
Gila CMS Documentation

Release 24 Nov 2017

Vasileios Zoumpourlis

Jun 27, 2018

1	About	3
1.1	Why choose Gila?	3
1.2	Spreading the word!	3
1.3	Where you can get help	3
2	Installation	5
2.1	Preparation	5
2.2	Installer	5
3	Content	9
3.1	Pages	9
3.2	Posts	11
3.3	Categories	12
3.4	Media	12
4	Administration	15
4.1	Users	15
4.2	Widgets	15
4.3	Packages	15
4.4	Themes	17
4.5	Settings	18
5	Structure	21
5.1	Packages	21
6	Api	25
6.1	Core Classes	25
7	Indices and tables	29

Contents:

Gila CMS is an open-source and free content management system built with php7. Built with MVC architecture, is very easy to develop on it any customized solution. It is licensed under BSD 3-Clause License. The website is gilacms.com

1.1 Why choose Gila?

Gila CMS is a good option for self-hosted blogs or startup websites

- Installs by default a blogging system with social integrations.
- No coding skills are required to install or maintain the website
- Themes that includes are responsive, that makes it accessible in all devices.
- It is fast, and compresses the content where it needs to.

1.2 Spreading the word!

You can help us spread the word about Gila CMS! We would surely appreciate it!

- Follow our [Facebook Page](#)
- [Retweet](#) us!
- Give a star on [Github](#)

1.3 Where you can get help

- Join [Slack](#) or [Discord](#)
- Ask on stackoverflow using the tag **gilacms**

2.1 Preparation

Before beginning with installation make sure that your web host or local server meets these requirements:

- Apache 2 server
- MySQL / MariaDB server
- PHP 5.4+ with the following extensions *mysqli*, *zip*, *mysqlnd*, *json* and *mod_rewrite* enabled

If you want to install gila cms in your local machine and not sure how to prepare your server don't hesitate to ask for help on [Slack](#)

First unzip gila in a public html folder e.g */var/www/html/gila* and make sure that the folder is writable from the application.

In order to proceed with the installation, you will need your database settings. If you do not know your database settings, please contact your host and ask for them. You will not be able to continue without them. More precisely you need the database hostname, the database name, the database username and password.

2.2 Installer

We access in installation page with the browser e.g *http://localhost/gila/install*

Gila CMS Installation

Hostname	Admin Username
<input type="text" value="localhost"/>	<input type="text"/>
Database	Admin Email
<input type="text"/>	<input type="text"/>
DB Username	Admin Password
<input type="text"/>	<input type="text"/>
DB Password	Base URL
<input type="text"/>	<input type="text" value="//localhost/gila/"/>
<input type="submit" value="Submit"/>	

In the installation page we must fill all the fields

Hostname: the hostname of the database, usually it is *localhost*

Database: name of the database

DB Username, DB Password: the username and the password in order to connect to the mysql

Admin Username, Admin Email, Admin Password: a user will be created for the website as administrator with these data

Base Url: the web address of the website must finish with '/' e.g. *mywebsite.com/*

After filling the data and submit them, we wait a few seconds untill the installation is finished.

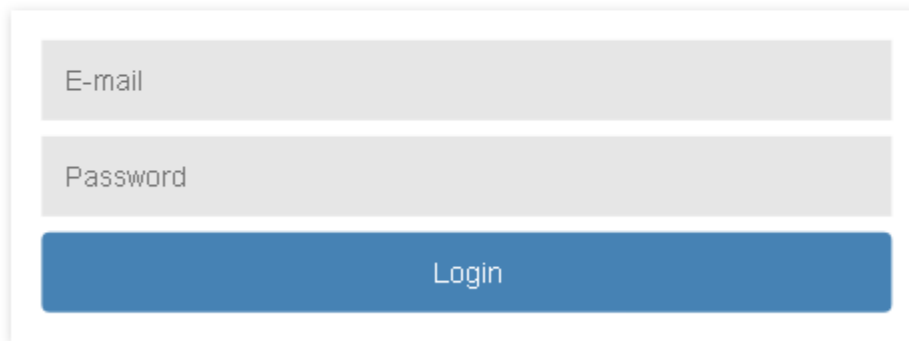
Installation finished successfully!

Visit the website
Login to admin panel

When installation is finished we can enter on the admin panel using the admin email and password that we wrote before.



Login



E-mail

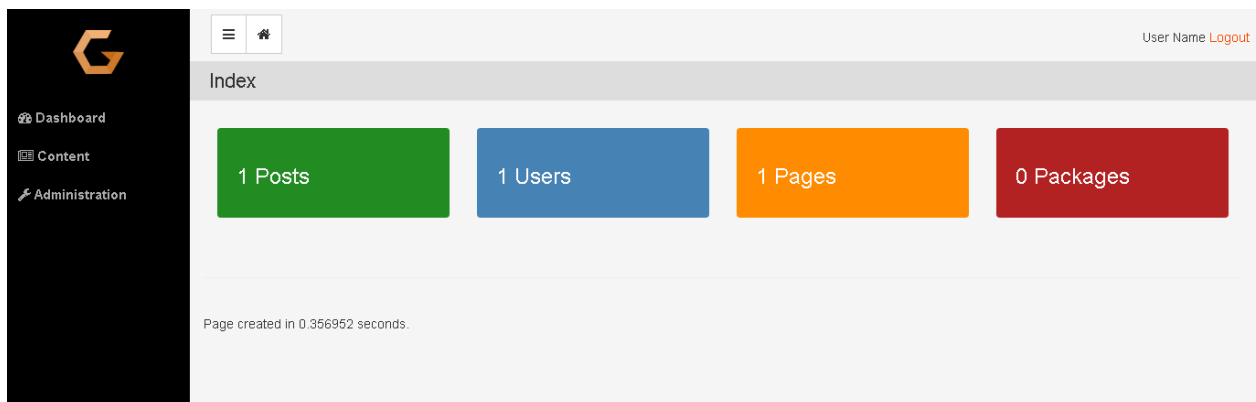
Password

Login

[Forgot password?](#)

We can always access in the login page from these links *mysite.com/ login* it redirects to the front page of the website *mysite.com/ admin* it redirects to the administration

We enter in the administration dashboard.



