# MVC Introduction – Exercise

[Model-View-Controller](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) architectural design pattern is one of the most used patterns in the architectural toolset chosen for web applications in the past 5 years.

## Part I: Routing

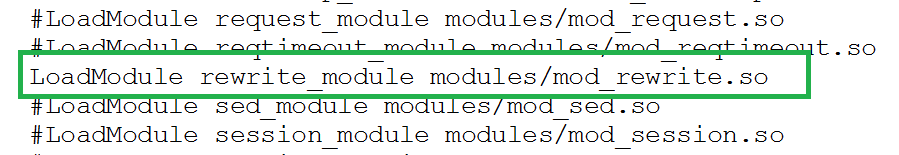
### URL Redirect

Executable applications have **one entry point**, which is the standard event loop executed. An application container manages the lifecycle of the application and **sends user input to the application event loop**.

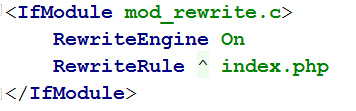
This is **not the case when using standard web servers** which execute files instead of managing applications. We need to **simulate** it. One way to do it is to **redirect** all requests to one file, which will **take the role of an entry point**.

In order to make it we need to use web server **runtime** configurations. In Apache Webserver this could be done by .[htaccess](http://httpd.apache.org/docs/current/howto/htaccess.html).

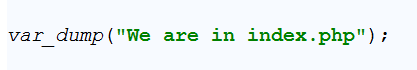
In the .htaccess files we will use the [mod\_rewrite](http://httpd.apache.org/docs/current/mod/mod_rewrite.html) module. We need to make sure that in the **%apache\_dir%/conf/httpd.conf** (e.g. c:/xampp/apache/conf/httpd.conf) file we have the **mod\_rewrite** line uncommented:



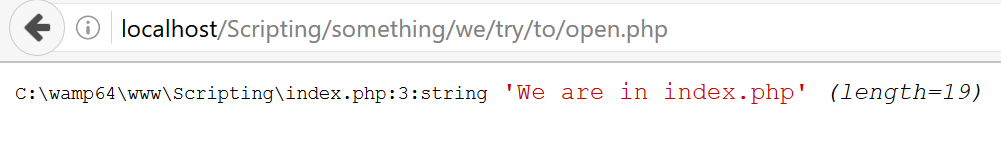
Now we can create our .htaccess file using the rewrite module and set a conditions – everything sent to web server to be redirected to one file – index.php.



We can test that whatever we write in the URI will redirect us to index.php. Let’s assume our app resides in folder Scripting . Create index.php and add the following line:



And the result, opening **http://localhost/Scripting/something/we/try/to/open.php**



But what if we need a JavaScript file. Will we be redirected to index.php again? The short answer is Yes. Let’s test it. Create a folder called js and add index.js file with some javascript there and try to invoke it from index.php

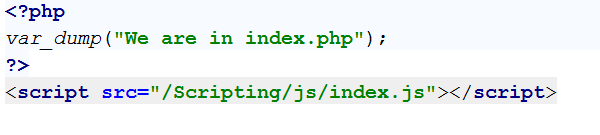


-

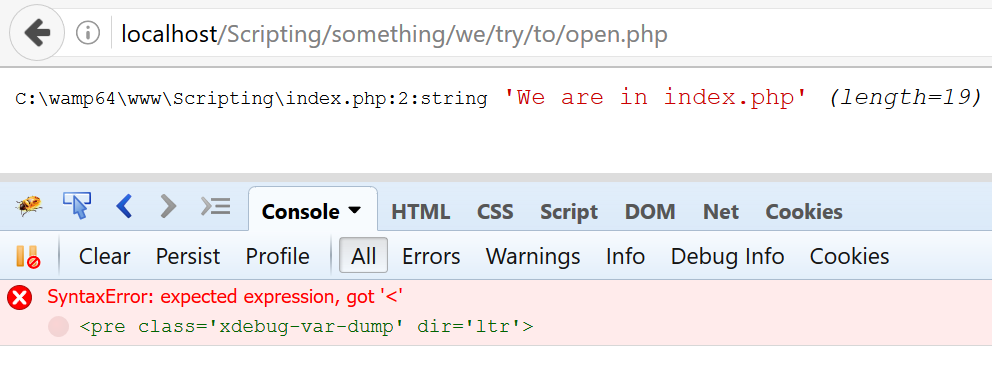


-

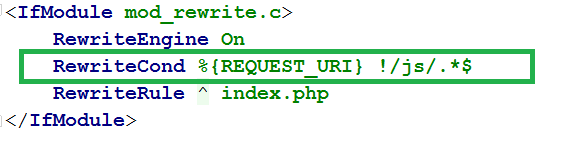
index.php:



