), z:qvalue()})} | {error, term() }

#validate_query_args {} properties: none

6.1.10 Installation requirements

Zotonic runs on Linux, Mac OS X and (not officially) on Windows.

Before running Zotonic, you must make sure your system meets the minimum requirements to do so. Zotonic needs the following software installed:

1. Erlang/OTP version **23** or newer. Build it from source, or use packages.
2. **ImageMagick** (version 6.5 or higher) for the `convert` and `identify` tools. Make sure that the `convert` and `identify` tools are in your path so that zotonic can find them. For auto-rotation to work you'll need the `exif` utility as well.
3. **PostgreSQL** version 9.5 or higher. Enable trust-authentication (username+password) in Postgres (see below).
4. **gcc** and **g++** For compiling dependencies like *erlexec*.
5. **git** Zotonic comes with a few subprojects which are pulled from the web with the `git` command.
6. **gettext** For translation support.

If you meet these requirements, head straight on to *Cloud-Init* (page 12), otherwise, read on below for the specifics on these.

Erlang

You can check if you have Erlang installed by typing the following command in a terminal:

```
$ erl
```

The output should be something like:

```
Erlang/OTP 23 [erts-11.0] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-  
→threads:1] [hipe]  
Eshell V11.0 (abort with ^G)  
1>
```

(Press ctrl+c twice to exit)

If your version is below release **23**, you need to upgrade. If you don't have Erlang installed, we recommend downloading a build for your operating system from the Erlang Solutions website:

<https://www.erlang-solutions.com/downloads/download-erlang-otp>

ImageMagick

You can check if you have ImageMagick installed by typing the following command in a terminal:

```
$ convert -version
```

This tells you which version of ImageMagick is installed:

```
Version: ImageMagick 6.8.9-9 Q16 x86_64 2018-09-28 http://www.imagemagick.org
```

PostgreSQL

You can check if you have PostgreSQL installed by typing the following command in a terminal:

```
$ psql -V
```

Returns the PostgreSQL version:

```
psql (PostgreSQL) 9.5.17
```

Enabling trust-authentication in PostgreSQL

To let Postgres users access the database from Zotonic, you need to make a configuration change in Postgres' configuration file `pg_hba.conf`.

On Linux, this file is located in `/etc/postgresql/<pg version>/main/pg_hba.conf`. Add the following lines:

```
# Zotonic settings
local all zotonic ident
host all zotonic 127.0.0.1/32 md5
host all zotonic ::1/128 md5
```

These settings assume that your Zotonic sites are going to connect with a PostgreSQL user called `zotonic`. For other user names, adjust accordingly. Do not forget to restart PostgreSQL after you've made this change.

Platform-specific notes

Ubuntu / Debian

We recommend you install Erlang from the Erlang solutions website:

<https://www.erlang-solutions.com/downloads/>

The other requirements are easily fetched with `apt`:

```
sudo apt-get install gcc g++ build-essential git imagemagick postgresql ffmpeg
```

FreeBSD

If you're running on FreeBSD, make sure you've got the 'GNU' 'make' (check with `make --version`, which should give you GNU, and version info). If you don't have GNU make, Zotonic will give an error when trying to compile.

macOS

With [Homebrew](#) you can install Erlang and ImageMagick using the following commands:

```
brew install erlang git imagemagick ffmpeg
```

Alternatively, with MacPorts:

```
sudo port install erlang +ssl
sudo port install ImageMagick
```

For PostgreSQL choose either:

- [Postgress.app](#)
- [EnterpriseDB](#)

Windows

Currently, Zotonic is not officially supported on the Windows platform. However, the main dependencies Erlang, PostgreSQL and ImageMagick do work on Windows, so, if you're adventurous, it should be possible to get it running.

6.1.11 Configuration

Global configuration

This section describes the location and contents of Zotonic's global configuration files `erlang.config` and `zotonic.config`. There's also *site-specific configuration* (page 626).

Config file locations

Zotonic depends on two global config files, called `zotonic.config` and `erlang.config`. On startup, Zotonic looks in the following places for these files:

- The init argument `zotonic_config_dir`
- The environment variable `ZOTONIC_CONFIG_DIR`
- The directory `$HOME/.zotonic`
- The directory `/etc/zotonic` (only on Unix)
- The OS specific directory for application config files

The OS specific directories are:

- On Unix: `~/.config/zotonic/config/`
- On macOS: `~/Library/Application Support/zotonic/config/`

In those directories the system searches for a `zotonic*` file in the following subdirectories (assuming the version of Zotonic is 1.2.3 and the node is called `zotonic001@foobar`):

- `zotonic001@foobar/`
- `zotonic001/`
- `1.2.3/`
- `1.2/`
- `1/`
- `.`

The default is the OS specific directory, with as subdirectory the major version number of Zotonic (in this case 1). For Linux this would be `~/.config/zotonic/config/1/`

The nodename is the name of the Zotonic Erlang node, which defaults to `zotonic001` (and can be set with `$SNAME` or `$LNAME` environment variable). Using the node name in the configuration path comes in handy when you want to run multiple Zotonic instances simultaneously.

By checking for *version* you can have separate configuration files for different versions of Zotonic which are running simultaneously.

If the Zotonic startup script finds a `zotonic*` file in one of the directories, it stops looking, so files in the other directories are ignored.

Zotonic will also load all configuration files in the `config.d` directory inside the Zotonic directory.

In the course of Zotonic starting up, it will print the locations of the global config files that it is using:

```

12:42:17.351 [info] zotonic_launcher_sup:36 =====
12:42:17.351 [info] zotonic_launcher_sup:37 Zotonic starting
12:42:17.351 [info] zotonic_launcher_sup:38 =====
12:42:17.351 [info] zotonic_launcher_sup:39 Init files used:
12:42:17.356 [info] zotonic_launcher_sup:40 - /home/user/.config/zotonic/config/1/
↳erlang.config
12:42:17.356 [info] zotonic_launcher_sup:41 Config files used:
12:42:17.357 [info] zotonic_launcher_sup:43 - /home/user/.config/zotonic/config/1/
↳zotonic.config
12:42:17.357 [info] zotonic_launcher_sup:44 =====

```

The used configuration files can be listed with `bin/zotonic configfiles`:

```

user$ bin/zotonic configfiles
Zotonic config files for zotonic001@foobar:
- /home/user/.config/zotonic/config/1.0/erlang.config
- /home/user/.config/zotonic/config/1.0/zotonic.config

```

The `zotonic.config` file

After installed for the first time, the `~/ .zotonic/1/zotonic.config` file is well annotated with comments about what each setting does. When in doubt, consult the stock `apps/zotonic_launcher/priv/config/zotonic.config.in` file for explanation about all config settings.

In the `zotonic.config` file you will find the password for the `zotonic_status` site where you can manage the server.

Zotonic configurations can also be fetched in the Erlang shell. For example, view the `zotonic_status` password:

```
z_config:get(password).
```

The Zotonic configuration files are read by the `zotonic_launcher` application before starting the core zotonic applications and all sites.

The zotonic configuration can be viewed with `bin/zotonic config`:

```

Zotonic config for zotonic001@aloha:
=====

zotonic:
  environment: production
  zotonic_apps: /home/user/zotonic/apps_user
  security_dir: /home/user/.config/zotonic/security
  password: Bthj3ruGbmGJxfmc
  timezone: UTC
  listen_ip: any
  listen_ip6: any
  listen_port: 8000
  ssl_listen_port: 8443
  port: 80
  ssl_port: 443
  max_connections: 20000
  ...