From the administration menu we choose Administration->Settings in order to fill more information about the website.

Title	Gila CMS
Description	An awesome website!
Website URL	//localhost/gila/
SSL URL	
Theme	Base ▼
Timezone	America/Mexico_City ▼
Default Controller	Blog ▼
Environment	Dev ▼

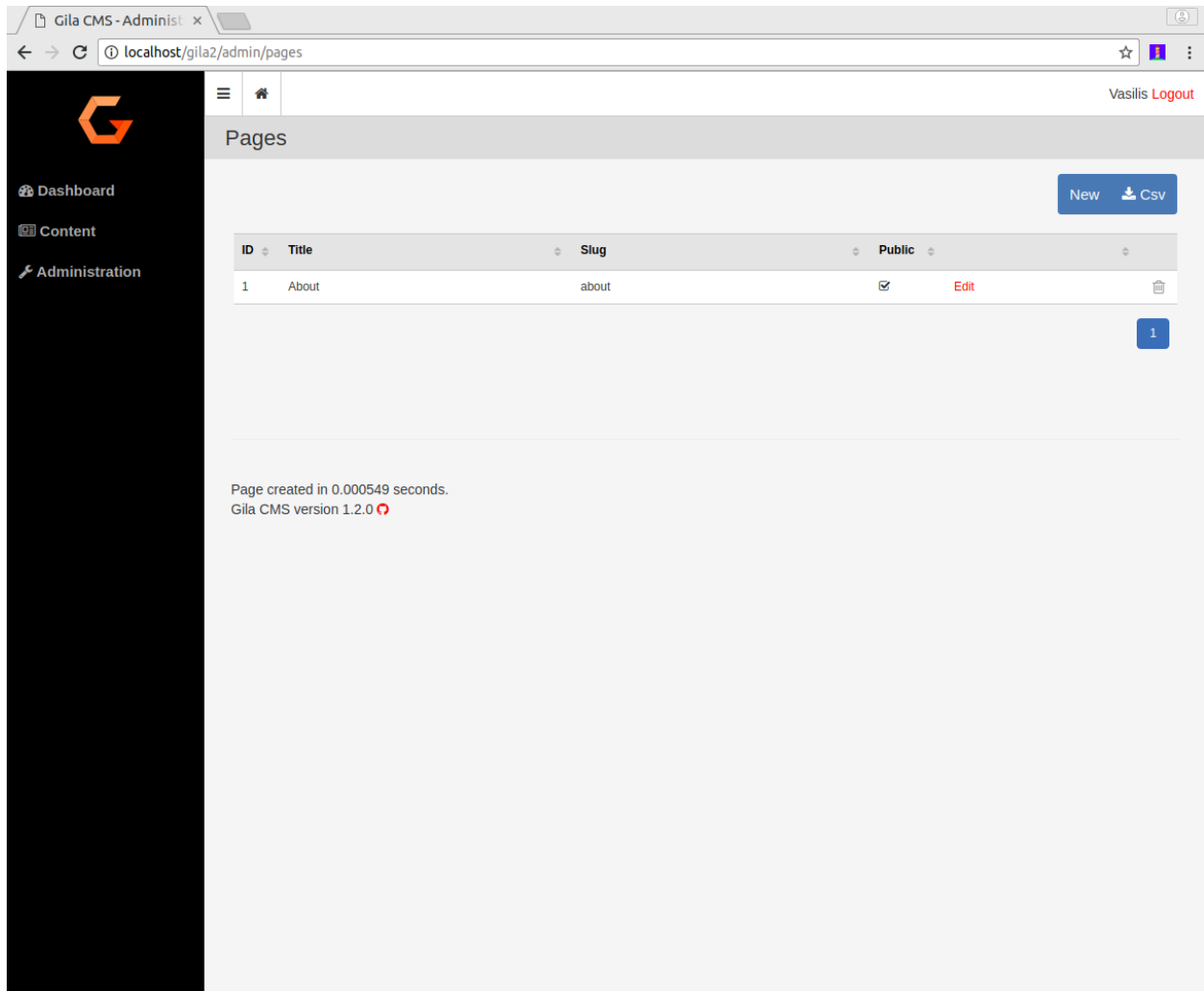
Submit

In the administration menu the Content option gives a submenu of the basic content types of Gila:

- *Pages*
- *Posts*
- *Categories*
- *Media*

3.1 Pages

Pages are the basic content type. A page can be just a text or have media. The information of a page is independent of time so you want them to be found by the visitor in the same place, like on the menu of the website.



