# PHP & Zend Framework Tutorial

I did a lot of googling, installing, re-installing, … but finally got my first php project to work. I used WAMP for the apache install, the Zend Framework as an MVC, and **PostGreSQL** as a backend database. Below you can find a basic tutorial on how to get started. Included are following steps:

1. Install & configure WAMP
2. Configure PHP
3. Install & configure PostGres
4. Set up the Zend Framework
5. Start programming

I'm gonna try to keep everything as simple as possible. Downside is that you'll have to try & configure some things on your own. On the other hand it'll keep this tutorial from becoming too big (and you probably didn't even got to reading this sentence)… Oh yeah, I'm doing this on my Windows XP system, I don't know how differently all of this should be done on other systems.

## 1. Install & configure WAMP.

Why WAMP? Well it's the Windows version of the popular LAMP distribution and the installer configures a lot of things for you. I started with installing & configuring everything (Apache, MySQL/PostGres & PHP) separately which worked (after some time), but I want to spare you the frustrations I had to endure.

I'm not gonna tell you how to download it and click next during the installer. The most important thing is the 'httpd.conf'-file which configures the server. Following parameters are quite important (but should be set correctly after installing WAMP):

- the parameter "documentRoot" points to your 'www'-folder, where you can drop your PHP-files that can be viewed by browser when the server's URL is given
- the parameter "loadModule" points to the PHP configuration that should be used (PHP runtime environment normally is a separate install – Apache can't handle PHP on his own)

Important! I installed WAMP in my "program files" folder, so I recommend you do this too, otherwise you will have to adjust the php.ini file, which is discussed later.

So, normally no configuration is need, and after installing you should see an "It works!" when you browse to your http://localhost. If you are having problems, make sure all service are running.

### *2. Configure PHP*

The WAMP install will do a lot for you, but you will still have to set your system variables:

- Path: add your php folder (e.g: …*existingPathValues*…;C:\Program Files\wamp\php)
- PHPRC (=PHP Runtime Conf): e.g. "C:\Program Files\wamp\php"

You can google if you don't know how to set system variables, but remember to restart your PC after having done this.

A very important file is php.ini! It configures your PHP runtime environment. I'll list the most important parameters (to me):
(remember to remove the ";" in front of a parameter to uncomment it)

- display_errors: set it to "On"
- display_startup_errors: set it to "On"
- log_errors: set it to "On"
  (this is good for a dev environment but remember it affects performance)
- extension=… (to add dll's – we'll need some to connect to PostGres, so I'll come back to it later)
- extension_dir: to tell PHP where it can find dll's that are added through the 'extension' parameters
- include_path: used to include external (php-)classes that can be used in your project (we'll include the Zend framework classes, so I'll come back to this later)

Also something nice is PDO, when learning PHP you'll stumble on to this – give it some extra attention!

## 3. Install & configure PostGres

Why PostGres? Well, because we're going to need it for a client. We'll have an expert setting it up, but it always to know a little bit about what (or who) you are 'doing'. Normally I would have gone for **MySQL** (which is part of and configured by **WAMP**).

I suggest you follow the installer. Then open PgAdminIII (part of the installer) and double-click on your server. Enter your password (the one you entered during install) , right-click on 'databases' and select "New database…". For this tutorial enter "firstphpdb" for the name and click "OK".

Then goto 'Tools' in the menu and select 'Query tool'. Enter the following query:

```
CREATE TABLE users (
	id serial PRIMARY KEY,
	firstname varchar(50) UNIQUE NOT NULL,
	lastname varchar(50) UNIQUE NOT NULL,
	dateCreated timestamp DEFAULT current_timestamp
);
```

Then click the 'Run' button. This will create the "Users" table for you (under Schema's → Public → Tables). I guess you can find out on your how to enter some names in there.

This cheat sheet is quite useful.

## 4. Set up the Zend Framework

OK, why **Zend**? Well it's quite popular and IBM also likes it. Since I'm a big fan of **Lotus Notes** I'll follow IBM on this. I'll leave the discussion on which framework to use to the real PHP freaks.

First, we need to make sure we can use the **Zend** classes. We do this in our php.ini through the 'include_path' parameter:

```
; Windows: "\path1;\path2"
include_path = ".;C:\Program Files\Zend\ZendFramework-1.0.1\library"
```

Of course this setting depend on where you extracted your **Zend Framework**. Don't forget to **download it**.