```

The `erlang.config` file

The `erlang.config` file contains application environment variables for the Erlang applications that Zotonic depends on. Here you can configure for instance the paths for the *log files* (page 82) (in the `kernel` section), `emqt` ports, et cetera.

This file is included as an *init* configuration option when starting `erl` via the command line script in `bin/zotonic`.

The erlang configuration can be viewed with `bin/zotonic config erlang`:

```
Erlang init for zotonic001@aloha:
=====

exometer:
  predefined:
    - {[erlang,memory],{function,erlang,memory,[],value,[],[]}}
    - {[erlang,system_info],
      {function,erlang,system_info,['$dp'],value,[process_count]}},
      []}
    - {[erlang,statistics],
      {function,erlang,statistics,['$dp'],value,[run_queue]}},
      []}
    - {[erlang,io],
      {function,erlang,statistics,[io],match,{['_',input},['_',output]}},
      []}
filezcache:
  data_dir: priv/filezcache/data
  journal_dir: priv/filezcache/journal
kernel:
...
```

Site configuration

This chapter describes the configuration options for your sites. There's also [global configuration](#) (page 624).

Site config locations

Configuration for sites is stored:

- in a `priv/zotonic_site.config` file in the site directory
- optionally, in files in the site's `priv/config.d/` directory. This is so that automated provisioning tools can easily override site configuration.

The `config.d` files will extend and/or override the configuration options from the `zotonic_site.config` file. So if the same key is present in both `zotonic_site.config` and `config.d/some-file`, the value from `some-file` will be used. The files under `config.d/` are read in alphabetical order.

Parameters

Important: After changing any of these configuration parameters, [restart the site](#) (page 20) for the change to take effect.

admin_password

The password for the admin user:

```
{admin_password, "top_secret"},
```

Note that if the `admin_password` is set to `admin` then only clients with local network addresses can logon as the admin. See also `ip_allowlist` below.

dbhost

Database host that the site connects to. Example:

```
{dbhost, "127.0.0.1"},
```

Defaults to the `dbhost` in the Zotonic config. Which defaults to `127.0.0.1`.

dbport

Port of the database server. Example:

```
{dbport, 5432},
```

Defaults to the `dbport` in the Zotonic config. Which defaults to `5432`.

dbuser

Database user. Defaults to the `dbuser` in the Zotonic config. Which defaults to `zotonic`.

dbpassword

Database password. Defaults to the `dbpassword` in the Zotonic config. Which defaults to `zotonic`.

dbdatabase

Database name. Use `none` to run Zotonic without a database:

```
{dbdatabase, none},
```

Defaults to the database configured in the Zotonic config (which defaults to `zotonic`).

If the `dbuser` does not have permission to create the database then you have to create it yourself:

```
CREATE DATABASE "some_database" ENCODING = 'UTF8' TEMPLATE template0;
GRANT ALL ON DATABASE some_database TO some_db_user;
```

Where *some_db_user* must be the user you configured with `dbuser` and *some_database* must be the database configured with `dbdatabase`.

dbschema

PostgreSQL Database schema. Defaults to either:

- the name of the site if the `dbdatabase` is the default database from the zotonic config
- `public` if the `dbdatabase` is set to something else than the default from the Zotonic config.

So if you configure a special database for your site then the default schema will be `public`.

Example:

```
{dbschema, "mysiteschema"},
```

In Zotonic, a single PostgreSQL database can host the data of multiple sites. This does not work using table prefixing, but instead, Zotonic uses PostgreSQL's native feature of database schemas.

A database schema is basically another database inside your database. If your dbuser does not have permission to create the schema, then you will need to create any schema other than `public` first:

```
CREATE SCHEMA some_site;
GRANT ALL ON SCHEMA some_site TO some_db_user;
```

Where *some_db_user* must be the user you configured with `dbuser`. After creating the schema, either restart the site or restart Zotonic.

db_max_connections

Maximum number of database connections used by the site. Unused connections are closed after some time.

This parameter should be enlarged if the following error occurs frequently:

```
Database pool usage is close to exhaustion, please increase the pool size.
```

And definitely increase the pool size if this error occurs frequently:

```
Database pool is exhausted, please increase the pool size.
```

The default is 20 connections:

```
{db_max_connections, 20},
```

depcache_memory_max

The maximum amount of memory a site may take (in MB). The depcache caches various results of function calls and database queries in memory. Example:

```
{depcache_memory_max, 100},
```

hostname

The hostname and port part of the site URL. This is used to determine to which site an incoming request should be routed. Example:

```
{hostname, "127.0.0.1"},
```

Note that the hostname does *not* specify on which port Zotonic will listen; this is *configured globally* (page 630).

hostalias

A list of alias hostnames for the site. By default, Zotonic redirects these to `hostname` (see `redirect`). Example:

```
{hostalias, [
  "example.com",
  "www.example.com",
  "example.net",
  "www.example.net"
]},
```


redirect

Whether or not to redirect the host-aliases (listed by the `hostalias` directives) to the main hostname. Defaults to `true`, to prevent content duplication: it is good web practice to let your content live on a single URL only:

```
{redirect, true},
```

skeleton

Set by the `zotonic addsite` command, this settings tells Zotonic which skeleton site to use. Example:

```
{skeleton, blog},
```

install_menu

Creates the initial main menu when installing *mod_menu* (page 381). A menu item is an Erlang tuple with a resource name and list of child menu items (if any): `{name, []}`. Example:

```
{install_menu, [
    {page_some_thing, []},
    {page_some_other_thing, []},
    {page_one_more_thing, []}
]},
```

modules

List of all modules that are *activated* (page 60) when the site is started. After adding a module here, *restart the site* (page 20) to load the module. Example:

```
{modules, [
    mod_admin,
    mod_menu,
    mod_your_custom_module
]},
```

ip_allowlist

List of TCP/IP addresses and their netmasks. The default admin user password (“admin”) will only be accepted for an IP in the allowlist. This protects development systems that are exposed to the internet. This can also be configured *globally* (page 624). Default:

```
{ip_allowlist, "127.0.0.0/8,10.0.0.0/8,192.168.0.0/16,172.16.0.0/12,::1,fd00::/8"}
```

smtphost

Hostname you want e-mail messages to appear from. See *E-mail handling* (page 74).

cookie_domain

The domain the Zotonic session-id and page-id cookies will be set on. Defaults to the main hostname.

installer

Override the default zotonic installer (`z_installer`). `<module>` should make sure that the database, if used, is setup properly along with any required data. Note that it is `z_installer` that is processing the `install_modules` and `install_menu` options, so if this module is not used then those menus and modules will not be installed unless the new module performs those operations. Example:

```
{installer, your_installer_erlang_module},
```

Setting module-specific config values in the site config

It is also possible to add [m_config](#) (page 560) values for modules to the site's `sitename/priv/zotonic_site.config` file. To do this, add clauses like this to the site's config:

```
{mod_foo, [{key, value}, ...]}
```

Using environment variables in the site config

Any variable in your site's `zotonic_site.config` file can be retrieved from the OS environment variables. To do so, wrap the config value in a `{env, ...}` tuple. For instance, to use the `DB_HOST` environment variable as the database host, put the following as the `dbhost` config value:

```
{dbhost, {env, "DB_HOST"}},
```

Besides `{env, "NAME"}` tuple, you can also specify `{env, "NAME", "default value"}` for the case the environment variable is not set:

```
{dbhost, {env, "DB_HOST", "localhost"}},
```

To convert environment variables to integer (e.g. for the database port), use `env_int`:

```
{dbhost, {env_int, "DB_PORT"}},
```

or, with default value:

```
{dbhost, {env_int, "DB_PORT", "5432"}},
```

Note that the default value needs to be a string in this case, not an int.