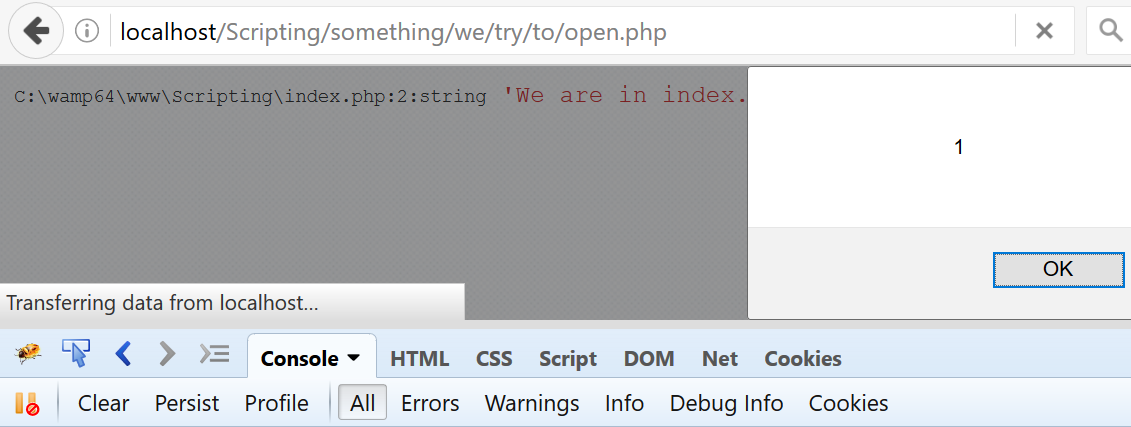
Result:



That’s right! We haven’t exclude from redirect our content files such as css and js. We can exclude the whole **js folder** in .htaccess:



And try index.php again ☺



We are done redirecting ☺

**Additional Resources**

1. Apache Web Server - mod\_rewrite see: <http://httpd.apache.org/docs/current/mod/mod_rewrite.html>
2. Apache Web server .htaccess files: https://httpd.apache.org/docs/current/howto/htaccess.html
3. NGINX web server - ngx\_http\_rewrite\_module: <http://nginx.org/en/docs/http/ngx_http_rewrite_module.html>
4. Microsoft IIS server rewrite rules: <https://docs.microsoft.com/en-us/iis/extensions/url-rewrite-module/creating-rewrite-rules-for-the-url-rewrite-module>

#### Extracting Significant URI Parts

In the standard we will set, there will be two general parts in the URI:

* Controller and Action
* Parameters

This means that <http://localhost/Scripting/users/hello/john/smith> means that:

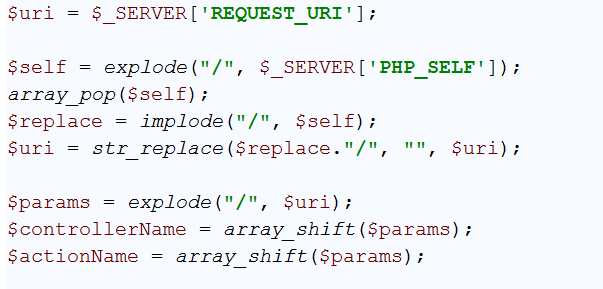
* **Users** is a **Controller**
* **Hello** is an **Action**
* **John** and **Smith** are **Parameters**

The Controller most probably is a class e.g. **UsersController**, the Action is a **method** in that class. The Parameters are arguments of the **class method** e.g. two string arguments $firstName and $lastName

We can extract these parts by introspecting the [$\_SERVER](http://php.net/manual/en/reserved.variables.server.php) superglobal.

