ПОБУДОВА PHP MVC ФРЕЙМВОРКА

ЛЕКЦІЇ З КУРСУ «ПРОГРАМУВАННЯ INTERNET»

Інженерний інститут Запорізького Національного Університету,
кафедра програмного забезпечення автоматизованих систем,
доцент Попівщий В.І.

2019 р.

ПЛАН

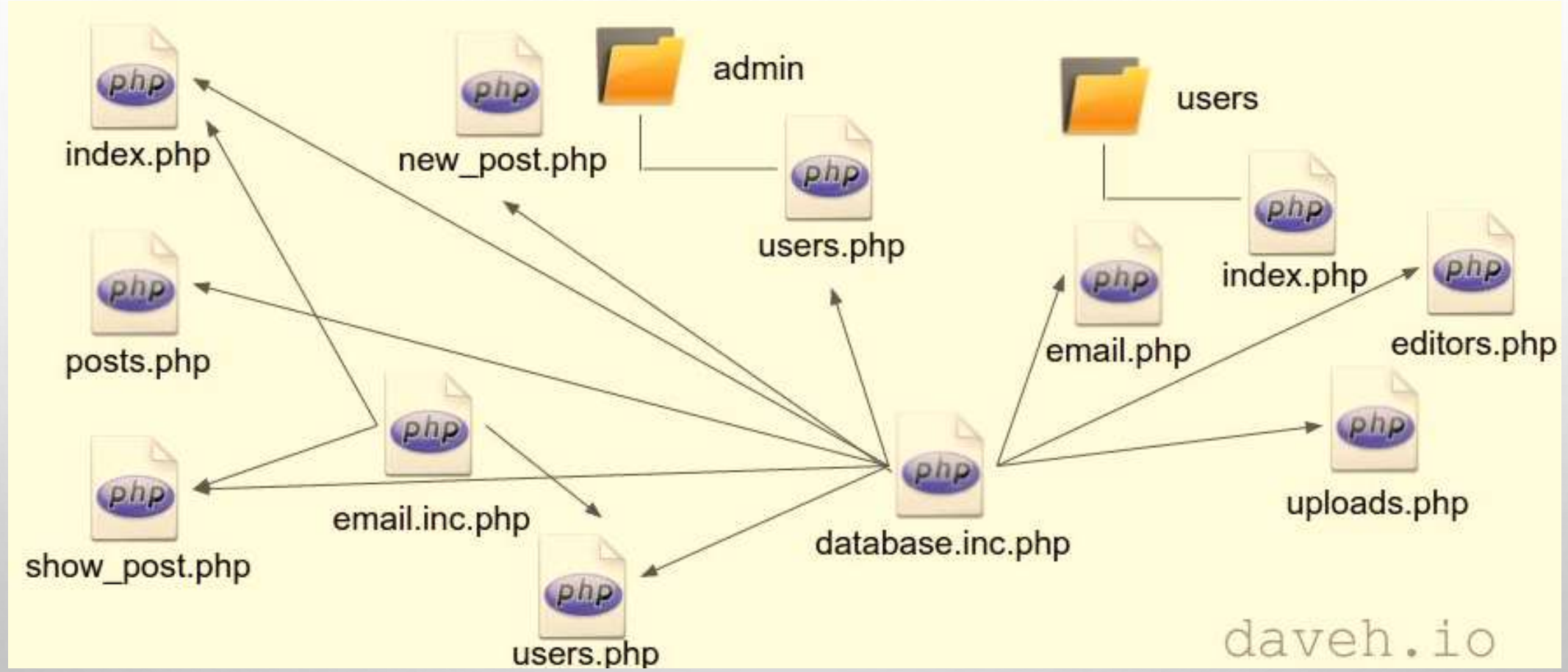
- ВСТУП ДО MVC І НАЛАШТУВАННЯ ПРОЕКТУ
- МАРШРУТИЗАЦІЯ І FRONT CONTROLLER
- ПРОДВИНУТИЙ РОУТИНГ
- КОНТРОЛЕРИ І ДІЇ (ACTIONS)
- ВИГЛЯДИ (VIEWS)
- УПРАВЛІННЯ ПАКЕТАМИ ЗА ДОПОМОГОЮ COMPOSER
- МОДЕЛІ (MODELS)
- КОНФІГУРУВАННЯ ДОДАТКУ І ОБРОБКА ПОМИЛОК

ПРОБЛЕМИ СТВОРЕННЯ WEB ДОДАТКІВ

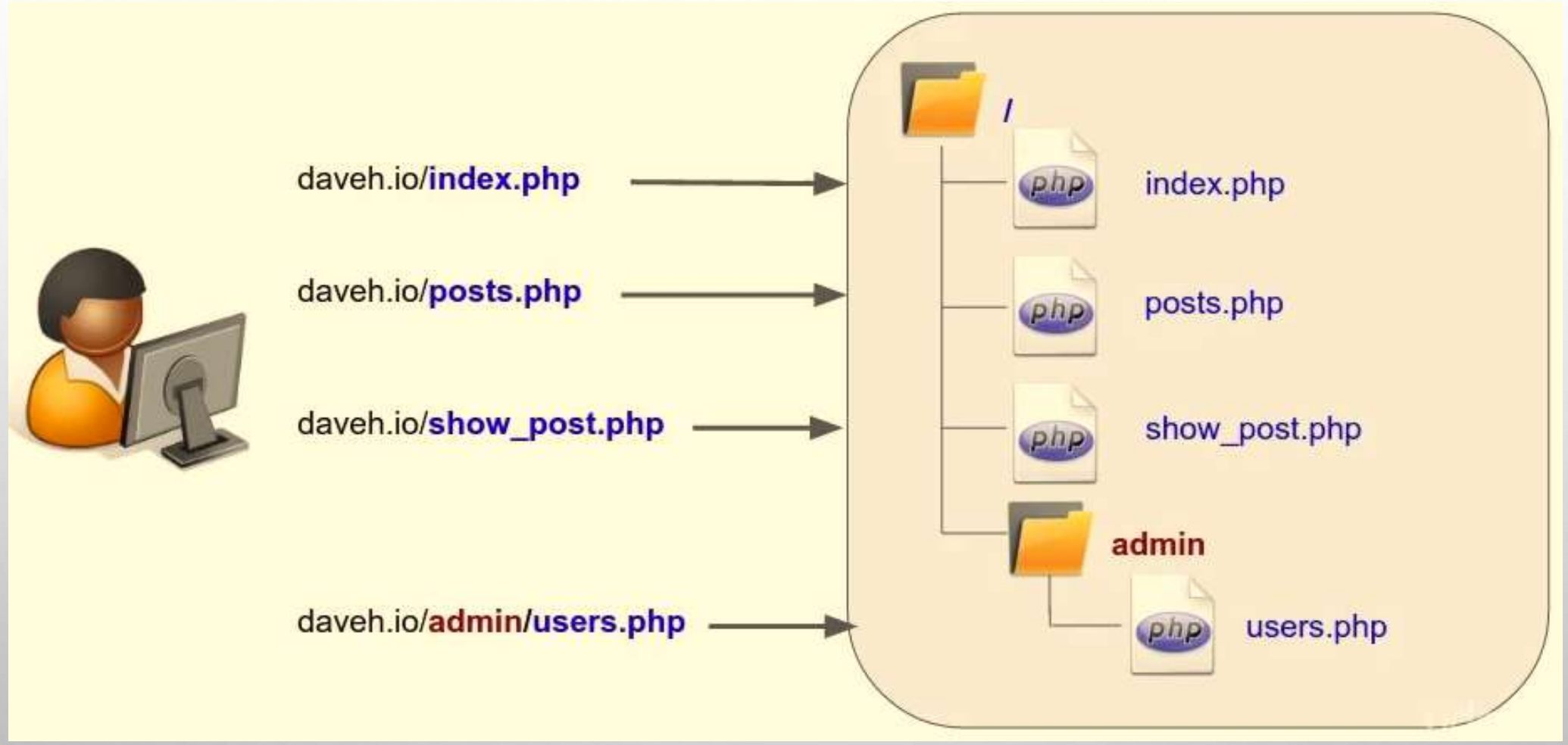
- НЕСТРУКТУРОВАНИЙ КОД.
- ЛОГІКА ЗАСТОСУНКУ ЗМІШАНА З РІВНЕМ ПРЕДСТАВЛЕННЯ

```
<?php

$db = new PDO("mysql:host=localhost;dbname=daveh_db", "daveh", "secret");
$posts = $db->query("SELECT * FROM posts")->fetchAll();

if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $stmt = $db->prepare('INSERT INTO posts (title, body) VALUES (....
}
?>
<html>
<body>
    <form method="post">
        ...
    </form>
    <?php foreach ($posts as $post): ?>
        <h2><?php echo $post["title"]; ?></h2>
    <?php endforeach; ?>
</body>
</html>
```

ФАЙЛИ ТЕЖ НЕСТРУКТУРОВАНІ



URL = file location



ПРОБЛЕМИ

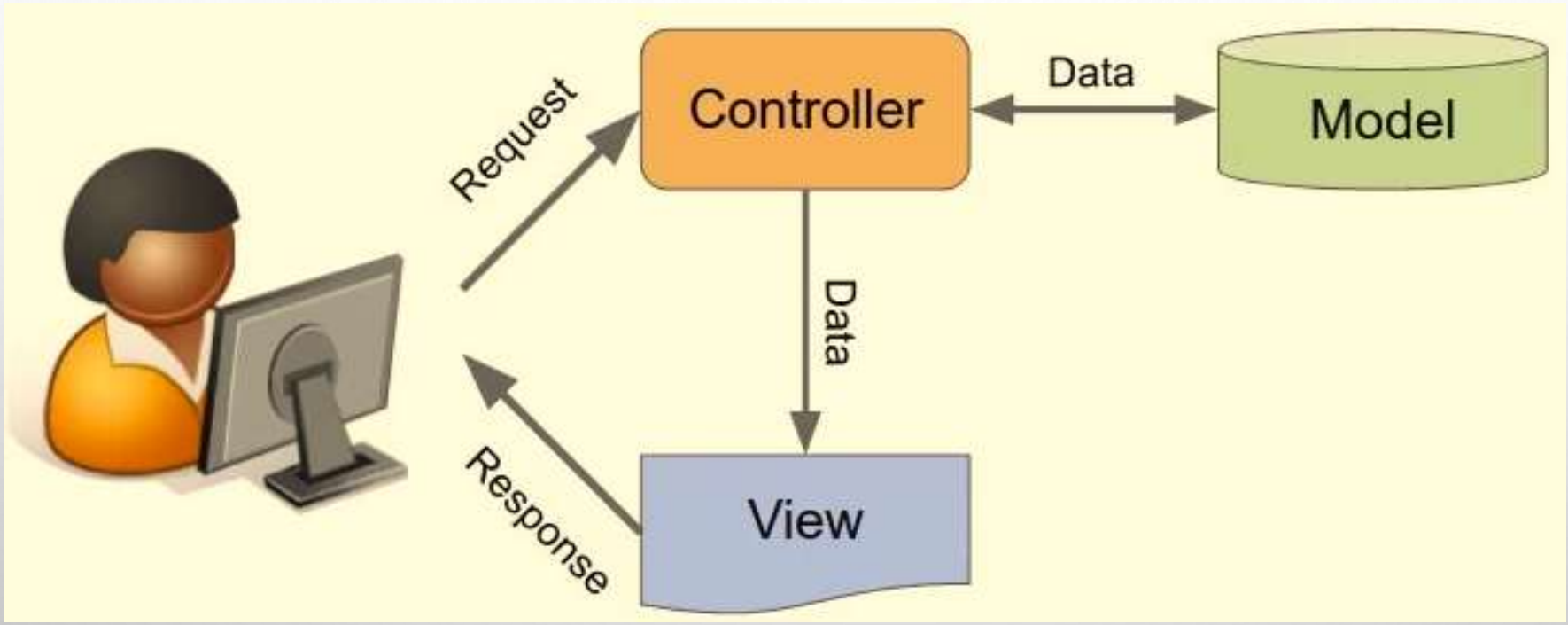
- ТРУДНОЩІ З СУПРОВОДОМ КОДУ: файли неструктуровані, зустрічається таке – `require '../../../../../../../../database.inc.php';`
- ТРУДНОЩІ РОЗРОБКИ: логіка застосунку змішана з презентаційним рівнем; програміст і дизайнер не можуть працювати з одним і тим же файлом
- ПОРУШЕННЯ БЕЗПЕКИ: паролі до баз даних містяться в папках публічного доступу

РІШЕННЯ: Використати Фреймворк

- ФРЕЙМВОРК – ЦЕ БІБЛІОТЕКА КОДУ. ВОНА ПРОПОНУЄ СТРУКТУРУ ПОБУДОВИ ДОДАТКУ.
- ВИ ЗМОЖЕТЕ КОДУВАТИ ШВИДШЕ
- БІЛЬШЕ ОДНІЄЇ ПЕРСОНИ МОЖЕ ПРАЦЮВАТИ З КОДОМ
- КОД СТАЄ МЕНШ СКЛАДНИМ, ЙОГО БУДЕ ЛЕГШЕ СУПРОВОДЖУВАТИ
- БІЛЬШЕ БЕЗПЕКИ: ПАРОЛІ ДО БАЗ ДАНИХ МОЖНА ЗБЕРІГАТИ ОКРЕМО ВІД ДИРЕКТОРІЙ ПУБЛІЧНОГО ДОСТУПУ

ШАБЛОН MVC

- The Model-View-Controller Pattern



УСТАНОВКА WEB СЕРВЕРА, СЕРВЕРА БД ТА PHP

ВАРІАНТИ:

- XAMPP (https://www.apachefriends.org/)
- AMPPS (http://www.ampps.com)

ampps

PRODUCTS APPS IT'S FREE PREMIUM FEATURES DOCS DOWNLOADS FORUMS TOUR Buy

Downloads

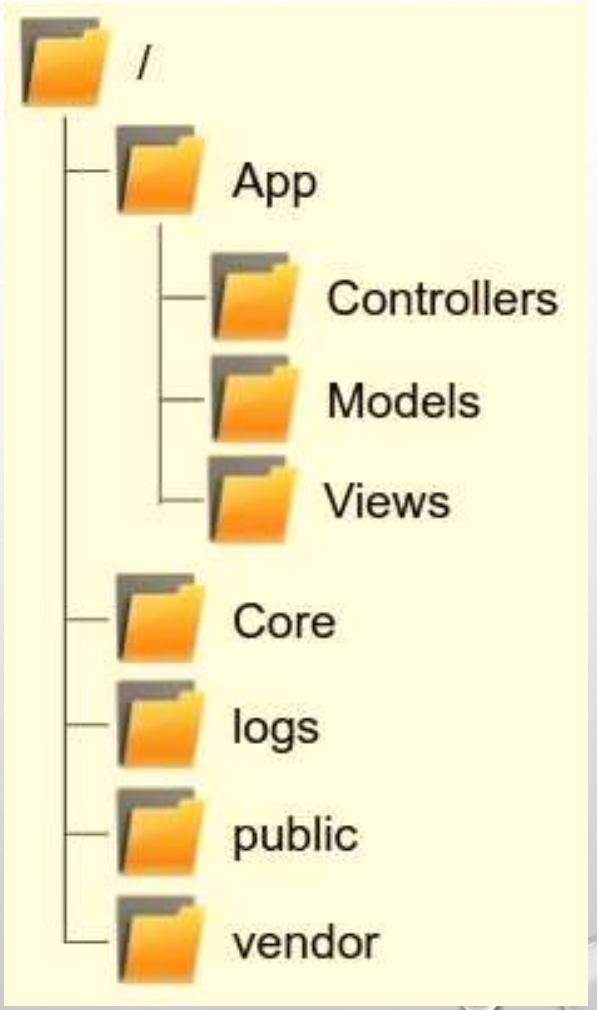
You must read and fully understand the License agreement before you download and use this software.

Current Version : 3.8

Includes	Windows	MAC OS X	Linux
Softaculous AMPPS	4.9.3	4.9.3	4.9.3
Apache	2.4.27	2.4.27	2.4.27
MySQL	5.6.37	5.6.37	5.6.37
PHP	7.1.8, 7.0.22, 5.6.31, 5.5.38, 5.4.45 and 5.3.29	7.1.8, 7.0.22, 5.6.31, 5.5.38, 5.4.42 and 5.3.29	7.1.8, 7.0.22, 5.6.31, 5.5.38, 5.4.45 and 5.3.29
PERL	5.16.3	5.26.0	5.26.0
Python	3.6.2	3.6.2	3.6.2
MongoDB(64-bit)	3.4.7	2.4.7	3.4.7

Start writing the framework: Create the folders and configure the web server

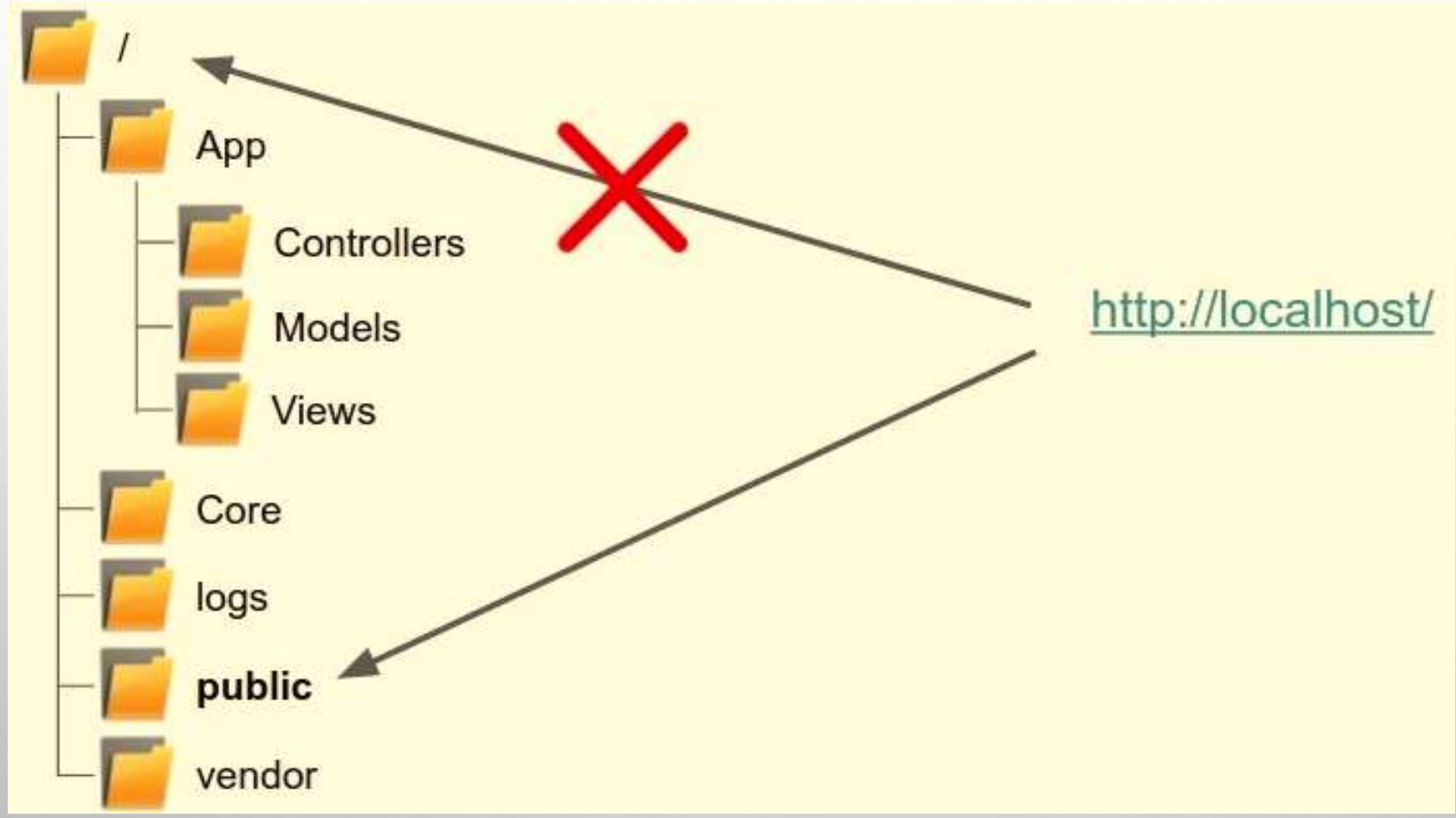
- FRAMEWORK FOLDERS



The **public** folder

- ЄДИНА ДИРЕКТОРІЯ, ДОСТУПНА З WEB
- КОРІНЬ (ROOT) ВЕБ СЕРВЕРА, ТОБТО НА ЦЮ ДИРЕКТОРІЮ НАПРАВЛЕНИЙ URL http://localhost
- ФРОНТ КОНТРОЛЕР (front controller) І ВСІ СТАТИЧНІ ФАЙЛИ (static files) – CSS, images etc. – ЗНАХОДЯТЬСЯ ТУТ
- ВЕЛИКОЮ ПЕРЕВАГОЮ Є ТЕ, ЩО БІЛЬША ЧАСТИНА КОДУ НЕ Є ДОСТУПНОЮ В ІНТЕРНЕТІ, І ТОМУ Є БІЛЬШ ЗАХИЩЕНА

Configure the root of the web server



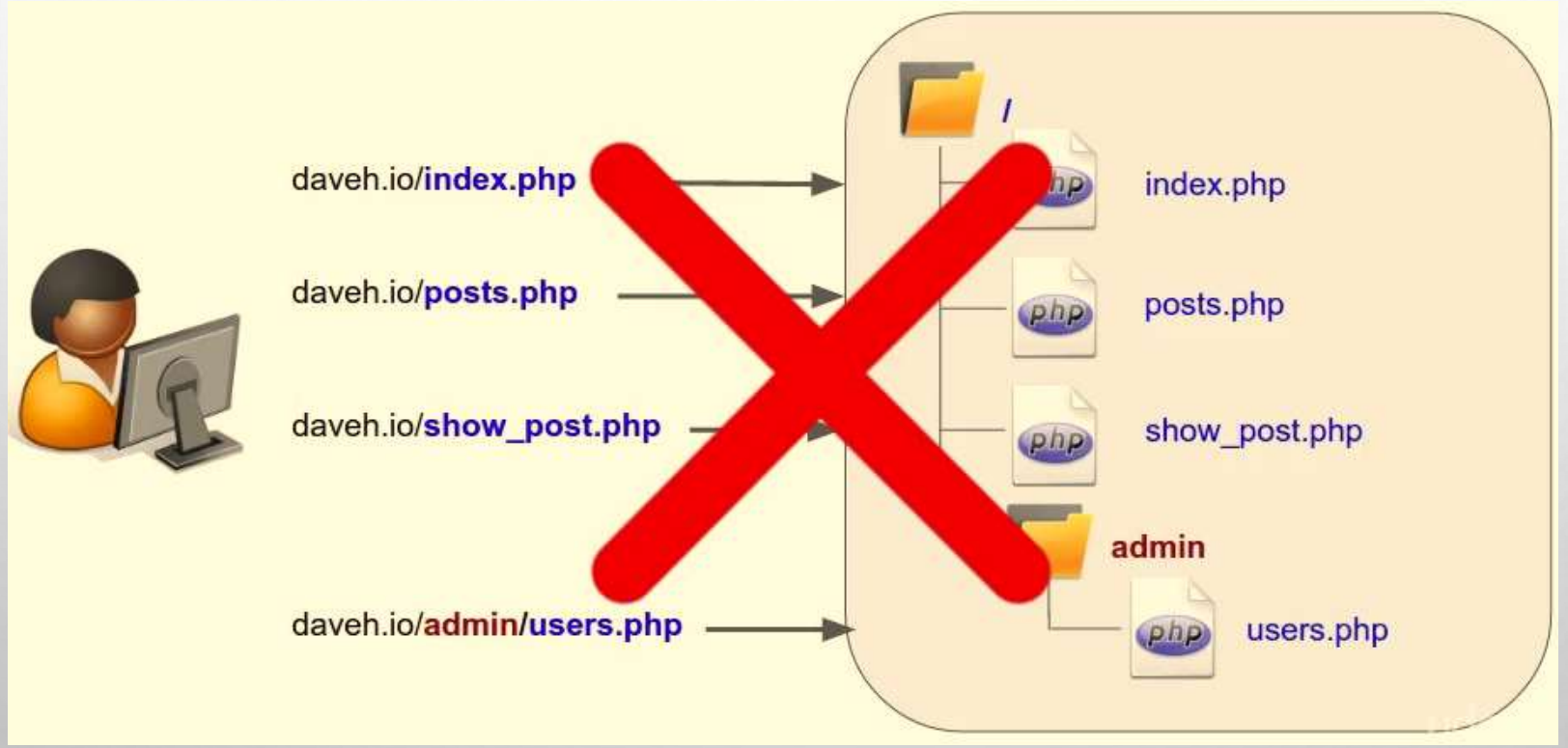
КОНФІГУРУВАННЯ СЕРВЕРА APACHE

- ДЛЯ AMPPS: файл **httpd.conf**
- Знайти секцію #### Localhost VirtualHost ####
- Змінити рядок **DocumentRoot** :

  DocumentRoot “{$path}/www/public”
- Перемістити файл **index.php** в директорію **public**

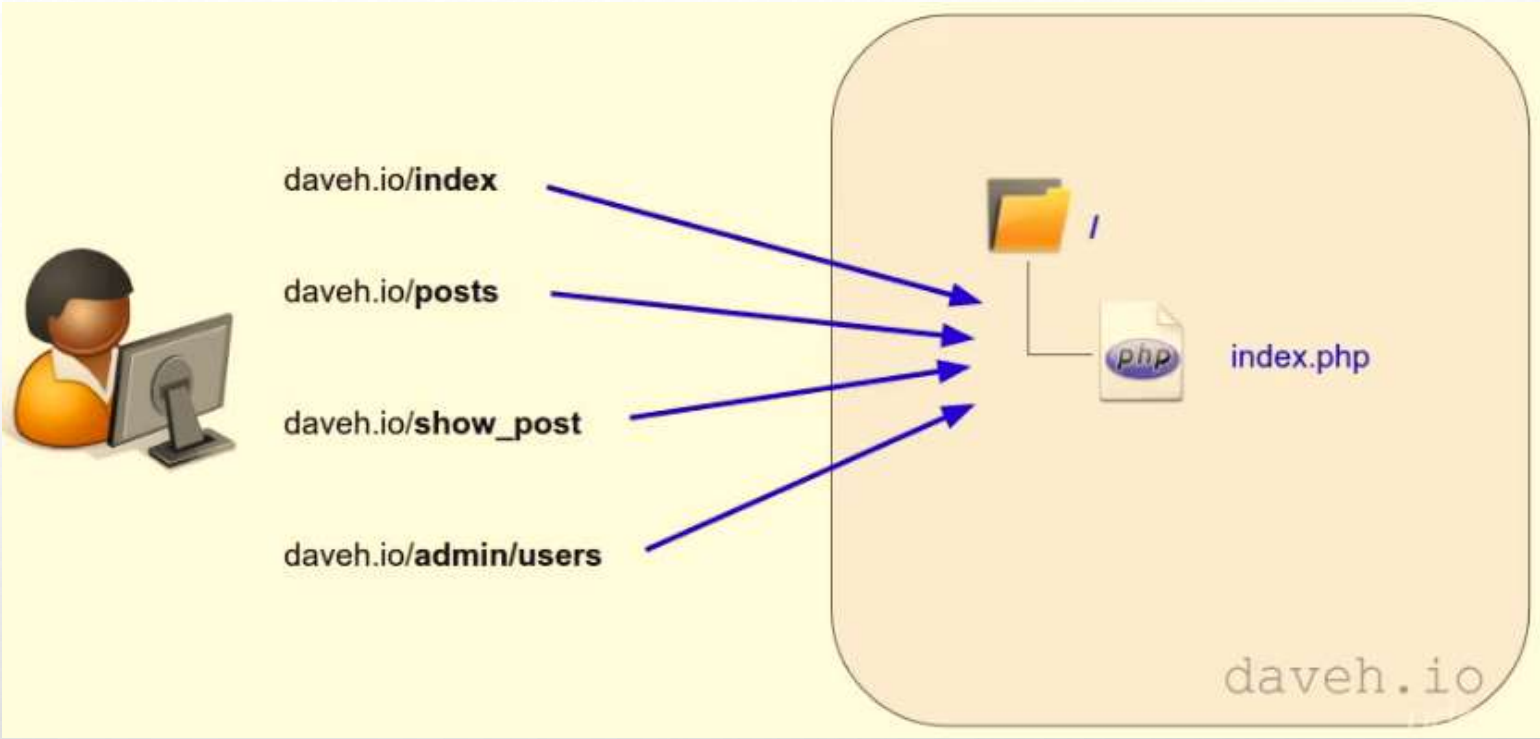
СТВОРЕННЯ ЦЕНТРАЛЬНОЇ ТОЧКИ ВХОДУ У ФРЕЙМВОРК: FRONT CONTROLLER

- URL = file location is a bad idea



FRONT CONTROLLER

Замість того, щоб відображати URL-адреси на окремі скрипти, URL-адреси відображаються на ДІЇ в КОНТРОЛЕРАХ.



- ALL REQUESTS ARE SENT THROUGH ONE PAGE, IN THIS CASE, INDEX.PHP

The request is in the query string

- QUERY STRING – це частина URL, що йде після першого знака питання

  `daveh.io/index.php?/home`

- Ми можемо використовувати це, щоб вирішити, куди направити запит, іншими словами, який контролер і яка дія.

  `daveh.io/index.php?/show_post/123`

  `daveh.io/index.php?/posts?page=2`

- Таким чином, весь рядок запиту буде нашою URL-адресою запиту (request URL) або маршрутом (route).

