
Gila CMS Documentation

Release 1.7.1

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	13
3.4	Media	13
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	Examples	25
6.1	Event: Post Tags	25
6.2	Widget: Twitter Timeline	26
7	Api	29
7.1	Core Classes	29
8	Indices and tables	33

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 include 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 [Gitter](#)
- 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*, *gd* 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.

On nginx server you will need to configure the redirects in */etc/nginx/sites-enabled/default* ([issue #1](#))

```
location / {
    index index.php index.html index.htm;
    rewrite gila/(?!install) (?!src) (?!themes) (?!lib) (?!assets) (?!tmp) (?!robots.txt) (.
↪*)$ /gila/index.php?url=$1 last;
}
```

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*

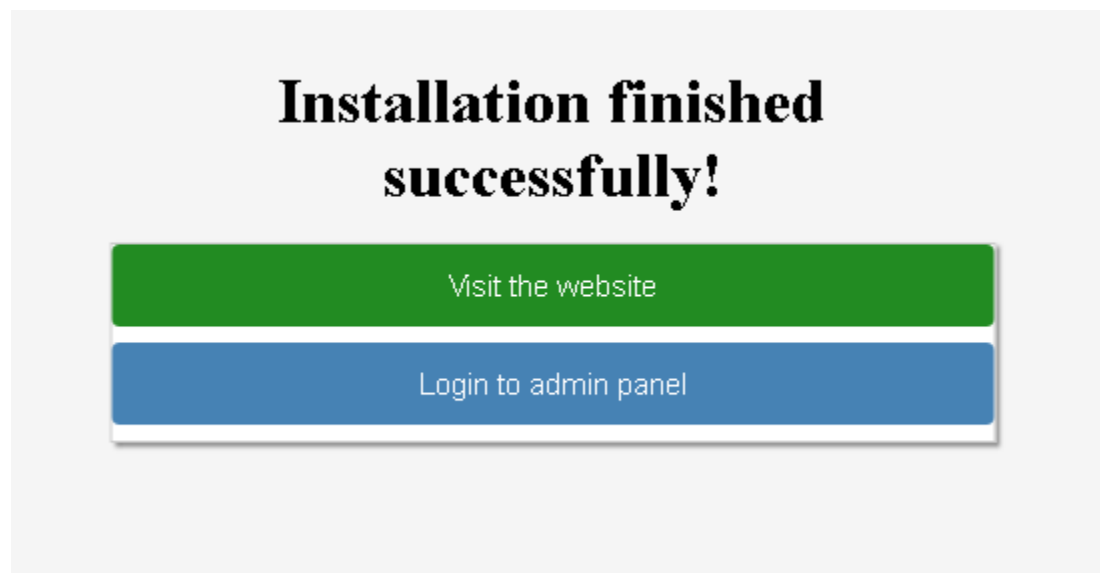
Gila CMS Installation

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

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/*

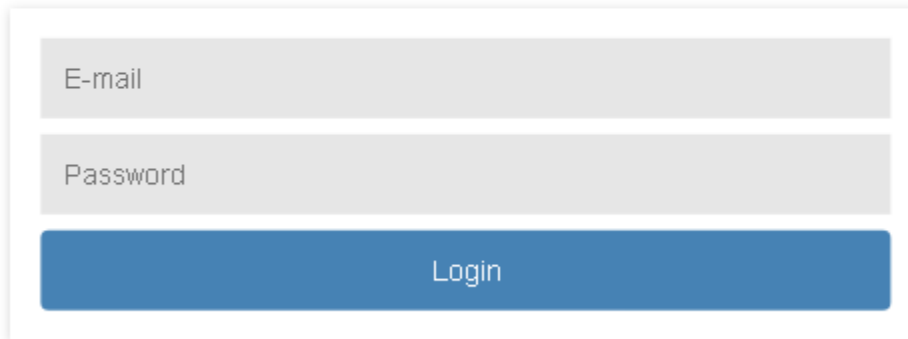
After filling the data and submit them, we wait a few seconds until the installation is finished.