* PHP\_SELF key from the $\_SERVER superglobal contains the current script name and its folders before
* REQUEST\_URI key from the $\_SERVER superglobal contains what the user has written e.g: **/Scripting/users/hello/john/smith**
* In most of the scenarios **/Scripting/** will not exist, but if it does, we need to take care to remove it, as it is irrelevant

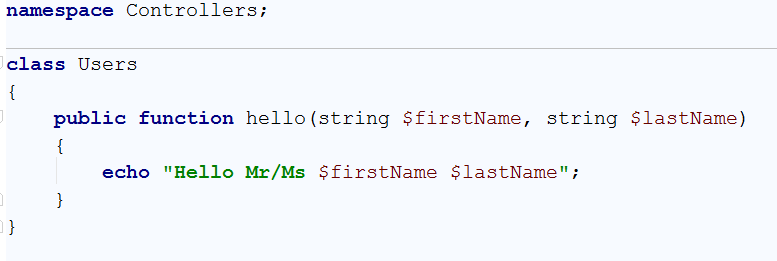
So we need to remove the intersection from PHP\_SELF and REQUEST\_URI in order to extract **/users/hello/john/smith**



### Dispatching

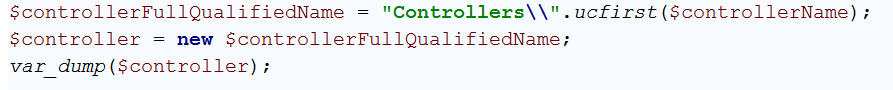
So far we have our significant parts. We can map to **classes**, **method calls** and **argument passing**. Luckily, PHP as a **scripting language** relies heavily on [reflection](https://en.wikipedia.org/wiki/Reflection_(computer_programming)).

Let’s create a Controllers folder and put there a Users class. Then define a hello() method accepting two arguments: $firstName and $lastName:

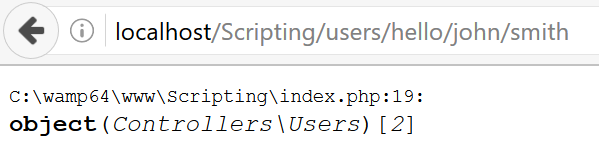


Now we can instantiate our Users controller by using native’s **PHP reflection** – instantiating classes from strings. Don’t forget to use the autoloader from one of our previous exercises.

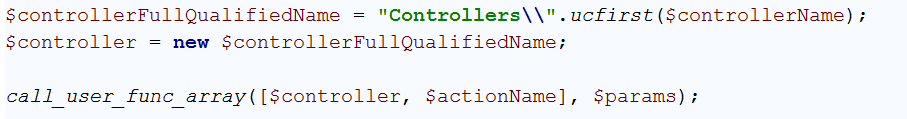
index.php:



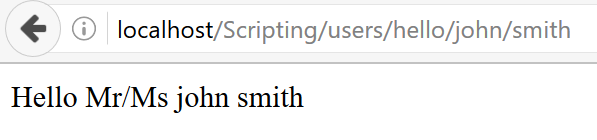
Result:



Now we can invoke the action (hello) by providing a set of arguments, using the [call\_user\_func\_array()](http://php.net/manual/en/function.call-user-func-array.php) built-in function.



Result:



This text printed means our echo was executed, so we have successfully dispatched the user request to the relevant methods ☺

### Error Page

For a usual website if the route in the URL is not right you need to display a **HTTP 404 code error page**. For SEO purposes it is bad to reload to the home page.

But if you are in a mode in which the page is not indexed (like an administration area) you may want to reload to an **index page** **(main controller and main action)**.

Program a mechanism to display the error page. Later reprogram your code to use all the **view functionality.**

## Part II: Advanced

Take a look at the next Part – [MVC in Depth](http://svn.softuni.org/admin/svn/php-web-dev/Oct-2016/06.%20PHP-Web-MVC-in-Depth/06.%20PHP-Web-Dev-MVC-in-Depth-Exercises.docx)

## Part III: Bonus\*

Play more with **mod\_rewrite**. Imagine that in a separate directory you want to have some open source software installed (like a free blogging system). How would you **change the .htaccess file** so that it might not reload to index.php for that particular folder?