

Create your WEB Application with MVC paradigm and OPP – PAP support

Tutorial 5

Let's RECAP: [http://controller](#) -> methods -> parameters

At this point your structure tree, need to be like the example image:

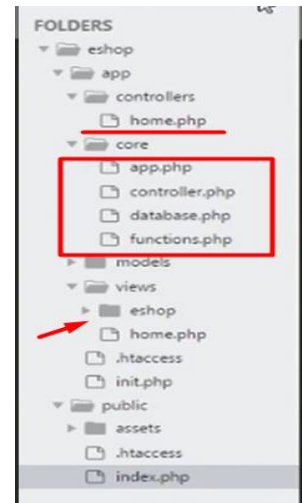
So, to recap:

Inside the folder controllers, you need to have a file named home.php, which will correspond to the home.php, or index.php.

Inside the core folder, you need to have 4 files. This 4 files will manage the backend logic of the app.

Inside the views folder, you need to have your template. You can either place it inside a additional folder (highly recommended) or directly inside inside the views folder.

Inside the public folder you need to have a assets folder, containing all of you lib's (css, js, img , ... so on).



If everthing is working well in choosen template, let's move on to the next step. If it is not, you need to solve the issues first.

CREATE DYNAMIC PATHS

Based on server parameters, we need to provide some structure to handle potential changes that may occur. Imagine that you are currently in development mode, but in the future, your work will move to production mode. Instead of configuring the connection line by line, it would be beneficial to provide additional configurations to simplify this process.

In your "index.php" file on public folder, you will now add the following code: `show($_SERVER)`. Then, reload your web application. Now, you have access to all configurations from your server at this moment.

```
[PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
[WINDIR] => C:\Windows
[SERVER_SIGNATURE] =>
Apache/2.4.29 (Win32) OpenSSL/1.1.0g PHP/7.2.2 Server at localhost Port 80

[SERVER_SOFTWARE] => Apache/2.4.29 (Win32) OpenSSL/1.1.0g PHP/7.2.2
[SERVER_NAME] => localhost
[SERVER_ADDR] => 127.0.0.1
[SERVER_PORT] => 80
[REMOTE_ADDR] => 127.0.0.1
[DOCUMENT_ROOT] => C:/xampp/htdocs
[REQUEST_SCHEME] => http
[CONTEXT_PREFIX] =>
[CONTEXT_DOCUMENT_ROOT] => C:/xampp/htdocs
[SERVER_ADMIN] => postmaster@localhost
[SCRIPT_FILENAME] => C:/xampp/htdocs/eshop/public/index.php
[REMOTE_PORT] => 50847
[GATEWAY_INTERFACE] => CGI/1.1
[SERVER_PROTOCOL] => HTTP/1.1
[REQUEST_METHOD] => GET
[QUERY_STRING] =>
[REQUEST_URI] => /eshop/public/
[SCRIPT_NAME] => /eshop/public/index.php
[PHP_SELF] => /eshop/public/index.php
[REQUEST_TIME_FLOAT] => 1611492263.907
[REQUEST_TIME] => 1611492263
)
```

Now, you have some important variables that you will use to construct a path to manipulate your application more easily. For example, consider that at this moment your `[SERVER_NAME]=>localhost`, but in the future, it might be something like www.potatos.com. If you don't handle this now, when you deploy your application, you may encounter a lot of problems.

To avoid this problems, let's construct a simple handler path. Inside your `init.php` put the code below:

```
eshop > public > index.php
1  <?php
2
3  session_start();
4
5  $path = $_SERVER['REQUEST_SCHEME'] . "://" . $_SERVER['SERVER_NAME'] . $_SERVER['PHP_SELF'];
6  $path = str_replace("index.php", "", $path);
7
8  //Define constant to ROOT folder
9  define('ROOT', $path);
10
11 //Define constant to ASSETS folder
12 define ('ASSETS', $path . "assets/");
13
14 //Define constant to TEMPLATE folder
15 define('THEME', 'eshop/');
16
17 include "../app/init.php";
18 //show($_SERVER);
19
20 $app = new App();
```

If you repair, at this moment, you have define two constants:

1. 'ROOT': This constant is defined based on a variable `$path`, which likely contains the root path of the project. It is used to represent the root path of the project across all files.
2. 'ASSETS': This constant is defined based on the ROOT constant concatenated with the directory "assets/". It is used to represent the path to the assets folder (such as CSS, JavaScript, images, etc.) within the project.