Every page have four values:

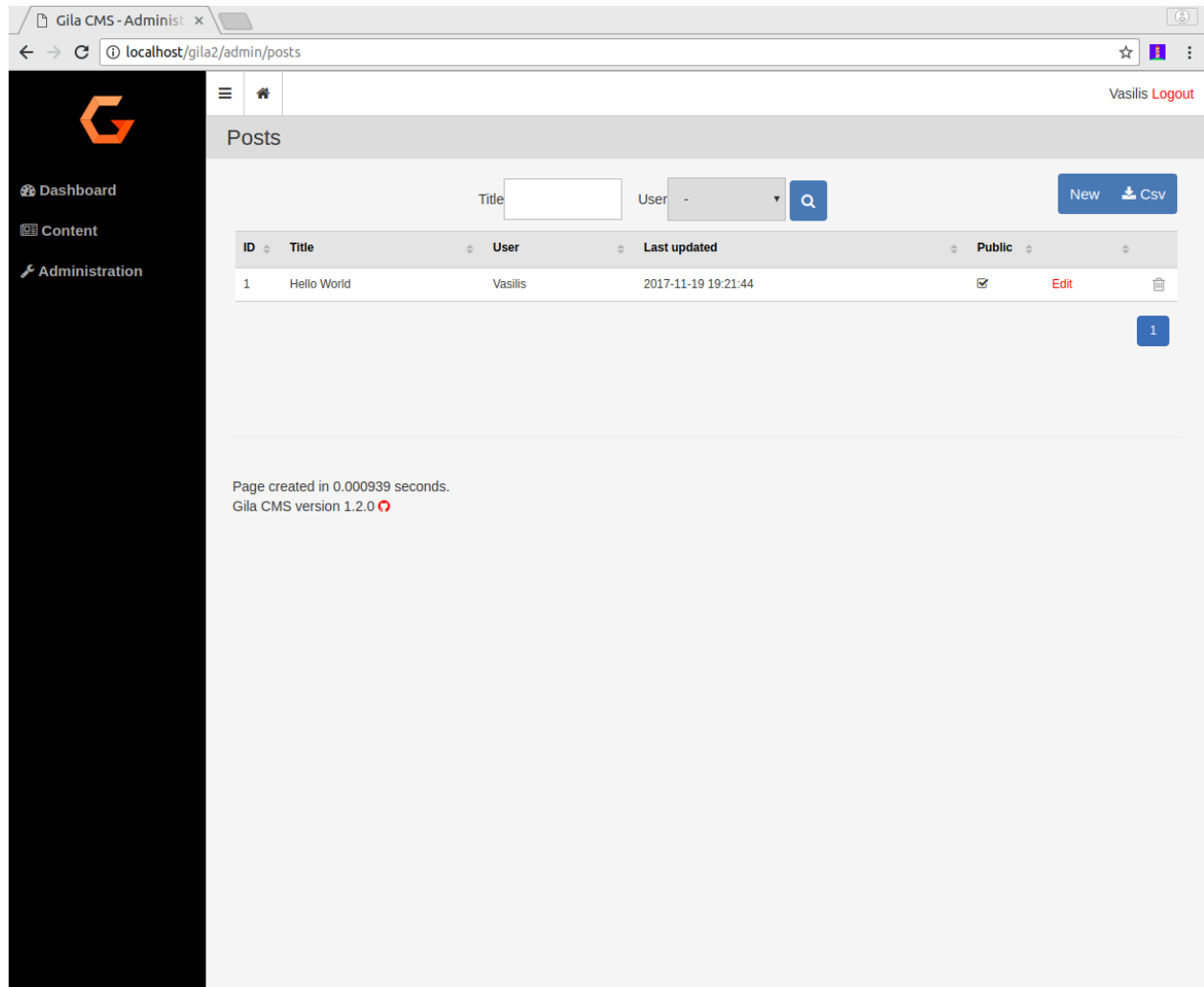
- **ID:** a unique identifier
- **Title:** the title of the page
- **Slug:** is the path of the page. For example the path of a page with title 'My Page' will be *mysite.com/my-page*
- **Public:** an on/off flag. If Public value is off for 'My Page' then *mysite.com/my-page* wont be accessible from the browser.

To **create a new page** click on button **New** that you see on the up-right corner of the table.

The screenshot shows the Gila CMS Admin interface in a web browser. The browser's address bar shows the URL `localhost/gila2/admin/pages/new`. The page title is "Pages". On the left, there is a dark sidebar with the Gila logo and navigation links: "Dashboard", "Content", and "Administration". The main content area has a form for creating a new page. It includes a "Title" input field, a "Public" toggle switch (which is currently turned on), and a rich text editor with a menu bar (File, Edit, Insert, View, Format, Table, Tools) and a toolbar with various formatting options. Below the editor is a "Submit" button. At the bottom of the page, a status message reads: "Page created in 0.001480 seconds. Gila CMS version 1.2.0".

3.2 Posts

The posts can be news or articles about your business or the interests of the web-site. They are organized in categories and are listed in chronological order.



