IPL
escola superior
de tecnologia e gestão
instituto politécnico
de leiria

Routing

PHP – MVC - Routing

Vitor Carreira & Marco Monteiro

Contributors

- Author(s):
  - Vitor Carreira (vitor.carreira@ipleiria.pt)
  - Marco Monteiro (marco.monteiro@ipleiria.pt)

Current Development State

- Solution provided as a demo of this presentation:
- "**09/09.start**" folder
- MVC pattern, as implemented on "06. PHP – MVC".
  - View, add, edit and delete users (using a simplified "users" table)

MVC Architecture / Structure

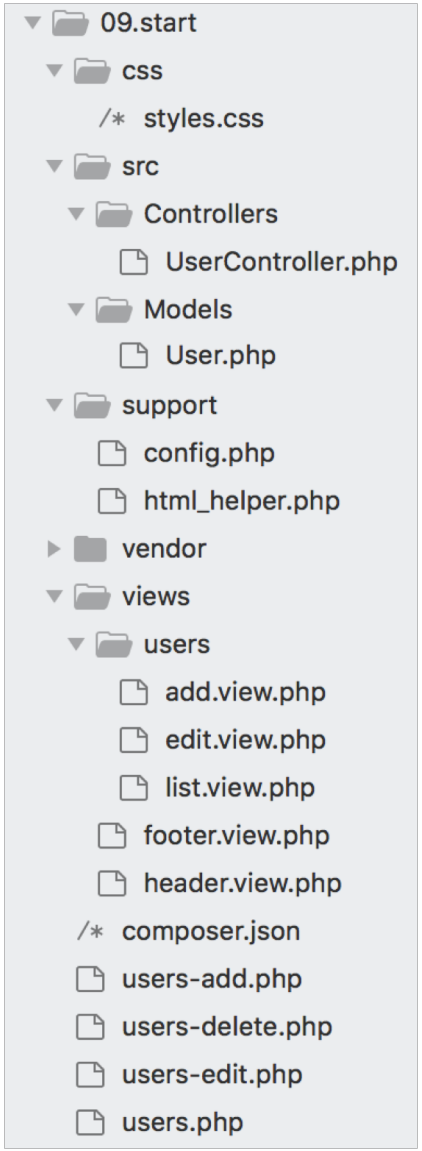
- 1 Model :

  `src/Models/User.php`

- 1 Controller:

  `src/Controllers/UserController.php`

- 3 Content views: `views/users/list.view.php`
  `views/users/add.view.php`
  `views/users/edit.view.php`
  - + 2 template views: `views/footer.view.php`
    `views/header.view.php`
- 4 "*Entry point*": `users.php`
  `users-add.php`
  `users-delete.php`
  `users-edit.php`
- 2 support file: `support/config.php`
  `support/html_helper.php`



“Entry Points”

Users.php

```
<?php
require_once "vendor/autoload.php";

use Controllers\UserController;

$controller = new UserController;
$controller->getUsers();
```

Users-add.php

```
<?php
require_once "vendor/autoload.php";

use Controllers\UserController;

$controller = new UserController;
$controller->addUsers();
```

Users-edit.php

```
<?php
require_once "vendor/autoload.php";

use Controllers\UserController;

$controller = new UserController;
$controller->editUsers();
```

Users-delete.php

```
<?php
require_once "vendor/autoload.php";

use Controllers\UserController;

$controller = new UserController;
$controller->deleteUsers();
```

"Entry Points" - DRY

Users.php

```
<?php
require_once "vendor/autoload.php";

use Controllers\UserController;

$controller = new UserController;
$controller->getUsers();
```

Users-add.php

```
<?php
require_once "vendor/autoload.php";

use Controllers\UserController;

$controller = new UserController;
$controller->addUsers();
```

Users-edit.php

```
<?php
require_once "vendor/autoload.php";

use Controllers\UserController;

$controller = new UserController;
$controller->editUsers();
```

Users-delete.php

```
<?php
require_once "vendor/autoload.php";

use Controllers\UserController;

$controller = new UserController;
$controller->deleteUsers();
```

DRY – Don't Repeat Yourself

Common code

Specific to each action

Routing

- Routing is the component responsible for mapping an **URI** to an **action**
  - Example:

    `http://www.mydomain.com/`**`user/add`**
  - The routing component will match "**user**" to a controller called "UserController" and "**add**" to the method add() of that controller
  - Mapping can be done by **<u>convention</u>** (as previous example), by **<u>configuration</u>** or by a mix of both
    - Convention – URI pattern matches class and method by name conventions
    - Configuration – Mapping URI to classes and methods is done by configuration on code, configuration files, database, etc.

