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

Tutorial 4

Let's RECAP: http://controller -> methods -> parameters

In the last tutorial we see how to adjust our url to access our methods within the appropriate controller.

Now will start to create our access to the views.

If you remember the views is provided by the controller, so when the user click to get something, the view send the response to the controller and the controller leads with this responsibility.

However the view is provided also by the controller. It's a little tricky :D

Let's create our method inside the file controller to lead with your views.

Inside your core folder you have a file with name controller. Inside the file put this method code:

```php
<?php

Class Controller
{

    public function view($path)
    {

        if(file_exists("../app/views/" . $path . ".php"))
        {
            include "../app/views/" . $path . ".php";
        }
    }
}
```

ASSIGMENT: put the comment that respond to the question above the function from the picture: **What is this view function?**

Ok, in this code we have a entry variable $path, will receive the path from where his the view. The if validation, check's if the file exist's, load the view.

Now we must to do some OPP operation and extend the home controller to this controller, because the index funtion need to access to the view function from controller to have ability the display the views to the user.

```php
<?php


class Home extends Controller
{
    //Public default metodo index, mesmo
    public function index()
    {
```

Now, create one file inside views folder and name it home.php.

Inside the file put a example like the picture:

```html
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
This is the home page
</body>
</html>
```

Then, go to your controller home.php file and add the view function inside the index:

```php
<?php

Class Home extends Controller
{

    public function index()
    {

        $this->view("home");
    }


}
```

Now, if you reload you browser, you get the message like you write in your view.

At this moment, you have a routing system full working, and when the user get the view to the controller, the response is statuscode 200 (OK).

See it in your network separator inside develop tools from your web browser.

We also can add data to page with the scope from the method.

Change the scope, like the image please. Add an empty array like the example.

Need to be a empty array to avoid error's in the future.

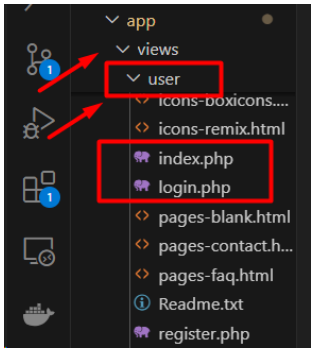
```php
public function view($path, $data = [])
{

    if (file_exists("../app/views/" . THEME . $p
```

ASSIGMENT: Inside the controller file, in view method create the follow logic:

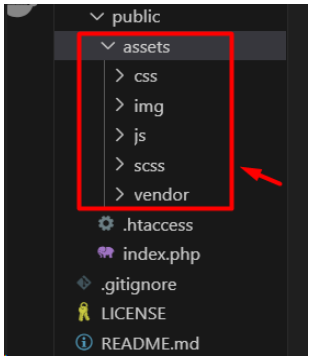
- if the file view doesn't exist, load a error 404 page.

Next step, you will need to obtain a template for your PAP presentation and download it. Please ensure that you choose a template that is freely available, and verify that its usage is allowed under the author's rights.



In your document, it's crucial to include a reference to the author at the end. This ensures proper acknowledgment of the author's work.

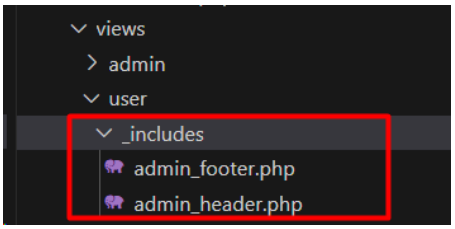
Inside the views folder you will put your html files, like the example. In my example I create a user folder a put all html files inside.



For the assets, you will create folders within the public folder and place all the template's folders inside them. It is very important to pay attention to the paths of the folders, as you may encounter issues when accessing the libraries that make your template work properly.

Next, you will organize your code into separate components. Imagine that you consistently use the same header and footer! It becomes a bit ambiguous to repeatedly use the same code over and over again... Let's utilize the same reference for all our views.

Now you will create a file header.php and a footer.php inside your views, like the image.



In my case I use a separated folder to divide the files, but you can put the new created files on the same place that the others.

Now in your index.html file name it to index.php and get the specific parts (header / footer) to the new files that you have created.

Inside the index.php, only stays the body, from the template.

In the first line you will call the header file, like the example below.

```
eshop > app > views > user > index.php
1  <?php $this->view( "_includes/user_header", $data); ?>
2
3
4  <!-- ======= Sidebar ======= -->
5    <aside id="sidebar" class="sidebar">
```

The user_header is my name file, and the $data variable will be the connection to send data to this part, like name or other stuff.

You will do the same to the footer, like the next example:

```
884        </div>
885      </section>
886
887    </main><!-- End #main -->
888    <?php $this->view( "_includes/user_footer", $data); ?>
```

Now, refresh your browser and check if is everything well with the template.

If you got a different view, from the original html, you need to check the path's for the libs.

END Tutorial 4

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