See also:

[mod_ssl_letsencrypt](#) (page 397), [mod_ssl_ca](#) (page 395), [Running on Port 80 and Port 443](#) (page 86)

Port configurations

Port configurations can be tricky, especially in combination with SSL. Here we explain all steps to come to a correctly configured installation with working SSL connectivity.

There are basically two sets of port configurations:

1. The ports Zotonic *listens* on
2. The ports an *outside* visitor *connects* to

The listen ports are configured with `listen_port` and `ssl_listen_port`.

The outside ports are configured with `port` and `ssl_port`. They default to the `listen_` variations if not defined.

Below are examples how to configure these.

Server direct on the Internet

Here you can use the methods described in *Running on Port 80 and Port 443* (page 86) to get your server on ports 80 and/or 443.

The port configurations would be:

Method	listen_port	ssl_listen_port	port	ssl_port	listen_ip	proxy_allowlist
authbind	80	443	80	443	any	none
setcap	80	443	80	443	any	none
iptables	8000	8443	80	443	127.0.0.1	none
http only	8000	none	80	none	127.0.0.1	none

For *Network Address Translation* (NAT), see the next section. The *proxy_allowlist* is explained in the section about proxies below.

In the case of *iptables* we restrict Zotonic to listen on the local 127.0.0.1 address. This to prevent that people can connect on port 8000 from their browsers.

Alternatively you can run your server on the *outside* port 8000, though then it is impossible to use Let's Encrypt certificates for SSL (as they require the server to run on the default http and https ports).

Server accessed via NAT

With *Network Address Translation* (NAT) the traffic is routed straight to the server using port mappings. This is typical for a situation where Zotonic runs on a local server behind a modem.

Proxy method	listen_port	ssl_listen_port	port	ssl_port	listen_ip	proxy_allowlist
NAT (eg. modem)	8000	8443	80	443	any	none

The *proxy_allowlist* is explained in the section about proxies below.

Server behind a proxy like Nginx or HAProxy

A proxy could be *haproxy* or *nginx*. The proxy terminates the https connection and handles the SSL certificates.

Typically the proxy connects to the default ports 8000 and 8443 on the Zotonic server. The proxy itself could be running on the local server or another server.

Proxy method	listen_port	ssl_listen_port	port	ssl_port	listen_ip	proxy_allowlist
localhost proxy	8000	none	80	443	127.0.0.1	local
proxy on LAN	8000	none	80	443	any	local
proxy on WAN	8000	none	80	443	any	see below

The proxy adds the hostname, address of the visitor and protocol information (http or https) to a HTTP header. Zotonic reads this header to know which site to serve and if the visitor was using https or not.

Everybody could add this header and then connect directly to the Zotonic server, which can then make wrong assumptions about the IP address of the visitor and if the visitor is on a secure connection.

To prevent the visitor spoofing the *Forward* header, Zotonic will check if the *inside* address of the proxy (as seen from Zotonic, not from the visitor) is on a list of allowed proxies.

This allowlist is specified in *proxy_allowlist* and can have the following values:

- `local` - default, only LAN addresses can connect
- `none` - no proxy exists, ignore proxy headers
- `any` - *insecure*, any server anywhere can be a proxy
- A tuple with a single ip address, for example: `{192,168,1,1}`
- A string with ip addresses with optional masks, for example: `"127.0.0.0/8,10.0.0.0/8,fe80:::/10"`

The `local` check is hardcoded and very fast.

SSL certificates

After the server's listen ports are correctly configured then the SSL connection can be tested.

Per default Zotonic will generate a self-signed certificate for all valid hostnames. Instead of these self-signed certificates a real certificate can be used. Check for these the modules [mod_ssl_letsencrypt](#) (page 397) and [mod_ssl_ca](#) (page 395)

The self-signed certificates are stored in the `self-signed` subdirectory of the security directory.

You can see the `security_dir` location using `bin/zotonic config`

Possible locations are:

- The environment variable `ZOTONIC_SECURITY_DIR`
- The `~/.zotonic/security` directory
- The `/etc/zotonic/security` directory (only on Linux)
- The OS specific directory for application data files

The OS specific directories are:

- On Unix: `~/.config/zotonic/security/`
- On macOS: `~/Library/Application Support/zotonic/security/`

The default is the OS specific directory.

HTTPS and security

Zotonic always redirects all incoming HTTP connections to HTTPS. This can not be disabled.

Secure cookies

All cookies related to authentication are set to *secure*. The cookie holding the authentication token has `SameSite` set to `Strict`. This can not be configured or changed.

Erlang SSL Configuration

The erlang ssl application is configured in the `erlang.config` file. It is stored next to the `zotonic.config` file. You can see the config files used with:

```
bin/zotonic configfiles
```

If this file is missing then it can be copied from `apps/zotonic_launcher/priv/erlang.config.in`. It contains a couple of important settings which we recommend you to change. The reason for this is that the default settings Erlang uses are unsuitable for web servers. The most important settings are listed below.

session_lifetime Sets the maximum lifetime of session data in seconds.

session_cache_server_max Sets the maximum number of client sessions cached by the server.

For more information on configuration options, please see [Erlang SSL App](#).

Adding your own SSL options or certificates

If you want to implement your own certificate handling you have to add a notification observer which returns the certificates to the underlying HTTPS server. This can be needed if you have a site with special hostname aliases, or if you want to implement automated certificate handling for a specific certificate authority.

The notification use by the SNI (Server Name Indication) handler is:

ssl_options{server_name=ServerName} Return the certificate, key or other ssl options. *ServerName* is a string (list) with the name of the server from the SSL handshake. You should return a proplist with Erlang `ssl:ssl_option()` terms. The proplist will override the default ssl options for this connection. For more information about the possible properties see [Erlang SSL](#). If undefined is returned the SSL handshake will try the next SSL module. If all modules return undefined then a self-signed certificate will be used.

6.1.12 Command-line

The `zotonic` command runs a number of utility commands which all operate on a Zotonic instance.

Tip: The `zotonic` command lives in the `bin/` folder of the Zotonic source. Putting this path into your `PATH` variable makes working with Zotonic a lot easier:

```
export PATH=$HOME/zotonic/bin:$PATH
```

The command determines where the Zotonic base dir is by looking at its path; it always assumes that its `zotonic` basedir is one dir up from where the binary itself is.

Shell environment variables

See [environment variables](#) (page 94) for the environment variables that can be set when using the Zotonic command line tools.