Routing

- **<u>Conventions for our example</u>**:
  - URI will always have the form:

`domain?<controller_name>/<action>&parameters`

  - The <controller_name> will be transformed to:
    - <Controller_name>Controller class
  - The <action> will be transformed to a:
    - **get**<Action>() method call (for GET requests)
    - **post**<Action>() method call (for POST requests)
    - <action> defaults to index
  - Example *(for a get method)*:
    - `domain?user/edit&id=3` -> Controller = `UserController`
      Method = `getEdit()`
      Parameters= `id = 3`

Routing – Route class

- Lets wrap everything into a class called Route :)

```
class Route
{
    // URL Format:index.php?controller/action&p1=v1&p2=v2&…
    private function __construct()
    {
        $this->root = dirname($_SERVER['SCRIPT_NAME']);
        $this->httpQueryString = $_SERVER['QUERY_STRING'];
        $queryStringParts = explode('&', $this->httpQueryString);
        $pathInfo = explode('/', $queryStringParts[0]);
        $this->httpQueryString = $queryStringParts[1] ?? "";
        if (count($pathInfo) > 0) {
            $this->httpMethod = strtolower($_SERVER['REQUEST_METHOD']);
            $this->controller = $pathInfo[0];
            $this->action = 'index';
            if (count($pathInfo) > 1) {
                $this->action = $pathInfo[1];
            }
        }
    }
```

A generic View


```
class View
{
    private $name;
    private $vars;

    private function __construct($name, $vars)
    {
        $this->name = $name;
        $this->vars = $vars;
    }

    public function render()
    {
        foreach ($this->vars as $name => $value) {
            $$name = $value;
        }
        include('views/header.view.php');
        include('views/'.str_replace('.', '/', $this->name).'.view.php');
        include('views/footer.view.php');
    }

    public static function create($name, $vars)
    {
        return new View($name, $vars);
    }
}
```

Adapting the controller (index)

From this:

```
public function getUsers()
{
    $users = User::all();
    $pagetitle = "List of Users";
    render_view('users.list', compact('users', 'pagetitle'));
}
```

To this:

```
public function getIndex()
{
    $users = User::all();
    $pagetitle = "List of Users";
    return View::create('users.list', compact('users', 'pagetitle'));
}
```

- Method name has changed to (getIndex) to follow conventions
- Uses the new View class
  - Instead of rendering the view within the controller, the View is passed to the outside of the controller – it will be rendered outside

Adapting the controller (add)

From this:

```
public function addUser()
{
    $pagetitle = "Add user";
    $errors = [];
    $user = new User;

    // The first time (GET), just show the page
    if (empty($_POST)) {
        return render_view('users.add', compact('pagetitle', 'user', 'errors'));
    }
    if (isset($_POST['cancel'])) {
        return $this->home();
    }
    $this->loadUserFromPost($user);
    $errors = $this->validateUser($user);
    if (count($errors) > 0) {
        return render_view('users.add', compact('pagetitle', 'user', 'errors'));
    }
    User::add($user);
    return $this->home();
}
```

Adapting the controller (add)

To this:

```
public function getAdd()
{
    $pagetitle = "Add user";
    $errors = [];
    $user = new User;
    return View::create('users.add', compact('pagetitle', 'user', 'errors'));
}

public function postAdd()
{
    $pagetitle = "Add user";
    $errors = [];
    $user = new User;
    if (isset($_POST['cancel'])) {
        return $this->home();
    }
    $this->loadUserFromPost($user);
    $errors = $this->validateUser($user);
    if (count($errors) > 0) {
        return View::create('users.add', compact('pagetitle', 'user', 'errors'));
    }
    User::add($user);
    return $this->home();
}
```

Adapting the controller (add)

- One controller method that handled both the get and post requests, is replaced by 2 controller methods:
- **getAdd()** – Responsible for rendering the web page with the form where user fills the new record data
- **postAdd()** – Responsible for saving (inserting) the data submitted by the user (submit -> post request)
- The method names follow the convention (get<Action> and post<Action>), for the add action

Adapting the controller

- editUser() is also replaced by two methods: getEdit() and postEdit()
- deleteUser() is replaced by postDelete()
- Also, home() method has to be adapted, because the URLs used in the application have changed

From this:

```
private function home()
{
    header('Location: users.php');
    exit(0);
}
```

To this:

```
private function home()
{
    header('Location: index.php?user');
    exit(0);
}
```

Adapting the Views

- All URLs within the views have to be adapted for the new URL pattern.
- Example on “users/list.view.php”