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



Login



E-mail

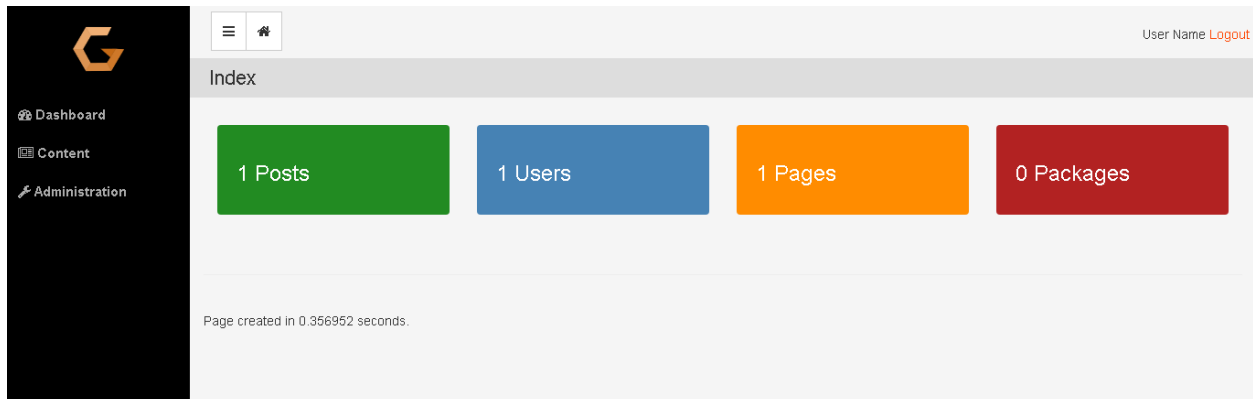
Password

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.

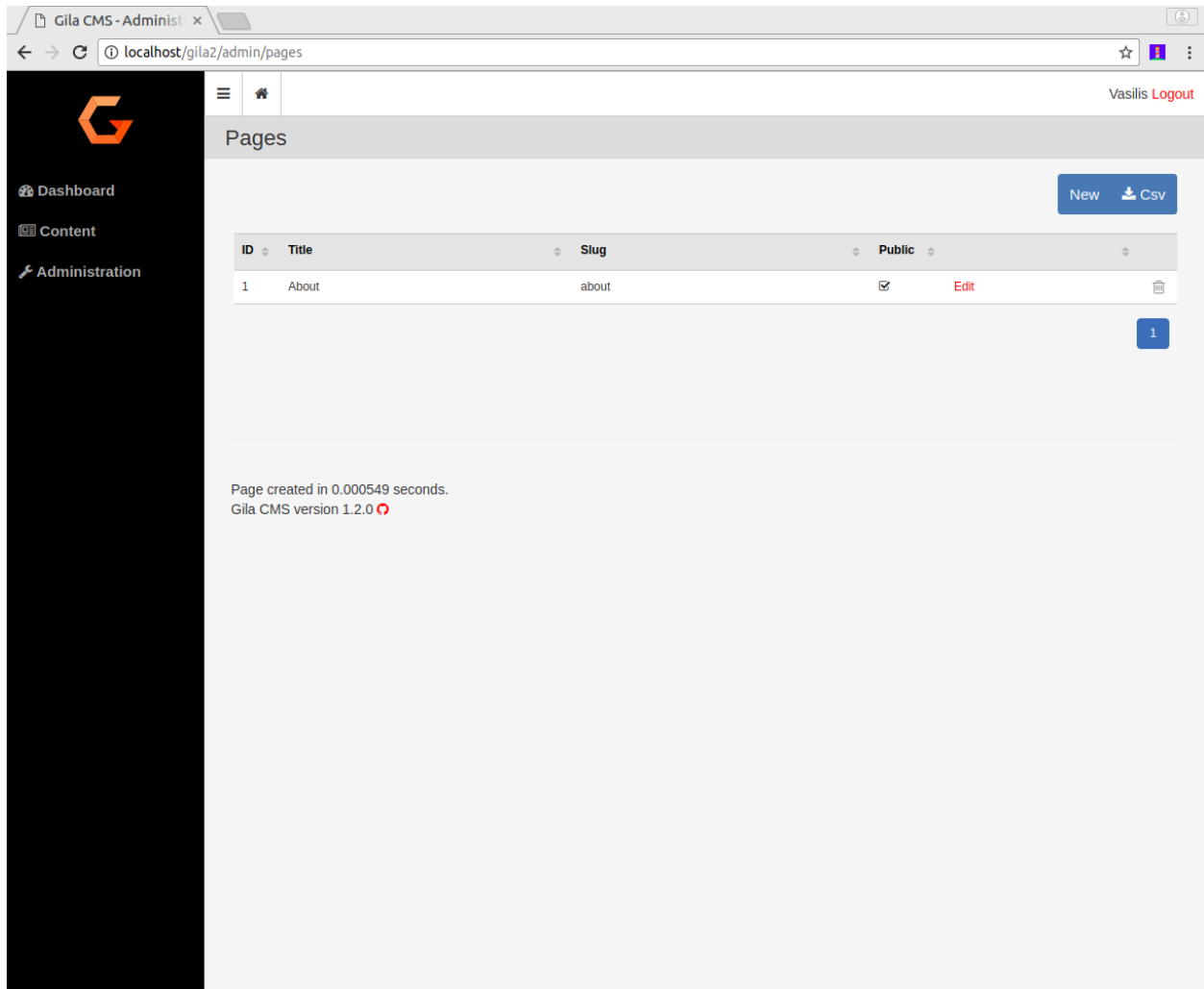
Title	Gila CMS
Description	An awesome website!
Website URL	//localhost/gila/
SSL URL	
Theme	Base ▼
Timezone	America/Mexico_City ▼
Default Controller	Blog ▼
Environment	Dev ▼