Commands

Currently, the following subcommands are implemented:

zotonic start Start the background Zotonic server instance.

zotonic stop Stop the Zotonic server instance.

zotonic debug Launch the Zotonic server interactively and get an EShell on the running instance. See [Command-line shell](#) (page 79). The `start.sh` command in the root folder is a shortcut for this command. The Zotonic instance can be stopped with twice `ctrl-C`.

zotonic start_nodaemon Start the Zotonic server instance as a foreground process, but without the interactive EShell. This is useful when running Zotonic in a Docker container or other process.

zotonic restart Restart the Zotonic server instance.

zotonic wait [timeout] Wait `timeout` seconds (defaults to 30 seconds) for Zotonic to be started, then return.

zotonic shell Get an EShell on the running Zotonic instance. See [Command-line shell](#) (page 79).

zotonic status List all sites on the running Zotonic instance and their current status.

zotonic configfiles List all config files used for Zotonic.

zotonic configtest Read all config files and check if they are syntactically correct.

zotonic config [**all** | **zotonic** | **erlang**] Prints the configuration as defined in the configuration files. Taking into account all shell environment variables. Defaults to showing the *zotonic* config.

zotonic rpc Send an RPC request to the running Zotonic instance. Example: *zotonic rpc "zotonic ping"*

zotonic addsite [**options**] **<site_name>** Creates a new site with [site_name] as its name. See *guide-cli-addsite* for a full overview of this command.

zotonic startsite **<site_name>** Start the site with name [site_name].

zotonic stopsite **<site_name>** Stop the site with name [site_name].

zotonic restartsite **<site_name>** Restart the site with name [site_name].

zotonic sitedir **<site_name>** Get the absolute path for a site based on [site_name]

zotonic siteconfig **<site_name>** Prints the configuration of site [site_name] as defined in its configuration files.

zotonic siteconfigfiles **<site_name>** List all configuration files for of site [site_name]

zotonic dispatch **<url>** Dispatch an URL. Finds the site matching the hostname and shows the dispatch information for the path. Example:

```
bin/zotonic dispatch https://mysite.test:8443/en/page/1/foobar
```

zotonic dispatch **<site_name>** [**detail**] List all dispatch rules for a site. Add the *detail* option to show all controller options for each dispatch rule.

zotonic dispatch **<site_name>** **<path>** Show the dispatch information for a specific path and site. Example:

```
bin/zotonic dispatch mysite /en/page/1/foobar
```

zotonic etop Show the processes that consume the most CPU. Stop with twice ctrl-C.

zotonic logtail [**console** | **error** | **crash**] Show the last 50 entries of the *console.log*, *error.log* or *crash.log* file. Defaults to the console log.

zotonic flush Flush the caches of all sites.

zotonic flush **<site_name>** Flush the caches of the site with name [site_name].

zotonic createdb **<site_name>** Create a database called *zotonic_[site_name]* with the basic setup in place to host a Zotonic datastore. This script will likely need to be run as postgres unless zotonic has been granted CREATEDB in postgres as follows:

```
ALTER ROLE zotonic WITH CREATEDB
```

zotonic compilefile **<path/to/filename.erl>** Compiles and reloads a single *Erlang module* within the Zotonic folder. This runs very fast and works very well on a save-hook of your text editor. In Emacs, it would be called like this:

```
(add-hook 'erlang-mode-hook
  (lambda ()
    (add-hook 'after-save-hook '
      (lambda ()
        (call-process "/path/to/your/bin/zotonic" nil "*scratch*" nil
          ↪ "compilefile" buffer-file-name)
        )
      )
  ))
```

Tip: Install *fswatch* or *inotify-tools* to automatically recompile files when they are changed. These tools will also enable automatic loading of changed templates, dispatch rules, and translations.

zotonic compile Compiles all the Zotonic Erlang source files, modules and sites, including those in the user directory (see [Global configuration](#) (page 624)).

zotonic update Like `zotonic compile` but also flushes caches and rescans all modules and sites for new templates etc.

zotonic load Reloads all (changed) beam files from disk.

zotonic runtests Starts Zotonic in the foreground and runs all (enunit) tests. Stops after completion of the tests.

zotonic sitetest <site_name> Runs all tests for the given site. Zotonic must be running. See [Testing sites](#) (page 82).

6.1.13 EDoc reference

There are reference docs built from the source code using edoc.

- [genindex](#)
- [Glossary](#) (page 637)

7.1 Glossary

Action An action is functionality that can be attached to a HTML element or event. Actions are wired to an element or event. Think of showing dialogs, posting forms, hiding elements etc.

See also [Actions](#) (page 56) in the Developer Guide.

Category The data model has a hierarchical tree for the categorization of resources. Every resource is part of one category. The categorization is used amongst others to decide which template to show when displaying a resource. A category is a *resource* of the category *category*. For more information, see [Categories](#) (page 7).

Context The context is the current request context. It contains all the request data, the current site, the handle to the database and the results (scripts or templates) you will be sending back. The context is commonly passed along in Zotonic as the last argument of a function.

Controller A controller is the main entry point where a request is handled. Controllers are referenced from a dispatch rule. Commonly used controller is `controller_template`, which serves a template on the URL for which the controller configured. See [Controllers](#) (page 20).

Data model *Zotonic's generic data model* (page 5) of (categorized) resources which connect to other resources using labelled edges. This data model is loosely based on the principles of the semantic web.

Delegate A reference to a module which will be used to call a callback function on. Used in the templates when attaching actions like a `:term:postback` to a DOM Event. See [Actions](#) (page 56).

Dispatch rule A dispatch rule maps URL patterns to controllers. Dispatch rules are defined in files in the `.dispatch.` folder of a Zotonic module. The dispatch rule definitions are also used to generate the urls for resources and other pages. See [Dispatch rules](#) (page 22).

Domain model A particular configuration of resource categories and predicates, which dictate how resources of certain categories relate to each other. For example, a blog-type site might need *person*, *article* and *keyword* categories, where persons and articles are connected using the *author* predicate to indicate article authorship, and articles might be connected to keywords with *has_keyword* predicates. See [The Zotonic data model](#) (page 5).

Edge A *resource* can connect to other resources. These connections are called edges. Edges contain no information other than where they are linked to and from, and what their predicate is. Edges have a single direction, from the subject to the object.

Erlang module Not to be confused with a Zotonic module, an Erlang module is a single .erl file which contains Erlang functions.

Filter A template mechanism which is used inside a template to transform data before it is output. For instance: the .lower. filter transforms its input to lowercase. Filters are implemented as Erlang modules, exporting a single filter function.

Media Media are files, embed codes etc. They are attached to a resource. Every resource can hold a single medium. The resource is usually within the category *media*. See: [Media](#) (page 32).

Model An Erlang module which is the main accessor for retrieving data. The Erlang modules are prefixed with *m_*; in the templates they are accessible using .m... For instance, the model to access *resources* is called *m_rsc.erl*; in the template this model lets you access resources by id as `{{ m_rsc[id] }}`.

Non Informational URI The non informational uri is the base url of a resource. It always redirects to a representation of the resource. Think of a HTML page, image or JSON download. The chosen representation depends on the .Accept. HTTP request header. The non informational uri of a resource is always like <http://example.com/id/1234>

Page Another word for .resource.; used in the admin.

Page connection Another word for .edge.; used in the admin.

Postback An AJAX or Websocket request from the browser to the server. It is handled on the server by event/2 Erlang functions. A postback is normally sent to the controller that generated the page, but can be changed by specifying a delegate, which must be the name of an Erlang module.

Predicate Each edge has a *label* attached to it to determine what the meaning of the edge is. For instance, when an article is linked to a person, the predicate (label) might read *author*, to indicate that that person is the author of the article. A predicate is a *resource* of the category *predicate*.

Property A field in a resource. Examples are title and summary. Properties are dynamically defined. Although some property names are reserved, you can set any other property, which will be stored in the resource.

Resource The main building block of the *data model* (page 5). For simplicity of communication, a resource is often referred to as a page. Every resource usually has its own page on the web site. See [Resources](#) (page 26).

Scomp A scomp (from .Screen COMponent.) is a custom template tag, implemented by an Erlang module named after the scomp name, prefixed with *scomp_*. Scomps usually generate HTML. Zotonic modules can implement their own scomp in the module.scomps/ folder.

Session The session is an Erlang process. It is connected to the *session cookie* id on the browser. The session contains the id of the current user and more key/value pairs, called session variables. The session is also linked to page processes. For every open page on the browser we have a process on the server. This page process is used for the communication between the server and the user-agent (browser).