Змінимо файл index.php

```
1)<?php
  /**
   * front controller
   */
  echo ‘Requsted URL = “’ . $_SERVER[‘QUERY_STRING’] . ‘”’;
```

2) Наберемо в браузері: http://localhost/index.php?/home
   Отримаємо: Requsted URL = “/home”
3) Наберемо в браузері: http://localhost/index.php?/posts/index
   Отримаємо: Requsted URL = “/posts/index”
4) Наберемо в браузері: http://localhost/index.php?/posts/index?page=1
   Отримаємо: Requsted URL = “/posts/index?page=1”
5) Наберемо в браузері: http://localhost/?/posts/index?page=1
   Отримаємо: Requsted URL = “/posts/index?page=1”

Configure the web server to have pretty URLs

Remove the default page

The default page configured on your web server

- The default page is (most likely) index.php
- So it can be removed from the URL

Також можна видалити знак запитання: те, що позначає, де починається рядок запиту.

При цьому ми будемо мати pretty URLs.

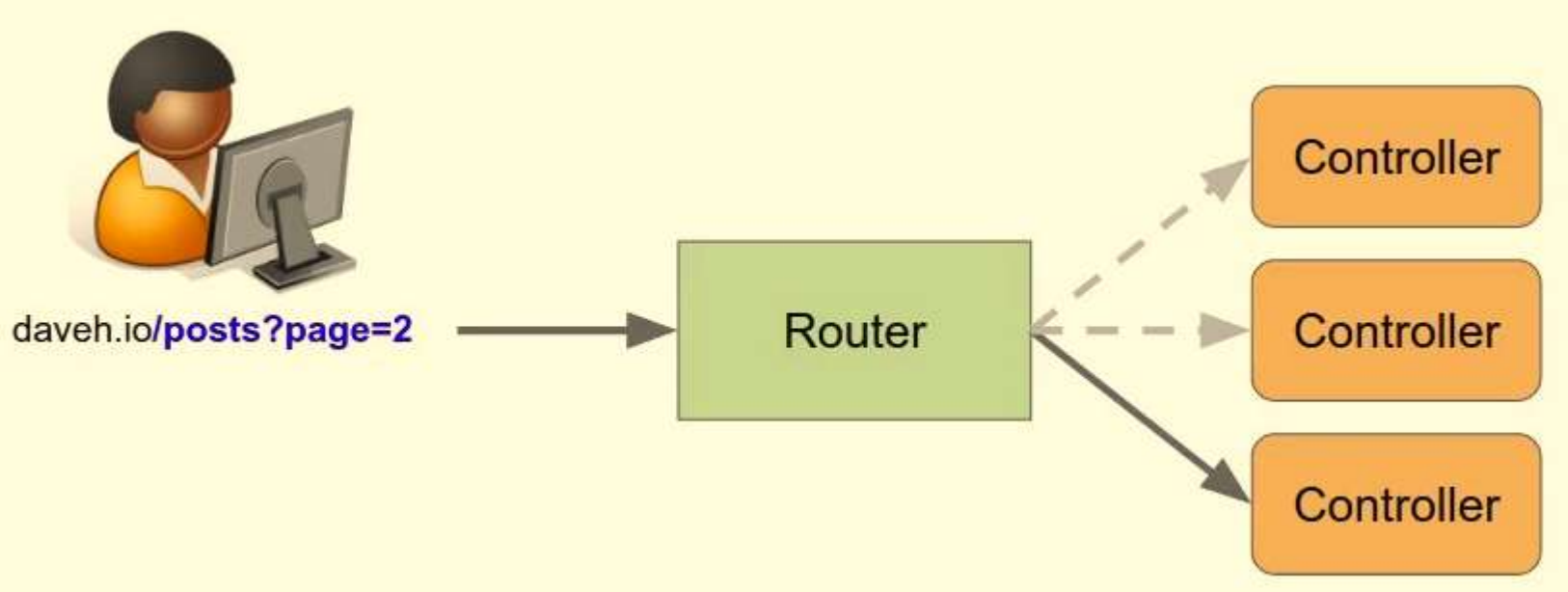
Зміни на веб-сервері

- В директорію public помістіть файл .htaccess наступного вмісту:

```
# Remove the question mark from the request but maintain the query string
RewriteEngine On
RewriteBase /
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-l
RewriteRule ^(.*)$ index.php?$1 [L,QSA]
```

Створення класу роутера

Takes the request URL or **route** and decides what to do with it:



Class Router

1) Помістіть в директорію Core файл Router.php :

```
<?php
 class Router
 {
 }
```

2) Змініть файл index.php :

```
<?php
 //echo 'Requsted URL = "' . $_SERVER['QUERY_STRING'] . '"';
 require '../Core/router.php';
 $router = new Router();
 echo get_class($router);
```

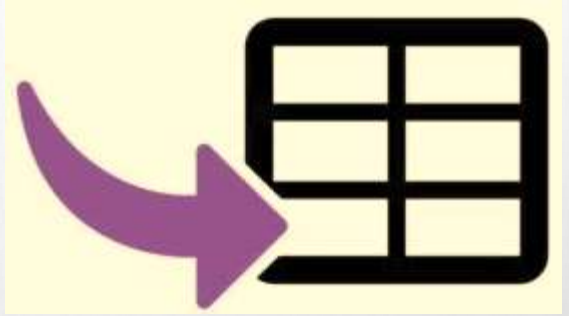
3) Наберіть у браузері http://localhost/

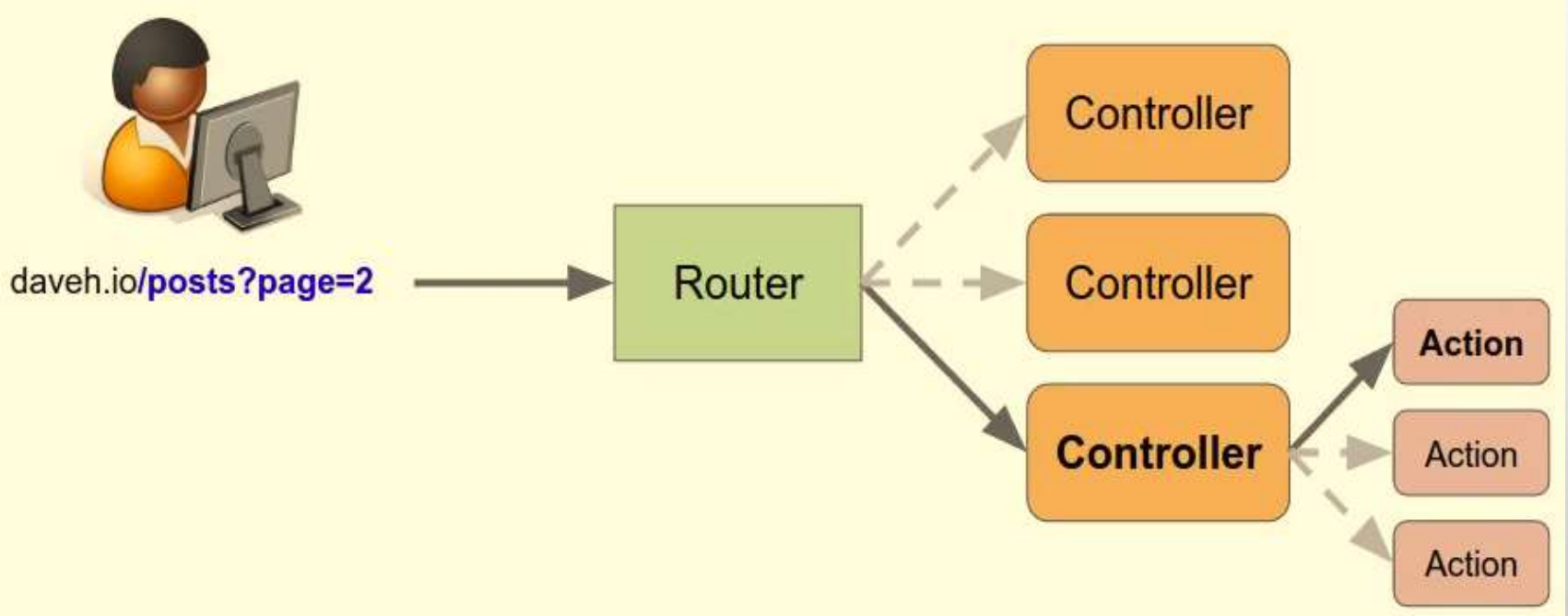
Отримаєте відповідь:

Router

Створення routing table в роутері і додавання деяких маршрутів

- пам'ятайте, що роутер вирішує, який контролер і дію буде виконано





The routing table

Route	Controller	Action
"/"	Home	index
"/posts"	Posts	index
"/show_post"	Posts	show
"/admin/users"	Admin\Users	index
...		

- Внесемо зміни в файли router.php та index.php нашого проекту

Файл router.php

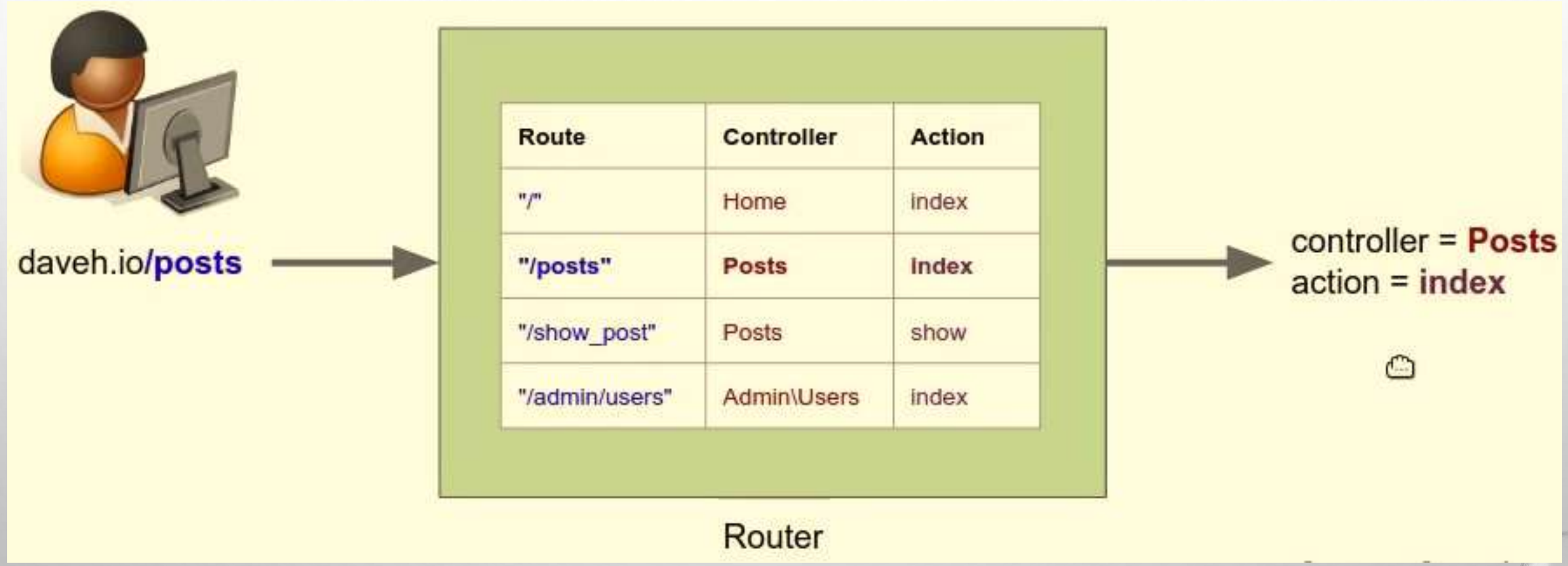
```
<?php
class Router{
    protected $routes = [];
    public function add($route, $params)    {
      $this->routes[$route] = $params;
    }
    public function getRoutes()    {
       return $this->routes;
    }
}
```

Файл index.php

```
<?php
require '../Core/Router.php';
$router = new Router();
$router->add('', ['controller' => 'Home', 'action' => 'index']);
$router->add('posts', ['controller' => 'Posts', 'action' => 'index']);
$router->add('posts/new', ['controller' => 'Posts', 'action' => 'new']);
 // Display the routing table
echo '<pre>';
var_dump($router->getRoutes());
echo '</pre>';
```

Наберіть в браузері http://localhost/ щоб побачити routing table

Відповідність URL-адреси запиту до маршруту



Файл router.php

Додамо властивість і два метода:

```
protected $params = [];
.   .   .   .
public function match($url)   {
        foreach ($this->routes as $route => $params) {
            if ($url == $route) {
                $this->params = $params;
                return true;
            }
        }
        return false;
}
public function getParams()   {
return $this->params;   }
```

Файл index.php

- Закоментуємо друк таблиці маршрутизації.
- Додамо:

```
$url = $_SERVER['QUERY_STRING'];
if ($router->match($url)) {
    echo '<pre>';
    var_dump($router->getParams());
    echo '</pre>';
} else {
    echo "No route found for URL '$url'";
}
```

Тестування

Наберіть у браузері http://localhost/
(для XAMPP http://localhost/mvcframework1/)
Отримаєте відповідь:

```
array(2) {
  ["controller"]=>
  string(4) "Home"
  ["action"]=>
  string(5) "index"
}
```

Наберіть у браузері http://localhost/posts
Отримаєте відповідь:

```
array(2) {
  ["controller"]=>
  string(5) "Posts"
  ["action"]=>
  string(5) "index"
}
```

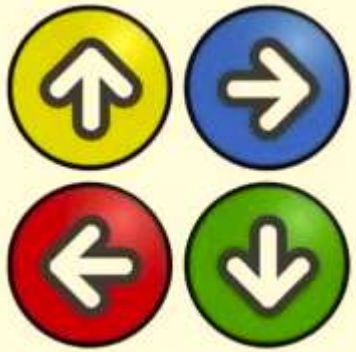
Наберіть у браузері http://localhost/posts/new

Наберіть у браузері http://localhost/products

(Дивись приклад mvcframework1 в Moodle)

03 Вступ до розширеної маршрутизації з використанням змінних маршруту.

Simple route matching



- Current routing table has an entry for **each route**:

Request URL	Routing table		
	URL	Controller	Action
daveh.io/**posts/index**	"posts/index"	Posts	index
daveh.io/**posts/new**	"posts/new"	Posts	new
daveh.io/**blog/index**	"blog/index"	Blog	index
daveh.io/**products/list**	"products/list"	Products	list

Advanced route matching

- Route **patterns**:

Request URL	Routing table	Controller	Action
	URL		
daveh.io/**posts/index**	" controller / action "	Posts	index
daveh.io/**posts/new**		Posts	new
daveh.io/**blog/index**		Blog	index
daveh.io/**products/list**		Products	list

Отримання контролера і дії з URL з фіксованою структурою

Matching routes with patterns

Замість простого порівняння рядків:

```
if ($url == $route) {
```

Шукати співпадання шаблону:

```
if (preg_match($reg_exp, $url)) {
```

Simple fixed URL structure

```
daveh.io/controller/action
```

```
daveh.io/posts/index
daveh.io/posts/new
daveh.io/blog/index
daveh.io/products/list
```

A regular expression for simple URL structure

```
daveh.io/controller/action
```

```
/^([a-z-]+)\/([a-z-]+)$/
```

```
preg_match($reg_exp, "posts/index", $matches)

[
  1 => "posts",
  2 => "index"
]
```

A regular expression for simple URL structure

```
daveh.io/controller/action
```

```
/^(?P<controller>[a-z-]+)\/(?P<action>[a-z-]+)$/
```

```
preg_match($reg_exp, "posts/index", $matches)

          [
            "controller" => "posts",
            "action" => "index"
          ]
```

ВИКОРИСТАННЯ ІМЕНОВАНИХ ПІДМАСОК

(https://www.php.net/manual/ru/function.preg-match.php)

```
<?php
  $str = ‘foobar: 2008’;
  preg_match('/(?P<name>\w+): (?P<digit>\d+)/', $str, $matches);
  print_r($matches);
?>
```

Результат:

```
Array
(
    [0] => foobar: 2008
    [name] => foobar
    [1] => foobar
    [digit] => 2008
    [2] => 2008
)
```

Файл Router.php, метод match

- Закоментуйте цикл foreach
- Додайте:

```
$reg_exp = "/^(?P<controller>[a-z-]+)\/(?P<action>[a-z-]+)$/";
if (preg_match($reg_exp, $url, $matches)) {
    $params = [];
    foreach ($matches as $key => $match) {
        if (is_string($key)) {
            $params[$key] = $match;
        }
    }
    $this->params = $params;
    return true;
}
```

Тестування

Наберіть у браузері http://localhost/ (для XAMPP http://localhost/mvcframework2/)
Отримаєте відповідь: No route found for URL ‘’

Наберіть у браузері http://localhost/home/index
Отримаєте відповідь:

```
array(2) {
  ["controller"]=>
  string(4) "home"
  ["action"]=>
  string(5) "index"
}
```

(Дивись приклад mvcframework2 в Moodle)

Отримання контролера і дії з URL зі змінною структурою

- Маршрути зі змінними

Fixed routes

- Adding a fixed route to the routing table:

```
$router->add('posts/index', [
    'controller' => 'Posts',
    'action' => 'index'
]);
```

Routes with variables

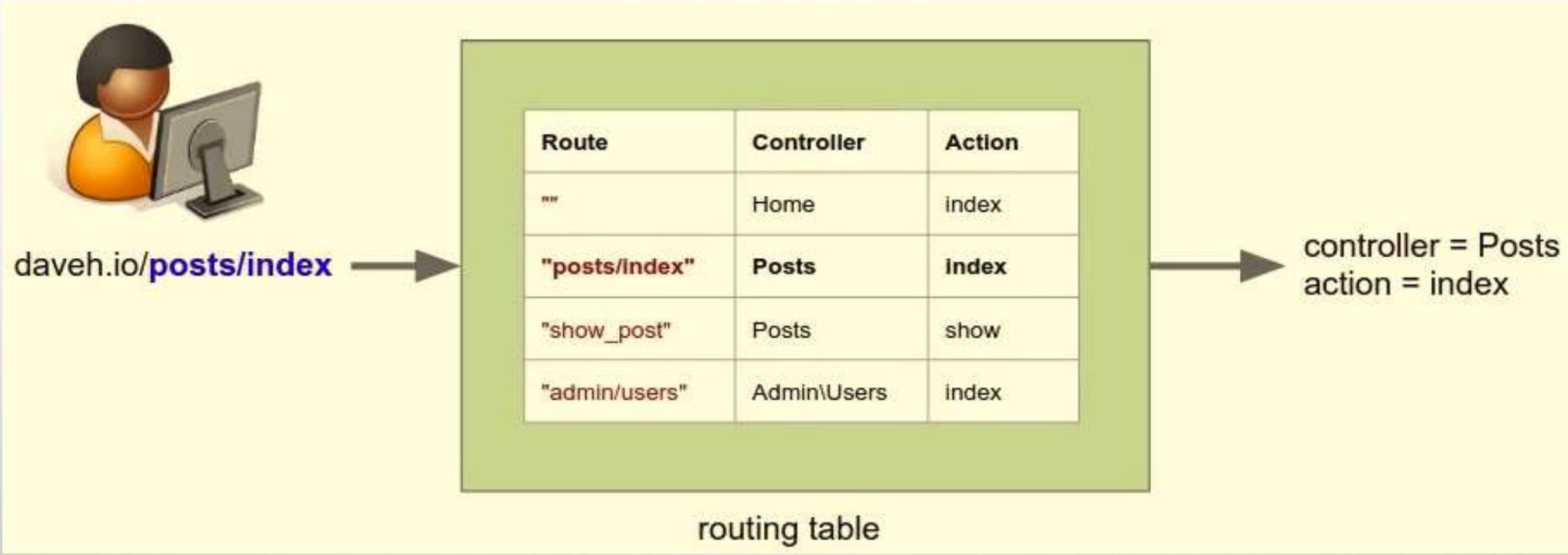
- Adding routes with **variables**:

```
$router->add('posts/index', [
    'controller' => 'Posts',
    'action' => 'index'
]);
```

```
$router->add('{controller}/{action}');
```

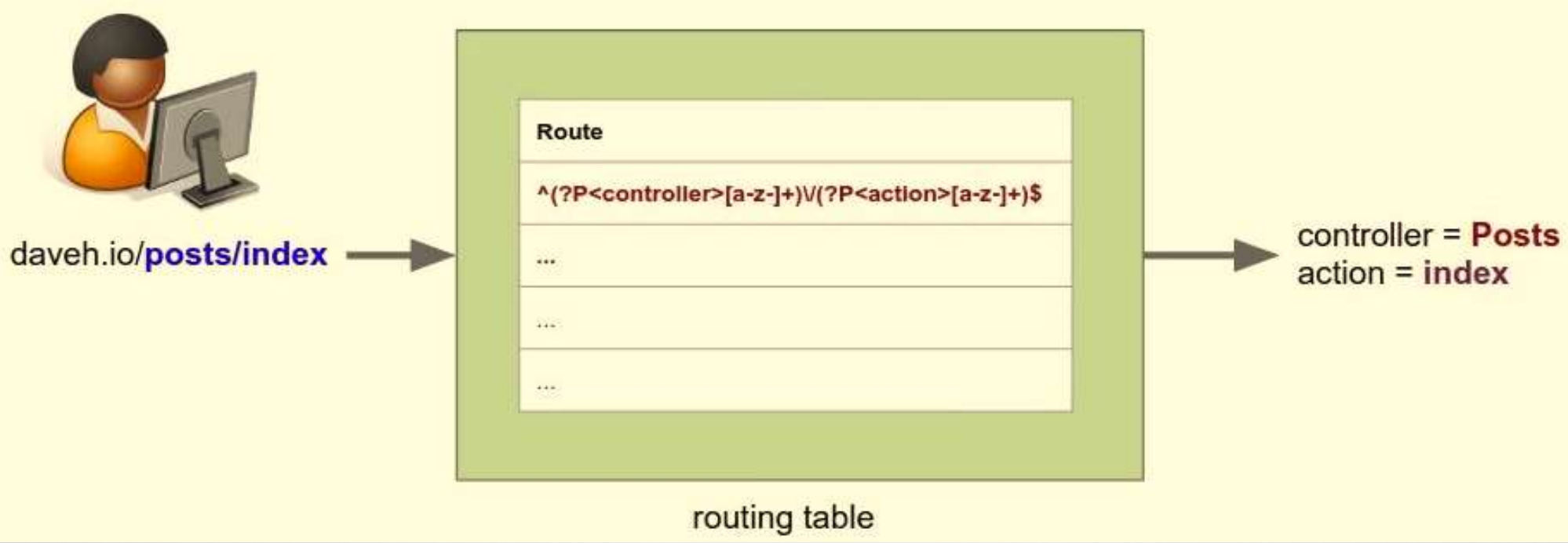
- Для додавання маршрутів із змінними нам потрібен спосіб вказати змінну при додаванні маршруту.
- Отже, припустимо, що змінні в маршруті – це рядок, що міститься в фігурних дужках.
- До речі, це не нотація регулярних виразів, ми просто вибрали фігурні дужки, щоб обмежити змінну частину маршруту.
- Таким чином, замість виконання рядкового порівняння URL-адреси запиту з кожним записом у таблиці маршрутизації, шукаючи відповідність, ми зробимо відповідність регулярних виразів з маршрутами.

Matching the request URL to the route




```
if ($url == $route) {
```

Matching the request URL to the route



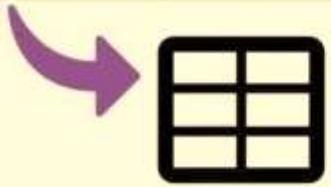

```
if (preg_match($reg_exp, $url)) {
```

Processing the route containing variables

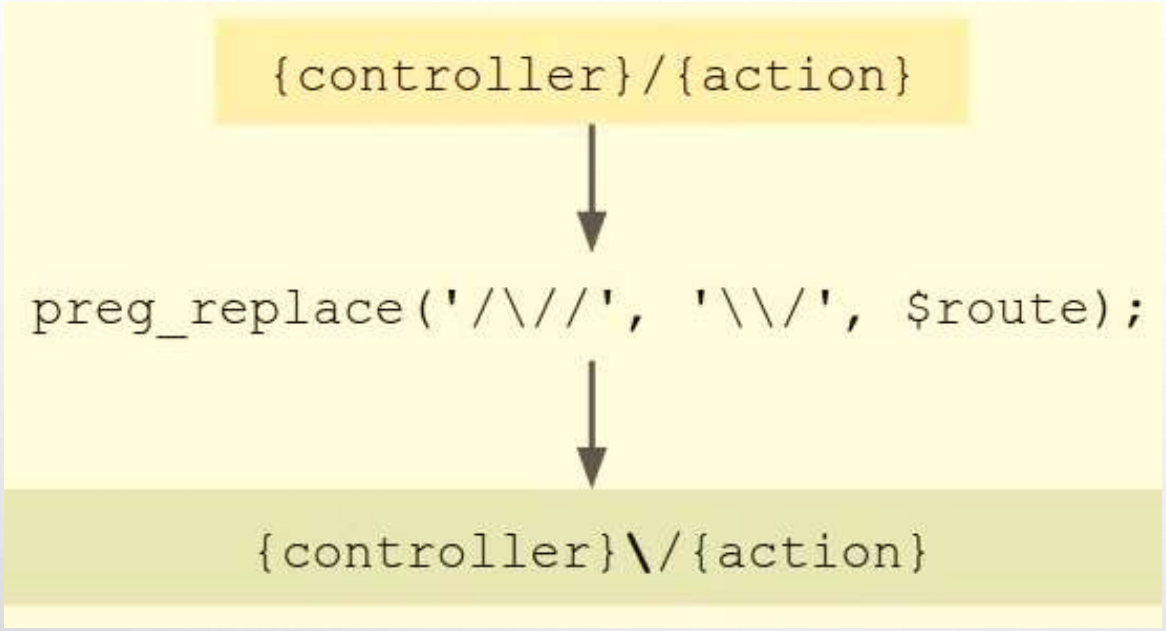
- Таким чином, маршрути в таблиці маршрутизації тепер будуть регулярними виразами замість фіксованих рядків.
- Таким чином, щоб відповідати маршрутам, які містять змінні, нам необхідно перетворити їх на регулярні вирази.
- Потім ми можемо додати їх до таблиці маршрутизації.

```
$router->add('{controller}/{action}');
```

```
/^(?P<controller>[a-z-]+)\/(?P<action>[a-z-]+)$/
```

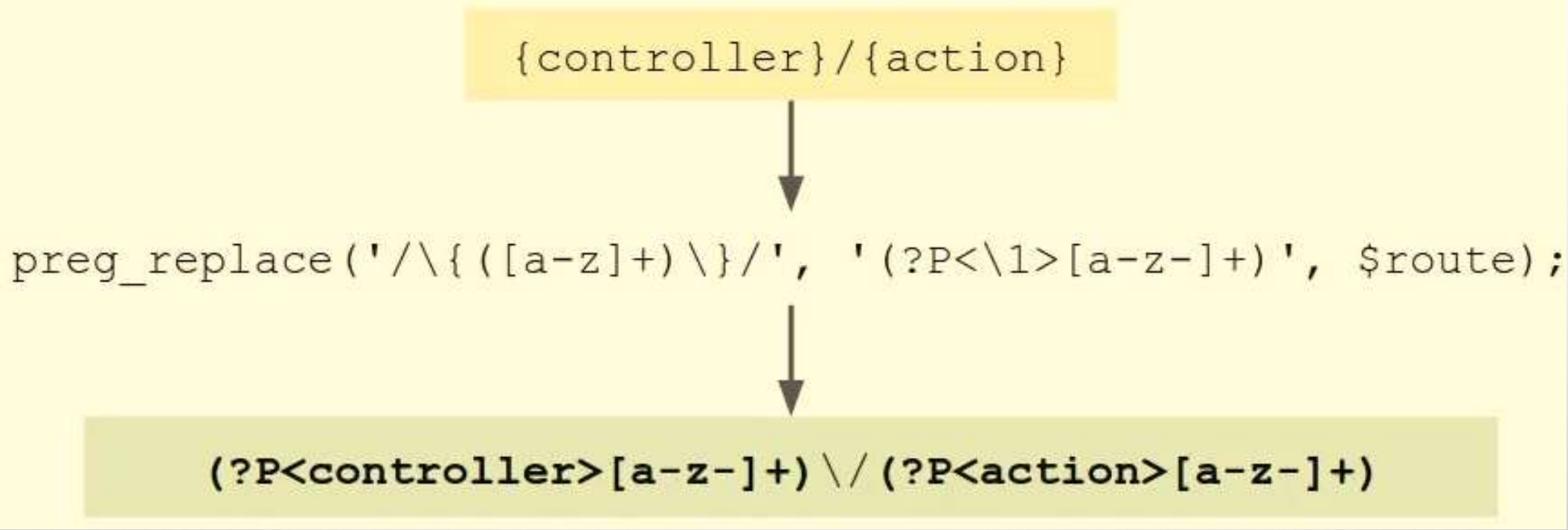


Turning the route into a regular expression



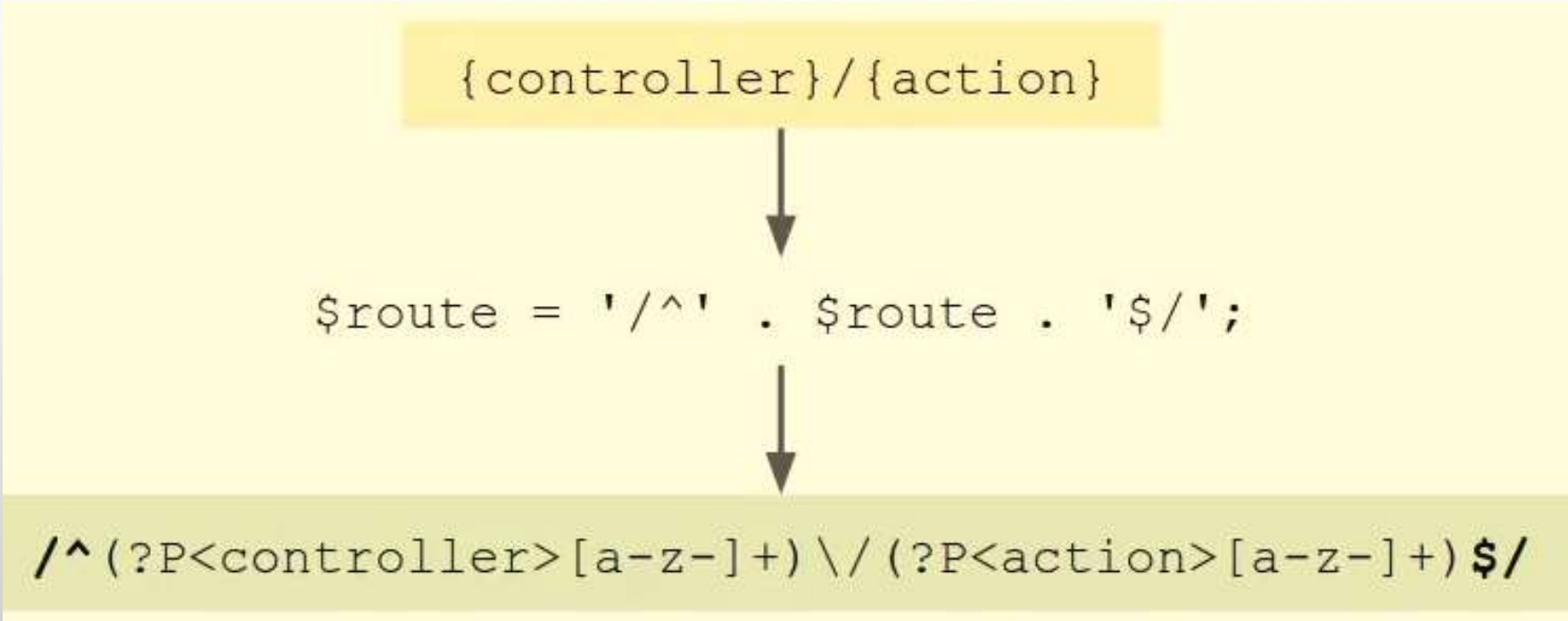
- По-перше, нам потрібно уникнути прямих слешів, тому що це спеціальні символи у регулярних виразах.