assets/new-post.png

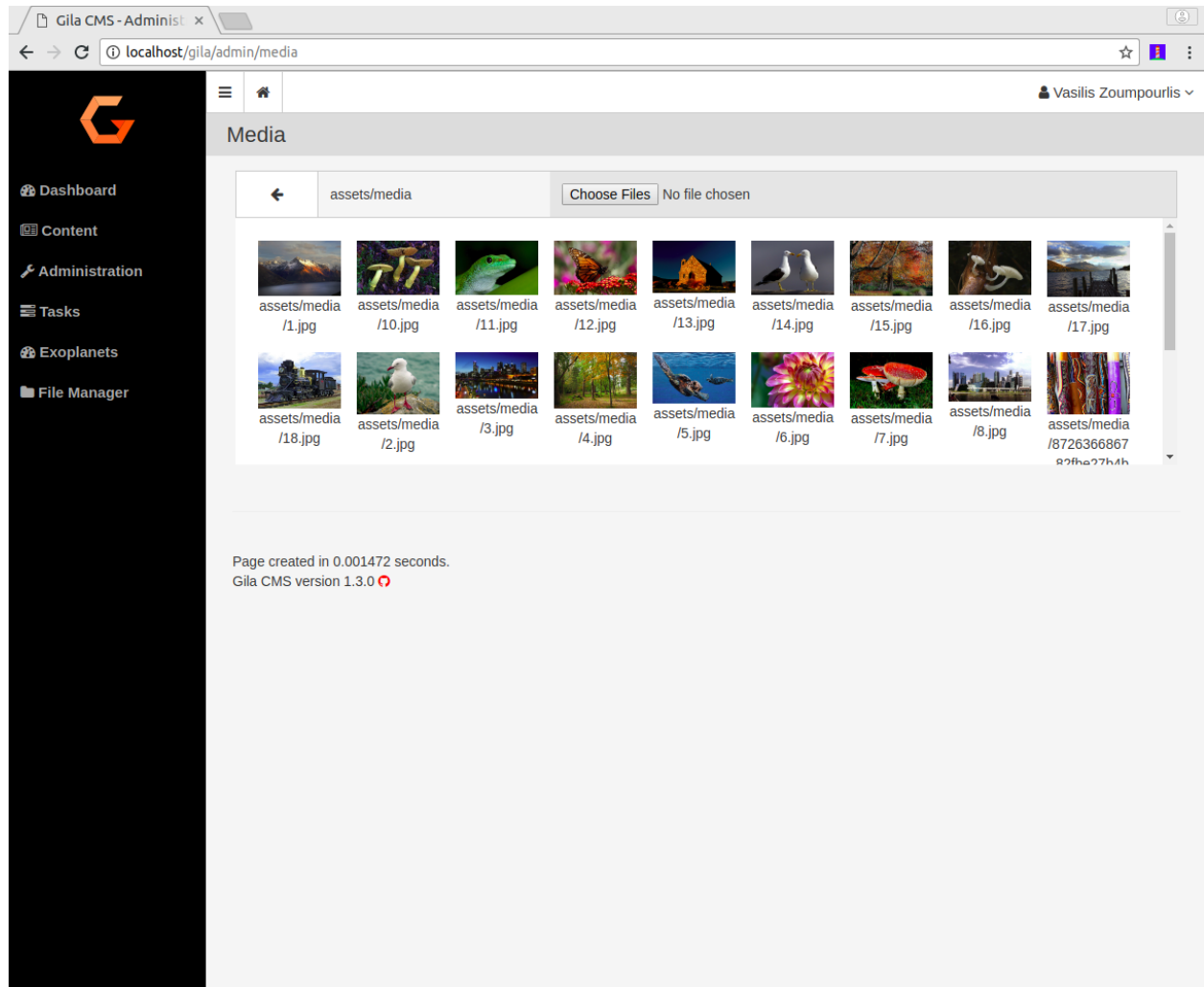
To **create a new post** click on button **New** that you see on the up-right corner of the table.

3.3 Categories

Categories are used to categorize posts or maybe other popular content that you could use later. You only add or edit the names of the categories.

3.4 Media

Media are the images that you want to use for your posts. They are saved as files and not in the database like the other content types. The root directory of media is */assets*. The files and subfolders of */assets* are visible in the public by the path *mysite.com/assets* so you should not upload files or images that you don't want to be found from search engines.



In the administration menu you the Administration option gives a submenu of the basic administration areas

- *Users*
- *Widgets*
- *Packages*
- *Themes*
- *Settings*

4.1 Users

Users are the persons that you can grant access to website and give theme privileges to create or edit content.

...Under development...

4.2 Widgets

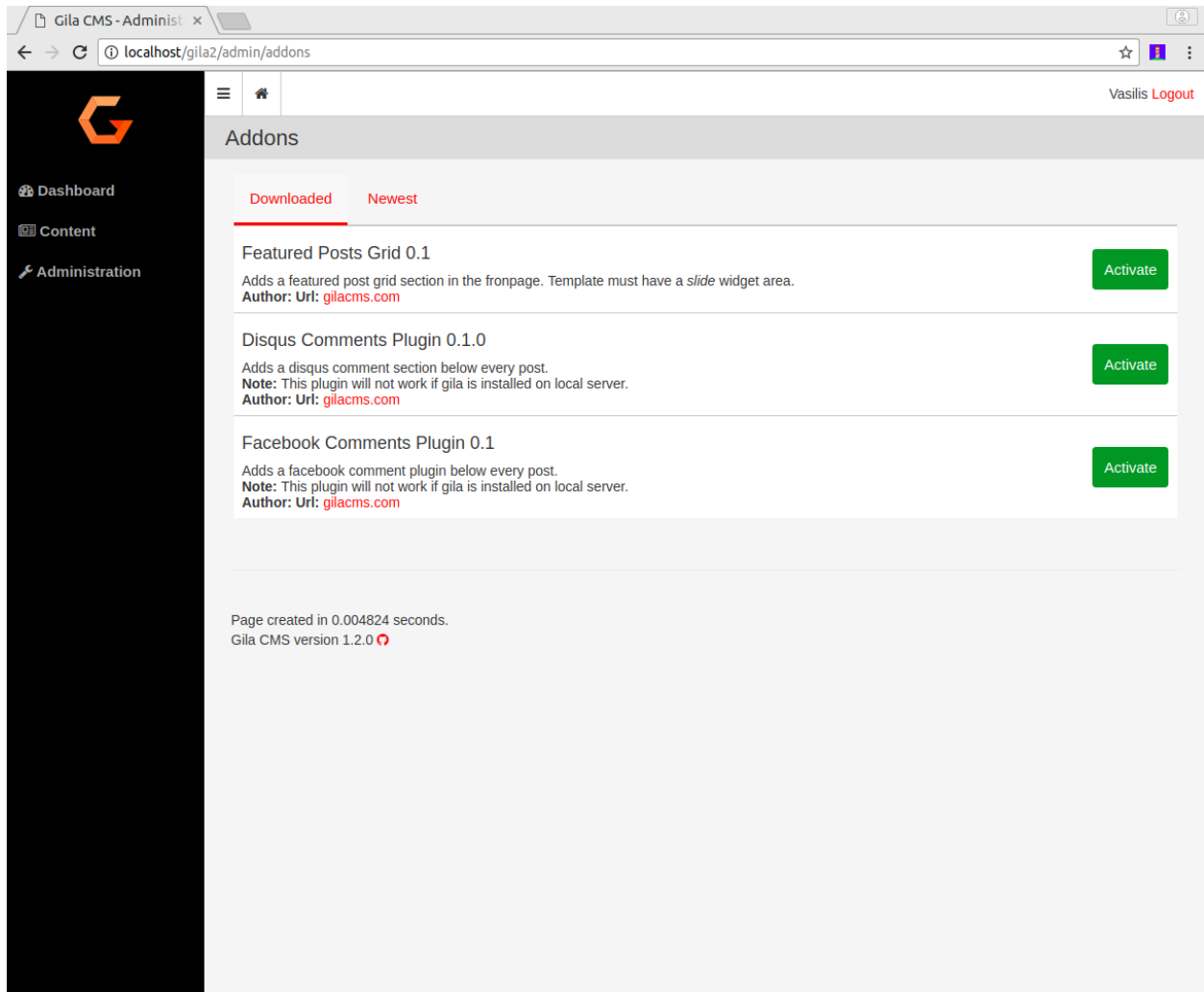
Widgets are some blocks that you can show them on the layout of the website and improve the user experience of the visitors. Widgets can be for example *menus*, *comment sections*, *text blocks*, *lists of links*.

