MY PHP FRAMEWORK DOCUMENTATION

Contents

Views

How to create view

All view files are located in views folder (app/views). View files must have .view.php extension. Middlewear allows you to use few syntactic sugars in your view, such as printing variables, including other views etc...

Example I: home view

File: **home.view.php**

```
<h1>Test</h1>

<br>
```

Variables

Printing variables in views is also possible. (variables are defined in controllers).

For printing variables you have to use *{{var}}*.

Example II: home view with variables

File: **home.view.php**

```
<h1>Test</h1>

<br>

My name is: {{ir['username']}}
```

Loops

Example III: home view with for loop

File: **home.view.php**

```
@for i in range(1,5)
        Line: {{i}}<br>
@endfor
```

You get:

```
Line: 1
Line: 2
Line: 3
Line: 4
```

```
@for i in range(5)
        Line: {{i}}<br>
@endfor
```

You get:

```
Line: 0
Line: 1
Line: 2
Line: 3
Line: 4
```

```
@for i in range(1,10,2)
        Line: {{i}}<br>
@endfor
```

You get:

```
Line: 1
Line: 3
Line: 5
Line: 7
Line: 9
```

Including views

Example IV: home view with more views included

File: **home.view.php**

```
[include]app/views/base.view.php[/include]

<h1>Test</h1>

<br>

My name is: {{ir['username']}}

@for i in range(1,5)

        Line: {{i}}<br>

@endfor

 [include]app/views/footer.view.php[/include]
```

CSRF

Framework offers basic CSRF protection. You can get CSRF token value in view using variable CSRF .

If you want to include CSRF field in form, use CSRFFORM variable.

Example V: Using CSRF token in views

File: **home.view.php**

```
[include]app/views/base.view.php[/include]
<h1>Test</h1>
<br>
My name is: {{ir['username']}}<br>
@for i in range(1,5)

        Line: {{i}}<br>

@endfor

<br>

Reading from cache: {{cVal}} <br>
CSRF TOKEN: {{CSRF}} <br>
<form action='' method='post'>
{{CSRFFORM}}<br>
Username: <input type='text' name='username'>
<br>
<input type='submit'>
</form>
[include]app/views/footer.view.php[/include]
```

Models

Controllers

All controllers extend main controller called Controller.

Example VI: Making controller

```
<?

require_once 'Controller.php';

class Main extends Controller{

}
```

All processed data in controller is passed to view using *show(<view>, <data>)* function.

All data sent to view is put into dictionary.

Example VII: Sending data to view

```
<?

$reloadTimer = 5;

$data = array("reloadTimer" => $reloadTimer);

$this->show("base.view.php", $data);
```

Framework also offers you functions to prevent CSRF attacks.

It is recommended to use CSRF tokens in all forms. That way forms can not get submitted from other sites.

Generating CSRF token is as easy as this

```
$token = $this->generateCSRF();
```

Note that we are still in our controller (but not in main controller called Controller).

If you want CSRF protection to be effective, you must pass it to view and then use it in form. Remember this {{CSRFFORM}}?

Checking if CSRF token is legit is also piece of cake.

```
$this->checkCSRF($_POST['csrf']);
```

This is called in your controller function (post or get). If you want your page to stop loading after it detects that CSRF token is invalid then you have to set another argument.

```
checkCSRF($_POST['csrf'], $forceStop = true);
```

If you don't want to stop loading page and just want to be notified about it then you can use it without *$forceStop* argument or set it to false. If CSRF token is invalid and *$forceStop* is set to false, function will return **false** otherwise if everything is ok it returns **true**.

Controller example:

```
<?php

require_once 'Controller.php';

class Main extends Controller{
	public function post() {
	}

	public function get() {
	}
}
```

Routing

Routing through pages is easy. All you have to do is to add one line in **index.php** file.

```
Router::home('Main', 'app/controllers/Main.php');
```

If you want to set default/home page use function home(<page name> , <controller>).

Note: <page name> is case insensitive, <controller> is case sensitive

```
Router::set(array('page_name', 'path_to_controller'));
```

At the end you must call function route(). This function navigates through pages. Navigation through pages is done using page GET parameter ($_GET['page']).

Example VIII: Routing

File: **index.php**

```
Router::home('Main', 'app/controllers/Main.php');

Router::set(array('Main', 'app/controllers/Main.php'));

Router::route();
```

Executing code after receiving GET request just write function get()

and post() for POST request. These functions must be written in your controller.

Functions

You can use functions implemented in framework by including Functions.php from core.

Example IX: Including functions

```
<?php

require_once 'core/Functions.php';

class Main extends Controller{

        public function __construct() {

                $allInputData = Functions::input();

                $postInputData = $postInputData["POST"];

                $gettInputData = $postInputData["GET"];

                $gettInputData = Functions::input("GET");


                $reloadTimer = 5;

                Functions::reload("index.php", $reloadTimer);

        }

}
```

Functions

input($type="ALL")

It returns dictionary with GET and POST values.
array("GET" => GET_VALUES, "POST" => POST_VALUES)

It is possible to manually select what values you want it to return.
input("GET") returns dictionary with GET values only.

redirect($toUrl)

When this function is called, website gets redirected to *$toURL* url

reload($toUrl, $time=0)

Reload function reloads website to *$toUrl* url after *$time* seconds.