Turning the route into a regular expression



- Тут нам потрібно перетворити змінні в рядках між фігурними дужками.

Turning the route into a regular expression



- Нарешті, ми додамо початкові та кінцеві роздільники регулярного виразу.

Файл Router.php, метод add

```
public function add($route, $params = [])
{
  $route = preg_replace('/\//', '\\/', $route);
  $route = preg_replace('/\{([a-z]+)\}/', '(?P<\1>[a-z-]+)', $route);
  $route = '/^' . $route . '$/i';
  $this->routes[$route] = $params;
}
```

Файл Router.php, метод match

```
public function match($url)
{
  foreach ($this->routes as $route => $params) {
     if (preg_match($route, $url, $matches)) {
         foreach ($matches as $key => $match) {
             if (is_string($key)) {
                 $params[$key] = $match;
             }
         }
         $this->params = $params;
         return true;
     }
  }
  return false;
}
```

Файл index.php

```
<?php
  require '../Core/Router.php';
  $router = new Router();
  $router->add('', ['controller' => 'Home', 'action' => 'index']);
  $router->add('posts', ['controller' => 'Posts', 'action' => 'index']);
   //$router->add('posts/new', ['controller' => 'Posts', 'action' => 'new']);
  $router->add('{controller}/{action}');
  $router->add('admin/{action}/{controller}');
  // Display the routing table
  echo '<pre>';//var_dump($router->getRoutes());
  echo htmlspecialchars(print_r($router->getRoutes(), true));
  echo '</pre>';
   // Match the requested route
  $url = $_SERVER['QUERY_STRING'];
   if ($router->match($url)) {
      echo '<pre>';
      var_dump($router->getParams());
      echo '</pre>';
   } else {    echo "No route found for URL '$url'";
}
```

Тестування http://localhost/ (для XAMPP http://localhost/mvcframework3/)

localhost/

```
Array
(
    [/^$/i] => Array
        (
            [controller] => Home
            [action] => index
        )

    [/^posts$/i] => Array
        (
            [controller] => Posts
            [action] => index
        )

    [/^(?P<controller>[a-z-]+)\/(?P<action>[a-z-]+)$/i] => Array
        (
        )

    [/^admin\/(?P<action>[a-z-]+)\/(?P<controller>[a-z-]+)$/i] => Array
        (
        )

)

array(2) {
  ["controller"]=>
  string(4) "Home"
  ["action"]=>
  string(5) "index"
}
```

Тестування http://localhost/products/show

```
Array
(
    [/^$/i] => Array
        (
            [controller] => Home
            [action] => index
        )

    [/^posts$/i] => Array
        (
            [controller] => Posts
            [action] => index
        )

    [/^(?P<controller>[a-z-]+)\/(?P<action>[a-z-]+)$/i] => Array
        (
        )

    [/^admin\/(?P<action>[a-z-]+)\/(?P<controller>[a-z-]+)$/i] => Array
        (
        )

)

array(2) {
  ["controller"]=>
  string(8) "products"
  ["action"]=>
  string(4) "show"
}
```

(Дивись приклад mvcframework3 в Moodle)

Додавання до URL потрібних змінних будь-якого формату

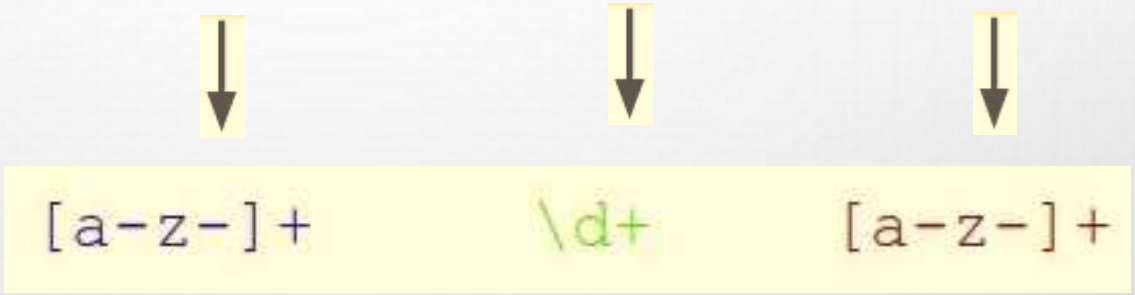
```
daveh.io/posts/123/edit
daveh.io/users/456/show
daveh.io/products/789/delete
```


```
$router->add('/{controller}/{id}/{action}');
```

Custom regular expression

- Define an optional regular expression for the variable:

```
$router->add('{controller}/{id:\d+}/{action}');
```



Turning the route into regular expression

```
/{controller}/{id:\d+}/{action}
```



```
preg_replace('/\{([a-z]+):([^\}]+)\}/', '(?P<\1>\2)', $route);
```



```
/{controller}/(?P<id>\d+)/{action)
```

Зміни в файлах Router.php, index.php

- Файл Router.php, метод add: (додати рядок після другого рядка)

```
$route = preg_replace('/\{([a-z]+):([^\}]+)\}/', '(?P<\1>\2)', $route);
```

- Файл index.php, додати маршрут

```
$router->add('{controller}/{id:\d+}/{action}');
```

- Файл index.php, закоментувати маршрут

```
//$router->add('admin/{action}/{controller}');
```

Тестування: http://localhost/ (для XAMPP http://localhost/mvcframework4/)

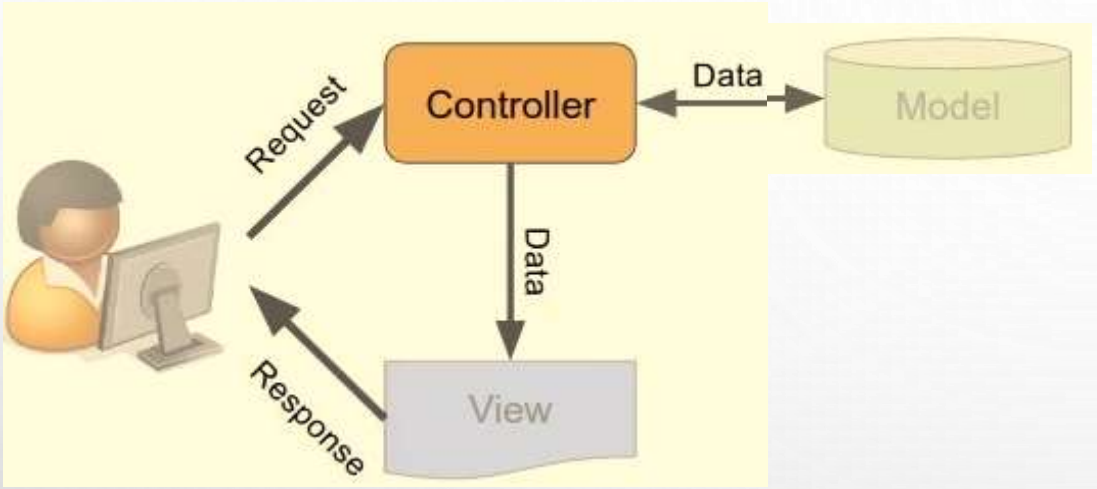
```
Array
(
    [/^$/i] => Array
        (
            [controller] => Home
            [action] => index
        )

    [/^posts$/i] => Array
        (
            [controller] => Posts
            [action] => index
        )

    [/^(?P<controller>[a-z-]+)\/(?P<action>[a-z-]+)$/i] => Array
        (
        )

    [/^(?P<controller>[a-z-]+)\/(?P<id>\d+)\/(?P<action>[a-z-]+)$/i] => Array
        (
        )

)

array(2) {
  ["controller"]=>
  string(4) "Home"
  ["action"]=>
  string(5) "index"
}
```

Тестування: http://localhost/posts/123/edit

```
    [/^$/i] => Array
        (
            [controller] => Home
            [action] => index
        )

    [/^posts$/i] => Array
        (
            [controller] => Posts
            [action] => index
        )

    [/^(?P<controller>[a-z-]+)\/(?P<action>[a-z-]+)$/i] => Array
        (
        )

    [/^(?P<controller>[a-z-]+)\/(?P<id>\d+)\/(?P<action>[a-z-]+)$/i] => Array
        (
        )

)

array(3) {
  ["controller"]=>
  string(5) "posts"
  ["id"]=>
  string(3) "123"
  ["action"]=>
  string(4) "edit"
}
```

(Дивись приклад mvcframework4 в Moodle)

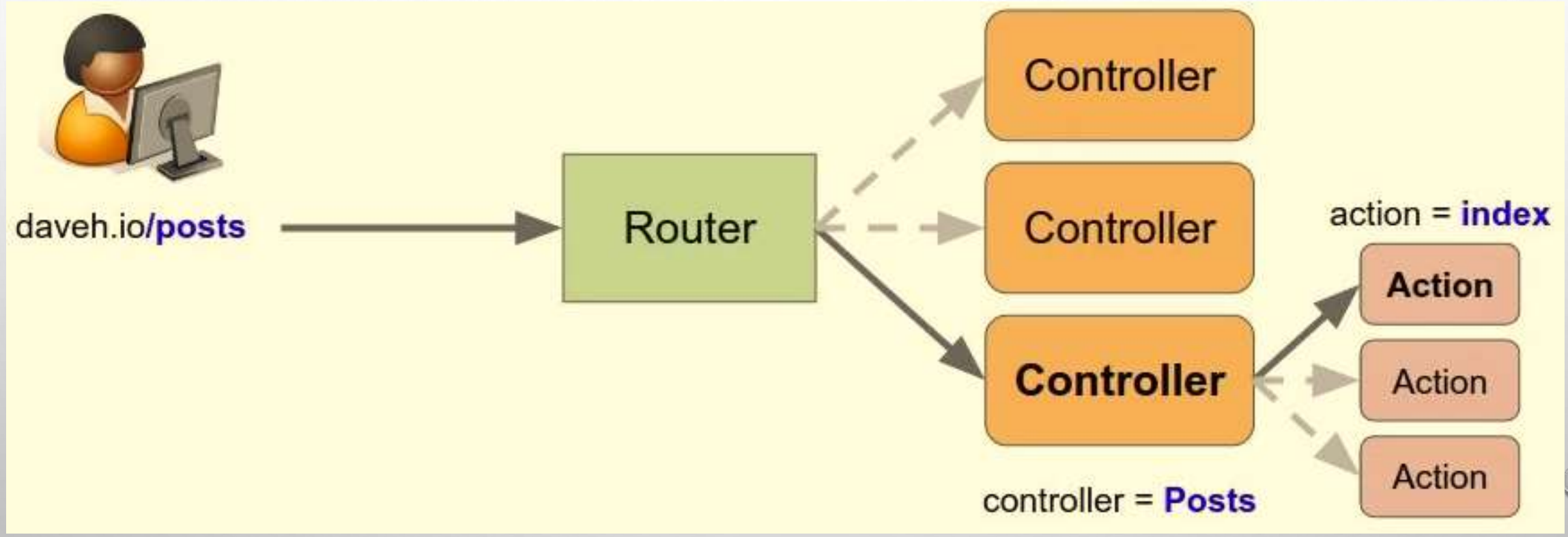
Контролери і дії



- Controllers are what the user **interacts** with.
- They receive a **request** from the user, decide what to do, and send a **response** back.

Controller and action

- The **router** has matched a URL to a **route** to get a **controller** and an **action.**



Controllers and actions

- Controllers are **classes**.
- They contain methods that are the **actions**.

Controller	Action
Home	index
Posts	index
Posts	new
Posts	edit

```
class Posts
{
    public function index()
    {
        // Show list of posts
        ...
    }
    public function new()
    {
        // Create a new post
        ...
    }
}
```

Dynamically creating objects

- To create a new object:

```
$post = new Post();
```

```
class Post
{
   ...
}
```

- To create an object based on a variable:

```
$class_name = "Post";
$post = new $class_name();
```

Dynamically calling methods

- To call a method:

```
$post = new Post();
$post->save();
```

- To call a method **based on a variable**:

```
$method = "save";
$post->$method();
```

```
class Post
{
    public function save()
    {
        ...
    }
}
```

Passing parameters

- To call a method and **pass parameters** to it:

```
class Post
{
    public function save($arg1, $arg2)
    {
        ...
    }
}
```

```
$post = new Post();
call_user_func_array([$post, “save”], [123, “abc”]);
```

Error handling

- To check if a **class exists** before creating an object:

```
$class_name = "Post";

if (class_exists($class_name)) {

    $post = new $class_name();

}
```

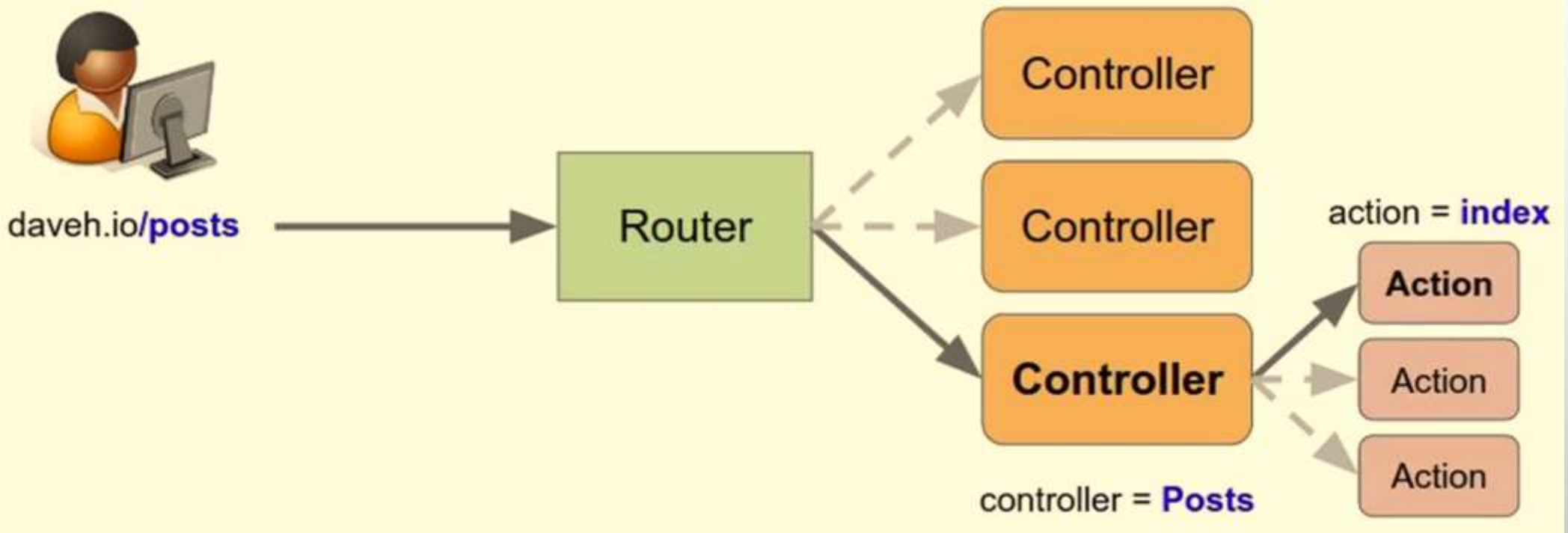
Error handling

- To check the **method exists**, and is **public** before calling it:

```
$post = new Post();

$method = "save";

if (is_callable([$post, $method])) {

    $post->$method();

}
```

Dispatch the route: create the controller object and run the action method

Controller and action



Dispatching

- Routing = **asking for directions**
- Dispatching = **following those directions**

The dispatching step is going to **create a controller object** and **run the action method**

Отримання класу контролера з маршруту

- Маршрутизатор (router) надасть **параметр** контролера, взятого з URL-адреси
- У URL-адресі слова розділені дефісами.
- Controller classes are named using StudlyCaps (PSR-1 coding standard)

Route (e.g. `/{controller}/{action}`)	Controller	
	Parameter	Class
`/posts/index`	`posts`	`Posts`
`/products/add-new`	`products`	`Products`
`/post-authors/list`	`post-authors`	`PostAuthors`

Get the action method name from the route

- Router will provide an action **parameter**, taken from the URL
- Words separated in the URL by hyphens
- Action **methods** are named using **camelCase**

Route (e.g. `/{controller}/{action}`)	**Action**	
	Parameter	**Method name**
`/posts/index`	`index`	`index`
`/products/add-new`	`add-new`	`addNew`
`/post-authors/list`	`list`	`list`

Convert parameters to classes and methods

Controller	
Parameter	Class
posts	Posts
products	Products
post-authors	PostAuthors

Action	
Parameter	Method name
index	index
add-new	addNew
list	list

Файл Router.php, метод dispatch

```
public function dispatch($url)
{
    if ($this->match($url)) {
        $controller = $this->params['controller'];
        $controller = $this->convertToStudlyCaps($controller);
        if (class_exists($controller)) {
                $controller_object = new $controller();
                $action = $this->params['action'];
                $action = $this->convertToCamelCase($action);
                if (is_callable([$controller_object, $action])) {
                    $controller_object->$action();
```

```
        } else {
                    echo "Method $action (in controller $controller) not found";
                }
            } else {
                echo "Controller class $controller not found";
            }
        } else {
            echo 'No route matched.';
        }
    }
 protected function convertToStudlyCaps($string)    {
        return str_replace(' ', '', ucwords(str_replace('-', ' ', $string)));
 }
protected function convertToCamelCase($string)    {
        return lcfirst($this->convertToStudlyCaps($string));
}
```

Контролер Posts

```
<?php
class Posts{
  public function index()    {
        echo 'Hello from the index action in the Posts controller!';
  }
  public function addNew()    {
        echo 'Hello from the addNew action in the Posts controller!';
  }
}
```

Front Controller (файл index.php)

```
<?php
require '../App/Controllers/Posts.php';
require '../Core/Router.php';

$router = new Router();

$router->add('', ['controller' => 'Home', 'action' => 'index']);
$router->add('{controller}/{action}');
$router->add('{controller}/{id:\d+}/{action}');

$router->dispatch($_SERVER['QUERY_STRING']);
```

Тестування

- URL: http://localhost/

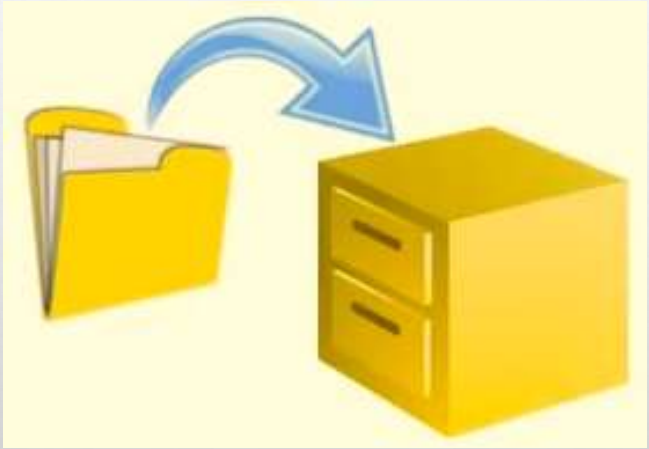
  (для XAMPP http://localhost/mvcframework5/)
- Відповідь: Controller class Home not found
- URL: http://localhost/mvcframework5/posts/index

  Відповідь: Hello from the index action in the Posts controller!
- URL: http://localhost/mvcframework5/posts/add-new

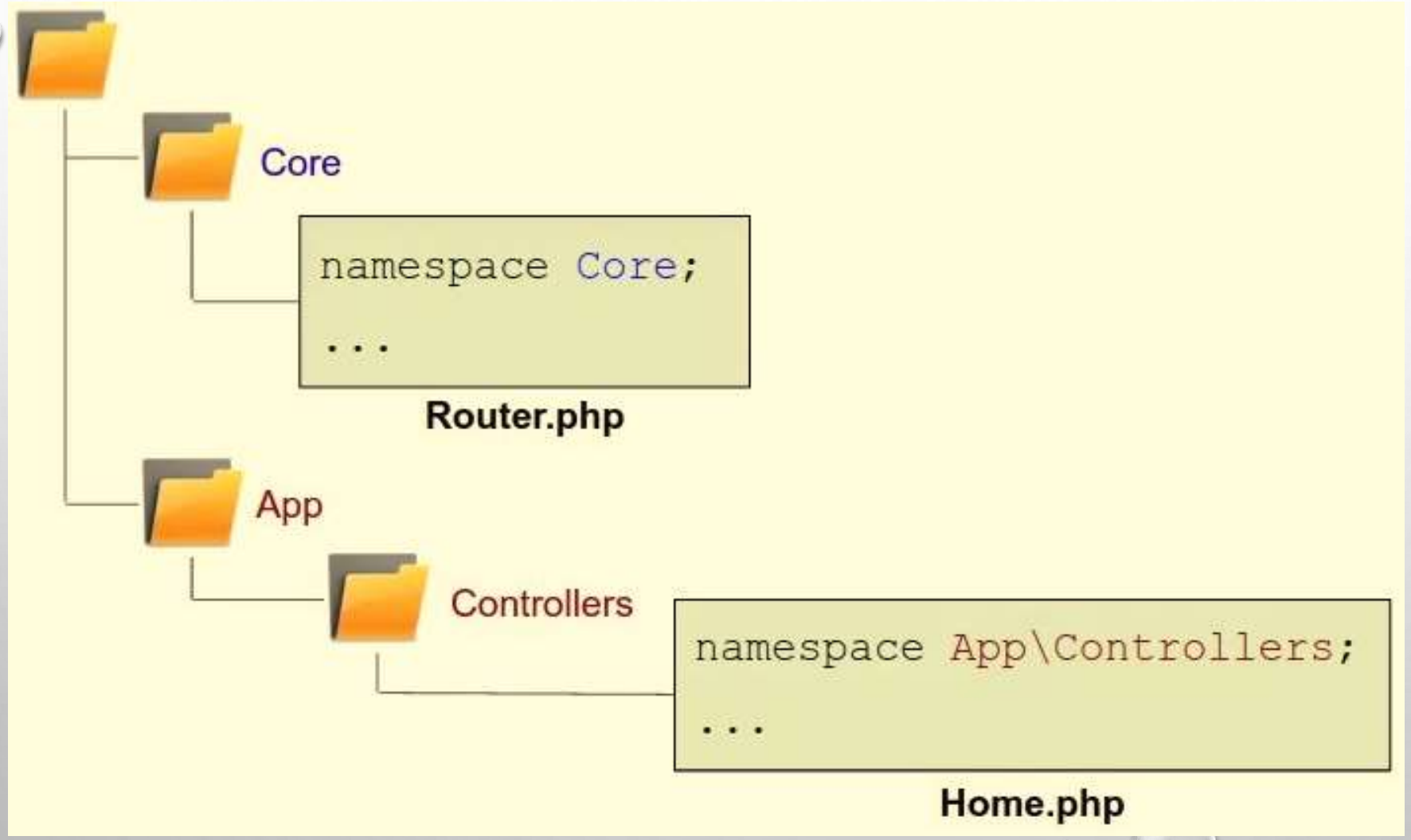
  Відповідь: Hello from the addNew action in the Posts controller!

(Дивись приклад mvcframework5 в Moodle)

Load classes automatically: add namespaces and an autoload function



Adding class namespaces



Adding the autoload function

- We want to require class files relative to the **root** of the site
- Front controller (index.php) is in the **public** folder
- The root folder is the parent directory of the **public** folder

index.php	__DIR__	dirname(__DIR__)
/var/www/public/index.php	/var/www/public	/var/www

Зміни в коді

- Файл Router.php
  1) На початку файла додайте:
     namespace Core;
  2) В методі dispatch додайте:
     $controller =
“App\Controllers\\$controller”;
- Файл Posts.php
  1) На початку файла:
     namespace App\Controllers;

- Додайте контролер Home (Файл Home.php):

```
<?php

 namespace App\Controllers;

 class Home
 {
    public function index()
    {
       echo 'Hello from the index action in the
Home controller!';
     }
  }
```

Файл index.php

```
<?php
 //require '../App/Controllers/Posts.php';
 spl_autoload_register(function ($class) {
     $root = dirname(__DIR__);  // get the parent directory
     $file = $root . '/' . str_replace('\\', '/', $class) . '.php';
     if (is_readable($file)) {
         require $root . '/' . str_replace('\\', '/', $class) . '.php';
     }
 });
 $router = new Core\Router();
$router->add('', ['controller' => 'Home', 'action' => 'index']);
$router->add('{controller}/{action}');$router->add('{controller}/{id:\d+}/{action}');

$router->dispatch($_SERVER['QUERY_STRING']);
```

Тестування

- URL: http://localhost (для XAMPP http://localhost/mvcframework6/)

  Відповідь: Hello from the index action in the Home controller!

- URL: http://localhost/posts/index

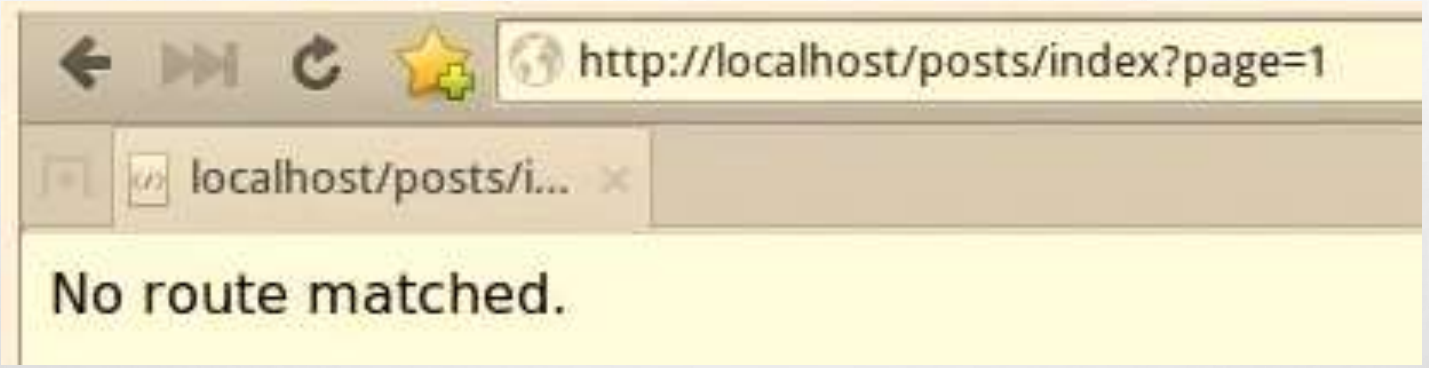
  Відповідь: Hello from the index action in the Posts controller!