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

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

3.1 Pages

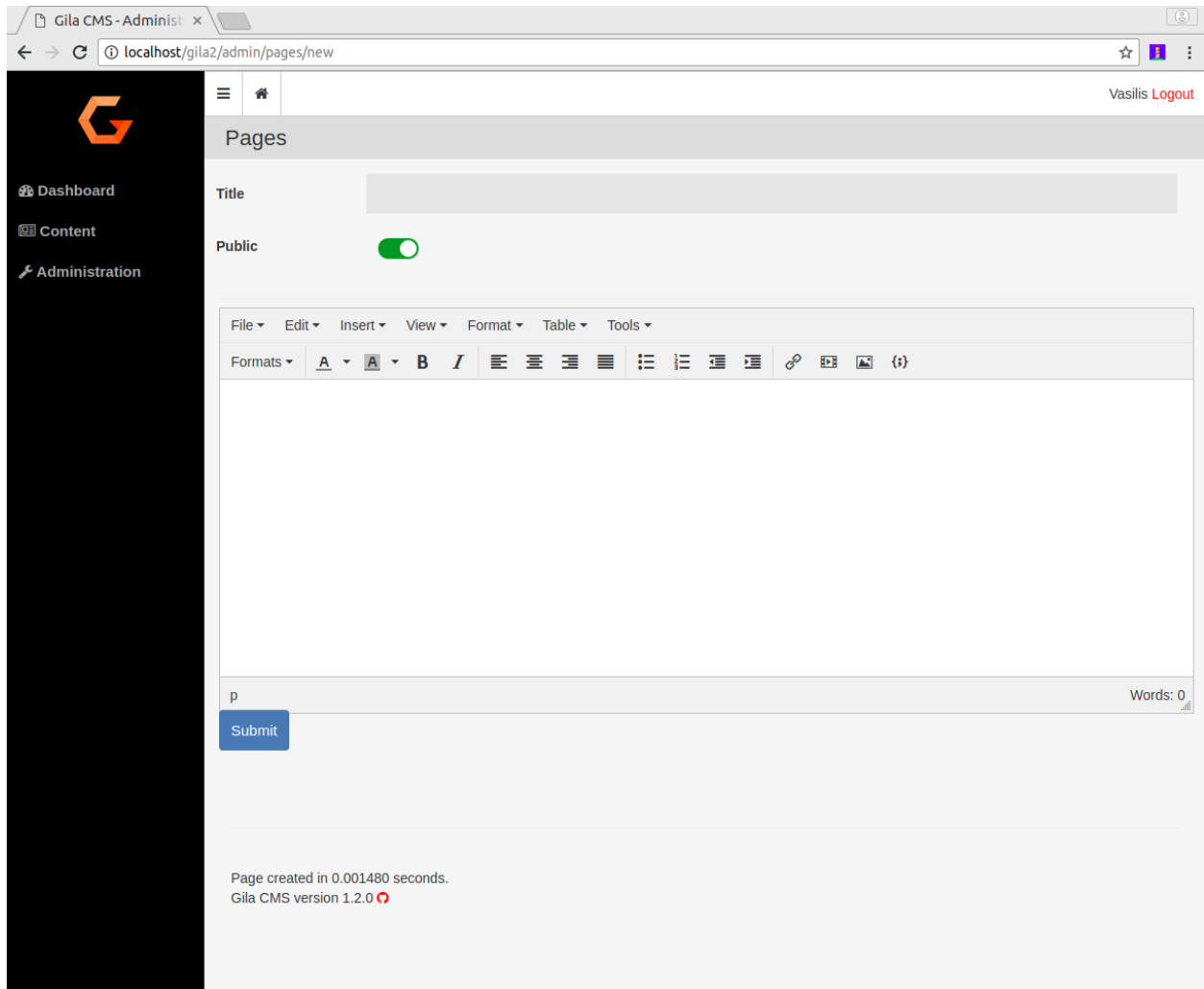
Pages are the basic content type. A page can be just a text or have media. The information of a page is independent of time so you want them to be found by the visitor in the same place, like on the menu of the website.