For example, if the `$path` is `/var/www/html/project/`, then ROOT will be defined as `/var/www/html/project/` and ASSETS will be defined as `/var/www/html/project/assets/`.

These constants are useful to avoid repeating paths throughout the code and make the code easier to maintain, especially when moving the project between different environments (e.g., from development to production).

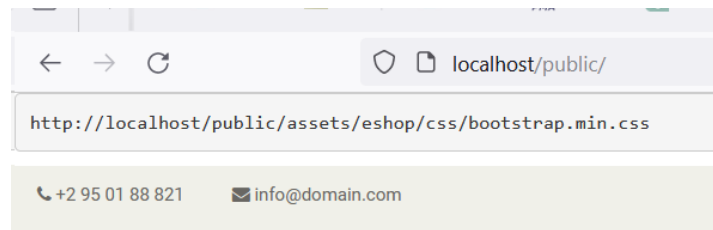
Now go to your header and update you path from your assets, like the image.

```
<link href="<?= ASSETS . THEME ?>css/bootstrap.min.css" rel="stylesheet">
<link href="<?= ASSETS . THEME ?>css/font-awesome.min.css" rel="stylesheet">
<link href="<?= ASSETS . THEME ?>css/prettyPhoto.css" rel="stylesheet">
<link href="<?= ASSETS . THEME ?>css/price-range.css" rel="stylesheet">
<link href="<?= ASSETS . THEME ?>css/animate.css" rel="stylesheet">
<link href="<?= ASSETS . THEME ?>css/main.css" rel="stylesheet">
<link href="<?= ASSETS . THEME ?>css/responsive.css" rel="stylesheet">

<?php show(ASSETS . THEME . "css/bootstrap.min.css")?>
```

If you repair, I have also another constant for THEME. It is because I have a multiples templates.

With the show information if you reload, you will see the debug message:



You can also do this: imagine in the future you want to use multiple templates, and this way is more versatile to handle the location of the folders.

Ok, at this moment let's recap what you've done:

- Added a template for your PAP presentation.
- Updated your code to have more flexibility when you need to introduce new features.
- ADD a path to handle the location folders.
- CREATE a constant variables to handler the files / folders locations

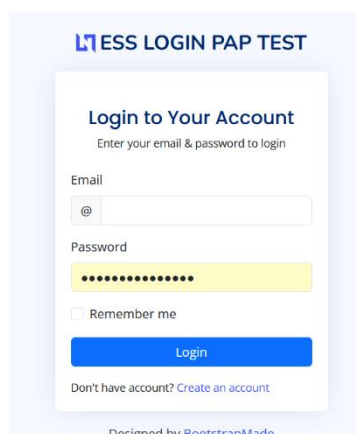
Next step:

ASSIGNMENT: prepare two controllers inside the folder controllers, and two views inside the folder views, one for the signup and another to the login application.

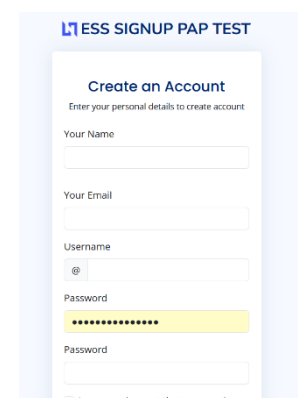
Test the views. If you have any issue, please look to the path of the files and the libraries location.

Example template:

<http://localhost/public/login>



<http://localhost/public/register>



END Tutorial 5

Reference: **PHP Ecommerce website development | Create the app class | MVC OOP - Quick programming**

Template: **NICEADMIN, design by BOOTSTRAP**