(Дивись приклад mvcframework6 в Moodle)

Remove query string variables from the URL before matching to the route

Ситуація зараз

- Роутер не працює, якщо додати до маршруту **query string variable**:

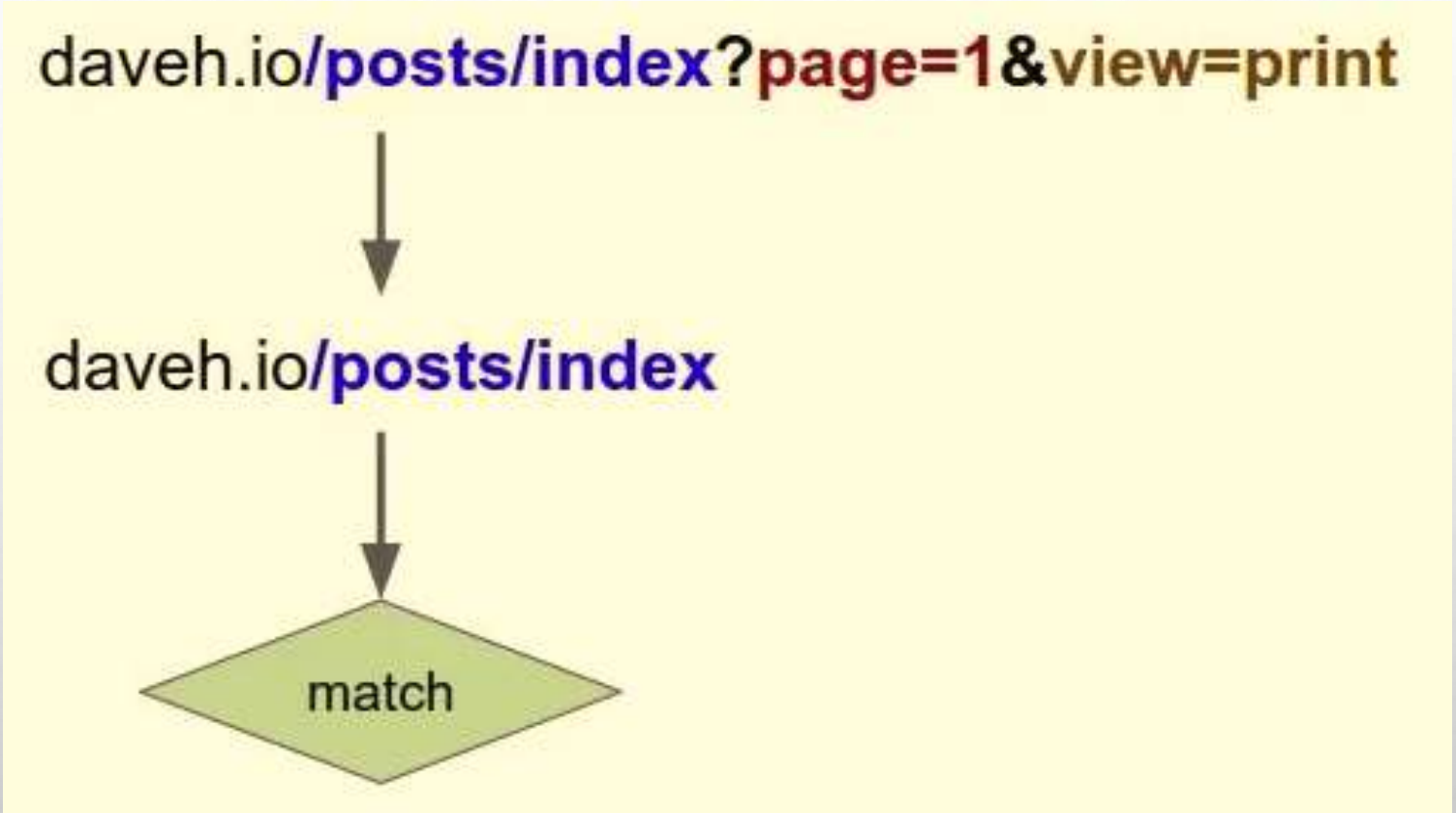


- Чому ?:

Routing using the query string

URL	$_SERVER['QUERY_STRING']	Route
localhost		
localhost/?		
localhost/?page=1	page=1	
localhost/posts?page=1	posts&page=1	posts
localhost/posts/index	posts/index	posts/index
localhost/posts/index?page=1	posts/index&page=1	posts/index

Removing query string variables from the URL



Файл Router.php

1)
```
protected function removeQueryStringVariables($url)    {
    if ($url != '') {
        $parts = explode('&', $url, 2);
        if (strpos($parts[0], '=') === false) {
            $url = $parts[0];
        } else {
            $url = '';
        }
    }
    return $url;
}
```

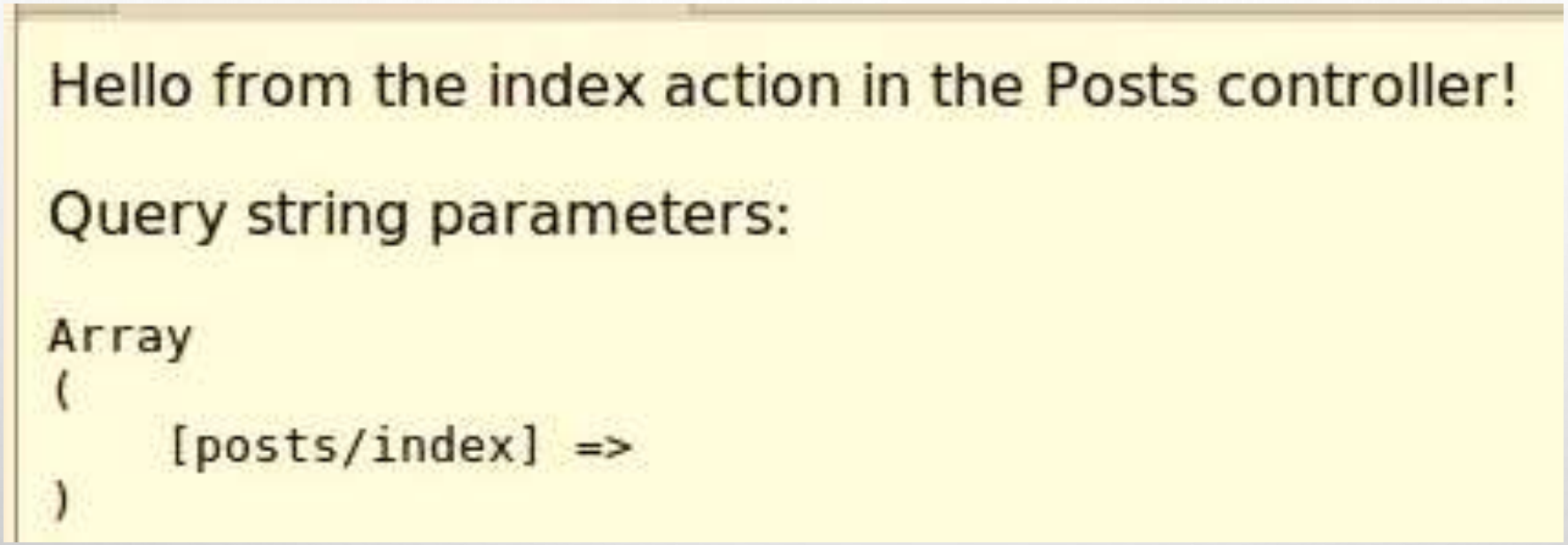
2) В методі dispatch вставити першим рядком:

```
$url = $this->removeQueryStringVariables($url);
```

Файл Posts.php

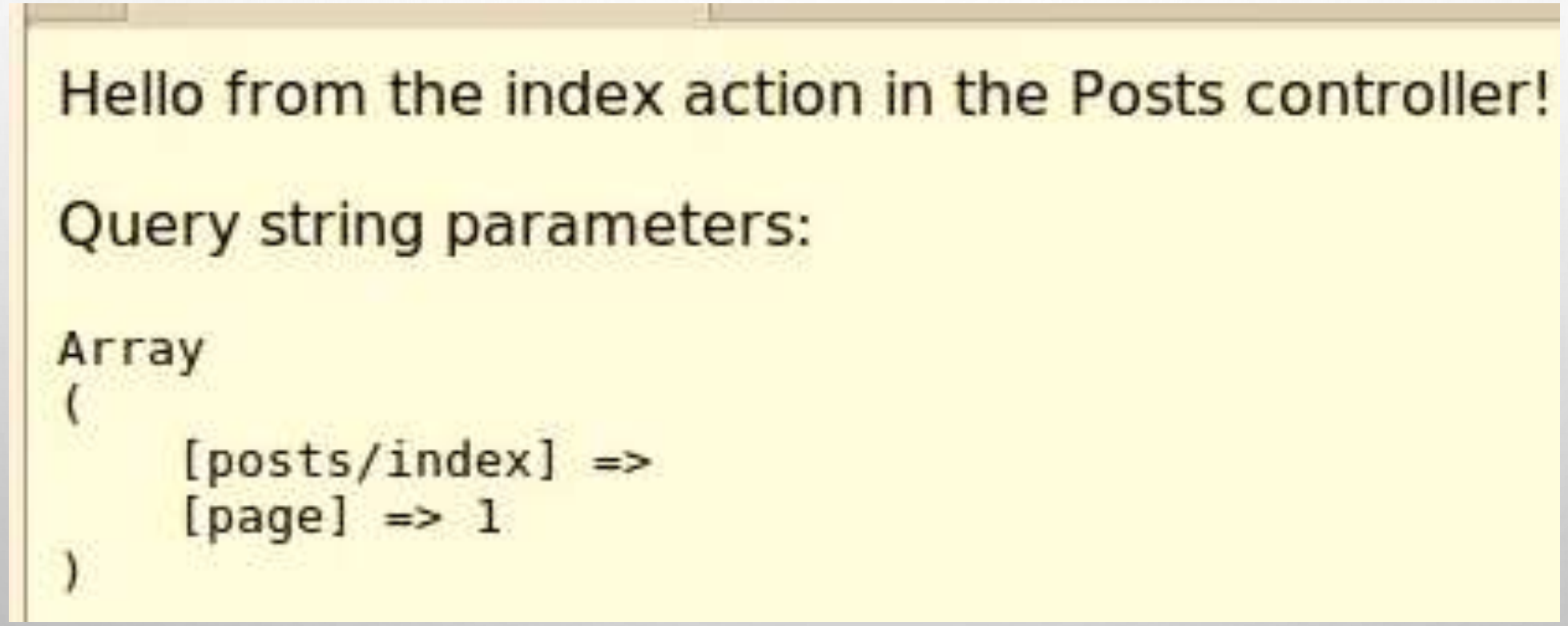
```
public function index()
{
    echo 'Hello from the index action in the Posts controller!';
    echo '<p>Query string parameters: <pre>'.
        htmlspecialchars(print_r($_GET, true)) . '</pre></p>';
}
```

Тестування http://localhost/posts/index
(для XAMPP http://localhost/mvcframewprk7/posts/index)



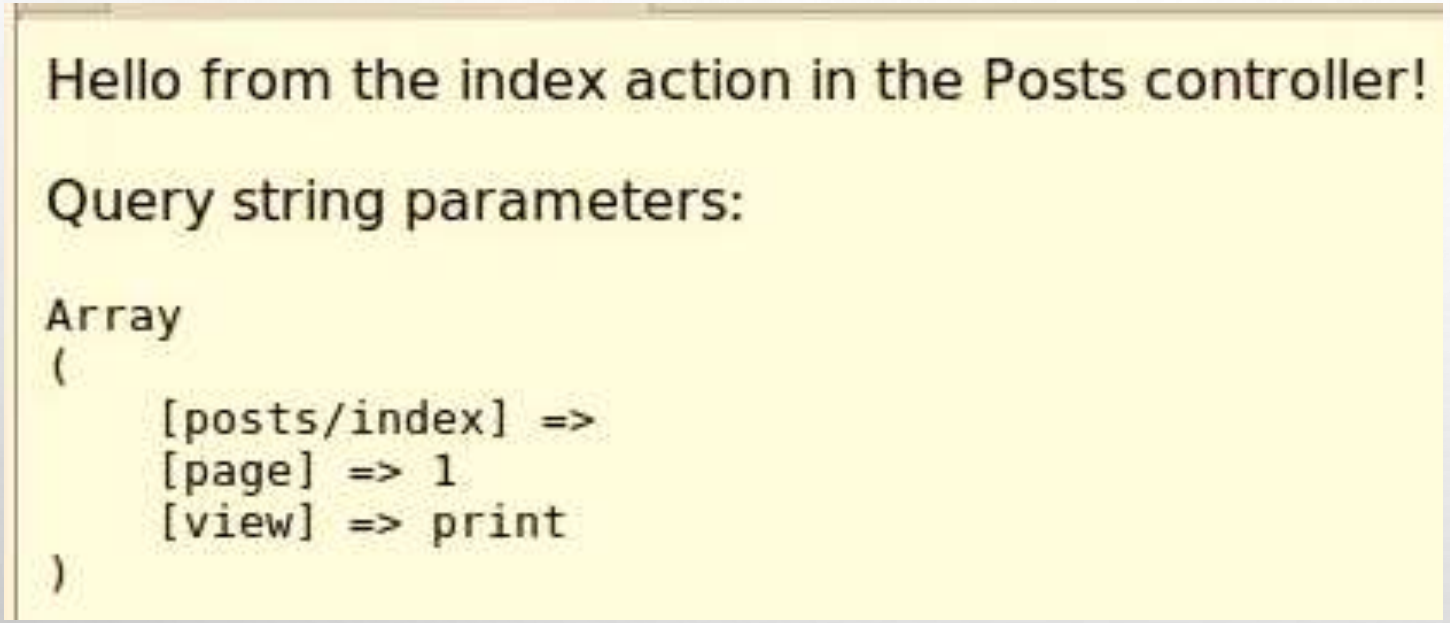
Тестування

http://localhost/posts/index?page=1



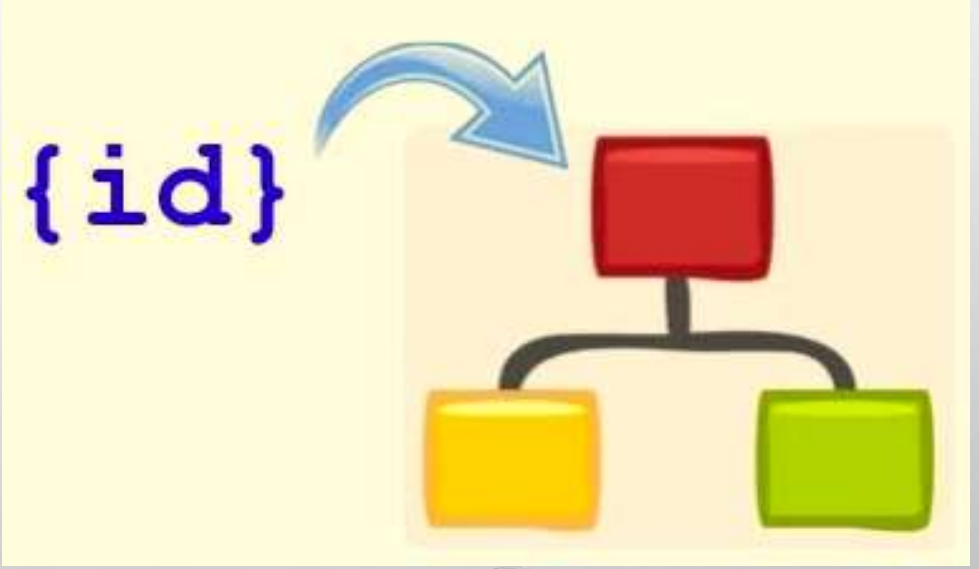
Тестування

http://localhost/posts/index?page=1&view=print



(Дивись приклад mvcframework7 в Moodle)

Pass route parameters from the route to all controllers



Custom route variables

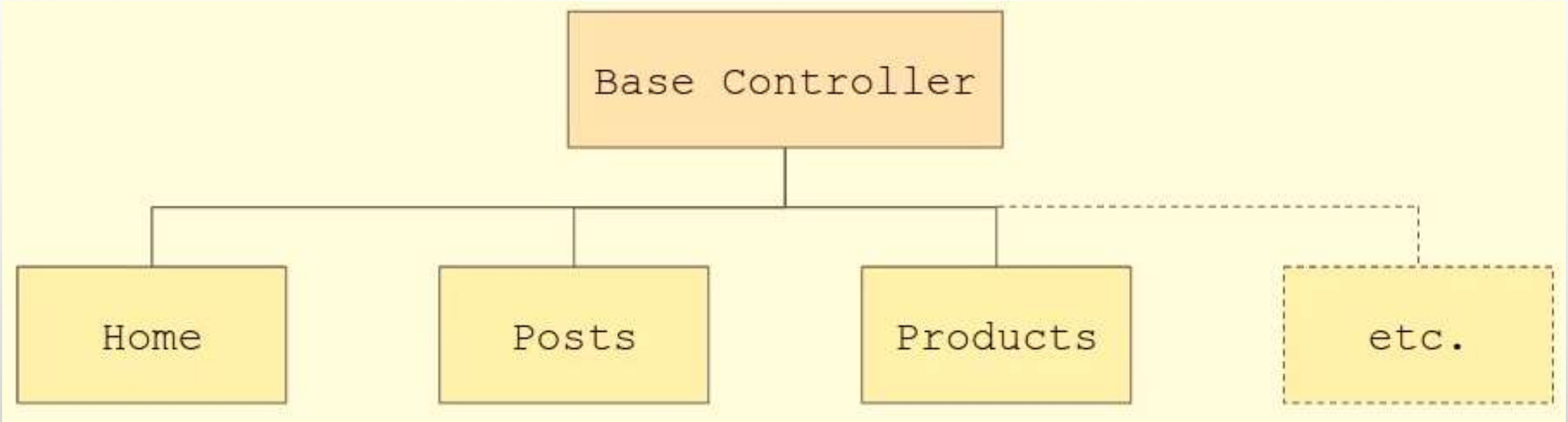
- Було раніше:

```
daveh.io/posts/123/edit
daveh.io/users/456/show
daveh.io/products/789/delete
```

```
$router->add('/{controller}/{id}/{action}');
```

- Щоб мати можливість доступу до цих змінних в контролері, нам необхідно передати їх до об'єктів контролерів.

Base controller class



- Давайте створимо таку ієрархію класів.

Каталог Core, файл Controller.php

```php
<?php    //  Base controller
  namespace Core;

  abstract class Controller
  {
    protected $route_params = []; // Parameters from the matched route

    public function __construct($route_params)   {
      $this->route_params = $route_params;
    }
  }
```

Каталог Controllers, файл Home.php

```
<?php     //Home controller
 namespace App\Controllers;
 class Home extends \Core\Controller
 {
   public function index()    {
     echo 'Hello from the index action in the Home controller!';
   }
 }
```

Каталог Controllers, файл Posts.php

- Зміни:

1) class Posts extends \Core\Controller
2) Додайте новий метод:

```
public function edit()    {
    echo 'Hello from the edit action in the Posts controller!';
    echo '<p>Route parameters: <pre>' .
    htmlspecialchars(print_r($this->route_params, true)) . '</pre></p>';
}
```

Каталог Core, файл Router.php

- Зміни в методі dispatch:

```
$controller_object = new $controller($this->params);
```

Тестування в браузері

- URL: http://localhost (для XAMPP http://localhost/mvcframework8/)

  Відповідь: Hello from the index action in the Home controller!

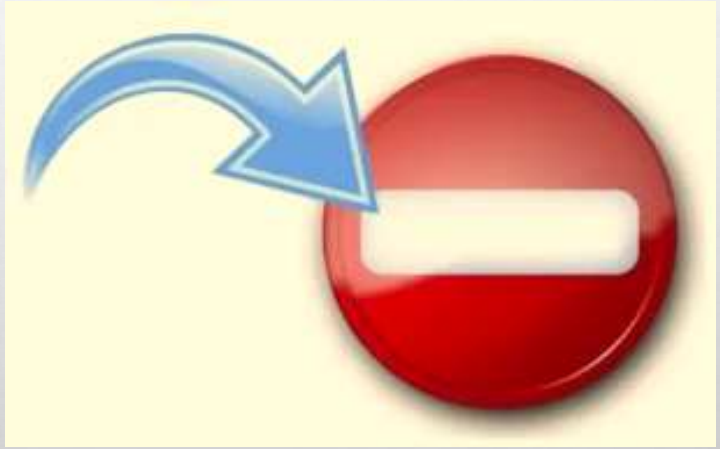
- URL: http://localhost/posts/123/edit

  Відповідь: Hello from the edit action in the Posts controller!

```
Route parameters:
Array
{
  [controller] => posts
  [id] => 123
  [action] => edit
}
```

(дивись приклад mvcframework8 в Moodle)

The `__call` magic method: how to call inaccessible methods in a class



Classes, objects and methods

- A class can have many **methods**
- **Public** methods can be run on **objects** of that class:

```
class Product
{
  public function save() {
  }
  public function load() {
  }
  private function modify() {
  }
}
```

```
$product = new Product();

$product->load();
$product->save();
```

daveh.io

Visibility

- `public` methods are available from **outside** the class
- `protected` and `private` methods are not available from **outside** the class

```
class Product
{
  public function save() {
  }
  private function modify() {
  }
}
```

```
$product = new Product();

$product->save();
$product->modify();
```

Fatal error: Call to private method Product::modify()

- Чи є спосіб доступу до цих недоступних методів?

The `__call` magic method

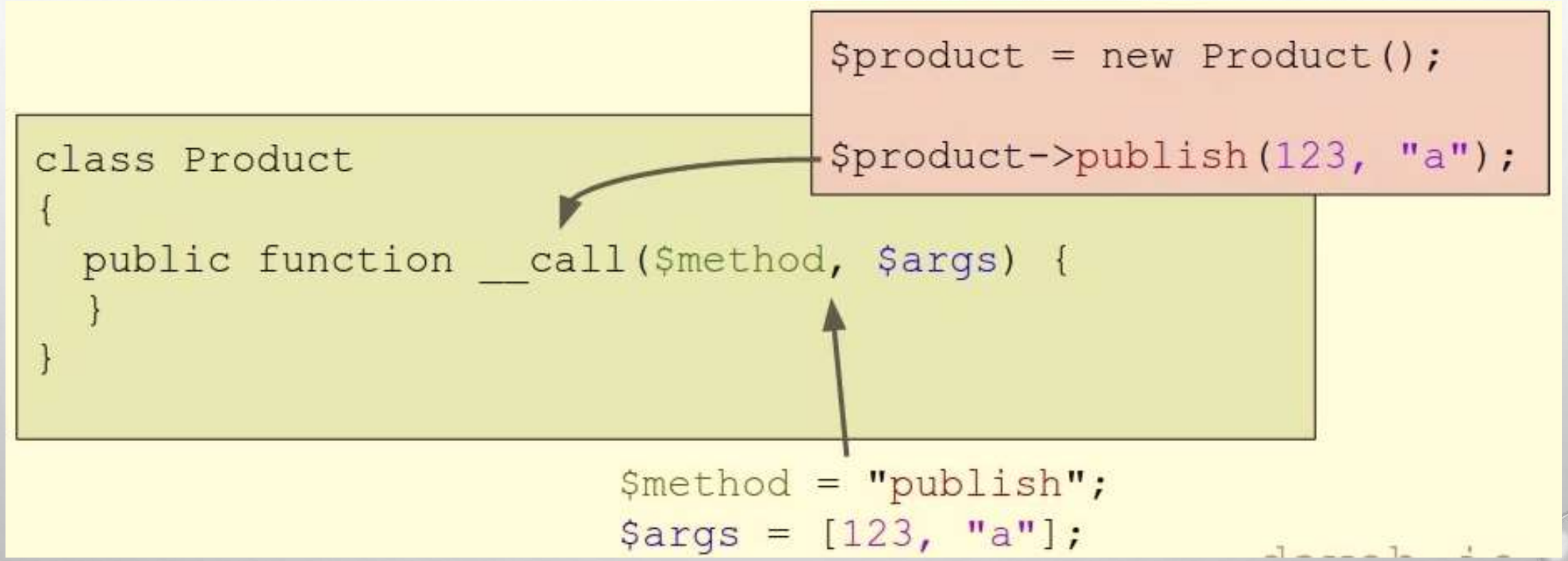
- `__call` is a PHP *magic method*
- Called whenever a **non-existent** or **non-public** method is called on an object

```
class Product
{
  public function __call($name, $arguments) {
  }
  private function modify() {
  }
}
```

```
$product = new Product();

$product->modify();
$product->publish();
```

Adding the `__call` magic method

- The method is passed the **name** of the method and the **arguments**



Adding the `__call` magic method

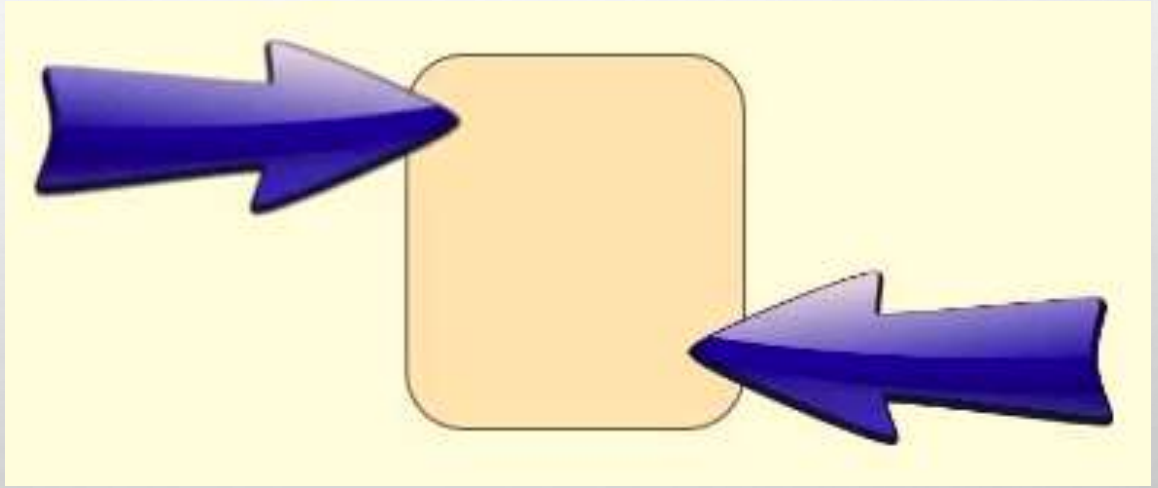
- The method is passed the **name** of the method and the **arguments**

```php
$product = new Product();

$product->publish(123, "a");
```

```php
class Product
{
  public function __call($method, $args) {
    call_user_func_array([$this, $method], $args);
  }
}
```

```php
$method = "publish";
$args = [123, "a"];
```

Action filters: call a method before and after every action in a controller



Controller actions

- A controller can have **many actions**:
- Actions will generally:
  - write out **content** (i.e. HTML)
  - **redirect** to another action
  - etc.

```
class Posts
{
  public function index()
  {
    // show all posts
  }
  public function show()
  {
    // show a single post
  }
}
```

Action filters

- How to execute some code **before** or **after** every action inside a controller?

For example:

- to check that a user has logged in, or has the correct permissions
- or to write a message to a log
- set the language

Ця функціональність присутня в більшості популярних фреймворків MVC і зазвичай називається фільтрами дій (action filters).

Filtering actions

- `__call` is executed for a **non-existent** or **non-public** method call
- By executing `__call` first, we could run code before and after a method, running the original method using the **call_user_func_array** function

```
class Posts
{
  public function __call($name, $args)
  {
    // run code before
    call_user_func_array([$this, $name], $args);
    // run code after
  }
}
```

Option 1: make the action methods private

```
class Posts
{
  private function index()
  {
    // show all posts
  }
  private function show()
  {
    // show a single post
  }
}
```

Option 2: add a suffix to the method name

```
class Posts
{
  public function indexAction()
  {
    // show all posts
  }
  public function showAction()
  {
    // show a single post
  }
}
```

- Тоді при виклику методу index він не буде існувати, і треба застосовувати __call
- Ми будемо застосовувати цей спосіб (як у популярних фреймворках Symfony, Phalcon)

Actions with a suffix

```php
$controller = new Posts();
$controller->index();
```

```php
class Posts
{
  public function __call($name, $args)
  {
    // run code before
    call_user_func_array([$this, "$nameAction"], $args);
    // run code after
  }
  public function indexAction()
  {
  }
}
```

Каталог Core, файл Controller.php

```
public function __call($name, $args)    {
    $method = $name . 'Action';
    if (method_exists($this, $method)) {
        if ($this->before() !== false) {
            call_user_func_array([$this, $method], $args);
            $this->after();
        }
    } else {
        echo "Method $method not found in controller " . get_class($this);
    }
}
protected function before()    {    }
protected function after()    {    }
```

Ще зміни в коді

- **Файл Posts.php**: додайте суфікс ‘Action’ до методів index, addNew, edit
- **Файл Home.php**:

1)
```
protected function before()    {
        echo "(before) ";
        //return false;
    }
    protected function after()    {
         echo " (after)";
    }
```

2) Додайте суфікс ‘Action’ до методу index

Тестування в браузері

- URL: http://localhost (для XAMPP http://localhost/mvcframework9/)

  Відповідь: (before) Hello from the index action in the Home controller! (after)

Розкоментуємо рядок в методі

```
protected function before()    {
    echo "(before) ";
    //return false;
}
```

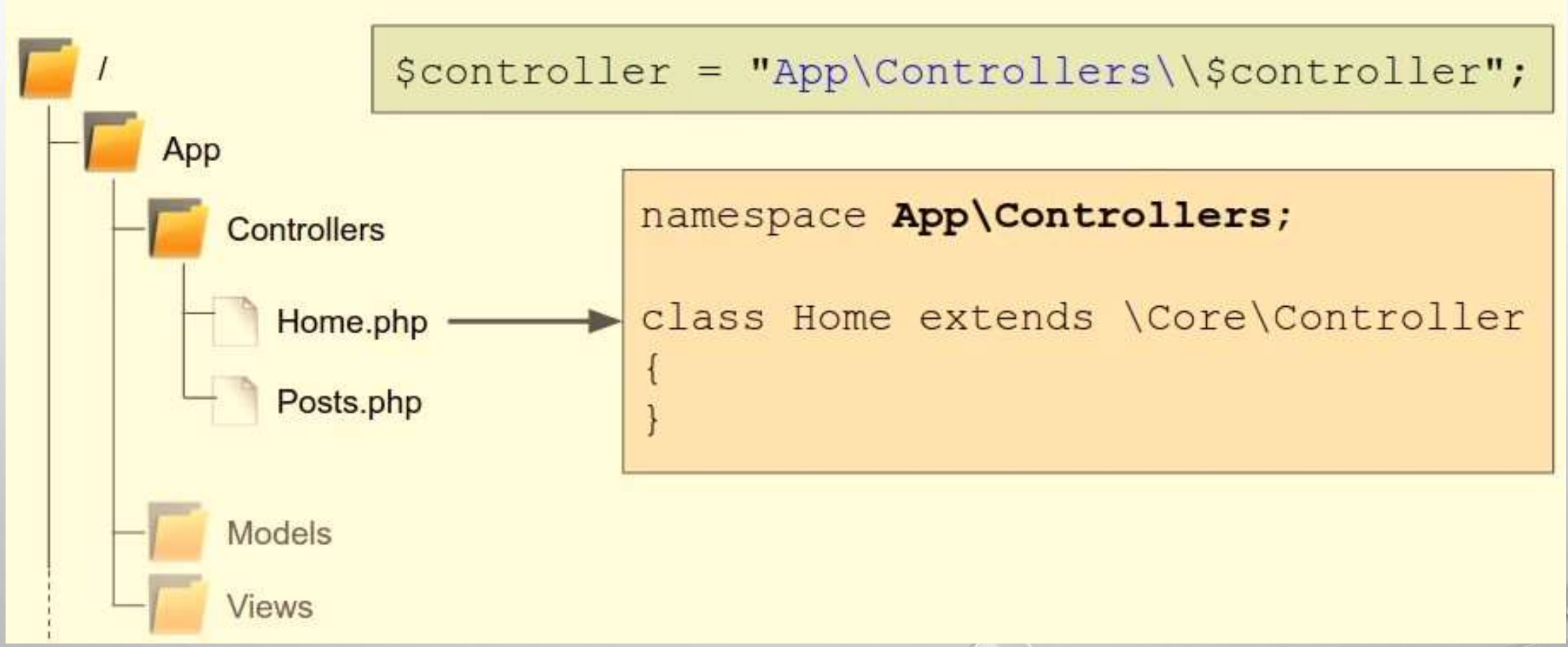
- URL: http://localhost (для XAMPP http://localhost/mvcframework9/)