Secondly, you'll need to understand how it works and should be used. I'll just explain the **MVC** setup, but **Zend** can do much more (I haven't tried the **other features**, I'll probably some more when I do). I'll try to explain the **MVC** on a step-by-step basis:

1. Remove everything from you document root (the www-folder of your **WAMP** installation). First we're going to create an .htaccess file in this folder. This file… Tip: You can't create a file that starts with a dot in Explorer, use the "Save as…" function in Notepad.
2. Now, we'll create an index.php file and drop it in our root folder (the www-folder of your **WAMP** installation). This file will launch when users browse to your server. You can find the file **here**. Everything is explained in the file itself:

```
<?php
try {
	// full error reporting (you can actually omit this because we also set this in php.ini)
	error_reporting(E_ALL);

	//includes: (remember the include_path in your php.ini? it's used here)
	include_once('Zend/Loader.php');

	//load the classes
	//the following translates into 'include(Zend/Controller/Front.php)'
	Zend_Loader::loadClass('Zend_Controller_Front');
	Zend_Loader::loadClass('Zend_Db');
	Zend_Loader::loadClass('Zend_Db_Table');

	//configure the database
	$options = array(
		'host' => 'localhost' ,
```

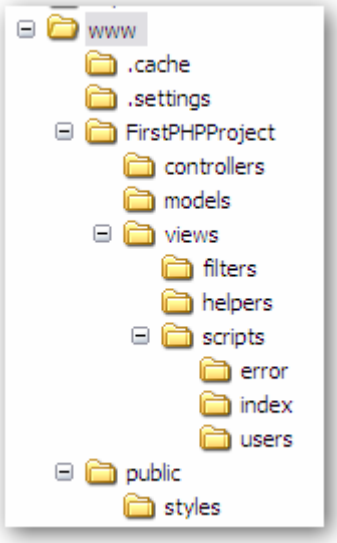
```
                'username' => 'postgres' ,
                'password' => 'lotusnotes' ,
                'dbname' => 'firstphpdb'
        );
        $db = Zend_Db::factory( 'PDO_PGSQL' , $options );
        //Using Zend_Db_Table is not mandatory,
        //But it will help write models because it assists in 'fast-writing' sql-queries
        //If we omit Zend_Db_Table we should use the PDO model
        Zend_Db_Table::setDefaultAdapter( $db );

        //configure the front controller (everything start here!)
        $controller = Zend_Controller_Front::getInstance();
        $controller->setControllerDirectory( './FirstPHPProject/controllers');

        //disable automatic view rendering (important, but look at Zend docs for the explanation)
        $controller->setParam('noViewRenderer' , true);

        //run the controller
        $controller->dispatch();
} catch (Exception $e) {
                echo $e->getMessage() . ' in ' . $e->getFile() . ' on line ' . $e->getLine() . "\n";
}
```

3. Now, because of the Zend_Controller in the index.php file, we need to create a directory structure for our project that the Zend_Controller can use. It is very important that you create it exactly like the Zend framework wants it:




**The Zend_Controller will translate the url's that are used. E.g. if you enter "localhost/index/index, Zend_Controller will look for the 'indexAction' in the 'IndexController' class in the controllers directory. If you enter "localhost/users/view", it will look for the 'viewAction' in the 'UsersController'.**

Now, you can ignore the ".cache" and ".settings" folder. Because I'm using Eclipse PDT as an editor, Eclipse created these folders.
You can't see the files, but .htaccess and index.php are at the same level as the folder "FirstPHPProject" and "public".

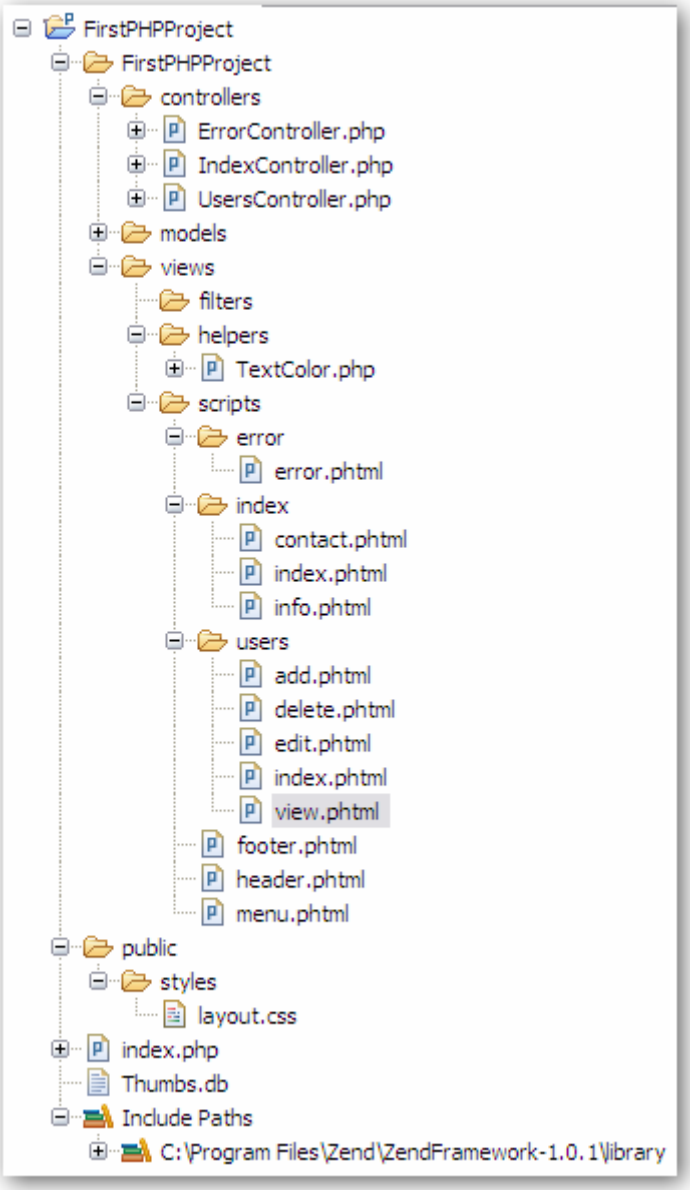
Now it's up to you to create the same folder structure.