...Under development...

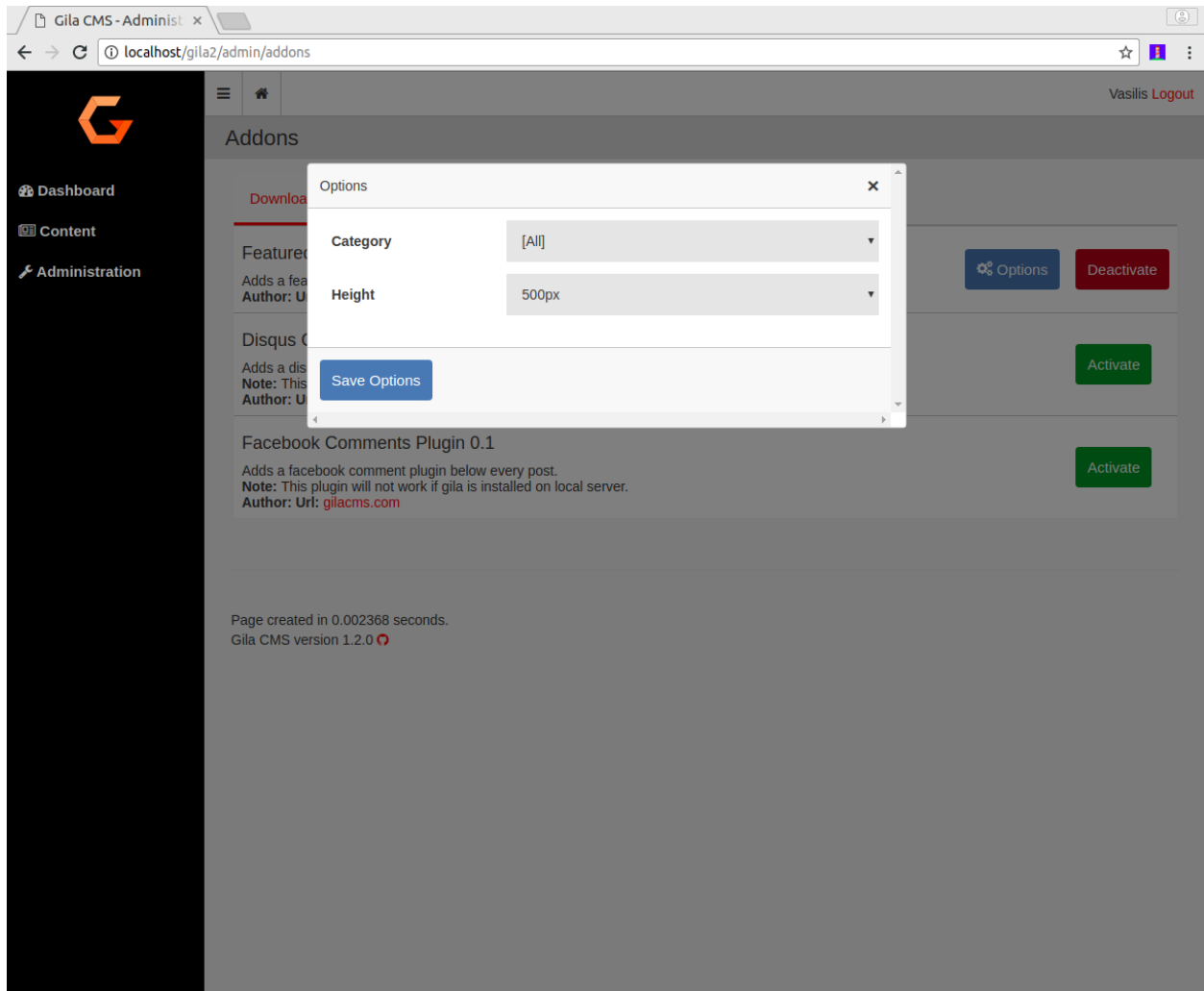
4.3 Packages

Packages give new functionalities on your web application. They may add a specific widget, a few new links in the administration menu or add new content and new templates to show the content. For example *Facebook Comments Plugin* add a facebook comments section below every page post. *Featured Posts Grid* show the thumbnails photos from featured posts in the front page of a blog theme.