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 for creating a new page. The browser address bar shows `localhost/gila2/admin/pages/new`. The user is logged in as Vasilis. The interface includes a sidebar with navigation options: Dashboard, Content, and Administration. The main content area is titled "Pages" and contains a form with the following elements:

- Title:** A text input field.
- Public:** A toggle switch that is currently turned on.
- Rich Text Editor:** A WYSIWYG editor with a menu bar (File, Edit, Insert, View, Format, Table, Tools) and a toolbar with icons for text formatting (bold, italic, underline, strikethrough), alignment, list creation, link insertion, and image insertion.
- Text Area:** A large text area containing the letter "p".
- Word Count:** A label "Words: 0" in the bottom right corner of the text area.
- Submit Button:** A blue button labeled "Submit" located below the text area.

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 website. They are organized in categories and are listed in chronological order.

The screenshot shows the Gila CMS Admin interface. On the left is a dark sidebar with the Gila logo and navigation links for Dashboard, Content, and Administration. The main content area is titled 'Posts' and features a search bar with 'Title' and 'User' filters, a search button, and 'New' and 'Csv' buttons. Below the search bar is a table with one row of data. The table has columns for ID, Title, User, Last updated, Public, and actions (Edit and Delete). At the bottom of the page, it says 'Page created in 0.000939 seconds. Gila CMS version 1.2.0'.

ID	Title	User	Last updated	Public	
1	Hello World	Vasilis	2017-11-19 19:21:44	<input checked="" type="checkbox"/>	Edit

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

Title	<input type="text"/>
Slug	<input type="text" value="Generate from title"/>
Description	<input type="text" value="A small description of 200 characters"/>
Thumbnail	<input type="text" value="Thumbnail image"/>
Categories	<input type="text"/>
Tags	<input type="text" value="values seperated by comma"/>
Public	<input checked="" type="checkbox"/>

File Edit View Insert Format Table

Formats A A B I [List Icons]

0 WORDS POWERED BY TINYMCE

Submit

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.

The screenshot shows the Gila CMS Admin interface. The browser address bar displays 'localhost/gila/admin/media'. The user is identified as 'Vasilis Zoumpourlis'. The main content area is titled 'Media' and shows a gallery of 18 image files. Each file is represented by a thumbnail and a filename starting with 'assets/media/'. The files are arranged in two rows of nine. The first row contains files /1.jpg through /17.jpg. The second row contains files /18.jpg, /2.jpg, /3.jpg, /4.jpg, /5.jpg, /6.jpg, /7.jpg, /8.jpg, and a file with a long alphanumeric name. A 'Choose Files' button and 'No file chosen' text are visible at the top of the gallery. A sidebar on the left contains navigation links: Dashboard, Content, Administration, Tasks, Exoplanets, and File Manager. At the bottom of the page, it states 'Page created in 0.001472 seconds. Gila CMS version 1.3.0'.

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 them privileges to create or edit content.

...Under development...