  Відповідь: (before)

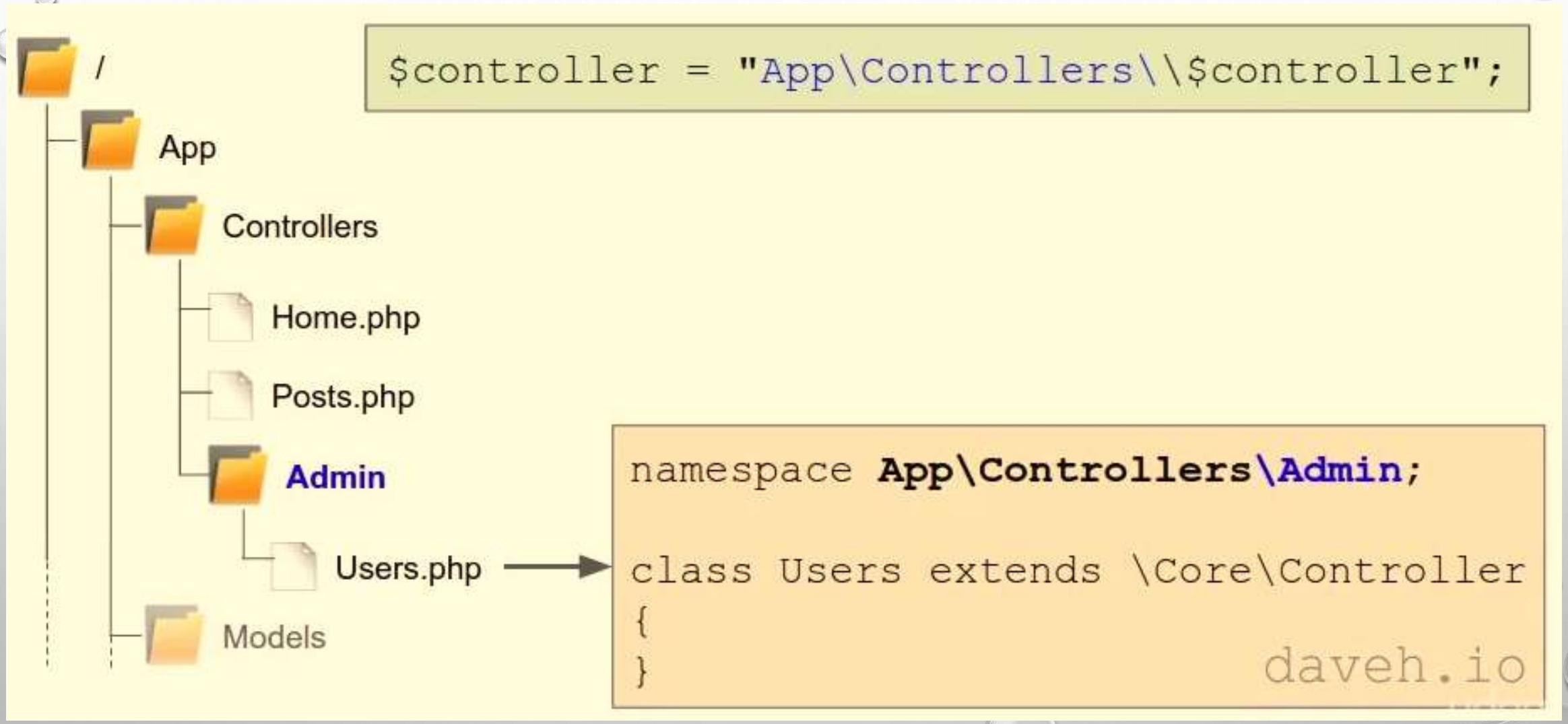
І це було б корисно, наприклад, для перевірки того, чи користувач увійшов або має правильні дозволи.

(Дивись приклад mvcframework9 в Moodle)

Організація контролерів в підкаталогах: додавання опції простору імен до маршруту



Controllers in subdirectories



Routes with namespace

- **Option** to specify the **namespace** in the route
- Defaults to App\Controllers is not specified

```
$router->add(
    '/admin/{controller}/{action}',
    ['namespace' => 'Admin']
);
```

Зміни в коді

- В каталозі Controllers створіть каталог Admin а в ньому – файл Users.php :

```
<?php
  namespace App\Controllers\Admin;
  class Users extends \Core\Controller
  {
     protected function before()
     {
     }
     public function indexAction()
     {
          echo 'User admin index';
     }
  }
```

Зміни в коді

- **Front Controller** (файл index.php):
  додайте маршрут
  $router->add('admin/{controller}/{action}', ['namespace' => 'Admin']);

- **Файл Router.php** : 1) додайте метод

```
protected function getNamespace()    {
    $namespace = 'App\Controllers\\';
    if (array_key_exists('namespace', $this->params)) {
        $namespace .= $this->params['namespace'] . '\\';
    }
    return $namespace;
}
```

2) в методі dispatch закоментуйте рядок

```
//$controller = "App\Controllers\\$controller"; і додайте рядок
$controller = $this->getNamespace() . $controller;
```

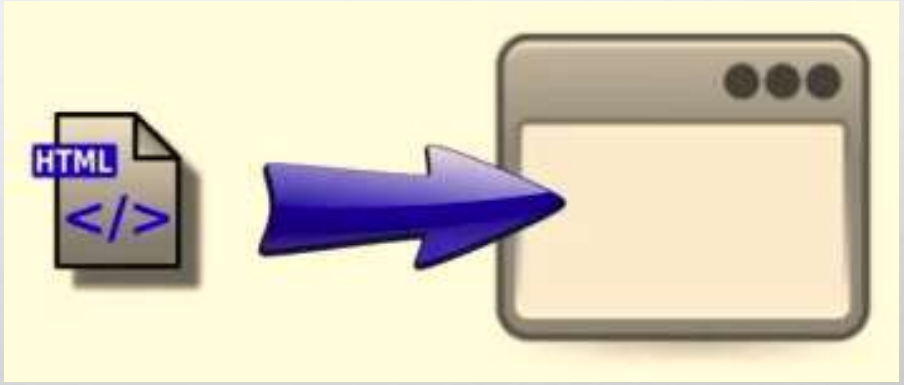
Тестування в браузері

- URL: http://localhost (для XAMPP http://localhost/mvcframework10/)
  Відповідь: (before) Hello from the index action in the Home controller! (after)

- URL: http://localhost/admin/users/index
  Відповідь: User admin index

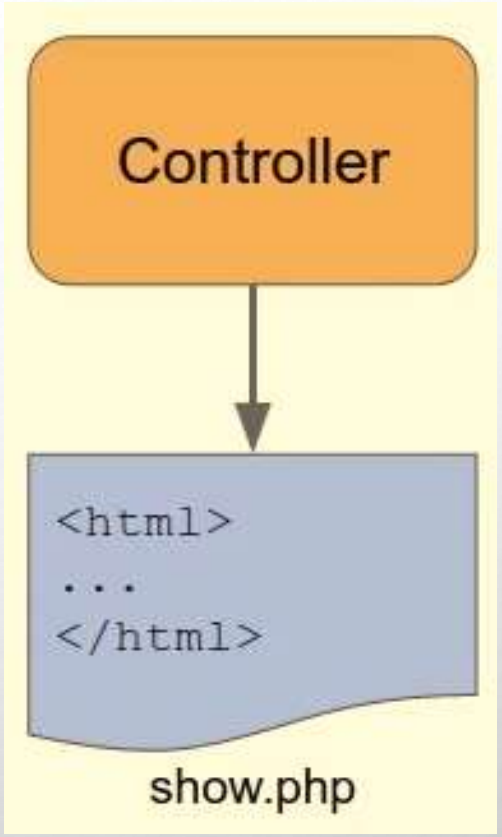
(Дивись приклад mvcframework10 в Moodle)

Display a view: create a class to render views and use it in a controller



Controllers and views

- The controller **doesn't write any output** (i.e. HTML)
- It loads and outputs a **view** file, which is what contains the content (HTML, JSON, XML etc.)



Views

- The view just **shows data**, so contains **minimum** amount of PHP: `echo`, `if`, `for`, **etc.**
- Has **no knowledge** of models, sessions, databases etc.

```
<!DOCTYPE html>
<html>
<body>
<h1>Welcome</h1>
<?php echo $message; ?>
</body>
</html>
```

App/Views/Home/index.php

Каталог Core, файл View.php

```
<?php
  namespace Core;
  class View
  {
     public static function render($view)    {
         $file = "../App/Views/$view";       // relative to Core directory
         if (is_readable($file)) {
              require $file;
         } else {
               echo "$file not found";
         }
     }
  }
```

Зміни в файлі Home.php

1) додати рядок

```
use \Core\View;
```

2) в методі indexAction:
   - Закоментувати рядок //echo ‘Hello from . . .
   - Додати рядок

```
View::render('Home/index.php');
```

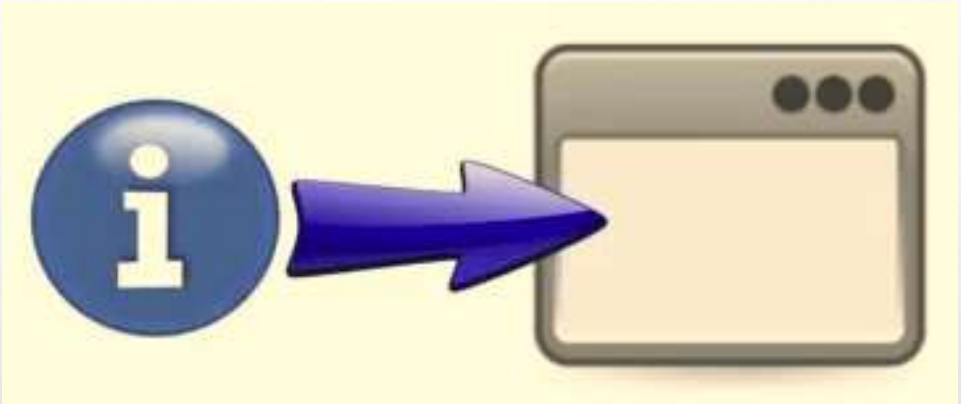
Новий файл Views\Home\index.php

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Home</title>
</head>
<body>
    <h1>Welcome</h1>
    <p>Hello from the view!</p>
</body>
</html>
```

Тестування в браузері: URL: http://localhost (для XAMPP http://localhost/mvcframework11/)

(Дивись приклад mvcframework11 в Moodle)

Pass data from the controller to the view



Extracting variables from an array

```
<html>
<?php echo $name; ?>
<?php echo $colour; ?>
</html>
```

```
$data = [
  'name' => 'Dave',
  'colour' => 'green'
];
```

```
extract($data);
```

```
$name   → 'Dave'
$colour → 'green'
```

Зміни у файлі Core\View.php

```
public static function render($view, $args = [] )
{
    extract($args, EXTR_SKIP);
     . . . .
```

Зміни у файлі App\Controllers\Home.php

```
public function indexAction()
{
    View::render('Home/index.php', [
        'name'    => 'Dave',
        'colours' => ['red', 'green', 'blue']
    ]);
}
```

Зміни в файлі App\Views\Home\index.php

```
<!DOCTYPE html>
<html>
<head> <meta charset="UTF-8"><title>Home</title></head>
 <body>
    <h1>Welcome</h1>
    <p>Hello <?php echo htmlspecialchars($name); ?>!</p>
   <ul>
       <?php foreach ($colours as $colour): ?>
           <li><?php echo htmlspecialchars($colour); ?></li>
       <?php endforeach; ?>
   </ul>
 </body>
 </html>
```

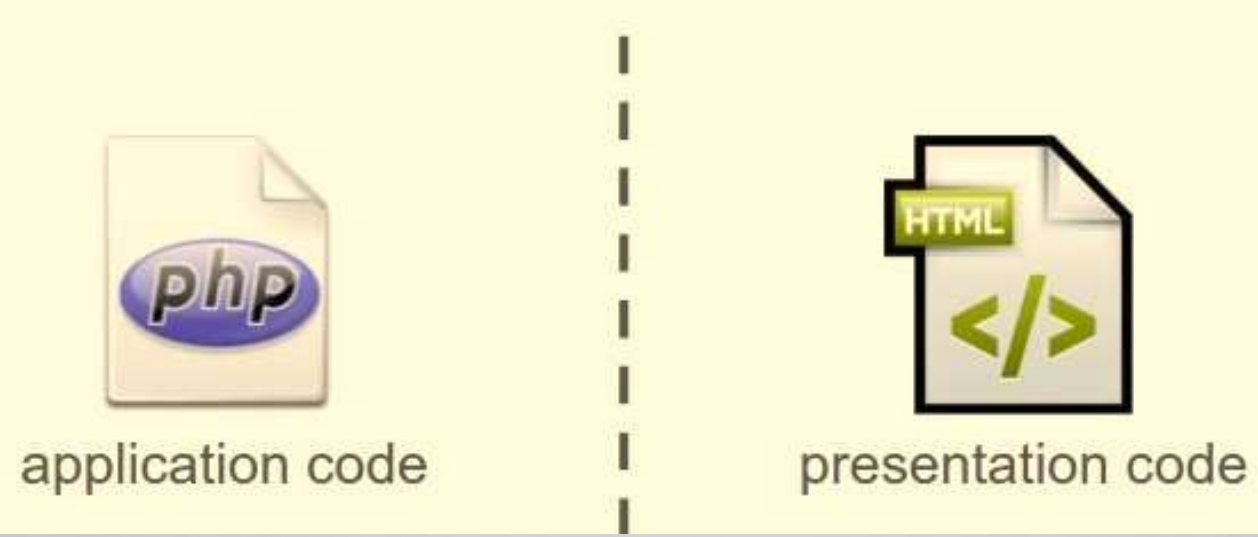
Тестування в браузері

(Дивись приклад mvcframework12 в Moodle)

Templating engines: what they are, and how they can improve your PHP code

What is a template engine?

- Tool that helps to separate **application code** from **presentation code.**
- Templates (views) contain **no PHP** at all: just **HTML** and **simple tags** to show the data.



What are the advantages?

- Simpler, easier syntax → easier to code and read, less prone to errors
- Autoescaping of variables → more secure
- Template inharitance → simpler templates, easier to maintain
- No PHP in the templates → designers don’t need to know PHP

Simpler syntax: displaying variables

- Для шаблонізатора Twig:

```
<p><?php echo $name; ?></p>
```

```
<p>{{ name }}</p>
```

Simpler syntax: displaying collections

```
<?php if ( ! empty($posts)): ?>
  <?php foreach ($posts as $post): ?>
    Show post...
  <?php endforeach; ?>
<?php else: ?>
  No posts.
<?php endif; ?>
```

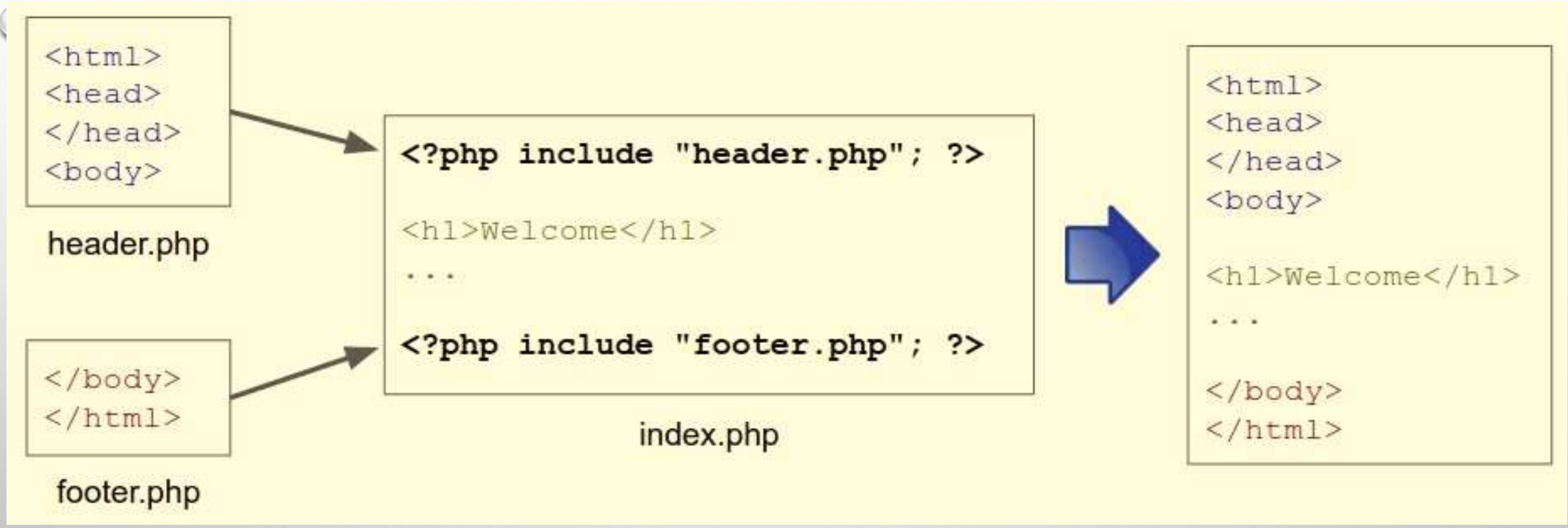
```
{% for post in posts %}
  Show post...
{% else %}
  No posts.
{% endfor %}
```

Autoescaping of variables

```
<p><?php echo htmlspecialchars($name); ?></p>
```

```
<p>{{ name }}</p>
```

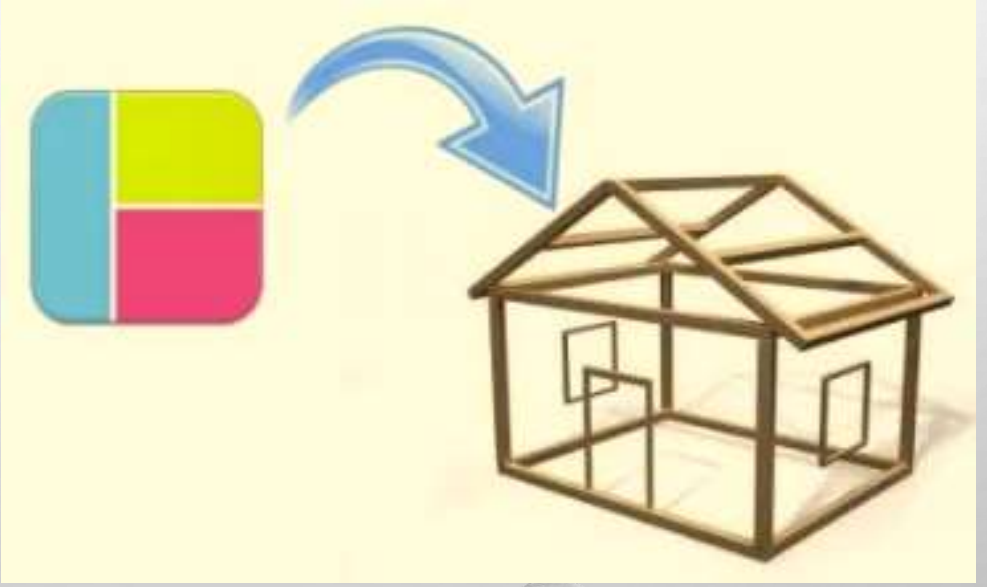
Including a common header and footer



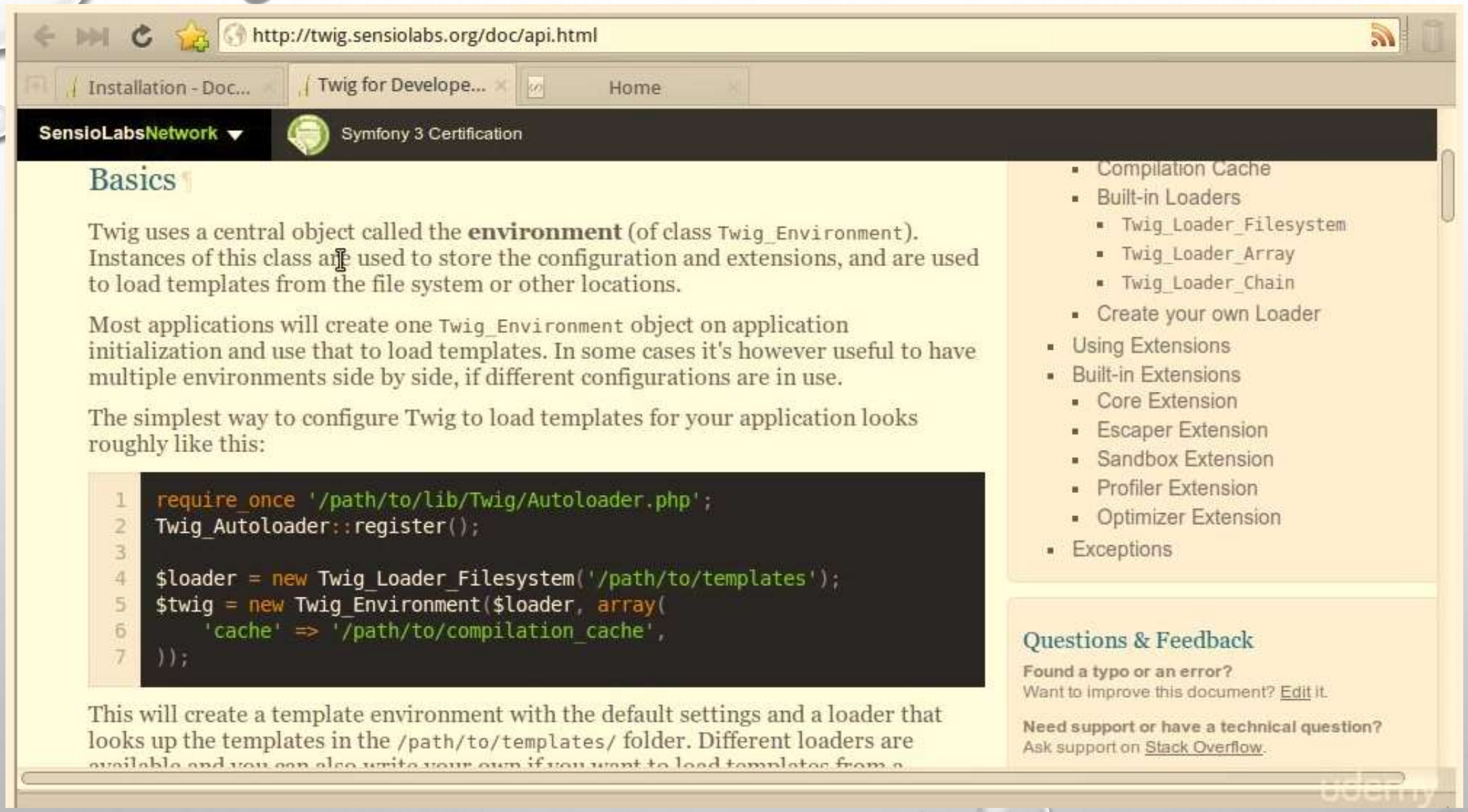
PHP template engines

- Twig http://twig.sensiolabs.org/
- Smarty http://www.smarty.net/
- Blade (Laravel) https://laravel.com/docs/blade
- Volt (Phalcon) https://docs.phalconphp.com/en/latest/reference/volt.html

Make views easier to create and maintain: add a template engine




http://twig.sensiolabs.org/doc/installation.html
Installation - Doc...
Twig for Develope...
Home
SensioLabsNetwork
SensioLabsInsight: regain control of your PHP applications
Installation
You have multiple ways to install Twig.
Installing the Twig PHP package
Installing via Composer (recommended)
Install Composer and run the following command to get the latest version:
composer require twig/twig:~1.0
Installing from the tarball release
Download the most recent tarball from the download page
Verify the integrity of the tarball http://fabien.potencier.org/article/73/signing-project-releases
Unpack the tarball
Table of Contents
Installation
Installing the Twig PHP package
Installing via Composer (recommended)
Installing from the tarball release
Installing the development version
Installing the PEAR package
Installing the C extension
Questions & Feedback
Found a typo or an error?
Want to improve this document? Edit it.
Need support or have a technical question?
Ask support on Stack Overflow.
License


http://twig.sensiolabs.org/doc/api.html
Installation - Doc...
Twig for Develope...
Home
SensioLabsNetwork
Symfony 3 Certification
Basics
Twig uses a central object called the environment (of class Twig_Environment). Instances of this class are used to store the configuration and extensions, and are used to load templates from the file system or other locations.
Most applications will create one Twig_Environment object on application initialization and use that to load templates. In some cases it's however useful to have multiple environments side by side, if different configurations are in use.
The simplest way to configure Twig to load templates for your application looks roughly like this:
require_once '/path/to/lib/Twig/Autoloader.php';
Twig_Autoloader::register();
$loader = new Twig_Loader_Filesystem('/path/to/templates');
$twig = new Twig_Environment($loader, array(
'cache' => '/path/to/compilation_cache',
));
This will create a template environment with the default settings and a loader that looks up the templates in the /path/to/templates/ folder. Different loaders are
Compilation Cache
Built-in Loaders
Twig_Loader_Filesystem
Twig_Loader_Array
Twig_Loader_Chain
Create your own Loader
Using Extensions
Built-in Extensions
Core Extension
Escaper Extension
Sandbox Extension
Profiler Extension
Optimizer Extension
Exceptions
Questions & Feedback
Found a typo or an error?
Want to improve this document? Edit it.
Need support or have a technical question?
Ask support on Stack Overflow.

Підключення Twig

1) Для спрощення: завантажили Twig і помістили в каталог **vendor**

2) Потім у фронт контролері (файл public**index.php**) додамо автозавантаження Twig:

```
require_once dirname(__DIR__) . '/vendor/Twig/lib/Twig/Autoloader.php';
Twig_Autoloader::register();
```

3) В файлі **Core\View.php** додамо метод renderTemplate (буде на наступному слайді)

Підключення Twig

```
//Файл Core\View.php
public static function renderTemplate($template, $args = [])
{
    static $twig = null;
    if ($twig === null) {
        $loader = new \Twig_Loader_Filesystem('../App/Views');
        $twig = new \Twig_Environment($loader);
    }
    echo $twig->render($template, $args);
}
```

Підключення Twig

```
//Файл Controllers\Home.php

public function indexAction()
{
    /*      View::render('Home/index.php', [
        'name'    => 'Dave',
        'colours' => ['red', 'green', 'blue']
    ]);
    */
    View::renderTemplate('Home/index.html', [
        'name'    => 'Dave',
        'colours' => ['red', 'green', 'blue']
    ]);
}
```

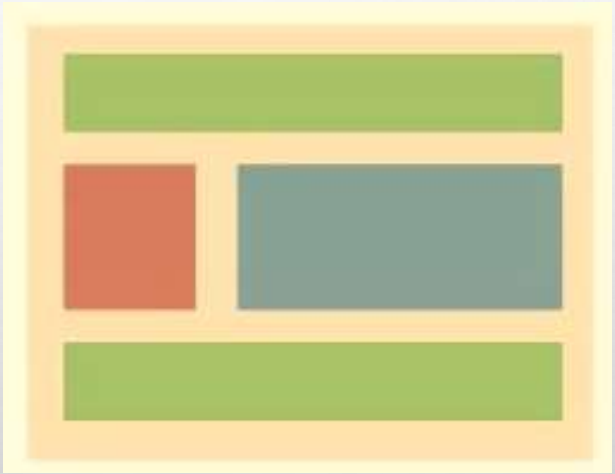
Підключення Twig