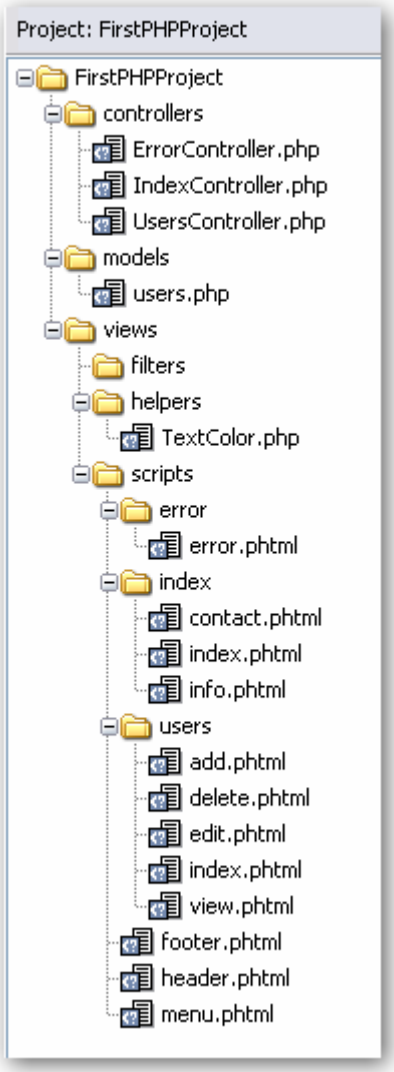
4. I hope you already know the basics of an MVC framework, because I'm not going to explain it. I'm just going to list the files you should create in each folder and explain some thing in the files themselves in the next chapter…

## 7. Start programming

Is started of using the Zend Editor, which is very nice, but **Sven Dens** recommended using the Eclipse PDT plugin which has recently got it's first 1-release. Since I've already used Eclipse and this is becoming our company 'standard' I went with it. So for no problems…

You can find my entire www-folder in the post, so I suggest you download it. This is how it should look like (left is in Eclipse, right in Zend Editor):




Most explanations are found in the files themselves. I'll discuss the basics here.

## *The Controllers*

As mentioned before, the Zend_Controller handles the requests and translates them into 'actions' from 'controller' classes in the "Controllers" folder.

Let's take the 'contactAction' 'IndexController' as an example which we can call through http://localhost/index/contact.

The 'init()' function as called every time an instance of the IndexController is created. In here, we initialize the view (which we will render at the end of the the 'contactAction'). We don't need to specify which view to use: Zend will look for a file called contact.phtml (=action name + .phtml) the in the "index" directory (=controller name) in the "scripts" folder. So make sure it exists. Note that everything is case sensitive!

Note: if you omit something, Zend will default to 'index'. E.g. if you enter http://localhost is will translate to http://localhost/index/index. If you enter http://localhost/users is will translate to http://localhost/users/index.

## *The Views*

As mentioned in the Controllers, Zend will look for a file with the same name as the action that's called (in the folder with the same name as the Controller that's called (in the "Script" folder)). So if you call http://localhost/users/view zend will execute the 'viewAction' in the 'UsersController' and when the 'viewAction' renders the view, zend will open the view.phtml file.

## *The Model*

A model represents the database model. In our example we created the model "users.php". This class inherits from the 'Zend_Db_Table' class. It has one constant '$_name' which represent the name of the table in the database. The initial connection to the database was made in index.php.

Now, we can use this model in our controllers. I suggest you look at the 'addAction' in the 'UsersController' to get started.