You can administrate packages from Administration->Packages



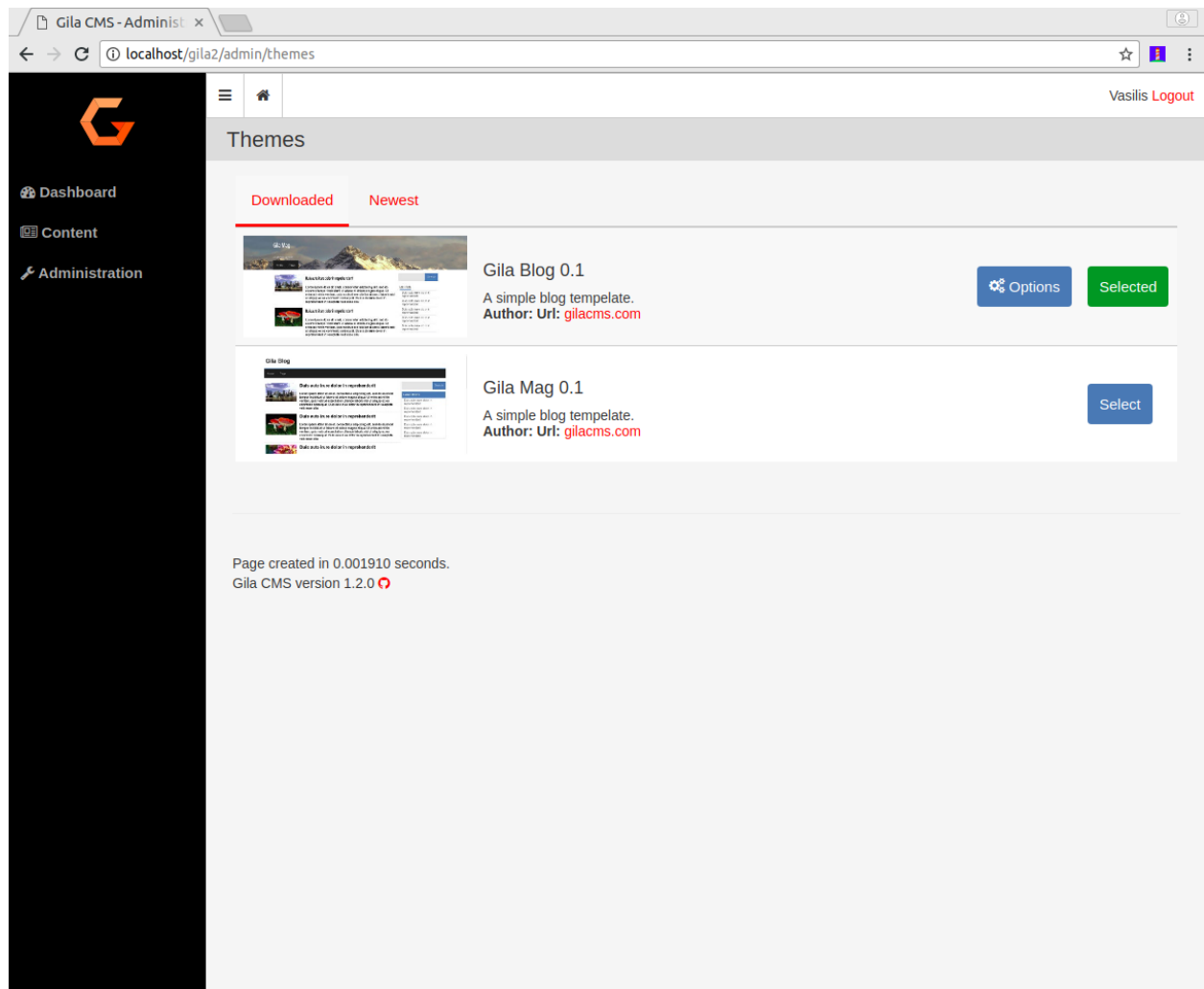
The installed packages usually show an **Options** button. By clicking this button you can change some parameters for the specific package. When you save the settings the changes will take effect by reloading the page.



4.4 Themes

Themes change the look and style of your website. They use different colors and fonts and helps your visitors identify your website and improve their user experience (UX).