From this:

```
…<a href="users-edit.php?id=<?= htmlspecialchars($user->id) ?>">
      Edit</a>…
<form action="users-delete.php" method="post">…
```

To this:

```
…<a href="index.php?user/edit&id=<?= htmlspecialchars($user->id) ?>">
Edit</a>…
<form action="index.php?user/delete" method="post">…
```

Executing a route

- Add code to the Route class to automatically execute the proper controller method from the URL pattern (controller and action)

```
class Route  {  . . .
    public function execute()
    {
        if (!$this->controller) {
            die("Should have a default route");
        }
        $className = 'Controllers\\'.ucfirst($this->controller).'Controller';
        $methodName = $this->httpMethod.ucfirst($this->action);

        $instance = new $className;
        // IMPROVE: use remaining pathinfo to pass parameters
        return $instance->$methodName();
    }

    private static $singleton;
    public static function defaultRoute()
    {
        if (is_null(self::$singleton)) {
            self::$singleton = new Route;
        }
        return self::$singleton;
    }
```

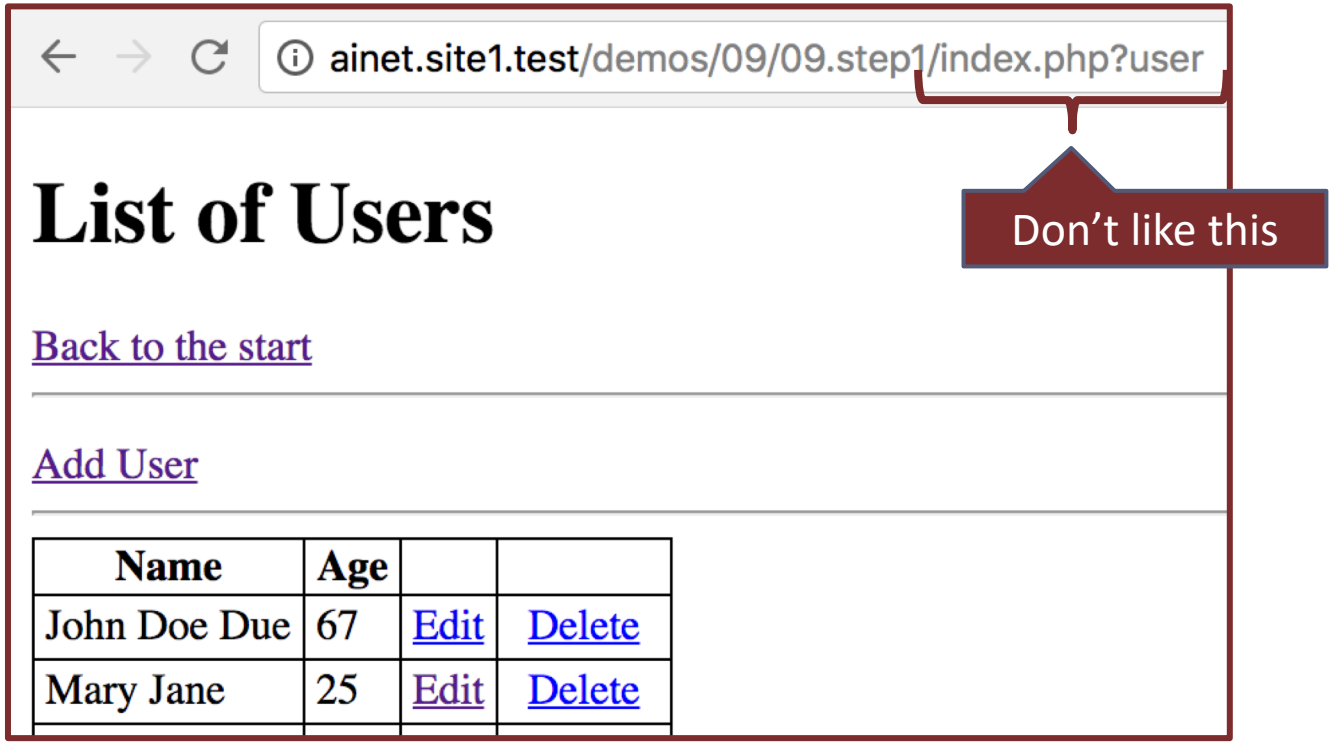
Single "*Entry Point*"

- users.php, users-add.php, users-edit.php and users-delete.php can be replaced with a single **index.php file**:

```
<?php

require 'vendor/autoload.php';

use Api\Route;

$route = Route::defaultRoute();
$route->execute()->render();
```

Let’s try it

- No need to create extra ”entry points” – just controllers, views and models.
- Follow the convention!
- What could be improved? URLs could be **”cleaner”**!