```
// Новий файл Views\Home\index.html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Home</title>
</head>
<body>
    <h1>Welcome</h1>
    <p>Hello from a Twig template, {{ name }}!</p>
    <ul>
        {% for colour in colours %}
          <li>{{ colour }}</li>
        {% endfor %}
    </ul>
</body></html>
```

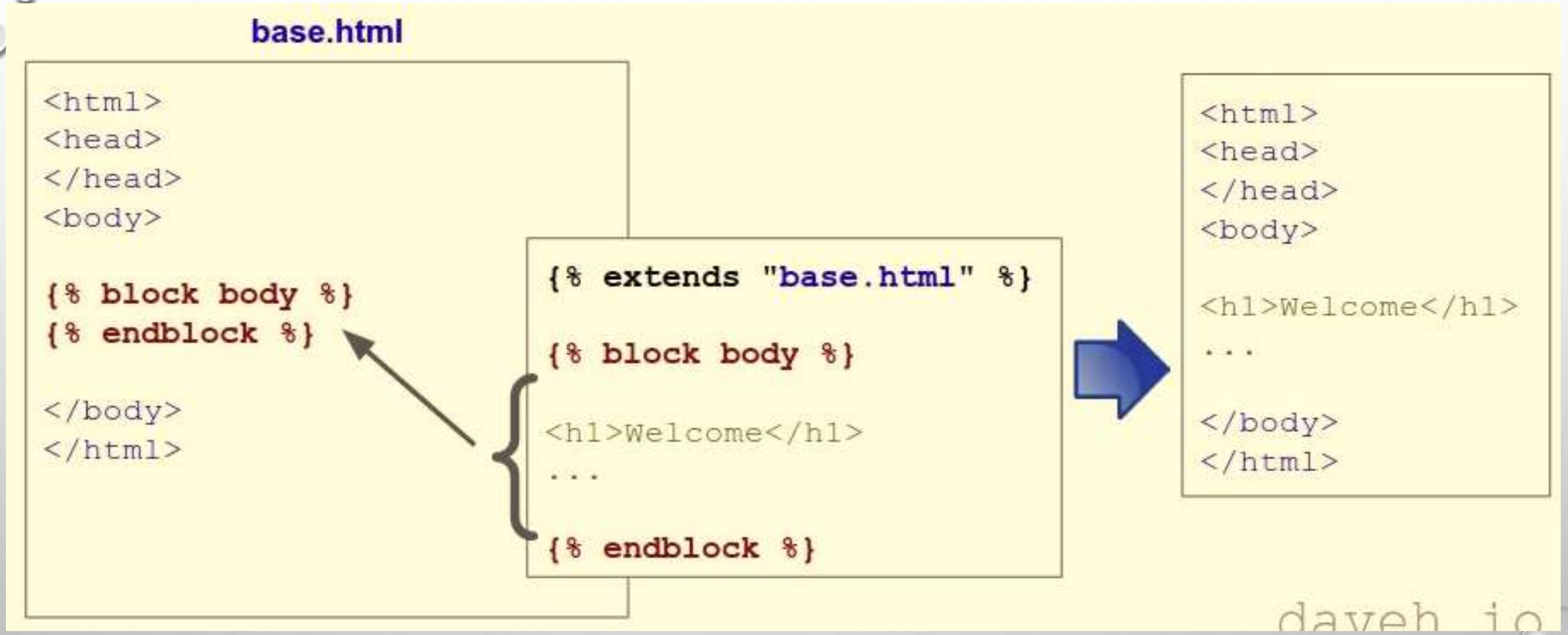
Тестування в браузері: URL: http://localhost (для XAMPP http://localhost/mvcframework13/)

(Дивись приклад mvcframework13 в Moodle)

Remove repetition in the view templates: add a base template to inherit from



Template inheritance



Новий файл Views\base.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    <nav>
        <a href="/">Home</a> |
        <a href="/posts/index">Posts</a>
    </nav>
    {% block body %}
    {% endblock %}
</body></html>
```

Файл Views\Home\index.html

```
{% extends "base.html" %}
{% block title %}Home{% endblock %}
{% block body %}
    <h1>Welcome</h1>
    <p>Hello from a Twig template, {{ name }}!</p>
    <ul>
        {% for colour in colours %}
            <li>{{ colour }}</li>
        {% endfor %}
    </ul>
{% endblock %}
```

Новий файл Views\Posts\index.html

```
{% extends "base.html" %}
{% block title %}Posts{% endblock %}
{% block body %}
    <h1>Posts</h1>
{% endblock %}
```

Зміни в файлі Controlles\Posts.php

1) Додати рядок

```
use \Core\View;
```

2)

```
public function indexAction()
{
  //echo 'Hello from the index action in the Posts controller!';
  View::renderTemplate('Posts/index.html');
}
```

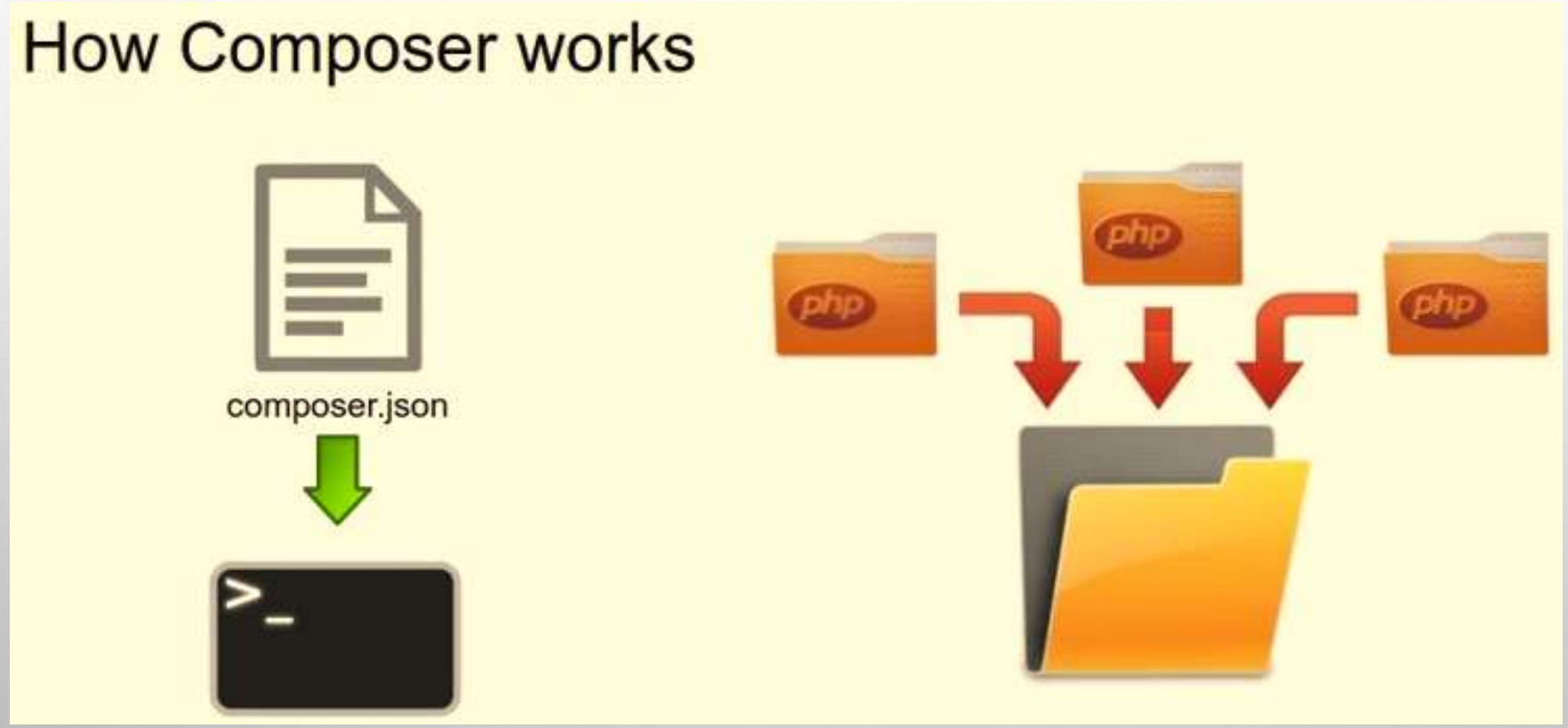
Тестування в браузері

- Дивись приклад mvcframework14 в Moodle

Install third-party PHP code libraries automatically using Composer

Composer – Dependency manager for PHP

- https://getcomposer.org/



The composer.json file

```
{
  "require": {
    "phpmailer/phpmailer": "~5.2",
    "twig/twig": "~1.0",
    "monolog/monolog": "1.0.*"
  }
}
```

Налаштування та робота з Composer

1) Завантажити Composer з сайту і установити.

2) Перевірити, чи Composer працює: запустити консоль і в командному рядку набрати: `composer about`

3) Через консоль зайти в каталог проекту (за допомогою команд `cd`) і створити файл `composer.json` з переліком всіх необхідних пакетів (бібліотек)

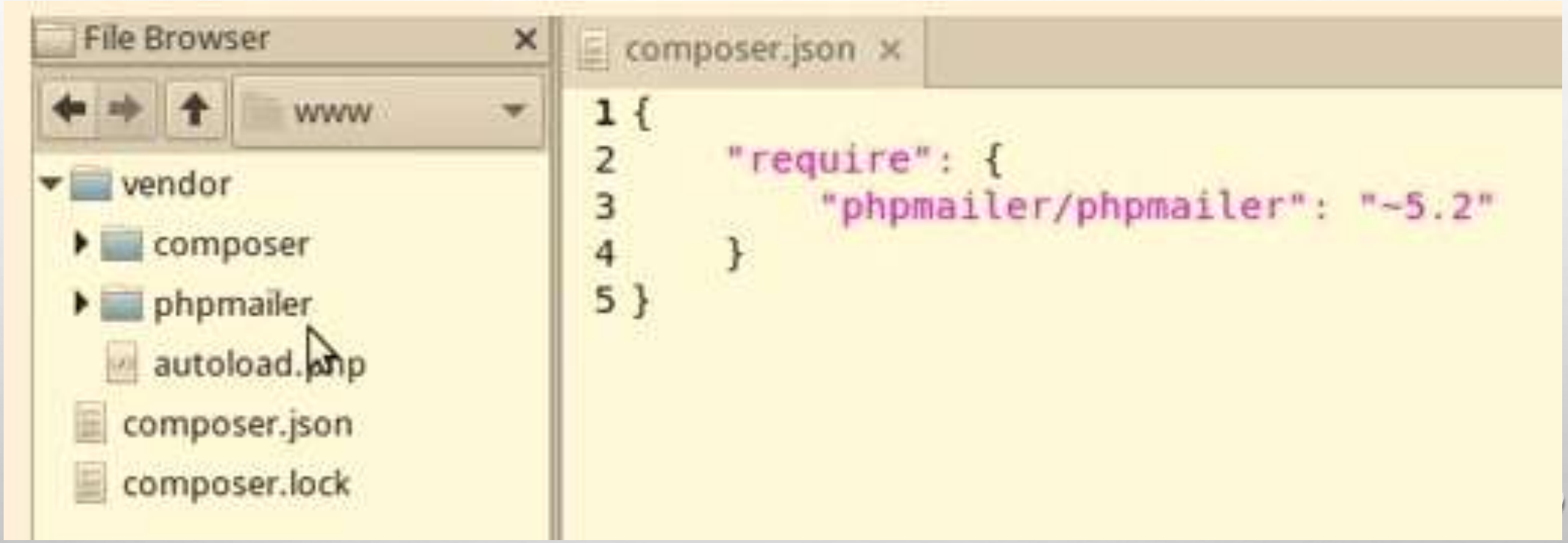
4) Приєднати (інсталювати) до проекту всі необхідні бібліотеки, подавши команду `composer install`

5) Як альтернативний варіант, можна приєднувати пакети до `composer.json` командами подібними до такої:

```
composer require "twig/twig: ~1.0"
```

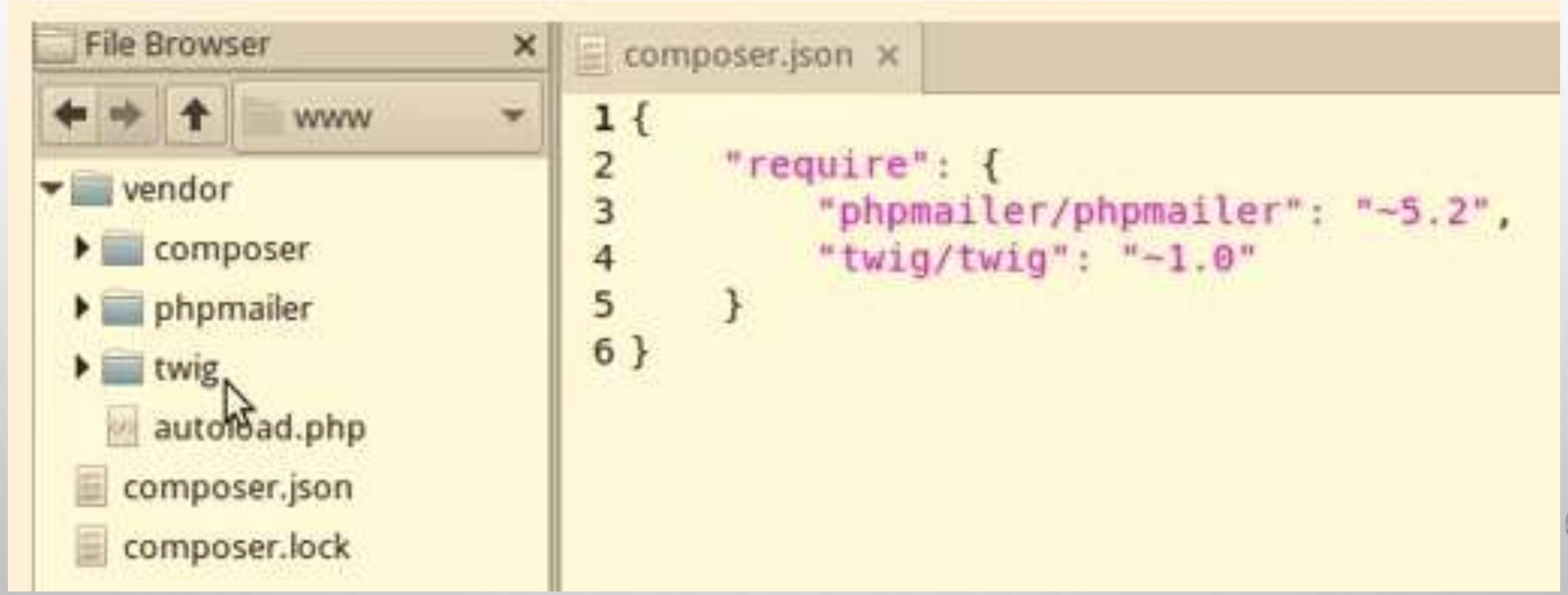
(якщо `composer.json` не існує, то він буде створений)

Приклад: після команди `composer install`

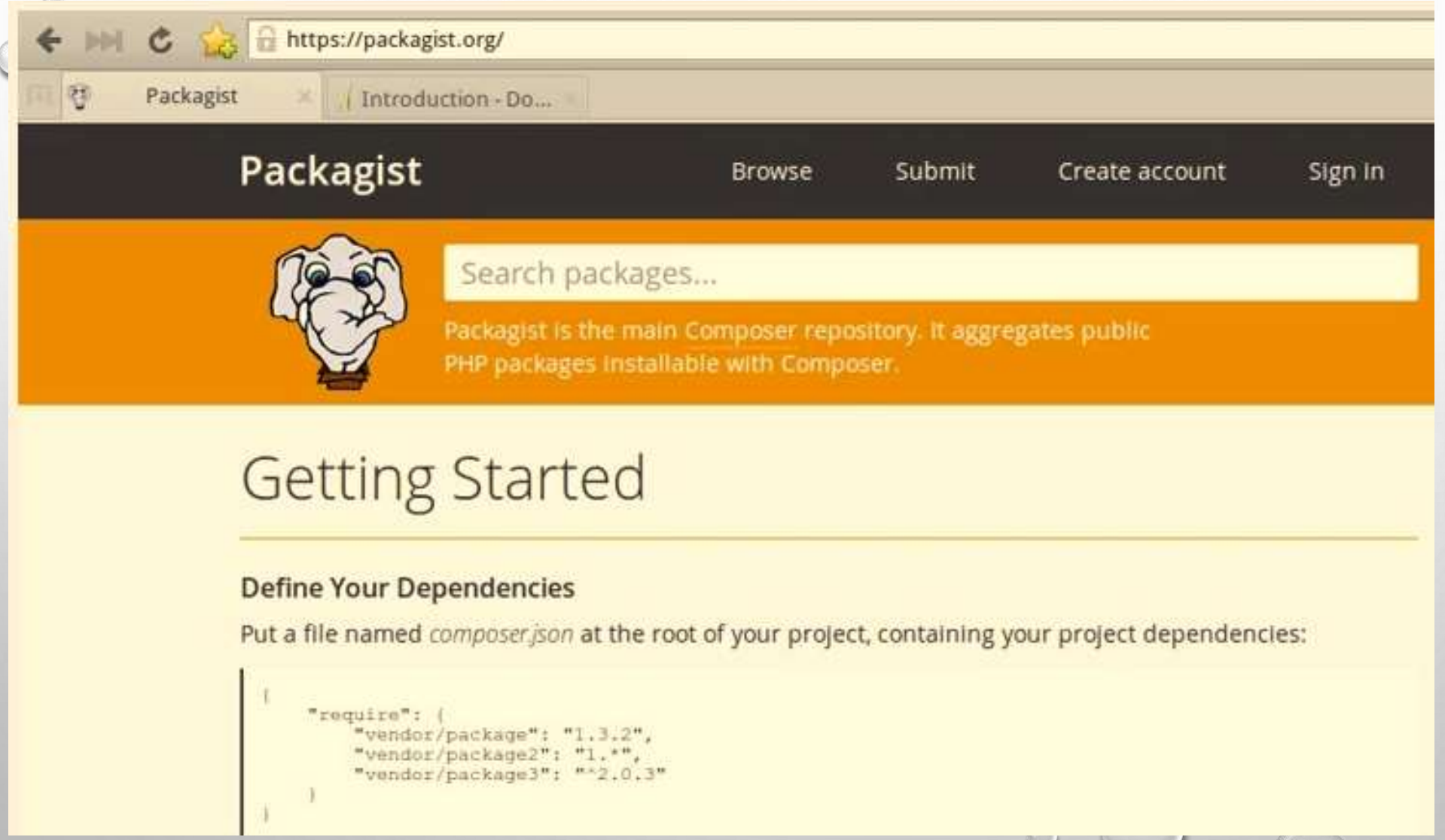


Приклад: після команди

`composer require “twig/twig: ~1.0”`



Де шукати пакети?



Install the template engine library using Composer

Треба виконати наступні дії:

1) З каталогу vendor видалити каталог Twig.
2) Через консоль (перебуваючи в кореневому каталозі проекту) подати команду
   `composer require "twig/twig: ~1.0"`
3) В результаті буде створено файл `composer.json` і інстальовано пакет Twig.
4) Виконати редагування рядка у фронт контролері (файл public\index.php)

require_once dirname(__DIR__) . '/vendor/twig/twig/lib/Twig/Autoloader.php'; :

5) Перевірити в браузері, чи все працює.

Use the Composer autoloader to load the template engine library

Виконайте наступні дії:

- Видаліть каталог Twig
- У фронт контролері (файл public\index.php) замініть рядок

require_once dirname(__DIR__) . '/vendor/twig/twig/lib/Twig/Autoloader.php';

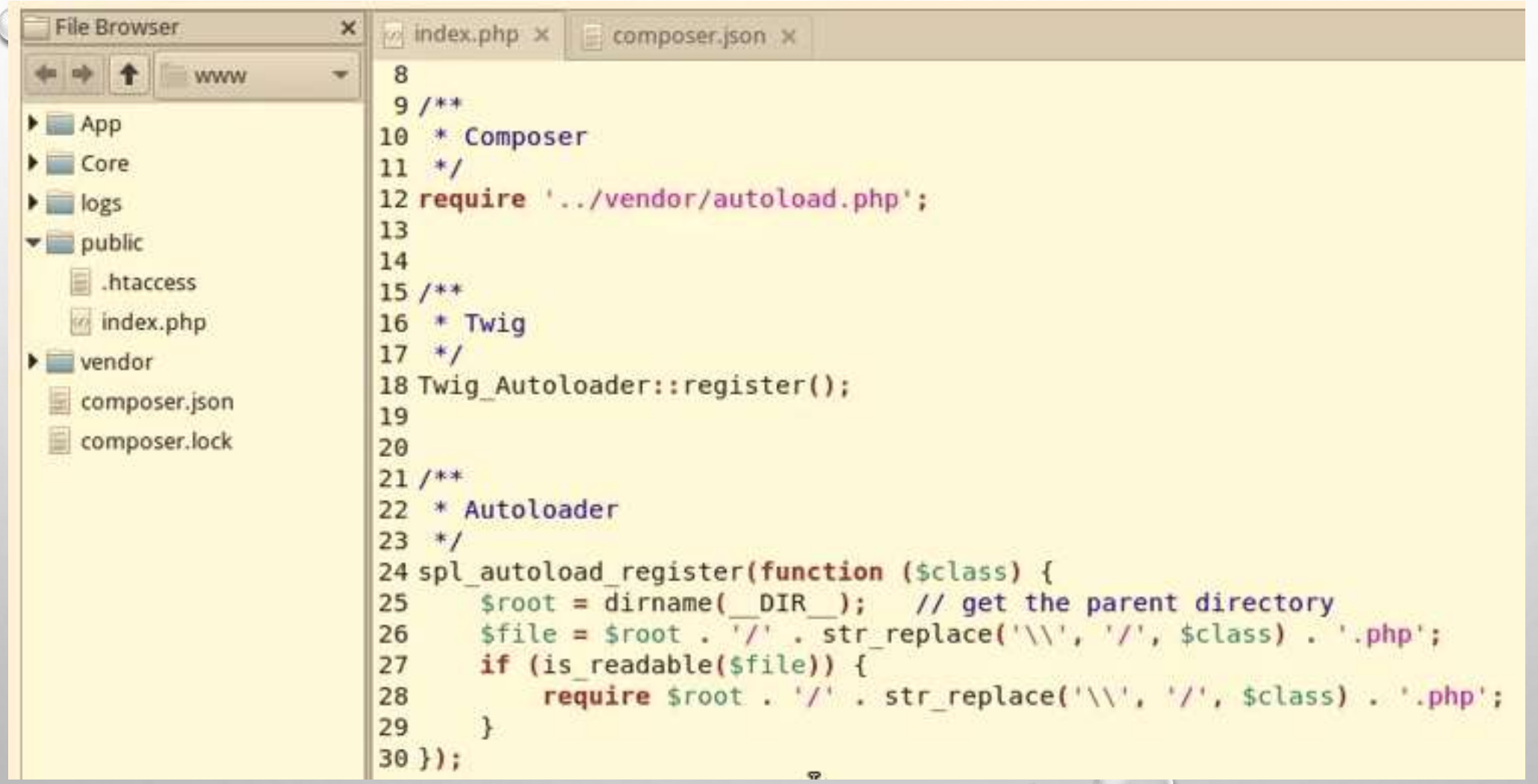
на рядок

```
require '../vendor/autoload.php';
```

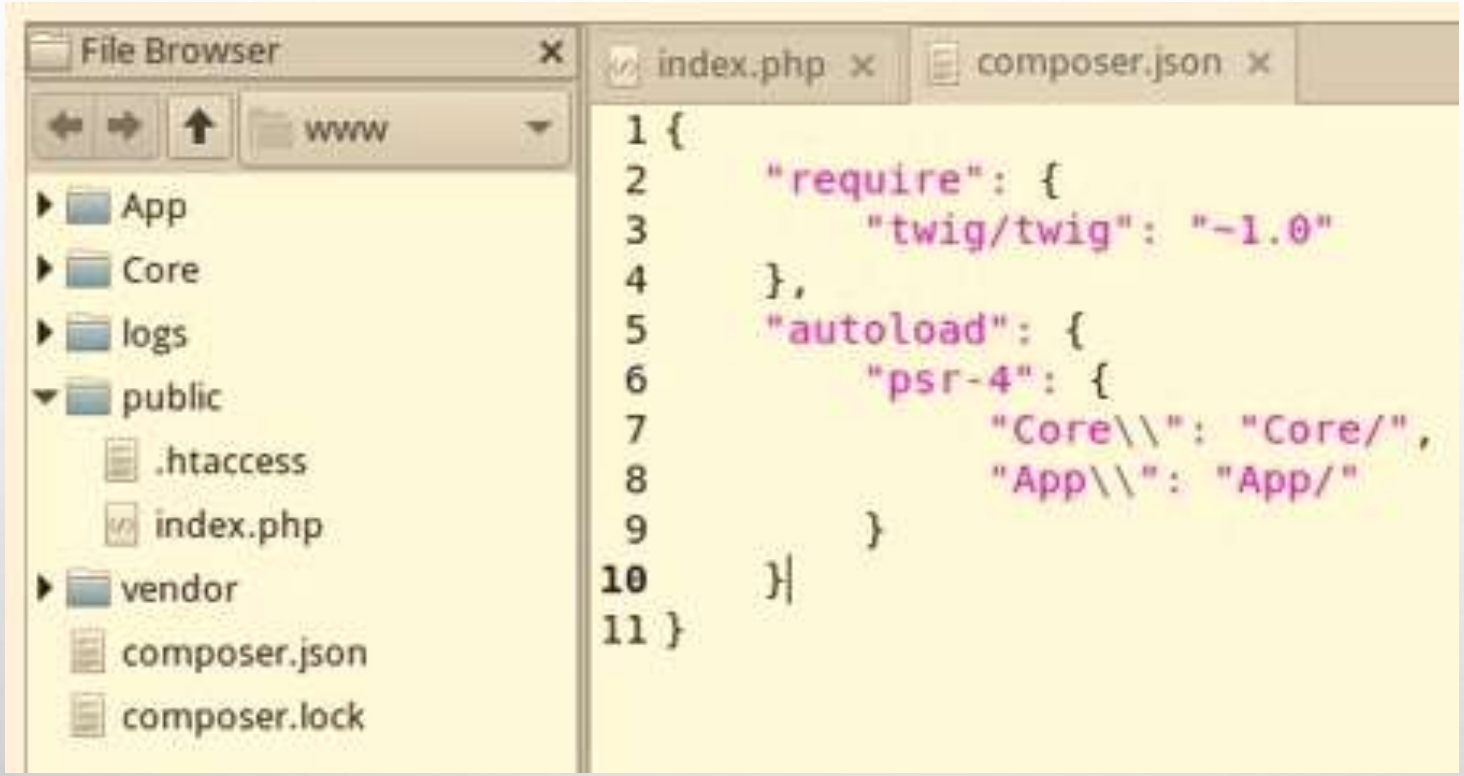
- Перевірте в браузері, чи все працює.
- Далі можна ще приєднувати інші пакети, не змінюючи файл index.php, всю роботу виконає автолоадер програми Composer.

Заміна функції автозавантаження на Composer autoloader

Зараз класи завантажує функція

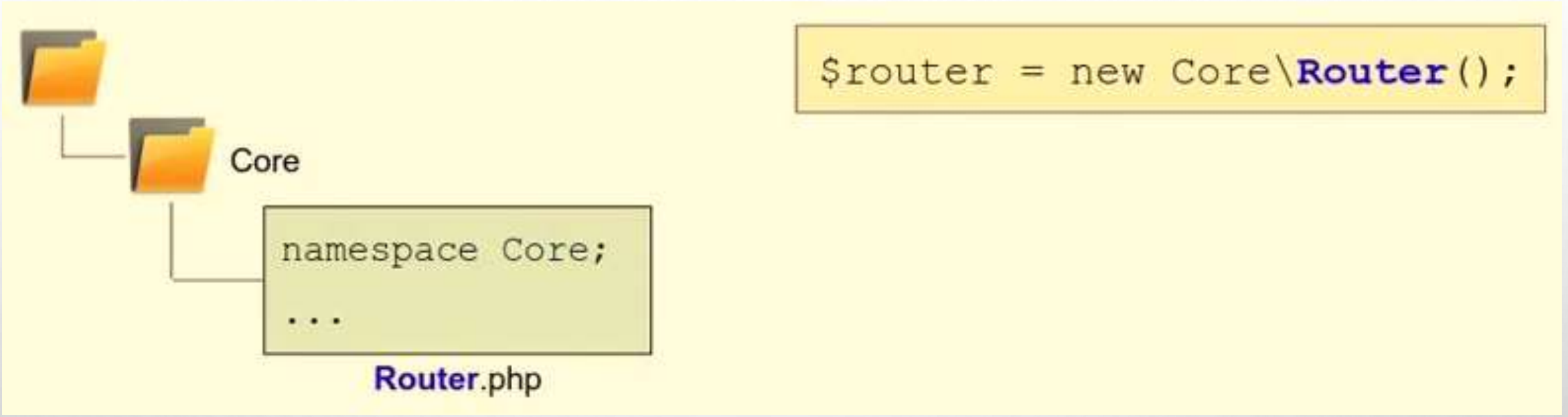


Змініть файл `composer.json`



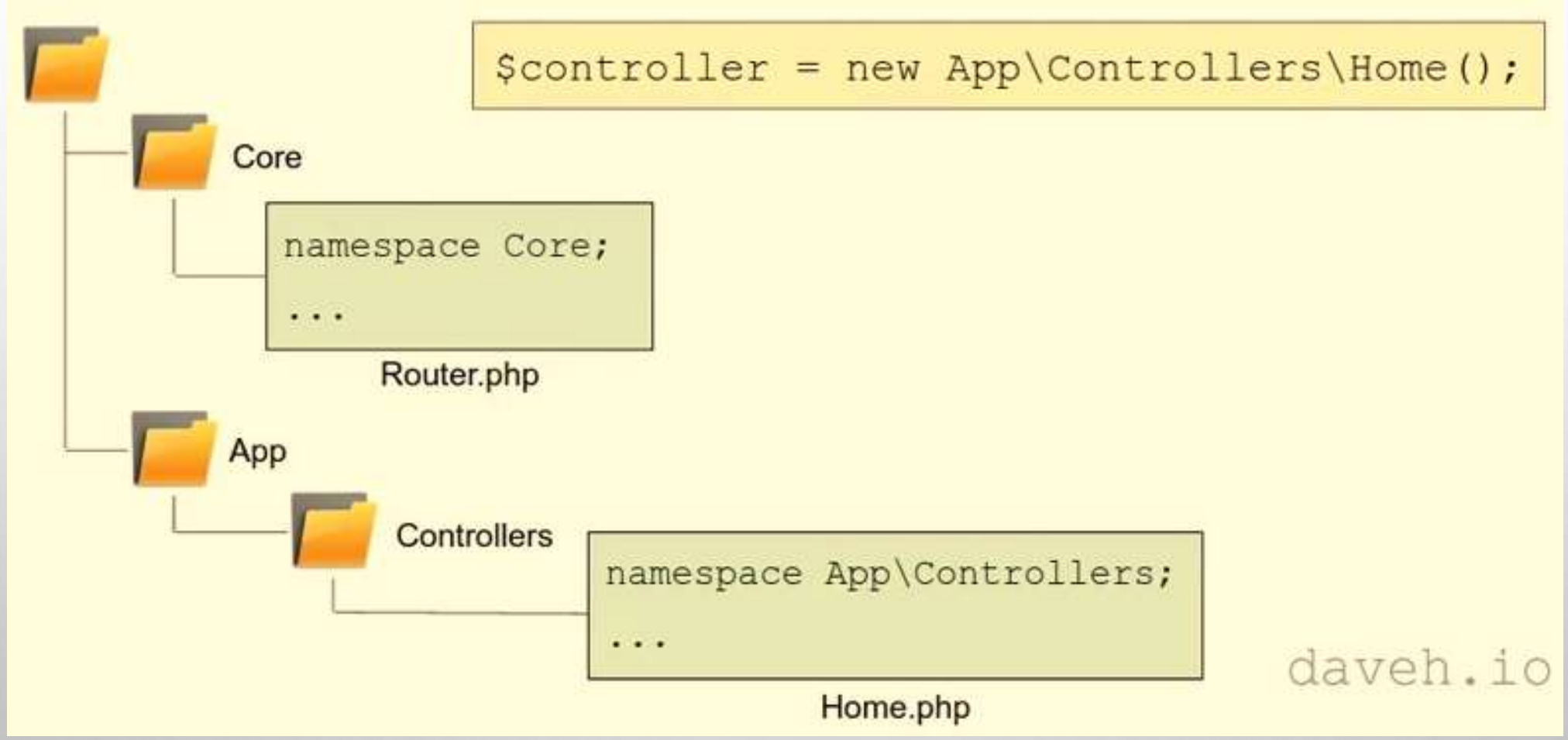
- Подайте команду `composer dump-autoload`
- Видаліть функцію автозавантаження з фронт контролера
- Перевірте в браузері, чи все працює.