4.2 Widgets

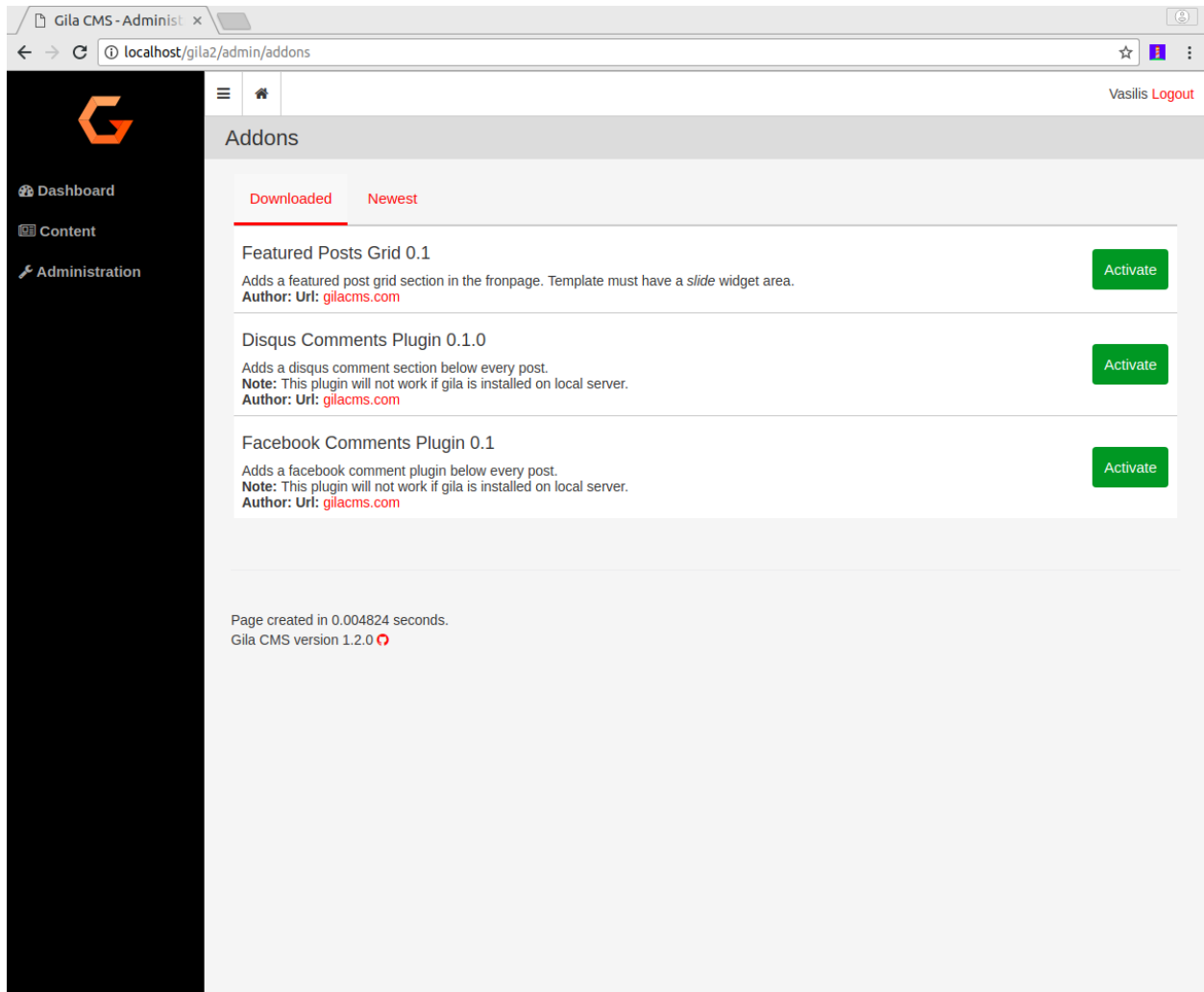
Widgets are some blocks that you can show them on the layout of the website and improve the user experience of the visitors. Widgets can be for example *menus*, *comment sections*, *text blocks*, *lists of links*.