Session cookie A cookie is a small piece of data sent from a website and stored in a user's web browser while the user is browsing that website. In contrast to persistent cookies, session cookies are created and kept only during the user's visit to the website, and deleted from the browser's cache when the user closes the session.

Tag The template systems provides tags which function as simple programming constructs. For instance, the if tag can be used for boolean tests and the for tag allows looping. The Zotonic templating system compiles the tags found in a template to Erlang byte code which will be called when the template is rendered. This is very efficient.

Template A snippet of code which is used to render a piece of content (usually HTML). Templates live under the templates/ folder of a module. The template is meant to express presentation logic.

Translation There are two kinds of translations. Texts in the templates and Erlang modules; and translations of resources. Templates and Erlang modules are translated using gettext. Resources are translated in the admin, any resource can have an arbitrary number of translations. Zotonic selects the shown language based on the preferred language of the visitor and the available languages of a resource.

Validator A validator is used to check input fields in a HTML form. A validator has two parts: the client side javascript and a server side check. You add validators to a form with the `{% validate %}` template tag. A validated query argument can be accessed on the server using `z_context:get_q_validated/2`.

Wire Connects actions and events to a HTML element. The wire scomp is the basis for most Ajax interaction on web pages. It allows to connected actions to HTML elements. Examples of actions are showing/hiding elements or postbacks to the server. `..todo:: hmm`. It is scomp - it is also a often used function in the Erlang code `z_render:wire/2`

Zotonic module A Zotonic module (just `.module.`, for short) is a collection of related functionality like scomp, filters, dispatch rules, controllers, templates, etc. Zotonic modules are OTP applications and are prefixed with `zotonic_mod_`. See [Modules](#) (page 59).

Zotonic site A Zotonic site is a collection of scomp, filters, dispatch rules for one website. It is a special kind of Zotonic module with has its own config file which allows one to set the hostname, admin password, database connection parameters. The config file contains site wide settings. Zotonic uses the settings to start the site on the right port and connect it to the right database. A Zotonic system can run multiple sites.

Zotonic user directory The directory in which user-installed Zotonic sites and modules are placed. Defaults to the path `apps_user` relative to the Zotonic installation, but can be adjusted by changing the `ZOTONIC_APPS` environment variable. See [Useful environment variables](#) (page 94).

A

Action, [637](#)

action

- [add_class, 408](#)
- [admin_tasks, 404](#)
- [alert, 425](#)
- [animate, 408](#)
- [auth_disconnect, 433](#)
- [backup_start, 405](#)
- [buttonize, 408](#)
- [config_delete, 405](#)
- [config_toggle, 405](#)
- [confirm, 426](#)
- [delete_media, 430](#)
- [delete_rsc, 430](#)
- [delete_username, 434](#)
- [development_templates_stream, 406](#)
- [dialog, 406](#)
- [dialog_close, 407](#)
- [dialog_config_delete, 405](#)
- [dialog_config_edit, 405](#)
- [dialog_config_new, 406](#)
- [dialog_delete_rsc, 430](#)
- [dialog_delete_username, 434](#)
- [dialog_duplicate_rsc, 430](#)
- [dialog_edit_basics, 431](#)
- [dialog_mail_page, 424](#)
- [dialog_mailing_page, 424](#)
- [dialog_media_upload, 431](#)
- [dialog_new_rsc, 431](#)
- [dialog_open, 407](#)
- [dialog_predicate_new, 428](#)
- [dialog_set_username_password, 434](#)
- [dialog_user_add, 434](#)
- [disable, 420](#)
- [editor_add, 417](#)
- [editor_remove, 418](#)
- [effect, 409](#)
- [enable, 420](#)
- [event, 421](#)
- [fade_in, 409](#)
- [fade_out, 409](#)
- [focus, 421](#)

- [form_reset, 421](#)
- [growl, 427](#)
- [hide, 409](#)
- [insert_after, 410](#)
- [insert_before, 410](#)
- [insert_bottom, 410](#)
- [insert_top, 411](#)
- [jquery_effect, 411](#)
- [link, 428](#)
- [logout, 434](#)
- [mailing_page_test, 425](#)
- [mailinglist_confirm, 425](#)
- [mailinglist_unsubscribe, 425](#)
- [mask, 411](#)
- [mask_progress, 412](#)
- [module_rescan, 404](#)
- [module_toggle, 404](#)
- [moreresults, 432](#)
- [move, 412](#)
- [overlay_close, 408](#)
- [overlay_open, 407](#)
- [postback, 419](#)
- [publish, 420](#)
- [redirect, 427](#)
- [redirect_incat, 404](#)
- [reload, 428](#)
- [remove, 412](#)
- [remove_class, 413](#)
- [replace, 413](#)
- [reset, 421](#)
- [script, 424](#)
- [set_class, 413](#)
- [set_value, 422](#)
- [show, 413](#)
- [slide_down, 414](#)
- [slide_fade_in, 414](#)
- [slide_fade_out, 414](#)
- [slide_toggle, 415](#)
- [slide_up, 415](#)
- [submit, 422](#)
- [template, 433](#)
- [toggle, 415](#)
- [toggle_class, 415](#)
- [trigger_event, 419](#)

- typeselect, 423
- unlink, 429
- unmask, 416
- update, 416
- update_iframe, 417
- validation_error, 423
- with_args, 403
- zlink, 418
- zmedia, 418
- zmedia_choose, 418
- zmedia_has_chosen, 418

automatic

- id, 516

Automatically generated ids, 526

C

Category, 637

content_group (mod_signup attribute), 393

Context, 637

Controller, 637

controller

- admin, 435
- admin_acl_rules_export, 435
- admin_backup, 435
- admin_backup_revision, 435
- admin_category_sorter, 435
- admin_comments, 436
- admin_comments_settings, 436
- admin_config, 436
- admin_edit, 436
- admin_mailing_preview, 437
- admin_mailing_status, 437
- admin_mailinglist, 438
- admin_mailinglist_recipients, 438
- admin_media_preview, 438
- admin_module_manager, 438
- admin_referrers, 439
- admin_seo, 439
- admin_statistics, 439
- api, 439
- authentication, 440
- export, 440
- export_resource, 440
- file, 440
- file_id, 442
- fileuploader, 442
- hello_world, 443
- http_error, 443
- id, 444
- keyserver_key, 444
- language_set, 444
- letsencrypt_challenge, 445
- letsencrypt_ping, 445
- log_client, 445
- logout, 445
- logon_done, 445
- mailinglist_export, 446
- mqtt_transport, 446

- nocontent, 446
- oauth2_access_token, 446
- oauth2_service_authorize, 447
- oauth2_service_redirect, 447
- page, 447
- ping, 449
- redirect, 449
- signup, 450
- signup_confirm, 451
- static_pages, 451
- template, 452
- website_redirect, 453

core

- model, acl, 555
- model, category, 557
- model, config, 560
- model, edge, 562
- model, hierarchy, 566
- model, identity, 566
- model, media, 567
- model, modules, 568
- model, predicate, 569
- model, req, 570
- model, rsc, 571
- model, rsc_gone, 577
- model, search, 578
- model, site, 579
- model, sysconfig, 581
- model, template, 581