Framework classes



- Буде працювати Composer autoloader

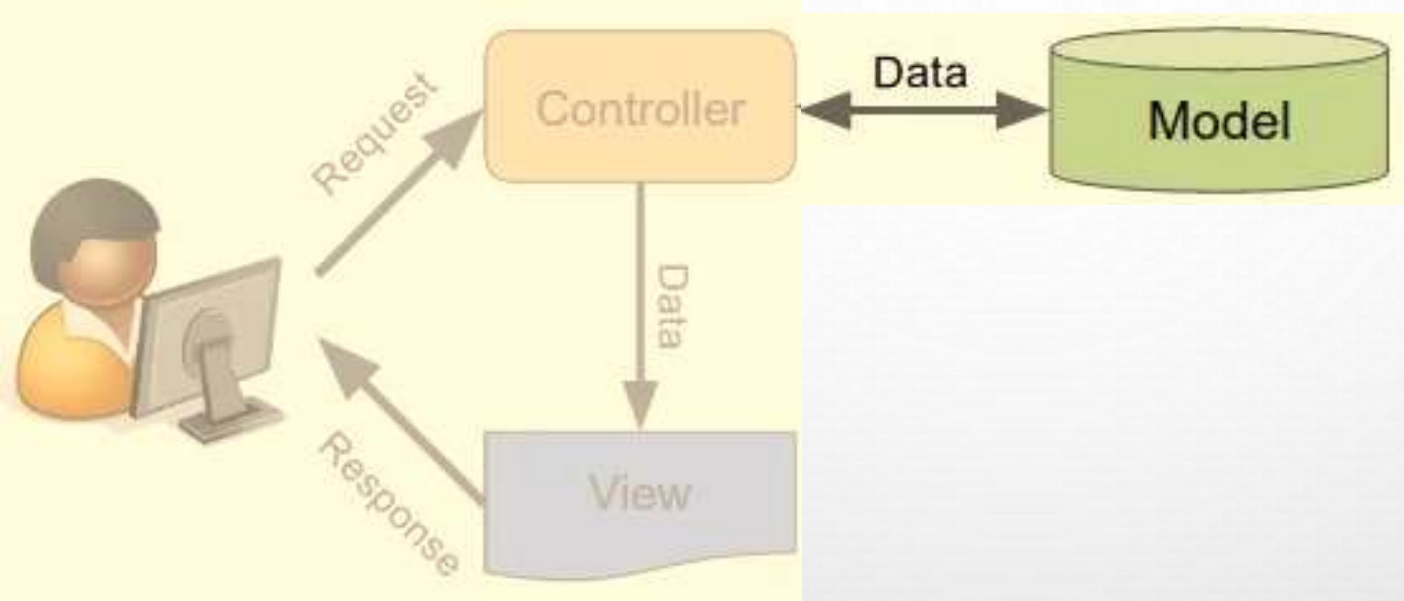
Framework classes



- Буде працювати Composer autoloader

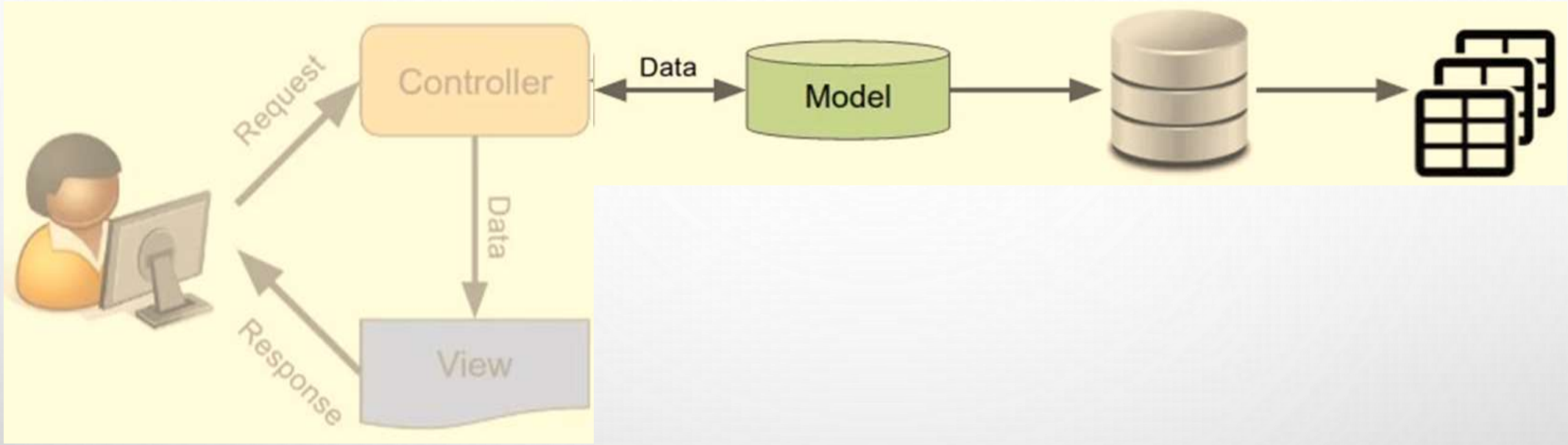
Models: an introduction

Models



- Models are where an application's **data** are stored.
- Responsible for **storing** and **retrieving** data.

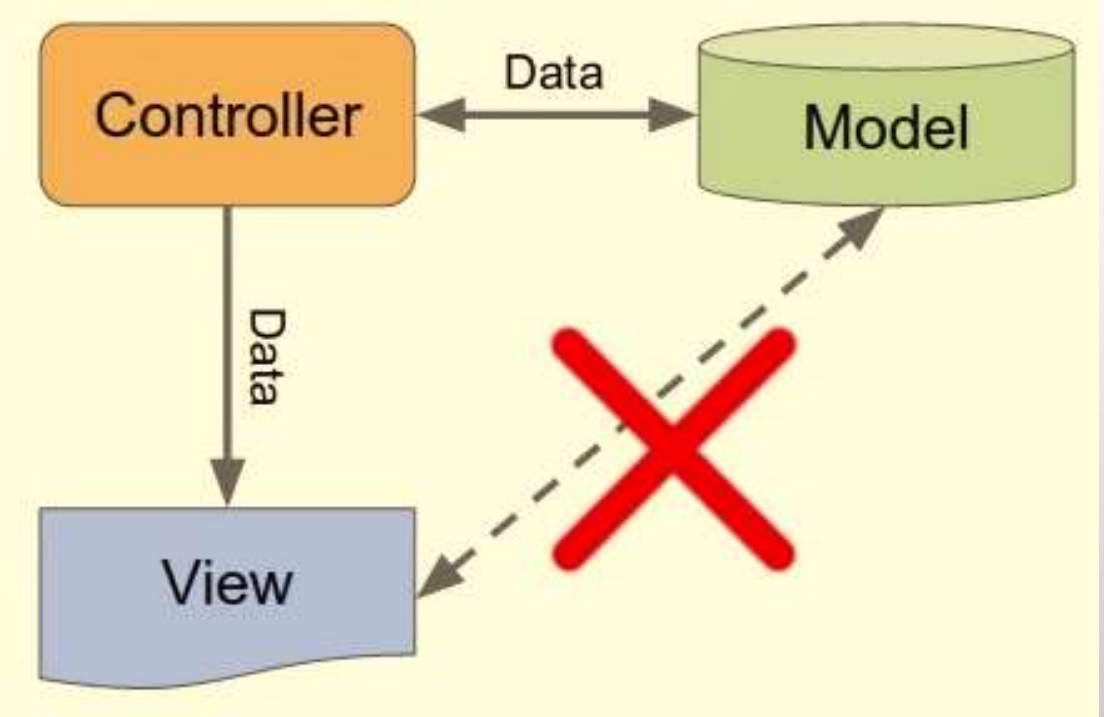
Models and databases



- Models commonly store data in a database.
- A **single** model also has an **equivalent database table**, e.g. Post model -> posts table

Separation of concerns

- Controllers pass data **to** the views.
- Controllers data **to** and **from** the models.
- The views **know nothing** about where the data comes from.



Without a framework

```
<?php

$db = new mysqli('localhost', 'user', 'pass', 'demo');
$result = $db->query('SELECT * FROM users');

?>
<html>
<body>
<h1>List of users</h1>
<?php while ($row = $result->fetch_assoc()): ?>
  <p><?php echo $row['name']; ?></p>
<?php endwhile; ?>
</body>
</html>
```

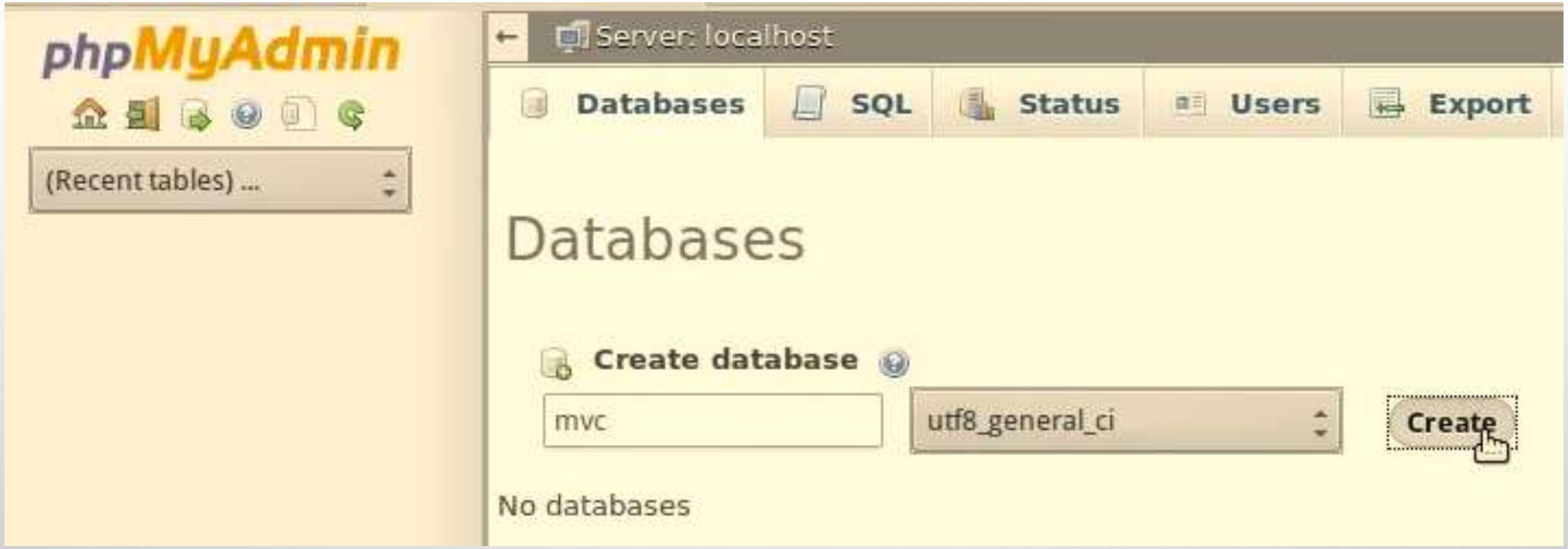
- **Все в одному файлі**: доступ до БД, витягнення деяких даних і відображення їх.

Create a database and check you can connect to it from PHP

Connecting to MySQL


http://localhost/ampps/
AMPPS - Powered...
http://phpmyad...
localhost/check_d...
ampps
PHP JavaScripts PERL Classes
Welcome soft
Search
Blogs
Micro Blogs
Portals/CMS
Forums
Image Galleries
Wikis
Social Networking
Ad Management
Calendars
Gaming
Mails
AMPPS
Configure
Secure AMPPS
Security Center
Status
Add Domain
Manage Domains
Database Tools
SQlite Manager
Add Database
phpMyAdmin
MySQL Password
RockMongo
http://localhost/ampps/index.php?act=listsoftwares&cat=mail

Створення бази даних



Sample database

posts
id
title
content
created_at

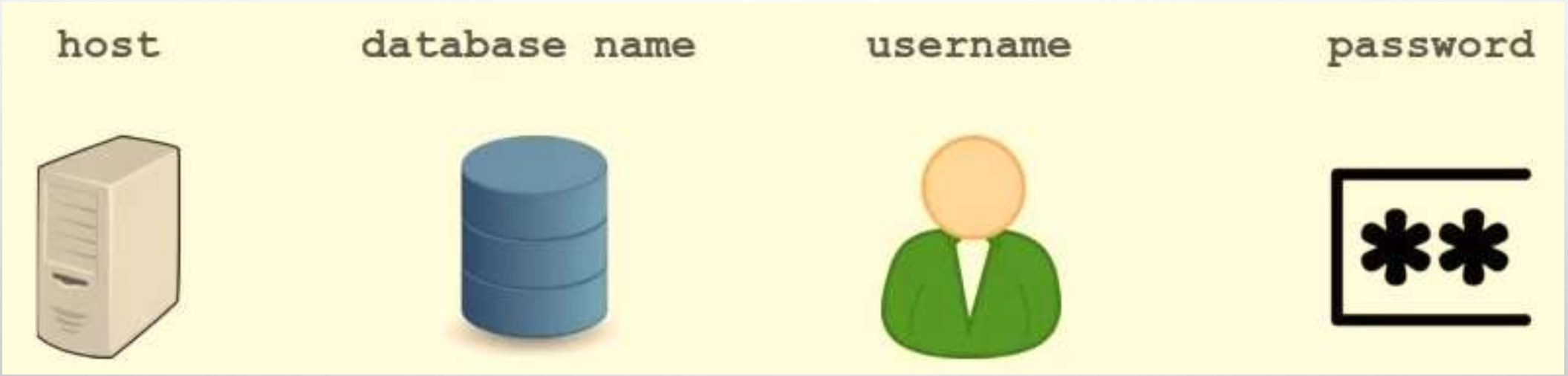
title	content
First post	This is a really interesting post.
Second post	This is a fascinating post!
Third post	This is a very informative post.

Файл sample_database.sql

```
CREATE TABLE posts (
   id int(11) NOT NULL AUTO_INCREMENT,
   title varchar(128) NOT NULL,
   content text NOT NULL,
   created_at timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY (id),
   KEY created_at (created_at)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

INSERT INTO posts (title, content) VALUES
('First post', 'This is a really interesting post.'),
('Second post', 'This is a fascinating post!'),
('Third post', 'This is a very informative post.');
```

Database connection data

- To connect to a database:



Файл check_database.php

```
<?php
  $host = "localhost";
  $db_name = "mvc";
  $user = "root";
  $password = "secret";

 $conn = new mysqli($host, $user, $password, $db_name);

 if ($conn->connect_error) {
      echo "Connection failed: " . $conn->connect_error;
 } else {
      echo "Connected successfully, connection data are ok.";
 }
```

An introduction to PDO: why it makes working with databases in PHP easier

Database-specific functions

```
$conn = new mysqli($host, $user, $password, $db_name);

$result = $conn->query("SELECT title FROM posts");

while($row = $result->fetch_assoc()) {
  echo $row["title"] . "<br/>";
}
```

What is PDO, and why use it?

- PDO (PHP Data Objects) is **a code library for accessing databases.**

Advantages:

- Not specific to any database platform → **faster to code**
- Named parameters in SQL statements → **more secure**
- Exceptions are used for error handling → **improved code quality**

Connecting to a database

```
try {
  $db = new PDO('mysql:host=local;dbname=mydb', 'user', 'password');
} catch(PDOException $e) {
  echo $e->getMessage();
}
```

Simple queries

```
$db = new PDO(...);

$sql = "SELECT title FROM posts WHERE id = 1";
$statement = $db->query($sql);
$row = $statement->fetchObject();
echo $row->title;
```

Simple inserts and updates

```
$count = $db->exec("INSERT INTO posts(title, content) ↲
                    VALUES ('Article 1', 'How interesting!')");
```

```
$count = $db->exec("UPDATE posts SET title = 'New' ↲
                    WHERE title = 'Old'");
```

Prepared statements

```
$id = 123;

$sql= "SELECT title FROM posts WHERE id = :postID";
$stmt = $db->prepare($sql);
$stmt->bindParam(':postID', $id, PDO::PARAM_INT);
$stmt->execute();
```

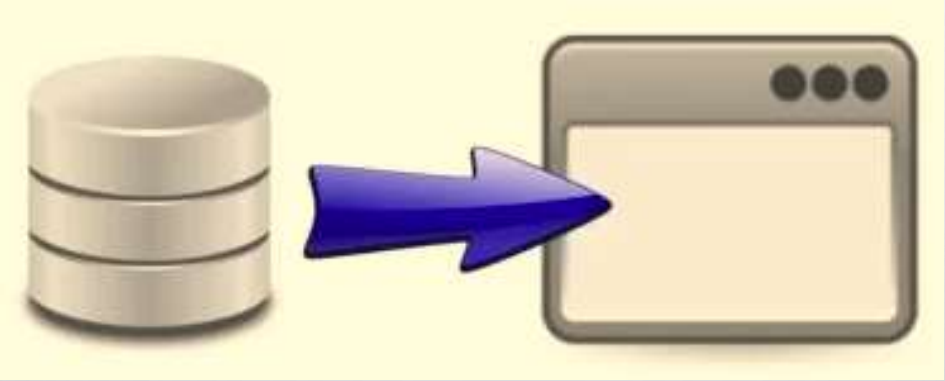
```
SELECT title, content FROM posts WHERE id = 123
```

Prepared statements

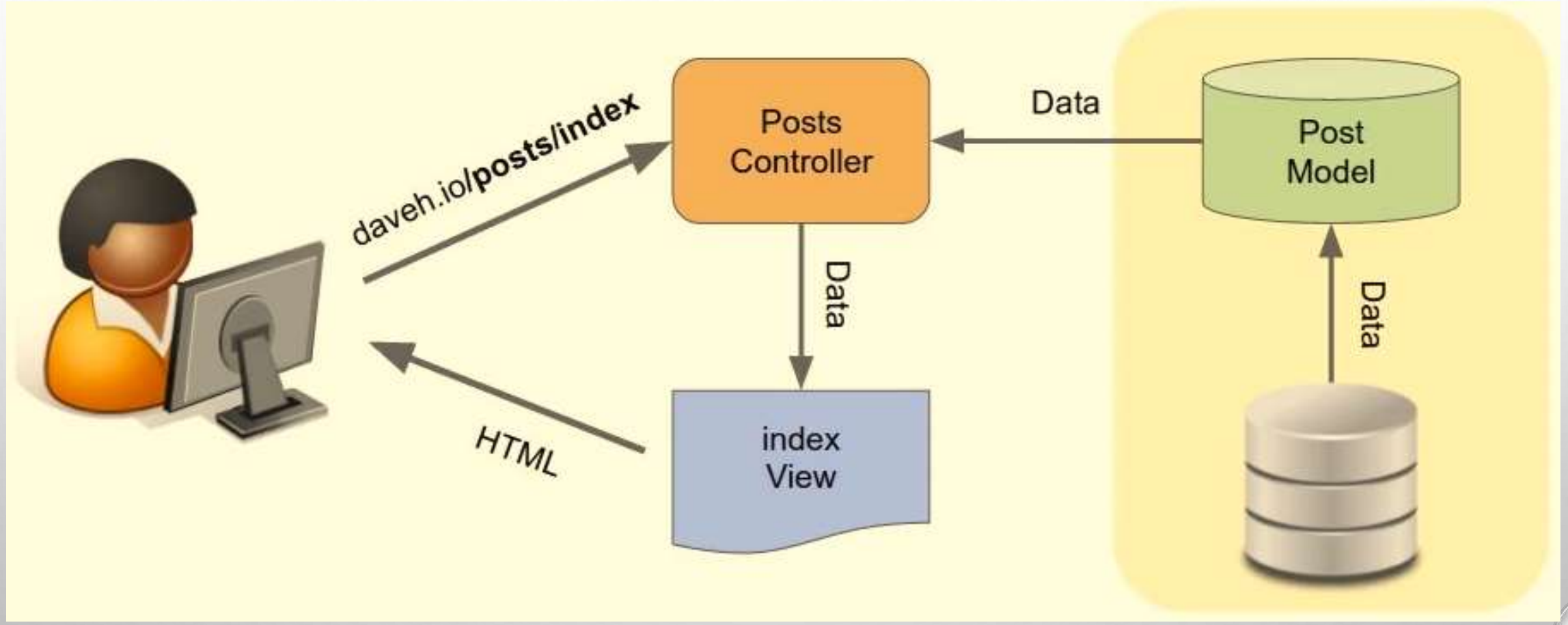
```
$sql= "INSERT INTO posts(title, content) ↲
       VALUES (:title, :content)";
$stmt = $db->prepare($sql);
$stmt->bindParam(':title', 'Article 1');
$stmt->bindParam(':content', "That's nice!");
$stmt->execute();
```

```
INSERT INTO posts(title, content)
VALUES ('Article 1', 'That\'s nice!')
```

Add a model, get data from the database and display it in a view



Displaying data from a model



Post Model (файл Models\Post.php)

```
<?php
  namespace App\Models;
  use PDO;
   class Post
   {
     public static function getAll()    {
         $host = 'localhost';
         $dbname = 'mvc';
         $username = 'root';
         $password = 'secret';
         try {
            $db = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8",
$username, $password);
          $stmt = $db->query('SELECT id, title, content FROM posts
ORDER BY created_at');
         $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
         return $results;                  } catch (PDOException $e) {          echo $e-
>getMessage();       }    }}
```

Post Controller (файл Controllers\Posts.php)

```
<?php
 namespace App\Controllers;
 use \Core\View;
 use App\Models\Post;
 class Posts extends \Core\Controller
 {
     public function indexAction()    {
         $posts = Post::getAll();
         View::renderTemplate('Posts/index.html', [
            'posts' => $posts
         ]);
     }
```

Файл Views\Posts\index.html

```
{% extends "base.html" %}
{% block title %}Posts{% endblock %}
{% block body %}
    <h1>Posts</h1>
    <ul>
        {% for post in posts %}
            <h2>{{ post.title }}</h2>
            <p>{{ post.content }}</p>
        {% endfor %}
   </ul>
{% endblock %}
```

```
    public function addNewAction()    {
            echo 'Hello from the addNew action in the Posts controller!';
    }

    public function editAction()    {
        echo 'Hello from the edit action in the Posts controller!';
        echo '<p>Route parameters: <pre>' .
            htmlspecialchars(print_r($this->route_params, true)) . '</pre></p>';
    }
}
```

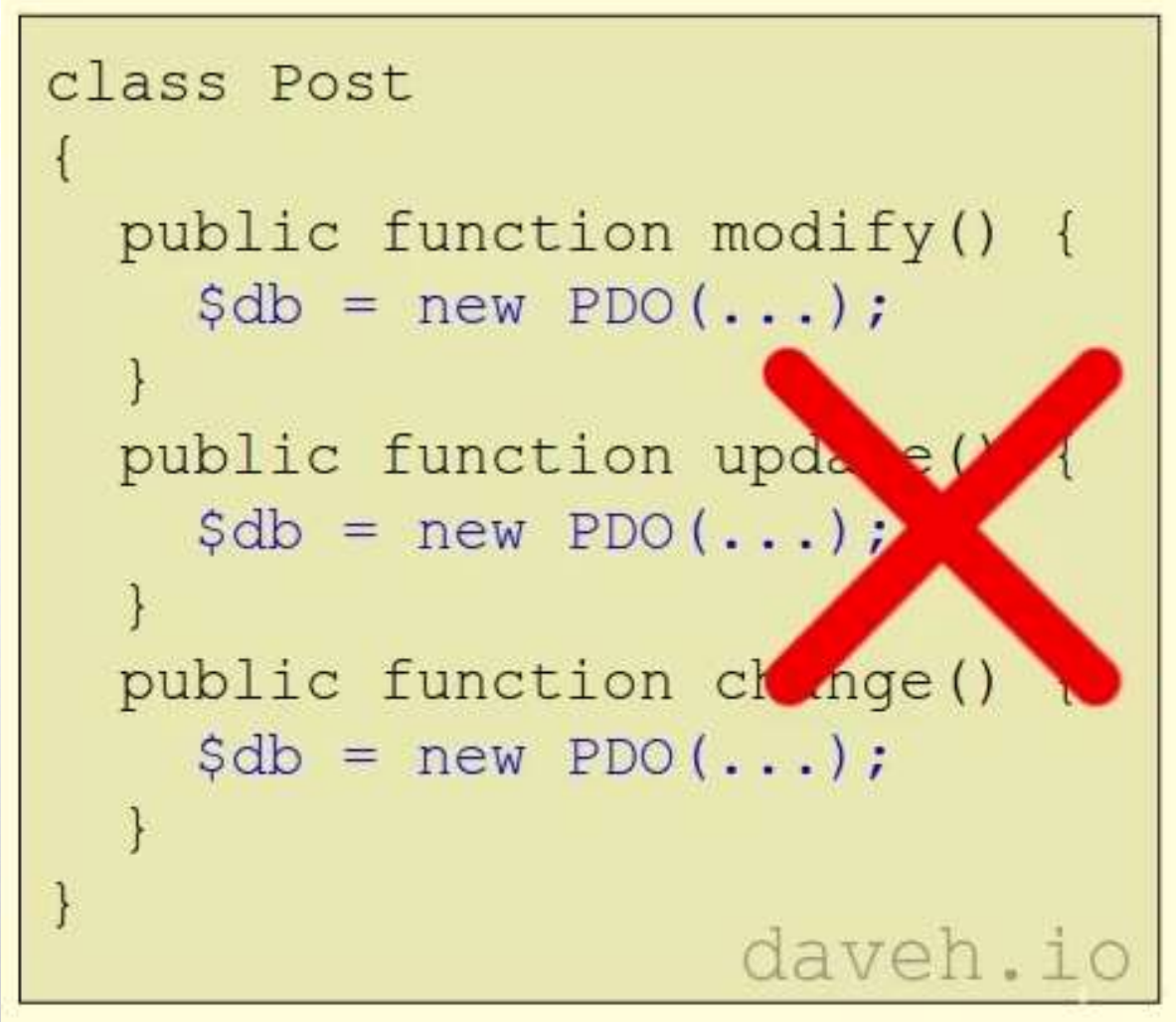
Тестування в браузері

- Дивись приклад mvcframework15 в Moodle

Optimise the database connection: connect only on demand and reuse it

Where to connect to the database?

- Connecting to the database is **costly** (time, server resources).
- A **single** request might contain multiple database queries.
- Ideally, we would only connect to the database **once** per request, **reusing** the same connection for multiple queries.



Where to connect to the database?

- Not all **models** will necessarily need to connect to the database.
- Not all **methods** in a model will necessarily need access to the database.

```
abstract class Model
{
  protected $db;

  public function __construct( ) {
    $this->db = new PDO(...);
  }

  ...
}
```

Where to connect to the database?