...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

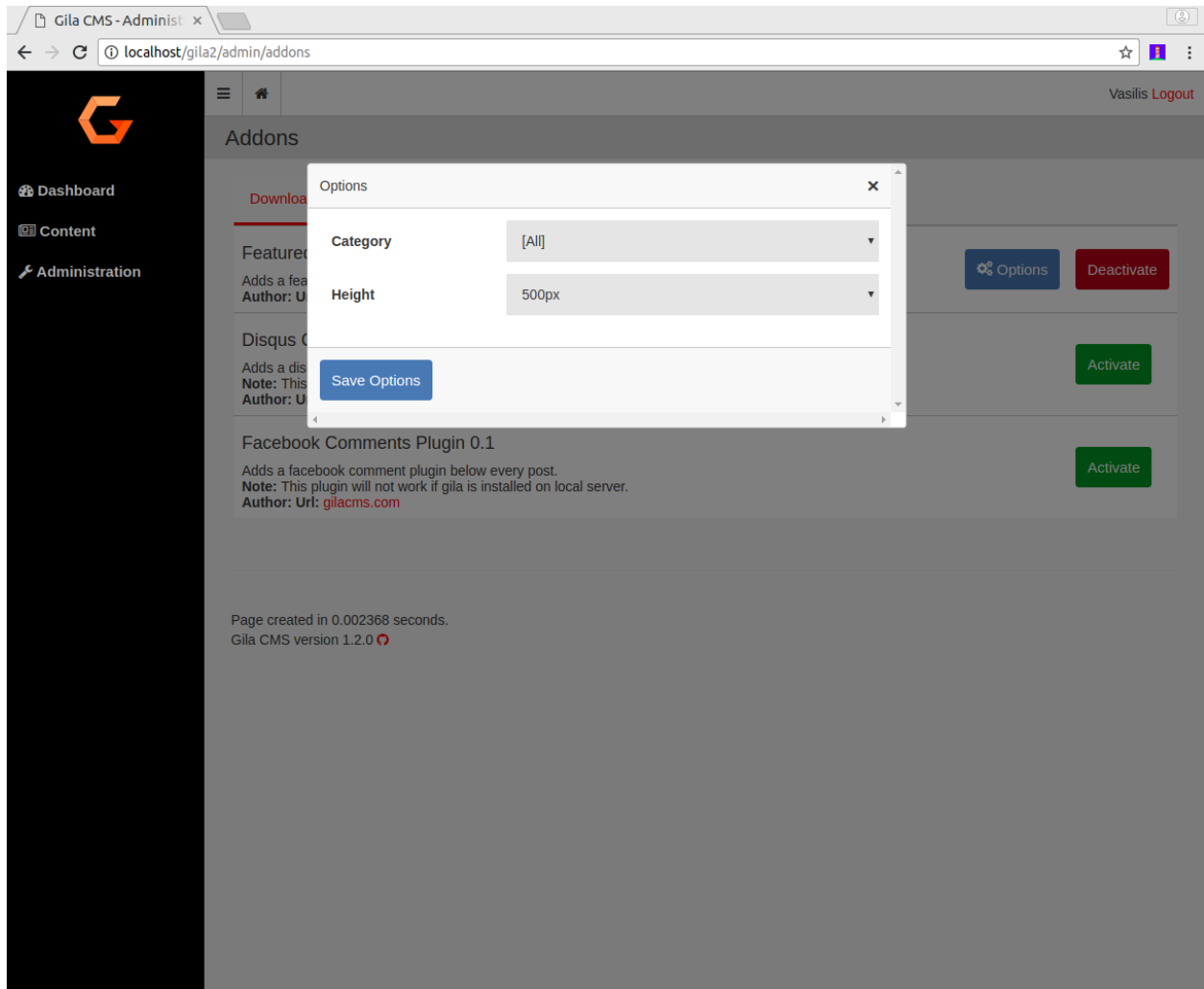


The screenshot shows a web browser window with the URL `localhost/gila2/admin/addons`. The page title is "Addons". On the left, there is a dark sidebar with a logo and navigation links: "Dashboard", "Content", and "Administration". The main content area has two tabs: "Downloaded" (selected) and "Newest". Below the tabs, there are three plugin cards, each with an "Activate" button:

- Featured Posts Grid 0.1**
Adds a featured post grid section in the fronpage. Template must have a *slide* widget area.
Author: [Uri: gilacms.com](http://gilacms.com)
- Disqus Comments Plugin 0.1.0**
Adds a disqus comment section below every post.
Note: This plugin will not work if gila is installed on local server.
Author: [Uri: gilacms.com](http://gilacms.com)
- Facebook Comments Plugin 0.1**
Adds a facebook comment plugin below every post.
Note: This plugin will not work if gila is installed on local server.
Author: [Uri: gilacms.com](http://gilacms.com)

At the bottom of the page, it says: "Page created in 0.004824 seconds. Gila CMS version 1.2.0".