D

Data model, 637

Delegate, 637

depiction_as_medium (mod_signup attribute), 393

Dispatch rule, 637

Domain model, 637

E

Edge, 638

Erlang module, 638

F

Filter, 638

filter

- add_day, 456
- add_hour, 456
- add_month, 456
- add_week, 457
- add_year, 457
- admin_merge_diff, 495
- after, 474
- append, 497
- as_atom, 509
- before, 474
- brlinebreaks, 465
- capfirst, 497
- center, 497
- chunk, 474

content_type_label, 495
content_type_urls, 496
date, 457
date_range, 459
datediff, 460
default, 509
element, 507
embedded_media, 473
eq_day, 460
escape, 465
escape_check, 465
escape_ical, 466
escape_link, 466
escapejs, 466
escapejson, 467
escapexml, 467
exclude, 474
filesizeformat, 487
filter, 475
first, 454
fix_ampersands, 467
flatten_value, 476
force_escape, 468
format_duration, 488
format_integer, 488
format_number, 488
format_price, 489
gravatar_code, 485
group_by, 476
group_firstchar, 492
group_title_firstchar, 493
if, 509
if_undefined, 509
in_future, 460
in_past, 460
index_of, 477
inject_recipientdetails, 482
insert, 498
ip2country, 485
ip2geo, 486
is_a, 494
is_defined, 510
is_even, 489
is_letsencrypt_valid_hostname, 487
is_list, 477
is_not_a, 494
is_number, 489
is_rtl, 504
is_site_url, 507
is_undefined, 510
is_valid_email, 498
is_visible, 495
join, 477
language, 505
language_dir, 505
language_sort, 505
last, 454
length, 455
linebreaksbr, 468
ljust, 498
log_format_stack, 498
lower, 498
make_list, 478
make_value, 510
match, 492
max, 489
md5, 464
member, 478
menu_expand, 483
menu_flat, 483
menu_is_visible, 483
menu_rsc, 484
menu_subtree, 484
menu_trail, 484
min, 490
minmax, 490
ne_day, 461
normalize_email, 499
nthtail, 478
parse_url, 508
pickle, 469
pprint, 511
rand, 490
random, 478
randomize, 479
range, 479
replace, 492
replace_args, 499
reversed, 479
rjust, 499
round, 491
round_significant, 491
sanitize_html, 472
sanitize_url, 472
sha1, 464
show_media, 469
slice, 480
slugify, 468
sort, 480
split, 481
split_in, 481
stringify, 500
striptags, 471
sub_day, 461
sub_hour, 461
sub_month, 462
sub_week, 462
sub_year, 463
summary, 496
survey_answer_split, 503
survey_any_correct_answer, 503
survey_any_wrong_answer, 503
survey_as_pages, 503
survey_is_stop, 503
survey_is_submit, 503
survey_prepare_matching, 504

- survey_prepare_narrative, 504
- survey_prepare_thurstone, 504
- survey_test_max_points, 504
- tail, 481
- temporary_rsc, 496
- timesince, 463
- to_binary, 455
- to_integer, 491
- to_json, 511
- to_name, 500
- toc, 500
- tokens, 501
- trans_filter_filled, 506
- translate, 501
- translation, 506
- trim, 502
- truncate, 502
- truncate_html, 472
- twitter, 485
- unescape, 468
- upper, 502
- url, 507
- url_abs, 508
- urldecode, 508
- urlencode, 469
- urlize, 472
- utc, 464
- vsplit_in, 482
- without, 482
- without_embedded_media, 473
- yesno, 455

I

id

- automatic, 516
- unique, 526

M

Media, 638

member_category (mod_signup attribute), 393

mod_acl_user_groups

- controller, admin_acl_rules_export, 435
- model, acl_rule, 556
- model, acl_user_group, 556

mod_admin

- action, admin_tasks, 404
- action, delete_media, 430
- action, delete_rsc, 430
- action, dialog_delete_rsc, 430
- action, dialog_duplicate_rsc, 430
- action, dialog_edit_basics, 431
- action, dialog_media_upload, 431
- action, dialog_new_rsc, 431
- action, link, 428
- action, redirect_incat, 404
- action, unlink, 429
- action, zlink, 418
- action, zmedia, 418

- action, zmedia_choose, 418
- action, zmedia_has_chosen, 418
- controller, admin, 435
- controller, admin_edit, 436
- controller, admin_media_preview, 438
- controller, admin_referrers, 439
- filter, temporary_rsc, 496
- model, admin, 556
- model, admin_blocks, 556
- model, admin_menu, 557
- model, admin_note, 557
- model, admin_status, 557

mod_admin_category

- controller, admin_category_sorter, 435

mod_admin_config

- action, config_delete, 405
- action, config_toggle, 405
- action, dialog_config_delete, 405
- action, dialog_config_edit, 405
- action, dialog_config_new, 406
- controller, admin_config, 436
- model, admin_config, 556

mod_admin_identity

- action, delete_username, 434
- action, dialog_delete_username, 434
- action, dialog_set_username_password, 434
- action, dialog_user_add, 434
- filter, normalize_email, 499
- model, admin_identity, 557
- validator, email_unique, 40
- validator, username_unique, 43

mod_admin_merge

- filter, admin_merge_diff, 495

mod_admin_modules

- action, module_rescan, 404
- action, module_toggle, 404
- controller, admin_module_manager, 438

mod_admin_predicate

- action, dialog_predicate_new, 428

mod_admin_statistics

- controller, admin_statistics, 439

mod_auth2fa

- model, auth2fa, 557

mod_authentication

- action, auth_disconnect, 433
- controller, authentication, 440
- controller, logoff, 445
- controller, logon_done, 445
- model, authentication, 557

mod_backup

- action, backup_start, 405
- controller, admin_backup, 435
- controller, admin_backup_revision, 435
- model, backup, 557
- model, backup_revision, 557

mod_base

- controller, api, 439
- controller, file, 440

controller, file_id, 442
 controller, http_error, 443
 controller, id, 444
 controller, keyserver_key, 444
 controller, mqtt_transport, 446
 controller, nocontent, 446
 controller, page, 447
 controller, ping, 449
 controller, redirect, 449
 controller, static_pages, 451
 controller, template, 452
 controller, website_redirect, 453
 filter, add_day, 456
 filter, add_hour, 456
 filter, add_month, 456
 filter, add_week, 457
 filter, add_year, 457
 filter, after, 474
 filter, append, 497
 filter, as_atom, 509
 filter, before, 474
 filter, brlinebreaks, 465
 filter, capfirst, 497
 filter, center, 497
 filter, chunk, 474
 filter, content_type_label, 495
 filter, content_type_urls, 496
 filter, date, 457
 filter, date_range, 459
 filter, datediff, 460
 filter, default, 509
 filter, element, 507
 filter, embedded_media, 473
 filter, eq_day, 460
 filter, escape, 465
 filter, escape_check, 465
 filter, escape_ical, 466
 filter, escape_link, 466
 filter, escapejs, 466
 filter, escapejson, 467
 filter, escapexml, 467
 filter, exclude, 474
 filter, filesizeformat, 487
 filter, filter, 475
 filter, first, 454
 filter, fix_ampersands, 467
 filter, flatten_value, 476
 filter, force_escape, 468
 filter, format_duration, 488
 filter, format_integer, 488
 filter, format_number, 488
 filter, format_price, 489
 filter, group_by, 476
 filter, group_firstchar, 492
 filter, group_title_firstchar, 493
 filter, if, 509
 filter, if_undefined, 509
 filter, in_future, 460
 filter, in_past, 460
 filter, index_of, 477
 filter, insert, 498
 filter, is_a, 494
 filter, is_defined, 510
 filter, is_even, 489
 filter, is_list, 477
 filter, is_not_a, 494
 filter, is_number, 489
 filter, is_site_url, 507
 filter, is_undefined, 510
 filter, is_visible, 495
 filter, join, 477
 filter, last, 454
 filter, length, 455
 filter, linebreaksbr, 468
 filter, ljust, 498
 filter, lower, 498
 filter, make_list, 478
 filter, make_value, 510
 filter, match, 492
 filter, max, 489
 filter, md5, 464
 filter, member, 478
 filter, min, 490
 filter, minmax, 490
 filter, ne_day, 461
 filter, nthtail, 478
 filter, parse_url, 508
 filter, pickle, 469
 filter, pprint, 511
 filter, rand, 490
 filter, random, 478
 filter, randomize, 479
 filter, range, 479
 filter, replace, 492
 filter, replace_args, 499
 filter, reversed, 479
 filter, rjust, 499
 filter, round, 491
 filter, round_significant, 491
 filter, sanitize_html, 472
 filter, sanitize_url, 472
 filter, sha1, 464
 filter, show_media, 469
 filter, slice, 480
 filter, slugify, 468
 filter, sort, 480
 filter, split, 481
 filter, split_in, 481
 filter, stringify, 500
 filter, striptags, 471
 filter, sub_day, 461
 filter, sub_hour, 461
 filter, sub_month, 462
 filter, sub_week, 462
 filter, sub_year, 463
 filter, summary, 496