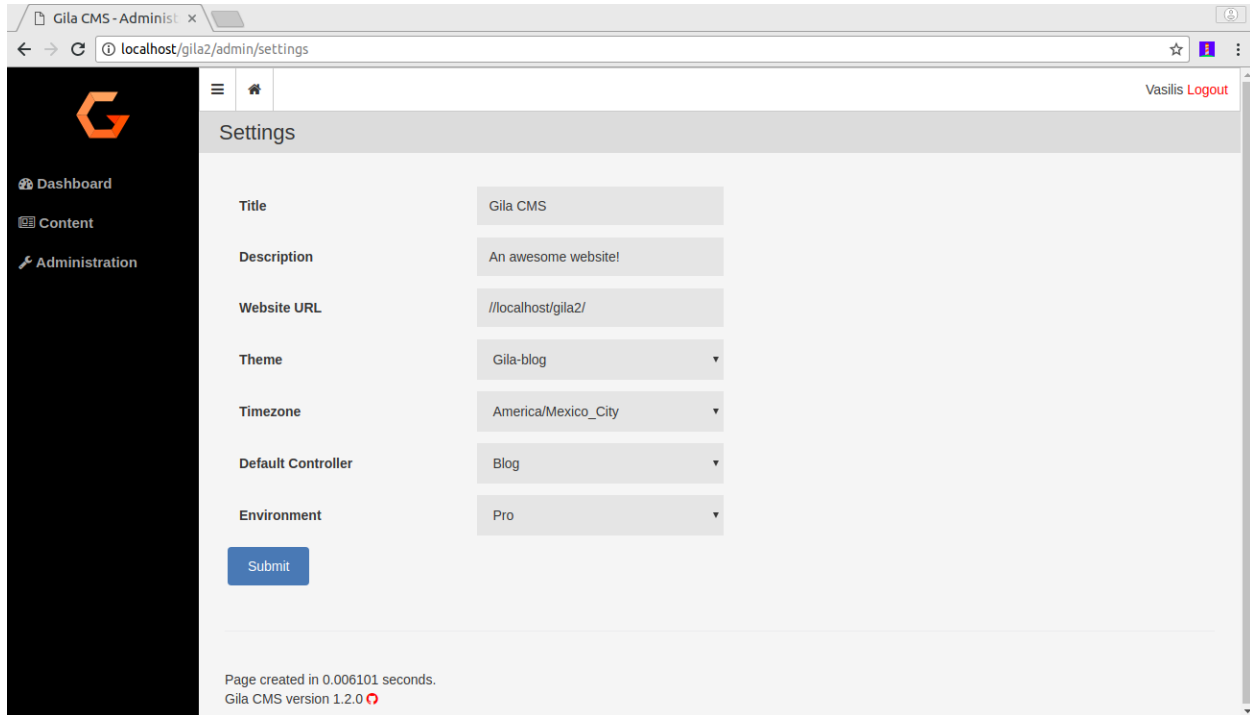
You can select the theme from Administration->Themes



The selected theme usually shows an **Options** button. By clicking this button you can change some options for the theme like the header image (logo) of the website or the main color.

4.5 Settings

On Administration->Settings page and we can make the following configurations



- **Title** is the website title. It will appear up from the menu if we don't use a logo from the theme options.
- **Description** is a small text that describes the website.
- **Website URL** the url path like 'www.mysite.com' or 'https://mysite.com'
- **Theme** changes the look and style of your website. You can select theme from *Administration->Themes*
- **Timezone** The dates and times saved in posts, logs and the rest of the content will be based on the selected timezone.
- **Default controller** The controller that will be used if the calling path do not provide it as first parameter. For example the **Admin** controller is used when we call `mysite.com/admin` but when we call `mysite.com/my-post` the default controller will be used, which is **Blog**, so these paths are equal with `mysite.com` and `mysite.com/my-post`. There is not need to change the default controller unless you want to change how the website will be used.

In the main folder we can see these folders and files.

```
assets/  
install/  
lib/  
log/  
tmp/  
src/  
themes/  
index.php  
config.php
```

assets/ A public folder where we upload our media files.

install/ A public folder used only for the installation.

lib/ A public folder. Third-party libraries are inside this folder.

log/ A private folder that save logs and the user sessions.

tmp/ A public folder with temporally files created.

src/ The folder of installed packages. Here is all the code of the system.

themes/ The folder of installed themes.

index.html The main index file. For any call, execution starts from here.

config.php The configuration file. It is generated after installation.

5.1 Packages

The source code of Gila CMS is split into packages, even the core files are part of the main package called *core*. The package folders are placed inside *src/* folder and desirably have a similar structure:

```
assets/  
controllers/  
models/  
views/  
package.json  
load.php
```

The folders are optional but very useful to organize better the code. The file **package.json** is a must have as it has the basic information of the package -without it the package is invisible- and the **load.php** is the file that will register new values and events of the package.

A simple **package.json** file:

```
{  
  "name": "Package Name",  
  "version": "1.0.0",  
  "description": "A short descriptive text of your package for what it does.",  
  "url": "package_url.com",  
  "author": "Your Name"  
}
```

You can also add another index in the object called *options*. It will be an array of objects, the objects are the options to be stored. The index is the option name and it can have optional values with the following indexes:

- **title** the option name to be displayed, if not specified, the index is used
- **type** select | postcategory
- **options** array of {value:display_text}, it is required if is set type:select

```
{  
  ...  
  "options": {  
    "option1": {},  
    "category": {  
      "type": "postcategory"  
    },  
    "lang": {  
      "title": "Language",  
      "type": "select",  
      "options": {  
        "en_US": "English", "es_ES": "Spanish", "gr_GR": "Greek"  
      }  
    }  
  }  
}
```

You can get the option values like that:

```
// options are saved using as prefix the package's folder name  
// for example if the package has the folder my_package/  
  
$option1 = gila::option("my_package.option1");  
$lang = gila::option("my_package.lang", "en_US"); // use default value
```

A simple **load.php** file could be:

```
<?php
// display text below any post
event::listen('post.after',function(){
    echo 'The post has ended';
})
```

IMPORTANT: The first line of the file should include only the opening tag, and not use later the closing tag.

For any question/feature proposal/help needed [Make a new issue](#)

6.1 Core Classes

class gila

Common methods for Gila CMS

controllers (*\$list*)

(static) Register new controllers.

Parameters **Array \$list** (*Assoc*) – Example: {'ctrl'=>'my_package/controllers/ctrl'}

widgets (*\$list*)

(static) Register new widgets.

Parameters **\$list** (*Array*) – Example: {'wdg'=>'my_package/widgets/wdg'}

amenu (*\$list*)

(static) Add new elements on administration menu.

Parameters **\$list** (*Array*) – Example: ['Item', 'controller/action', 'icon'=>'item-icon']

amenu_child (*\$h, \$item*)

(static) Add a child element on administration menu item.

Parameters

- **\$h** (*string*) – Index of the parent item.
- **\$item** (*Array*) – Example: ['SubItem', 'controller/action_1', 'icon'=>'item-icon']

config (*\$key, \$value = null*)

(static) Sets or gets the value of configuration element.