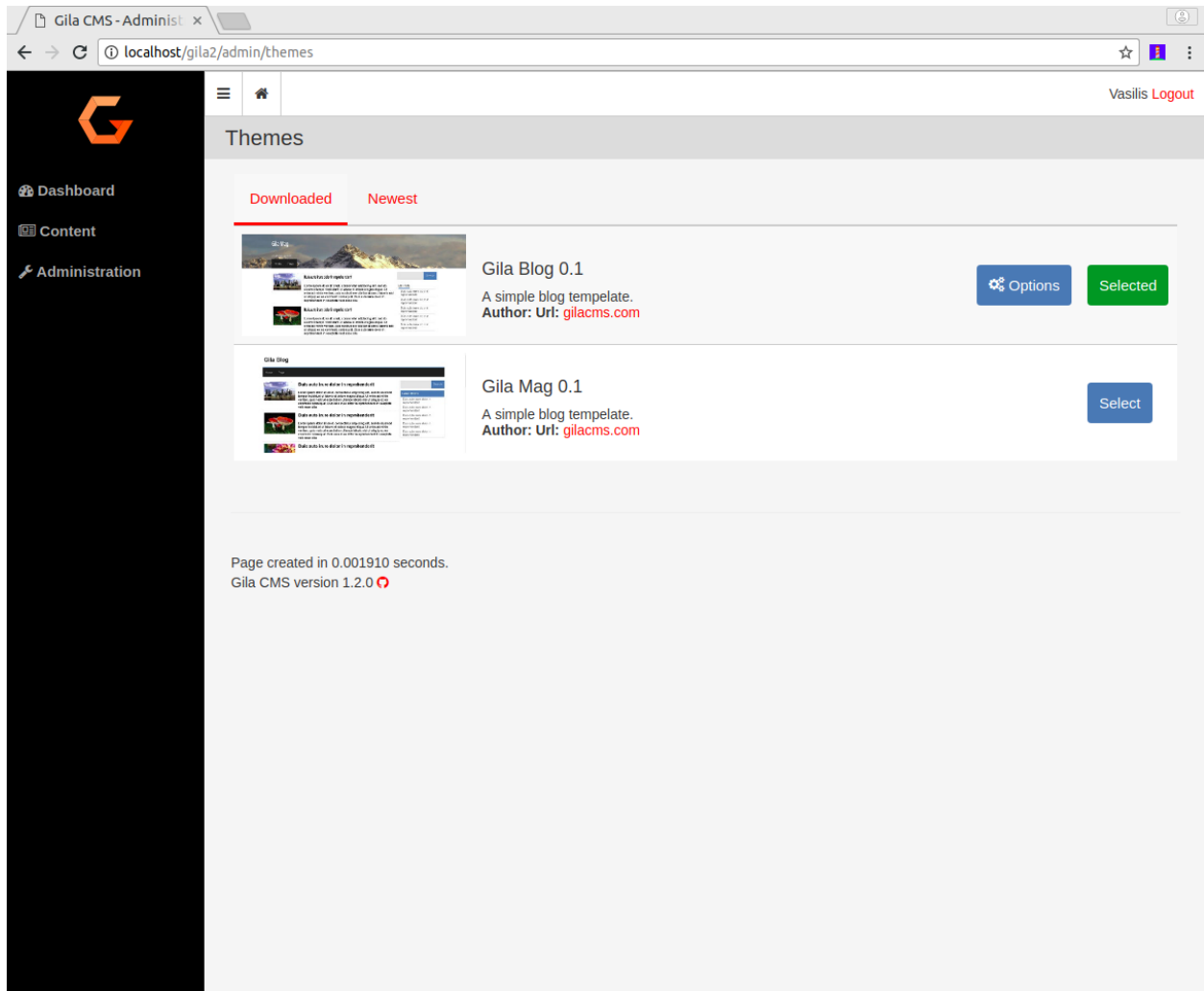
The installed packages usually show an **Options** button. By clicking this button you can change some parameters for the specific package. When you save the settings the changes will take effect by reloading the page.



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).

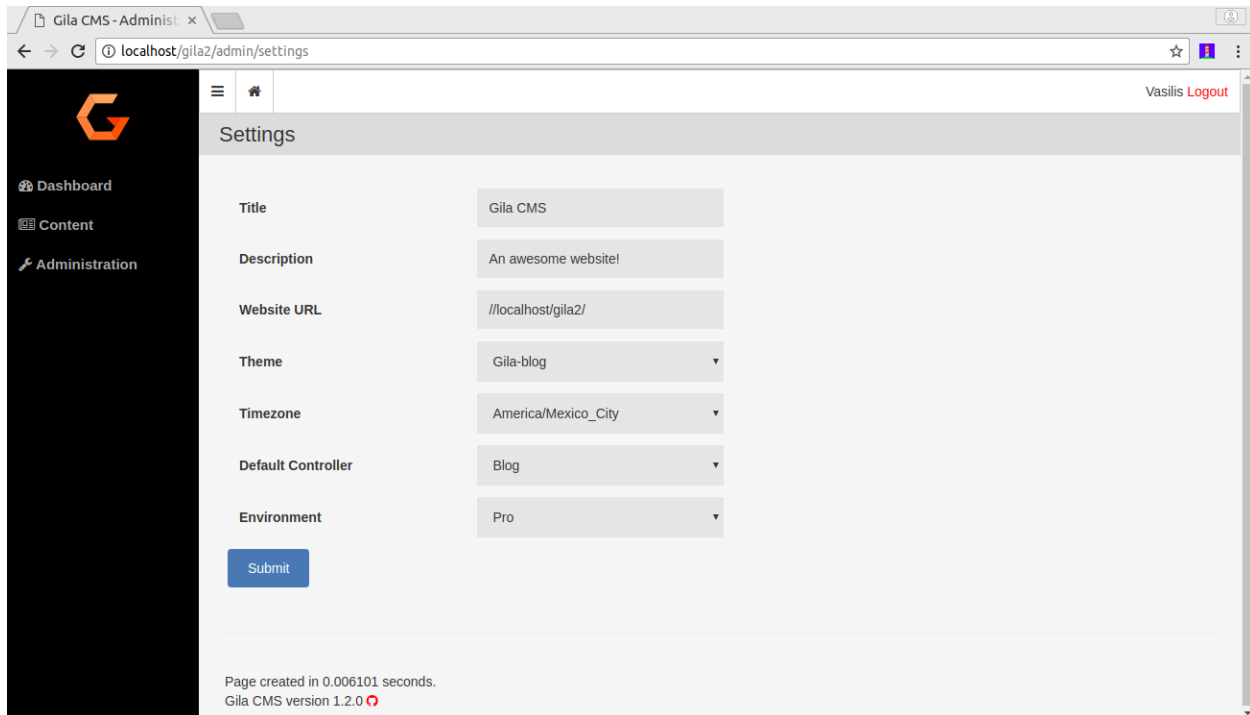
You can select the theme from Administration->Themes