- filter, tail, 481
- filter, timesince, 463
- filter, to_binary, 455
- filter, to_integer, 491
- filter, to_json, 511
- filter, to_name, 500
- filter, toc, 500
- filter, tokens, 501
- filter, trans_filter_filled, 506
- filter, translation, 506
- filter, trim, 502
- filter, truncate, 502
- filter, truncate_html, 472
- filter, unescape, 468
- filter, upper, 502
- filter, url, 507
- filter, url_abs, 508
- filter, urldecode, 508
- filter, urlencode, 469
- filter, urlize, 472
- filter, utc, 464
- filter, vsplit_in, 482
- filter, without, 482
- filter, without_embedded_media, 473
- filter, yesno, 455
- model, mqtt_ticket, 569
- scomp, chart_pie, 534
- scomp, chart_pie3d, 535
- scomp, cotonic_pathname_search, 536
- scomp, debug, 536
- scomp, google_chart, 539
- scomp, inplace_textbox, 541
- scomp, lazy, 541
- scomp, loremipsum, 543
- scomp, pager, 544
- scomp, spinner, 549
- scomp, tabs, 549
- scomp, worker, 554
- validator, acceptance, 38
- validator, confirmation, 38
- validator, custom, 38
- validator, date, 39
- validator, email, 39
- validator, format, 40
- validator, length, 41
- validator, name_unique, 41
- validator, numericality, 42
- validator, postback, 42
- validator, presence, 43
- mod_comment
 - controller, admin_comments, 436
 - controller, admin_comments_settings, 436
 - filter, gravatar_code, 485
 - model, comment, 559
- mod_content_groups
 - model, content_group, 561
- mod_custom_redirect
 - model, custom_redirect, 561
- mod_development
 - action, development_templates_stream, 406
 - controller, hello_world, 443
 - model, development, 561
- mod_editor_tinymce
 - model, editor_tinymce, 564
- mod_email_dkim
 - model, email_dkim, 564
- mod_email_receive
 - model, email_receive_recipient, 564
- mod_email_status
 - filter, is_valid_email, 498
 - model, email_status, 564
- mod_export
 - controller, export, 440
 - controller, export_resource, 440
- mod_facebook
 - model, facebook, 564
- mod_filestore
 - model, filestore, 564
- mod_fileuploader
 - controller, fileuploader, 442
 - model, fileuploader, 565
- mod_geoip
 - filter, ip2country, 485
 - filter, ip2geo, 486
- mod_image_edit
 - model, image_edit, 567
- mod_import_csv
 - model, import_csv_data, 567
- mod_l10n
 - model, l10n, 567
- mod_linkedin
 - model, linkedin, 567
- mod_logging
 - controller, log_client, 445
 - filter, log_format_stack, 498
 - model, log, 567
 - model, log_email, 567
 - model, log_ui, 567
- mod_mailinglist
 - action, dialog_mail_page, 424
 - action, dialog_mailing_page, 424
 - action, mailing_page_test, 425
 - action, mailinglist_confirm, 425
 - action, mailinglist_unsubscribe, 425
 - controller, admin_mailing_preview, 437
 - controller, admin_mailing_status, 437
 - controller, admin_mailinglist, 438
 - controller, admin_mailinglist_recipients, 438
 - controller, mailinglist_export, 446
 - filter, inject_recipientdetails, 482
 - model, mailinglist, 567
 - scomp, mailinglist_subscribe, 543
- mod_menu
 - filter, menu_expand, 483
 - filter, menu_flat, 483
 - filter, menu_is_visible, 483

- filter, menu_rsc, 484
 - filter, menu_subtree, 484
 - filter, menu_trail, 484
 - scomp, menu, 543
- mod_microsoft
 - model, microsoft, 568
- mod_mqtt
 - action, publish, 420
 - model, client_local_storage, 559
 - model, client_session_storage, 559
 - scomp, live, 542
- mod_oauth2
 - controller, oauth2_access_token, 446
 - controller, oauth2_service_authorize, 447
 - controller, oauth2_service_redirect, 447
 - model, oauth2, 569
 - model, oauth2_consumer, 569
 - model, oauth2_service, 569
- mod_ratelimit
 - model, ratelimit, 570
- mod_seo
 - controller, admin_seo, 439
 - model, seo, 578
- mod_seo_sitemap
 - model, seo_sitemap, 578
- mod_server_storage
 - model, server_storage, 578
- mod_signup
 - controller, signup, 450
 - controller, signup_confirm, 451
 - model, signup, 579
- mod_site_update
 - model, site_update, 580
- mod_ssl_letsencrypt
 - controller, letsencrypt_challenge, 445
 - controller, letsencrypt_ping, 445
 - filter, is_letsencrypt_valid_hostname, 487
 - model, ssl_letsencrypt, 580
- mod_survey
 - filter, survey_answer_split, 503
 - filter, survey_any_correct_answer, 503
 - filter, survey_any_wrong_answer, 503
 - filter, survey_as_pages, 503
 - filter, survey_is_stop, 503
 - filter, survey_is_submit, 503
 - filter, survey_prepare_matching, 504
 - filter, survey_prepare_narrative, 504
 - filter, survey_prepare_thurstone, 504
 - filter, survey_test_max_points, 504
 - model, survey, 580
 - scomp, poll, 546
- mod_tkvstore
 - model, tkvstore, 581
- mod_translation
 - controller, language_set, 444
 - filter, is_rtl, 504
 - filter, language, 505
 - filter, language_dir, 505
 - filter, language_sort, 505
 - filter, translate, 501
 - model, translation, 581
- mod_twitter
 - filter, twitter, 485
 - model, twitter, 582
- mod_wires
 - action, add_class, 408
 - action, alert, 425
 - action, animate, 408
 - action, buttonize, 408
 - action, confirm, 426
 - action, dialog, 406
 - action, dialog_close, 407
 - action, dialog_open, 407
 - action, disable, 420
 - action, editor_add, 417
 - action, editor_remove, 418
 - action, effect, 409
 - action, enable, 420
 - action, event, 421
 - action, fade_in, 409
 - action, fade_out, 409
 - action, focus, 421
 - action, form_reset, 421
 - action, growl, 427
 - action, hide, 409
 - action, insert_after, 410
 - action, insert_before, 410
 - action, insert_bottom, 410
 - action, insert_top, 411
 - action, jquery_effect, 411
 - action, logoff, 434
 - action, mask, 411
 - action, mask_progress, 412
 - action, moreresults, 432
 - action, move, 412
 - action, overlay_close, 408
 - action, overlay_open, 407
 - action, postback, 419
 - action, redirect, 427
 - action, reload, 428
 - action, remove, 412
 - action, remove_class, 413
 - action, replace, 413
 - action, reset, 421
 - action, script, 424
 - action, set_class, 413
 - action, set_value, 422
 - action, show, 413
 - action, slide_down, 414
 - action, slide_fade_in, 414
 - action, slide_fade_out, 414
 - action, slide_toggle, 415
 - action, slide_up, 415
 - action, submit, 422
 - action, template, 433
 - action, toggle, 415