Parameters

- **\$key** (*string*) – Index of the element.

- **\$value** (*) – (optional) The value.

Returns The value if parameter \$value is not sent.

updateConfigFile ()

(static) Updates the configuration file.

equal (\$v1, \$v2)

(static) Checks if two values are set and have the same value.

Parameters

- **\$v1** (*) – First value.
- **\$v2** (*) – Second value.

Returns True or false.

hash (\$pass)

(static) Generates a hash passwd from a string.

Parameters **\$pass** (*string*) – The string to be hashed.

Returns Hashed password.

option (\$option, \$default=)

(static) Gets an option value.

Parameters

- **\$option** (*string*) – Option name.
- **\$default** (*string*) – (optional) The value to return if there option has not saved value.

Returns The option value.

hasPrivilege (\$pri)

(static) Checks if logged in user has at least one of the required privileges.

Parameters **\$pri** (*string/Array*) – The privilege(s) to check.

Returns True or false.

make_url (\$c, \$action=, \$args=[])

(static) Generates a url.

Parameters

- **\$c** (*string*) – The controller.
- **\$action** (*string*) – The action.
- **\$args** (*Array*) – The parameters in array.

Returns The full url to print.

Examples:

```
$url1 = gila::make_url('blog','post',[1]); returns mysite.com/blog/post/1 $url1 =  
gila::make_url('blog','',['page1']); returns mysite.com/blog/page1
```

class event

Registers and fires events (hooks)

listen (\$event, \$handler)

(static) Sets a new function to run when an event is triggered later.

Parameters

- **\$event** (*string*) – The event name.
- **\$handler** (*function*) – The function to call.

fire (*string \$event* [, *Array \$params*])
(static) Fires an event and calls all handling functions.

Parameters

- **\$event** (*string*) – The event name.
- **\$params** (*function*) – (optional) Parameters to send to handlers.

class view

Have methods that outputs the HTML

function set (**\$param**, **\$value**)
(static) Sets a parameter from a controller action that can be used later from a view file.

Parameters

- **\$param** (*string*) – The parameter name.
- **\$handler** (*any*) – The value.

meta (*\$meta*, *\$value*)
(static) Sets a meta value that is printed later from `view::head()`.

Parameters

- **\$meta** (*string*) – The meta name.
- **\$value** (*string*) – The value.

stylesheet (*\$href*)
(static) Adds a new stylesheet link that is printed later from `view::head()`.

Parameters \$href (*string*) – The href attribute from the link.

script (*\$script*)
(static) Adds a new script to be included in the output HTML.

Parameters \$script (*string*) – The src attribute from the script.

getThemePath ()
(static) Returns the path of the current theme.

head (*\$meta=[]*)
(static) Prints all the head information in `<head>` tag.

Parameters \$file (*Array*) – (optional) Meta values to be printed.

findPath (*\$file*, *\$package = 'core'*)
(static) Returns the path of a file inside theme or package folder.

Parameters

- **\$file** (*Array*) – The file path.
- **\$package** (*string*) – (optional) The package folder where the file is located if is not found in theme folder.

Returns False if file is not found.

render (*\$file*, *\$package = 'core'*)
(static) Prints the view file adding the header.php and footer.php from theme.

Parameters

- **\$file** (*string*) – The file path.
- **\$package** (*string*) – (optional) The package folder where the file is located if is not found in theme folder.

renderAdmin (*\$file*, *\$package* = 'core')

(static) Prints the view file adding the admin/header.php and admin/footer.php from theme.

Parameters

- **\$file** (*string*) – The file path.
- **\$package** (*string*) – (optional) The package folder where the file is located if is not found in theme folder.

renderFile (*\$file*, *\$package* = 'core')

(static) Prints the view file alone from theme.

Parameters

- **\$file** (*string*) – The file path.
- **\$package** (*string*) – (optional) The package folder where the file is located if is not found in theme folder.

includeFile (*\$file*, *\$package* = 'core')

(static) Includes the view file without passing the.

Parameters

- **\$file** (*string*) – The file path.
- **\$package** (*string*) – (optional) The package folder where the file is located if is not found in theme folder.

widget_area (*\$area*, *\$div=true*)

(static) Prints the widgets of a specific area.

Parameters

- **\$area** (*string*) – The widget area name.
- **\$div** (*bool*) – (optional) Also print or not the widget inside a <div> tag with its title.

widget_area (*\$area*, *\$id*, *\$max=180*)

(static) Returns the path of a thumbnail image of specified dimensions. If thumbnail does not exist it will create one.

Parameters

- **\$src** (*string*) – The path of original image.
- **\$id** (*string*) – The name of the thumbnail.
- **\$max** (*int*) – (optional) The maximum width or height of thumbnail in pixels.

CHAPTER 7

Indices and tables

- `genindex`

A

amenu() (gila method), 25
amenu_child() (gila method), 25

C

config() (gila method), 25
controllers() (gila method), 25

E

equal() (gila method), 26
event (built-in class), 26

F

findPath() (view method), 27
fire() (event method), 27

G

getThemePath() (view method), 27
gila (built-in class), 25

H

hash() (gila method), 26
hasPrivilege() (gila method), 26
head() (view method), 27

I

includeFile() (view method), 28

L

listen() (event method), 26

M

make_url() (gila method), 26
meta() (view method), 27

O

option() (gila method), 26

R

render() (view method), 27
renderAdmin() (view method), 28
renderFile() (view method), 28

S

script() (view method), 27
stylesheet() (view method), 27

U

updateConfigFile() (gila method), 26

V

view (built-in class), 27

W

widget_area() (view method), 28
widgets() (gila method), 25