Rewrite engine to the rescue

- Both Apache and Nginx Web Servers have **rewrite engines** that allows incoming HTTP requests URLs to be transformed into other URLs
- With **Nginx**:
  - On the site configuration file add a section that is similar to:

```
location /subdomain/root/ {
   try_files $uri $uri/ /subdomain/root/index.php?$query_string;
}
```

  - /subdomain/root/ -> the domain or subdomain of the site we wish to transform
  - Configuration file location depends of the server and site, but usually (on a linux server) is located on:

`/etc/nginx/sites-available/site_name.conf`

Rewrite engine to the rescue

- With **Apache**:
  - On the site root, add a ".htaccess" file that is similar to:

```
DirectoryIndex index.php
<IfModule mod_rewrite.c>
     <IfModule mod_negotiation.c>
         Options -MultiViews -Indexes
     </IfModule>

     RewriteEngine On
     RewriteBase /subdomain/root/

     # Redirect Trailing Slashes...
     RewriteRule ^(.*)/$ $1 [L,R=301]

     # Handle Front Controller...
     RewriteCond %{REQUEST_FILENAME} !-d
     RewriteCond %{REQUEST_FILENAME} !-f
     RewriteRule ^ index.php [L]
 </IfModule>
```

Rewrite engine to the rescue

- With **Apache**:
  - On the site configuration file ensure that AllowOverride is activated

```
<Directory "/subdomain/root/">
   AllowOverride All
   Require all granted
</Directory>
```

  - Configuration file location depends of the server and site, but usually (on a linux server) is located on:

`/etc/apache2/sites-available/site_name.conf`

Adapting the Route class

- Route class has to be adapted:

```
class Route { . . .
private function __construct()
{
    $this->root = ltrim(dirname($_SERVER['SCRIPT_NAME']),'/');
    $requestUri = str_replace('/index.php', '', $_SERVER['REQUEST_URI']);
    $this->httpQueryString = $_SERVER['QUERY_STRING'];
    if ($this->httpQueryString) {
        $requestUri = str_replace('?'.$this->httpQueryString, '',
                                  $requestUri);
    }
    $pathInfo = explode('/', substr($requestUri, strlen($this->root) + 1));
    if (count($pathInfo) > 0) {
        $this->httpMethod = strtolower($_SERVER['REQUEST_METHOD']);
        $this->controller = $pathInfo[0];
        $this->action = 'index';
        if (count($pathInfo) > 1) {
            $this->action = $pathInfo[1];
        }
    }
}
```

Let’s try it again!

ainet.site1.test/demos/09/09.complete/user

List of Users

Back to the start

Add User

Name	Age	
John Doe Due	67	Edit
Mary Jane	25	Edit

index.php is hidden.
It does not appear on URL, but it is executed

Much Better ☺

ainet.site1.test/demos/09/09.complete/user/edit?id=3

Edit user

Back to the start

Name Edgar Smith
Age 23
Save Cancel

Generating URLs

- How to address old hard-coded routes?
  - One solution is to add helper methods to generate URL routes to controllers or actions
  - Add the following to Route class:

```
class Route { . . .
public function route($path)
{
    return $this->root . $path;
}
public function action($controller, $action = null, $params = null)
{
    $url = $this->root.'/'.$controller;
    if ($action) {
        $url .= '/'.$action;
    }
    if ($params) {
        $url .= '?'. http_build_query($params);
    }
    return $url;
}
```

Generating URLs

- Add the following Helper functions
  - On “support/html_helper” file

```
function route($path)
{
    $defaultRoute = Api\Route::defaultRoute();
    return $defaultRoute->route($path);
}

function action($controller, $action = null, $params = null)
{
    $defaultRoute = Api\Route::defaultRoute();
    return $defaultRoute->action($controller, $action, $params);
}
```

Generating URLs – use HTML helpers

- On the controller:

```
private function home()
{
    header('Location: ' . action('user'));
    exit(0);
}
```

- On the Views:

```
…<a href="<?= action('user', 'edit') ?>?id=<?=
                            htmlspecialchars($user->id) ?>">Edit</a>…
<form action="<?= action('user', 'delete') ?>" method="post">…
```

```
<link rel="stylesheet" href="<?= route('/css/styles.css') ?>">
```

The End

- Analyze all demos
  - *First, execute the database script to create the database*
- Available Demos:
- **09.start** – MVC without routing (starting point)
- **09.step1** – MVC with routing - using query string
- **09.complete** – MVC with routing and "*pretty*" URLs
  - This demo (09.complete) requires the configuration of Apache or Nginx web Server