MY PHP FRAMEWORK DOCUMENTATION

v0.8.3

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']}}
```

Including views

Example III: home view with more views included

Only 1 level of included files is supported. Included files can not include more files.

File: **home.view.php**

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

<h1>Test</h1>

<br>

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

[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 IV: 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>
<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

All models must extend class Model.

Example V: Basic model (in our case user)

File: **user.php**

```
<?php
require_once 'Model.php';

class user extends Model{
	var $table = "table_user";
	var $prefix = "user_";

	/* construction and helper functions */
	public function __construct() {

	}
}
```

In functions you can use some sort of ORM which helps you with some simple queries or you can write MySQL on your own. Note that framework only supports MySQLi.

Example VI: Simple select (one row)

File: **user.php**

```
<?php
require_once 'Model.php';

class user extends Model{
	var $table = "table_user";
	var $prefix = "user_";

	/* construction and helper functions */
	public function __construct() {

	}

	/* Function returns dictionary */
	/* columnNameFromDB => value */

	public function select ($whereKey, $whereValue){
		$data = $this->orm("select")->
			selectAll()->
			table($this->table)->
			where($whereKey, "=", $whereValue)->
			limit(1)->
			fetchRow();
		return $data;
	}
}
```

Example VII: Select (full table)

File: **user.php**

```
<?php
require_once 'Model.php';

class user extends Model{
	var $table = "table_user";
	var $prefix = "user_";

	/* construction and helper functions */
	public function __construct() {

	}

	/* Function returns array of dictionaries */
	/* [columnNameFromDB => value,...] */

	public function getAll(){
		$data = $this->orm("select")->
			selectAll()->
			table($this->table)->
			fetchArray();
		return $data;
	}
}
```

Example VIII: Simple select with where clauses

File: **user.php**

```
<?php
require_once 'Model.php';

class user extends Model{
	var $table = "table_user";
	var $prefix = "user_";

	/* construction and helper functions */
	public function __construct() {

	}

	/* Function returns dictionary */
	/* columnNameFromDB => value */

	public function select ($whereKey, $whereValue){
		$data = $this->orm("select")->
			selectAll()->
			table($this->table)->
			where($whereKey, "=", $whereValue)->
			limit(1)->
			fetchRow();
		return $data;
	}

	public function select And($wKey1, $wValue1, $wKey2, $wValue2){
		$data = $this->orm("select")->
			selectAll()->
			table($this->table)->
			whereAnd($wKey1, "=", $wValue1)->
			where($wKey2, "=", $wValue2)->
			limit(1)->
			fetchRow();
		return $data;
	}

	public function select Or($wKey1, $wValue1, $wKey2, $wValue2){
		$data = $this->orm("select")->
			selectAll()->
			table($this->table)->
			whereOr($wKey1, "=", $wValue1)->
			where($wKey2, "=", $wValue2)->
			limit(1)->
			fetchRow();
		return $data;
	}
}
```

Example IX: Your own MySQL

File: **user.php**

```
<?php
require_once 'Model.php';

class user extends Model{
	var $table = "table_user";
	var $prefix = "user_";

	/* construction and helper functions */
	public function __construct() {

	}

	/* Function returns dictionary */
	/* columnNameFromDB => value */

	public function removeFromQueue($hid){
		$sql = "DELETE FROM ".$this->table." WHERE hID = {$hid} LIMIT 1;";
		$this->execute($sql);
		return;
	}
}
```

Example X: Update

File: **user.php**

```
<?php
require_once 'Model.php';

class user extends Model{
	var $table = "table_user";
	var $prefix = "user_";

	/* construction and helper functions */
	public function __construct() {

	}

	/* Function returns dictionary */
	/* columnNameFromDB => value */

	public function debug(){
	$data = array(
	"user_name" => "Tralala",
	"user_email" => "my@new.email",
	"user_password" => "thisIsmynewhashedPassword"
	);

	$sql = $this->orm("update")->
		table("user")->
		update($data)->
		where("user_id", "=", "55")->
		commit();
	}
}
```

Example XI: Order

You can order by DESC or ASC. You can order by more columns if you order them in same order.

File: **user.php**

```
<?php
require_once 'Model.php';

class user extends Model{
	var $table = "table_user";
	var $prefix = "user_";

	/* construction and helper functions */
	public function __construct() {

	}

	/* Function returns array of dictionaries */
	/* [columnNameFromDB => value,...] */

	public function getAll(){
		$data = $this->orm("select")->
			selectAll()->
			table($this->table)->
			order("userID", "DESC")->
			fetchArray();
		return $data;
	}

	public function getAll2(){
		$data = $this->orm("select")->
			selectAll()->
			table($this->table)->
			order("userID, userEmail", "DESC")->
			fetchArray();
		return $data;
	}

}
```

Example XII: Debugging

File: **user.php**

```
<?php
require_once 'Model.php';

class user extends Model{
	var $table = "table_user";
	var $prefix = "user_";

	/* construction and helper functions */
	public function __construct() {

	}

	/* Function returns dictionary */
	/* columnNameFromDB => value */

	public function debug(){
	$data = array(
	"user_name" => "Tralala",
	"user_email" => "my@new.email",
	"user_password" => "thisIsmynewhashedPassword"
	);

	$sql = $this->orm("update")->
		table("user")->
		update($data)->
		where("user_id", "=", "55")->
		toSql();

		echo($sql);

		/* prints */
		/*
		UPDATE `user` SET `user_name` = 'Tralala', `user_email` = 'my@new.email',
		`user_password` = 'thisIsmynewhashedPassword' WHERE `user_id`='55' ;
		*/
	}
}
```

Controllers

All controllers extend main controller called Controller.

Your controller contains 2 functions, *post()* and *get()*. If user makes post request, *post()* function is called, if user makes get request then *get()* function is called.

You can show view with two different functions: *show()* or *render()*.

- show() function shows view without any additional functions.
- *render()* function shows view + uses cache and caches view after you change it. It also uses gzip compression.

Example XIII: Custom controller

```
<?php
require_once 'Controller.php';
class Home extends Controller{
        public function __construct(){
        }
        public function post() {
        }
        public function get() {
                $this->render("home.view.php");
        }
}
```

Example XIV: Showing view using show function

```
<?php
        $this->show("home.view.php");
```

Example XV: Showing view using render function

```
<?php

        $this->render("home.view.php");
```

If you want view to show variables, you can pass them to view, using additional argument.

Example XVI: Passing data to view

```
<?php

$reloadTimer = 5;

$moreInfo = "Hello";

$data = ["reloadTimer" => $reloadTimer,

            "extraInformation" => $moreInfo];

$this->render("home.view.php", $data);

// or $this->show("home.view.php", $data);
```

Routing

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

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

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

If you are going to have more pages then you have to use function *set(<dictionary>)*. Function accepts dictionary *pageName => pathToController*.

```
Router::set(array(
   'home' => 'app/controllers/Home.php',
   ));
```

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 XVII: Routing

File: **index.php**

```
/* routing */
Router::home('home', 'app/controllers/Home.php');
Router::set(array(
   'home' => 'app/controllers/Home.php',
   ));
Router::route();
```