- action, toggle_class, 415
- action, trigger_event, 419
- action, typeselect, 423
- action, unmask, 416
- action, update, 416
- action, update_iframe, 417
- action, validation_error, 423
- action, with_args, 403
- scomp, button, 533
- scomp, draggable, 537
- scomp, droppable, 538
- scomp, script, 546
- scomp, sortable, 547
- scomp, sorter, 548
- scomp, validate, 550
- scomp, wire, 551
- scomp, wire_args, 553
- Model, **638**
- model
 - acl, 555
 - acl_rule, 556
 - acl_user_group, 556
 - admin, 556
 - admin_blocks, 556
 - admin_config, 556
 - admin_identity, 557
 - admin_menu, 557
 - admin_note, 557
 - admin_status, 557
 - auth2fa, 557
 - authentication, 557
 - backup, 557
 - backup_revision, 557
 - category, 557
 - client_local_storage, 559
 - client_session_storage, 559
 - comment, 559
 - config, 560
 - content_group, 561
 - custom_redirect, 561
 - development, 561
 - edge, 562
 - editor_tinymce, 564
 - email_dkim, 564
 - email_receive_recipient, 564
 - email_status, 564
 - facebook, 564
 - filestore, 564
 - fileuploader, 565
 - hierarchy, 566
 - identity, 566
 - image_edit, 567
 - import_csv_data, 567
 - l10n, 567
 - linkedin, 567
 - log, 567
 - log_email, 567
 - log_ui, 567
 - mailinglist, 567
 - media, 567
 - microsoft, 568
 - modules, 568
 - mqtt_ticket, 569
 - oauth2, 569
 - oauth2_consumer, 569
 - oauth2_service, 569
 - predicate, 569
 - ratelimit, 570
 - req, 570
 - rsc, 571
 - rsc_gone, 577
 - search, 578
 - seo, 578
 - seo_sitemap, 578
 - server_storage, 578
 - signup, 579
 - site, 579
 - site_update, 580
 - ssl_letsencrypt, 580
 - survey, 580
 - sysconfig, 581
 - template, 581
 - tkvstore, 581
 - translation, 581
 - twitter, 582
- module
 - mod_acl_mock, 341
 - mod_acl_user_groups, 341
 - mod_admin, 346
 - mod_admin_category, 350
 - mod_admin_config, 351
 - mod_admin_frontend, 351
 - mod_admin_identity, 353
 - mod_admin_merge, 355
 - mod_admin_modules, 355
 - mod_admin_predicate, 356
 - mod_admin_statistics, 356
 - mod_artwork, 357
 - mod_audio, 358
 - mod_auth2fa, 359
 - mod_authentication, 359
 - mod_backup, 360
 - mod_base, 360
 - mod_bootstrap, 361
 - mod_clamav, 361
 - mod_comment, 361
 - mod_contact, 361
 - mod_content_groups, 361
 - mod_cookie_consent, 362
 - mod_cron, 363
 - mod_custom_redirect, 364
 - mod_development, 364
 - mod_editor_tinymce, 367
 - mod_email_dkim, 368
 - mod_email_receive, 369
 - mod_email_relay, 369

- mod_email_status, 370
- mod_export, 370
- mod_facebook, 370
- mod_filestore, 371
- mod_fileuploader, 375
- mod_geoip, 376
- mod_image_edit, 376
- mod_import_csv, 377
- mod_import_wordpress, 377
- mod_110n, 377
- mod_linkedin, 377
- mod_logging, 378
- mod_mailinglist, 378
- mod_media_exif, 380
- mod_menu, 380
- mod_microsoft, 381
- mod_mqtt, 381
- mod_oauth2, 386
- mod_oembed, 386
- mod_ratelimit, 387
- mod_search, 388
- mod_seo, 389
- mod_seo_sitemap, 390
- mod_server_storage, 391
- mod_signup, 393
- mod_site_update, 394
- mod_ssl_ca, 395
- mod_ssl_letsencrypt, 397
- mod_survey, 398
- mod_syslog, 400
- mod_tkvstore, 400
- mod_translation, 400
- mod_twitter, 401
- mod_video, 402
- mod_video_embed, 402
- mod_wires, 403
- mod_zotonic_site_management, 403

N

Non Informational URI, [638](#)
 notifications, [582](#)

O

object, [8](#)

P

Page, [638](#)
 Page connection, [638](#)
 Postback, [638](#)
 Predicate, [638](#)
 Property, [638](#)

R

request_confirm (mod_signup attribute), [393](#)
 Resource, [638](#)

S

Scomp, [638](#)

scomp

- button, [533](#)
- chart_pie, [534](#)
- chart_pie3d, [535](#)
- cotonic_pathname_search, [536](#)
- debug, [536](#)
- draggable, [537](#)
- droppable, [538](#)
- google_chart, [539](#)
- inplace_textbox, [541](#)
- lazy, [541](#)
- live, [542](#)
- loremipsum, [543](#)
- mailinglist_subscribe, [543](#)
- menu, [543](#)
- pager, [544](#)
- poll, [546](#)
- script, [546](#)
- sortable, [547](#)
- sorter, [548](#)
- spinner, [549](#)
- tabs, [549](#)
- validate, [550](#)
- wire, [551](#)
- wire_args, [553](#)
- worker, [554](#)

Session, [638](#)
 Session cookie, [638](#)
 subject, [8](#)

T

Tag, [638](#)

tag

- all catinclude, [511](#)
- all include, [512](#)
- autoescape, [513](#)
- block, [513](#)
- cache, [513](#)
- call, [514](#)
- catinclude, [515](#)
- comment, [515](#)
- cycle, [516](#)
- extends, [516](#)
- filter, [516](#)
- firstof, [517](#)
- for, [517](#)
- if, [518](#)
- ifchanged, [519](#)
- ifequal, [519](#)
- ifnotequal, [519](#)
- image, [520](#)
- image_data_url, [525](#)
- image_url, [525](#)
- include, [526](#)
- inherit, [527](#)
- javascript, [528](#)
- lib, [528](#)
- load, [529](#)

- media, 529
- now, 529
- overrides, 530
- print, 530
- raw, 530
- regroup, 531
- spaceless, 531
- templatetag, 531
- translate, 531, 532
- url, 532
- with, 533

Tags, 511

Template, **638**

Translation, **638**

U

unique

- id, 526

username_equals_email (mod_signup attribute), 393

V

Validator, **639**

validator

- acceptance, 38

- confirmation, 38

- custom, 38

- date, 39

- email, 39

- email_unique, 40

- format, 40

- length, 41

- name_unique, 41

- numericality, 42

- postback, 42

- presence, 43

- username_unique, 43

W

Wire, **639**

Z

Zotonic module, **639**

Zotonic site, **639**

Zotonic user directory, **639**