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](#)

Here are some examples to you get into the of package creation.

6.1 Event: Post Tags

In this example we will list the tags of a post just after it. Create a folder inside *src/* and name it *post-tags*. Inside it create the following files

```
package.json  
load.php  
logo.png
```

package.json is essential so the package can be seen from the package manager. Put these values in it:

```
{  
  "name": "Post Tags",  
  "version": "1.0.0",  
  "description": "A package to show tags below the post."  
}
```

load.php will run a code when the package is active. We will register a function to run right after the post display.

```
<?php  
  
event::listen('post.after', function(){  
    global $g; // $g will give us the post id  
    $tags = core\models\post::meta($g->id, 'tag'); // get the tag list of post  
    echo "<strong>TAGS:</strong> ";  
    foreach ($tags as $tag) echo " <a href='tag/$tag'>#$tag</a>";  
});
```

This function will run when the *post.after* event is dispatched. That happens with *event::fire('post.after')*; or *event::widget_area('post.after')*; from *single-post.php* view file.

logo.png is the package's logo and is displayed in the package list.

Activate the package in */admin/packages*. After that you should see the list of TAGS below any blog post.

6.2 Widget: Twitter Timeline

In this example we will create a widget that displays the last tweets of an account. Instead of using an event to run the code we let the user create instances of the widget choose in which widget area want to display the twitter plugin. Inside *src/* create a folder *twitter-timelines* and add the following files:

```
package.json
load.php
widgets/twitter-timeline/widget.php
widgets/twitter-timeline/twitter-timeline.php
```

package.json:

```
{
  "name": "Twitter Timelines",
  "version": "1.0.0",
  "description": "Installs a widget to display twitter timelines."
}
```

load.php:

```
<?php

// registers the widget name and its path
gila::widgets([
  'twitter-timeline' => 'twitter-timelines/widgets/twitter-timeline'
]);
```

widgets/twitter-timeline/widget.php will include the widget options we want to use. In this case we need the user account and the name to be displayed.

```
<?php

$options=[
  'accountID'=>[
    'title'=>'Twitter Account'
  ]
];
```

widgets/twitter-timeline/twitter-timeline.php is the view file of the widget, it will generate the html code. We use the embedding Twitter content from [here](#)

```
<?php
$account = gila::option('twitter-timelines.accountID', 'gilacms');
?>
<a class="twitter-timeline" data-height="400" href="https://twitter.com/<?=$account?>
↪">Tweets by <?=$account?></a>
<script async src="https://platform.twitter.com/widgets.js" charset="utf-8"></script>
```

gila::option() gets the option of the package that we set up in the package settings. A default value can be used if the option is null.

Activate the package. Now in `/admin/widgets` you can create a new widget with type `twitter-timeline` and set the widget area `sidebar` or `dashboard` to see it.

7.1 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 password 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 8

Indices and tables

- genindex

A

amenu() (gila method), 29
amenu_child() (gila method), 29

C

config() (gila method), 29
controllers() (gila method), 29

E

equal() (gila method), 30
event (built-in class), 30

F

findPath() (view method), 31
fire() (event method), 31

G

getThemePath() (view method), 31
gila (built-in class), 29

H

hash() (gila method), 30
hasPrivilege() (gila method), 30
head() (view method), 31

I

includeFile() (view method), 32

L

listen() (event method), 30

M

make_url() (gila method), 30
meta() (view method), 31

O

option() (gila method), 30

R

render() (view method), 31
renderAdmin() (view method), 32
renderFile() (view method), 32

S

script() (view method), 31
stylesheet() (view method), 31

U

updateConfigFile() (gila method), 30

V

view (built-in class), 31

W

widget_area() (view method), 32
widgets() (gila method), 29