- The value of the **static** variable is remembered between calls.
- The connection is only made **once**, and is reused after the first call.

```
abstract class Model
{
  static function getDB() {
    static $db = null;
    if ($db === null) {
      $db = new PDO(...);
    }
    return $db;
}
```

Новий файл Core\Model.php

```
<?php
  namespace Core;
  use PDO;
  abstract class Model
  {
    protected static function getDB()    {
      static $db = null;
      if ($db === null) {
          $host = 'localhost'; $dbname = 'mvc'; $username = 'root';
          $password = 'secret';
             try {
  $db = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8",
$username, $password);
        } catch (PDOException $e) {
           echo $e->getMessage();
        }      }      return $db;    }}
```

Файл Models\Post.php

```
<?php
  namespace App\Models;
  use PDO;
  class Post extends \Core\Model
  {
     public static function getAll()    {
          try {
              $db = static::getDB();
              $stmt = $db->query('SELECT id, title, content FROM posts
                                        ORDER BY created_at');
             $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
             return $results;
          } catch (PDOException $e) {
        echo $e->getMessage();       }    }}
```

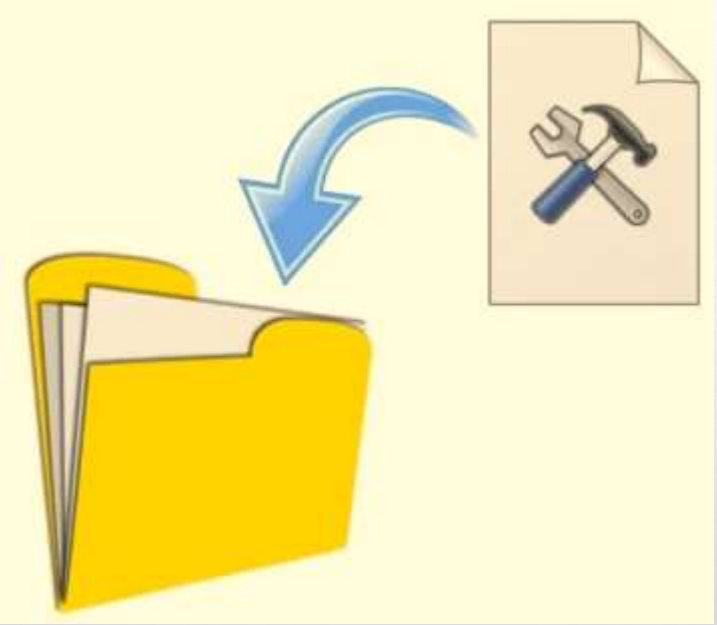
Тестування в браузері

- Дивись приклад mvcframework16 в Moodle

Put application configuration settings in a separate file

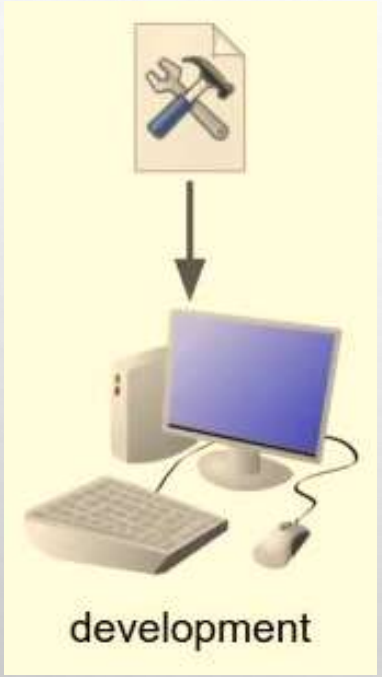
Application configuration

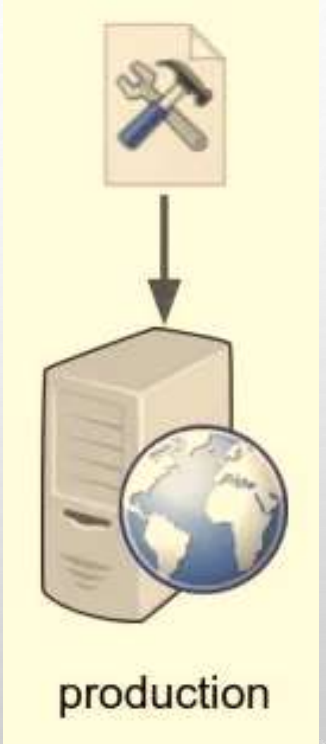
- Configuration settings should bi separate from the rest of the code:



Application configuration

- Configuration settings are different when **developing** and in **production**.
- Settings need to be **easily changed** when moving code between servers.





Новий файл App\Config.php

```
<?php
  namespace App;
  class Config
  {
     const DB_HOST = 'localhost';
     const DB_NAME = 'mvc';
     const DB_USER = 'root';
     const DB_PASSWORD = 'secret';
  }
```

Файл Core\Model.php

```
<?php
 namespace Core;
 use PDO;
 use App\Config;
abstract class Model{
    protected static function getDB()    {
        static $db = null;
        if ($db === null) {
        try {
            $dsn = 'mysql:host=' . Config::DB_HOST . ';dbname=' .
                Config::DB_NAME . ';charset=utf8';
            $db = new PDO($dsn, Config::DB_USER, Config::DB_PASSWORD);
} catch (PDOException $e) {
            echo $e->getMessage();          }       }
    return $db;    }}
```

Тестування в браузері

How PHP reports problems: errors, exceptions, and how to handle them

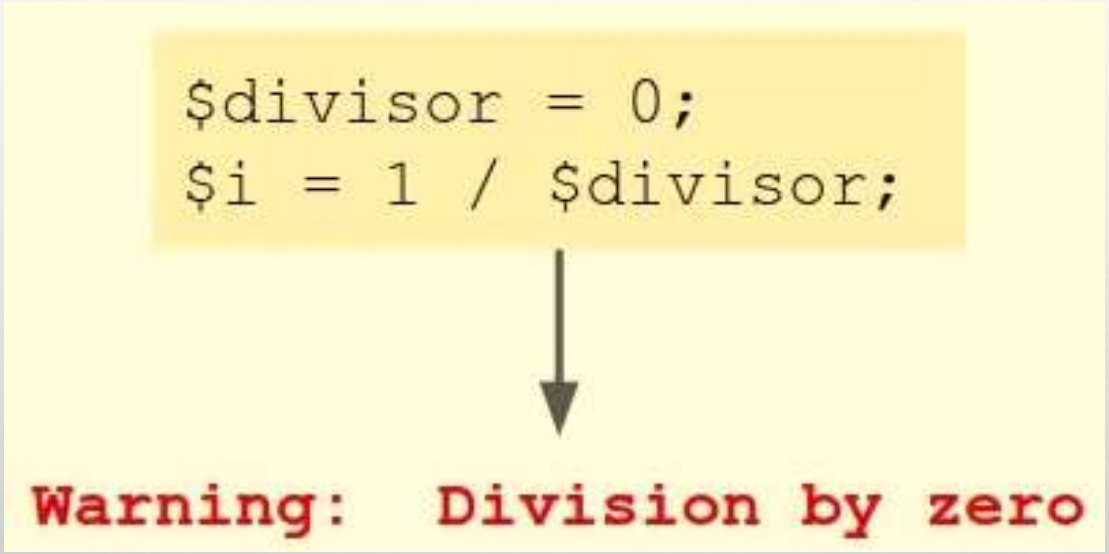
What are errors and exceptions?

- Errors and exceptions occur when **something goes wrong** in the code.
- PHP provides both **errors** and **exceptions** to tell us when there's a problem.
- If we don't write code that **handles** them, the program will stop.

Errors

- **Internal PHP functions** mainly use **errors** to signal a problem.
- Errors generally occur at the language level, for example a **syntax error** or an **invalid action** with a **variable**:

```
$divisor = 0;
$i = 1 / $divisor;
```



```
Warning:  Division by zero
```

Triggering errors manually

- Can be generated manually using the `trigger_error` function

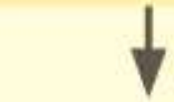
```
if ($divisor == 0) {
    trigger_error("Cannot divide by zero", E_USER_ERROR);
}
```

Fatal error: Cannot divide by zero

Handling errors

- You can register **a function to handle errors**, but errors are **unrecoverable**.

```php
function myErrorHandler($num, $str) {
  echo "Custom error: [$num] $str";
}

set_error_handler("myErrorHandler");

trigger_error("Something went wrong");
```



```
Custom error: [1024] Something went wrong
```

Exceptions

- **Exceptions** are the errors you get when dealing with **classes** and **objects**.
- Modern object-oriented PHP extensions generally use **exceptions** to signal a problem.

```
$datetime = new DateTime("invalid time string");
```

```
Fatal error: Uncaught exception ... Failed to parse time
```

Raising exceptions

- Can be generated or "`thrown`" manually using the `throw` statement, and passing in a new Exception object:

```
if ($divisor == 0) {
  throw new Exception("Cannot divide by zero");
}
```

```
Fatal error: Uncaught exception ... Cannot divide by zero
```

Handling exceptions

- Unlike errors, **exceptions** can be **caught** and dealt with, allowing the program to continue.

```
try {
   $datetime = new DateTime("invalid time string");
} catch (Exception $e) {
   echo "Problem: " . $e->getMessage();
}
```

```
Problem: Failed to parse time
```

Handling exceptions

- You can register **a function to handle exceptions** too:

```
function myExceptionHandler($e) {
  echo "Custom exception: " . $e->getMessage();
}

set_exception_handler("myExceptionHandler");

throw new Exception("Something went wrong");
```

```
Custom exception: Something went wrong
```

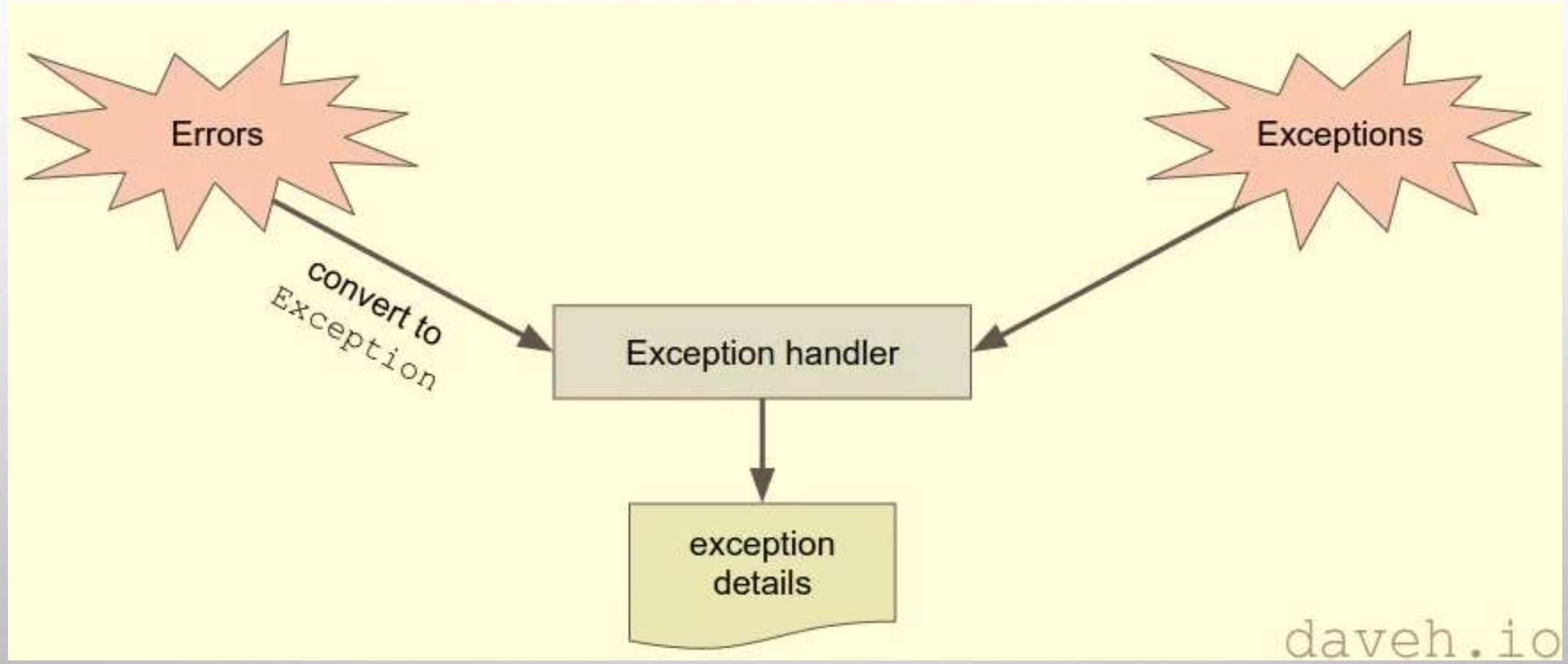
Converting errors to exceptions

- The easiest way to handle both types is to **convert errors to exceptions**, then only one handler is needed.
- So when an error occurs, an **exception** is raised instead.
- Exceptions have the added benefit of having a **stack trace**, which is helpful when debugging.

```
function myErrorHandler($level, $text, $file, $line) {
  throw new ErrorException($text, 0, $level, $file, $line);
}

set_error_handler("myErrorHandler");
```

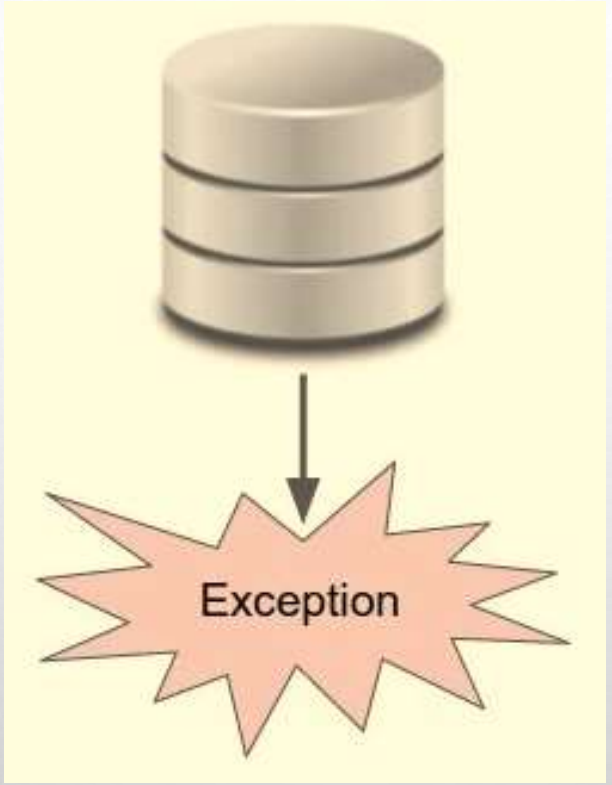
Handle errors: convert errors to exceptions, and add an exception handler

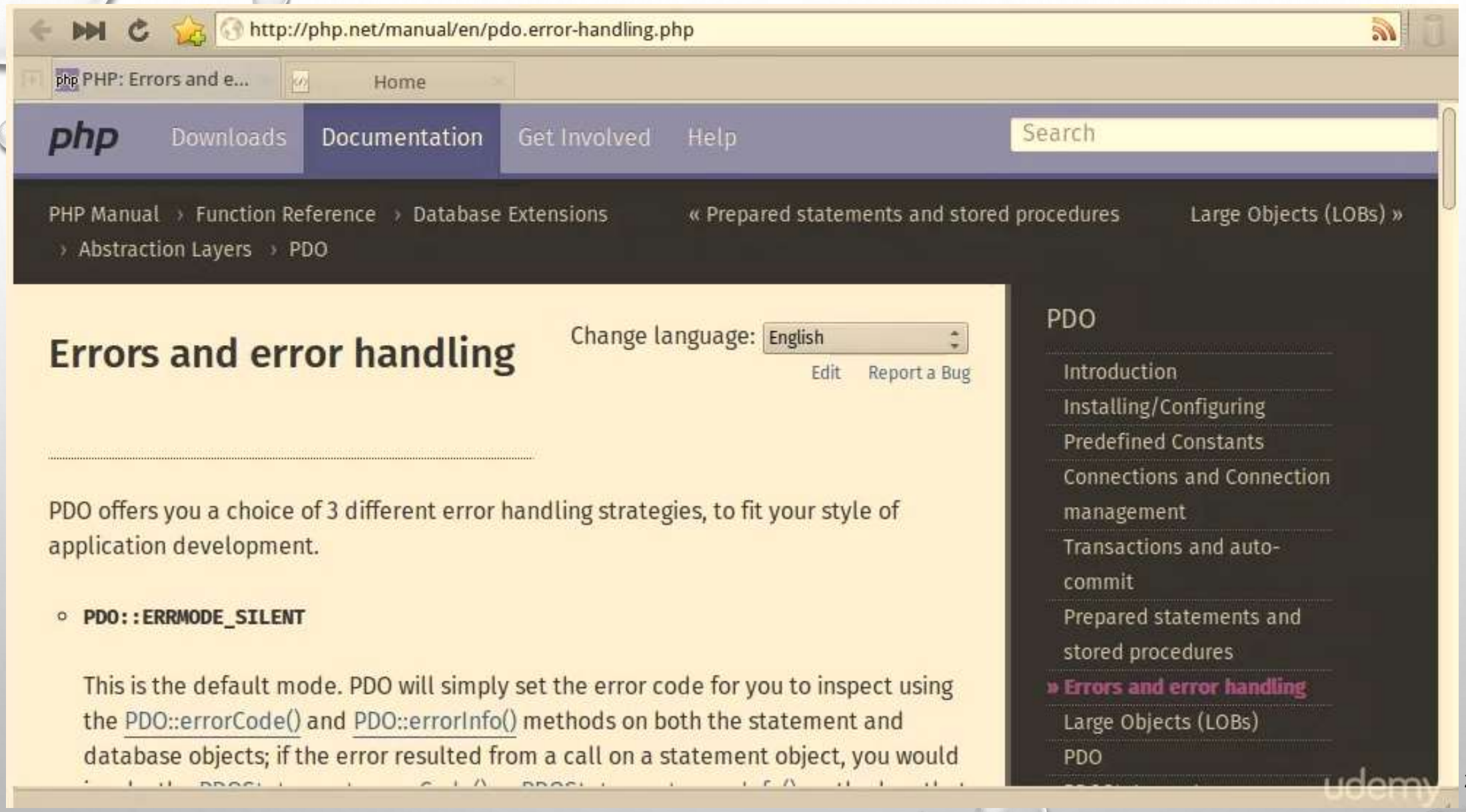
Handling errors and exceptions



Database errors when using PDO

- Throw an **Exception** when a database error occurs:




http://php.net/manual/en/pdo.error-handling.php
PHP: Errors and e...
Home
php
Downloads
Documentation
Get Involved
Help
Search
PHP Manual › Function Reference › Database Extensions › Abstraction Layers › PDO
« Prepared statements and stored procedures
Large Objects (LOBs) »
Errors and error handling
Change language: English
Edit
Report a Bug
PDO offers you a choice of 3 different error handling strategies, to fit your style of application development.
PDO::ERRMODE_SILENT
This is the default mode. PDO will simply set the error code for you to inspect using the PDO::errorCode() and PDO::errorInfo() methods on both the statement and database objects; if the error resulted from a call on a statement object, you would
PDO
Introduction
Installing/Configuring
Predefined Constants
Connections and Connection management
Transactions and auto-commit
Prepared statements and stored procedures
» Errors and error handling
Large Objects (LOBs)
PDO
udemy

Новий файл Core\Error.php

```
<?php
  namespace Core;
  class Error
  {
     public static function errorHandler($level, $message, $file, $line)
     {
        if (error_reporting() !== 0) {  // to keep the @ operator working
           throw new \ErrorException($message, 0, $level, $file, $line);
        }
     }
```

```
public static function exceptionHandler($exception)
{
    echo "<h1>Fatal error</h1>";
    echo "<p>Uncaught exception: '" . get_class($exception) . "'</p>";
    echo "<p>Message: '" . $exception->getMessage() . "'</p>";
    echo "<p>Stack trace:<pre>" . $exception->getTraceAsString() . "</pre></p>";
    echo "<p>Thrown in '" . $exception->getFile() . "' on line " .
                $exception->getLine() . "</p>";
}
}
```

Редагування файлів

- Фронт контролер (файл public\index.php)

  Додайте два рядки:

```
set_error_handler('Core\Error::errorHandler');
set_exception_handler('Core\Error::exceptionHandler');
```

- Файл Core\Model.php

  Після створення об’єкта PDO додати рядок:

```
// Throw an Exception when an error occurs
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

- Файл Core\Controller.php. В методі __call додати:

```
//echo "Method $method not found in controller " . get_class($this);
throw new \Exception("Method $method not found in controller " .
                     get_class($this));
```

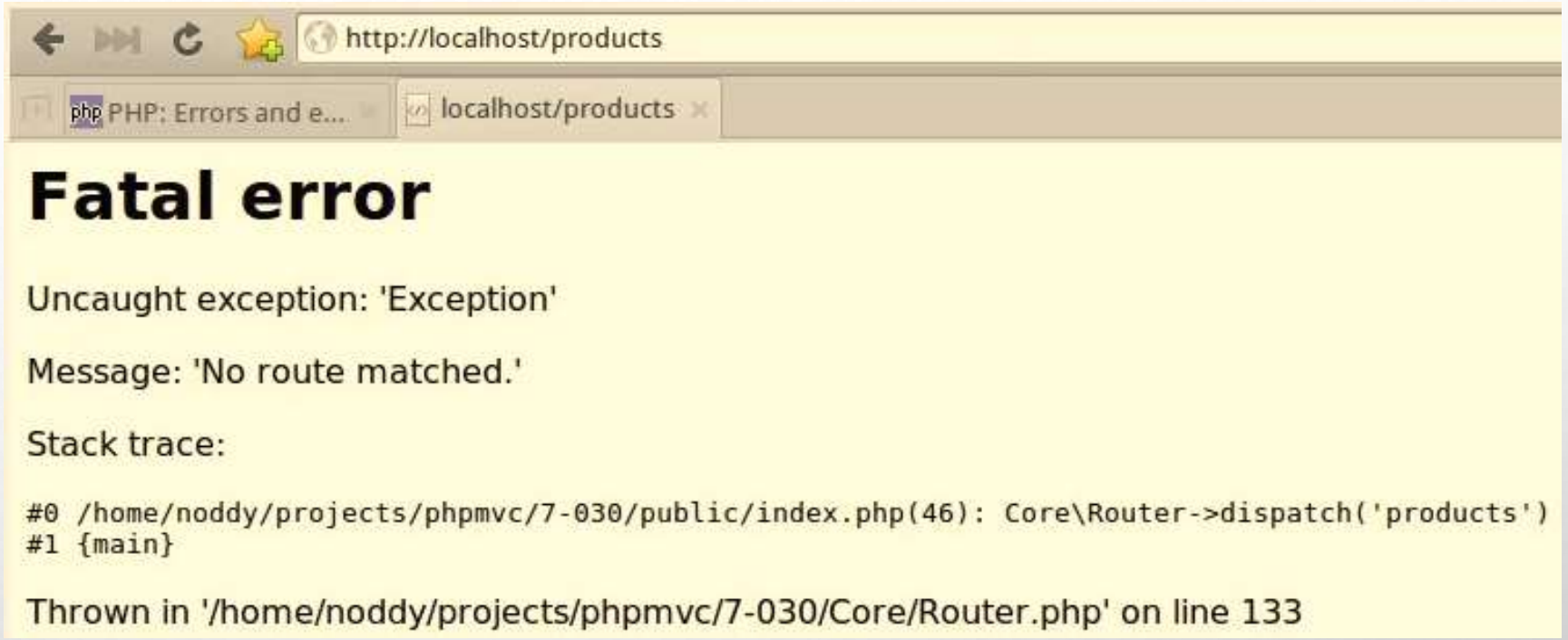
Файл Core\Router.php, метод dispatch

```
            } else {
                //echo "Method $action (in controller $controller) not found";
                throw new \Exception("Method $action (in controller $controller) not found");
            }
        } else {
            //echo "Controller class $controller not found";
            throw new \Exception("Controller class $controller not found");
        }
    } else {
        //echo 'No route matched.';
        throw new \Exception('No route matched.');
    }
}
```

Файл Core\View.php, метод render

```
        } else {
            //echo "$file not found";
            throw new \Exception("$file not found");
        }
    }
```

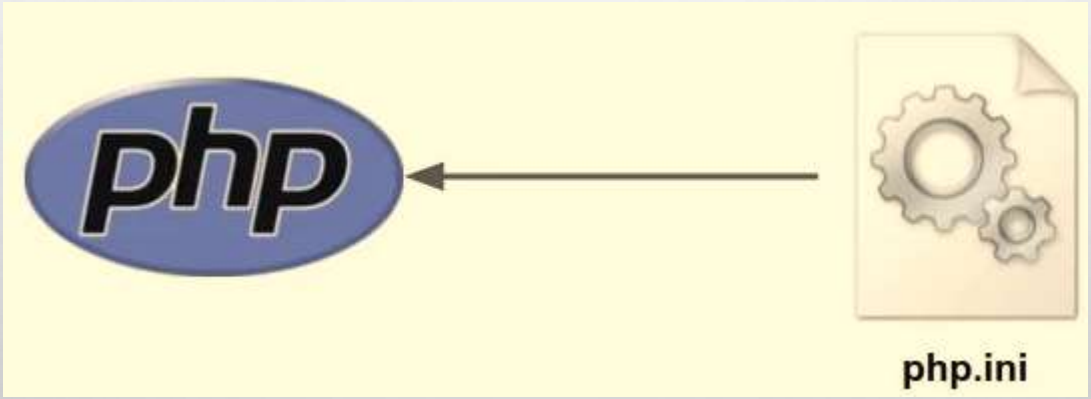
Тестування в браузері



PHP configuration settings: where to find them, and how to change them

PHP configuration

- PHP has **various settings** that can be configured, for example showing or hiding errors, the amount of memory to use and so on.
- **Current settings** can be viewed using the `phpinfo()` command.
- The settings are stored in the `php.ini` configuration file:



Changing PHP settings

php.ini

```
ini_set('setting', 'value');

error_reporting(...);
set_include_path(...);
```

Configure PHP to display error messages

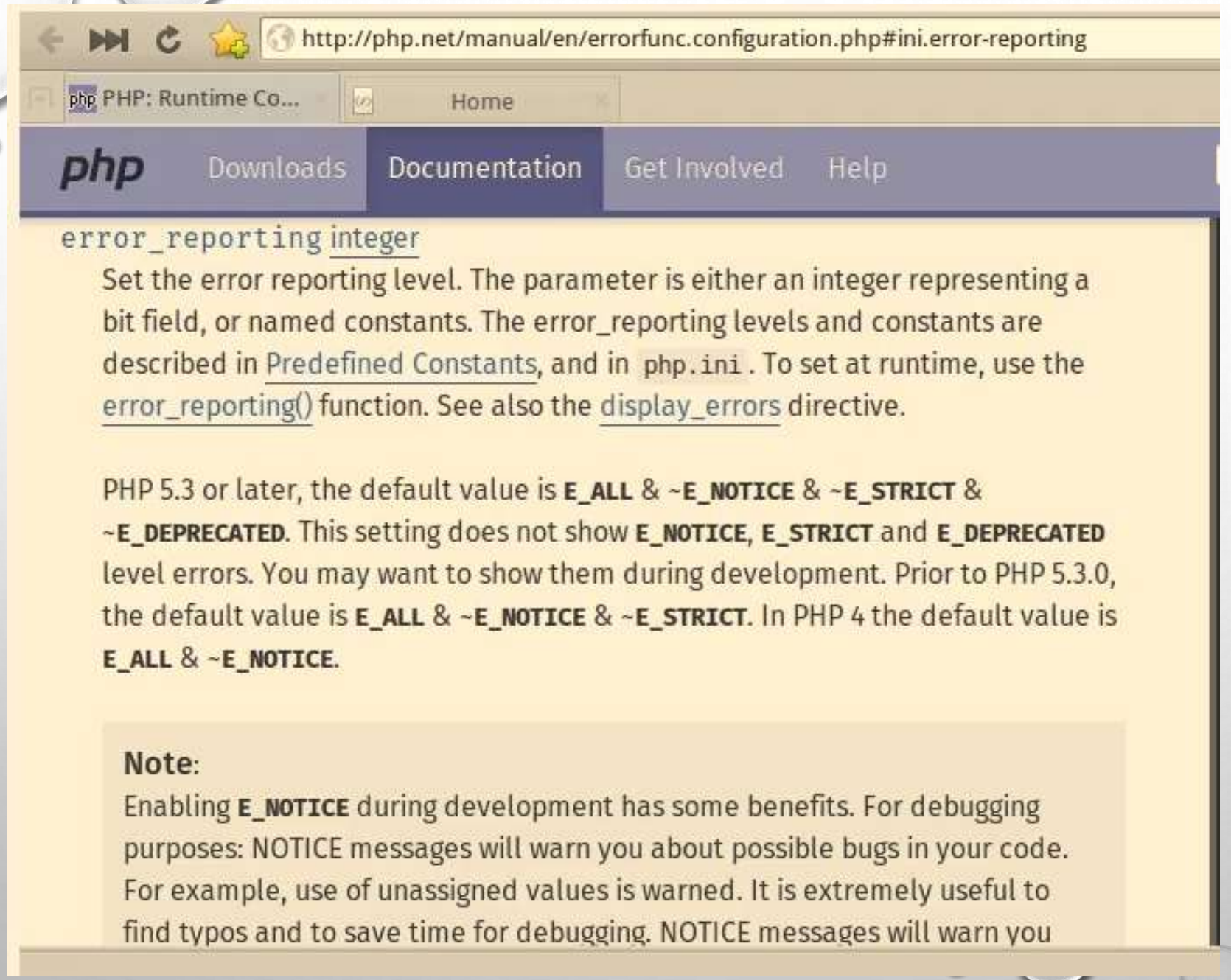
PHP error configuration

- Errors occur when **something goes wrong** in the code.
- PHP can be configured to **show** or **hide** errors or only show errors **of certain types**.
- We want to see errors **so we can fix them**.

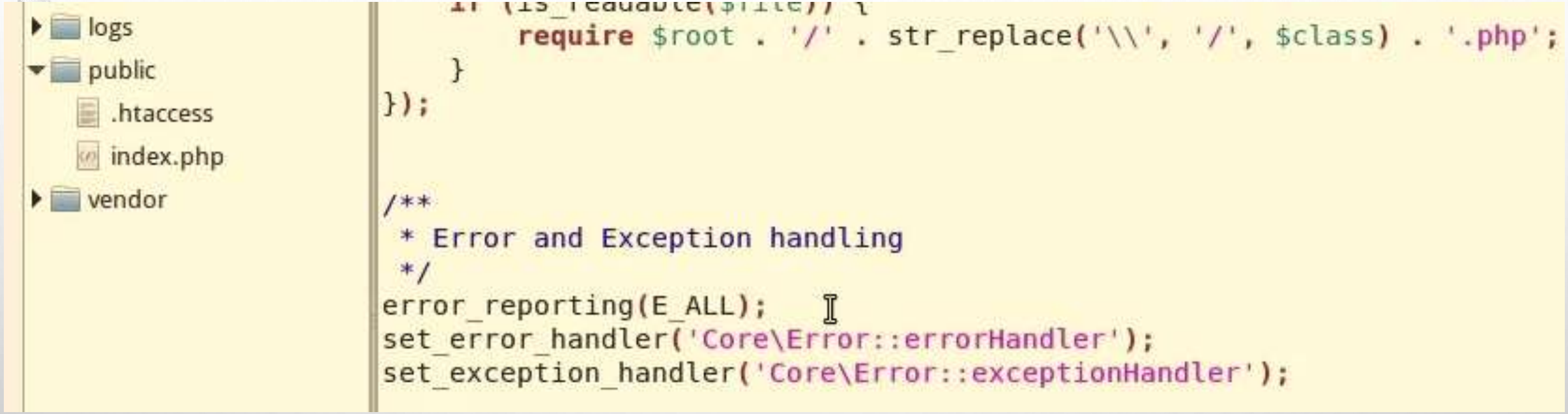
Display all types of errors

- Configure **which type** of error to display

```
error_reporting(E_ALL);
```

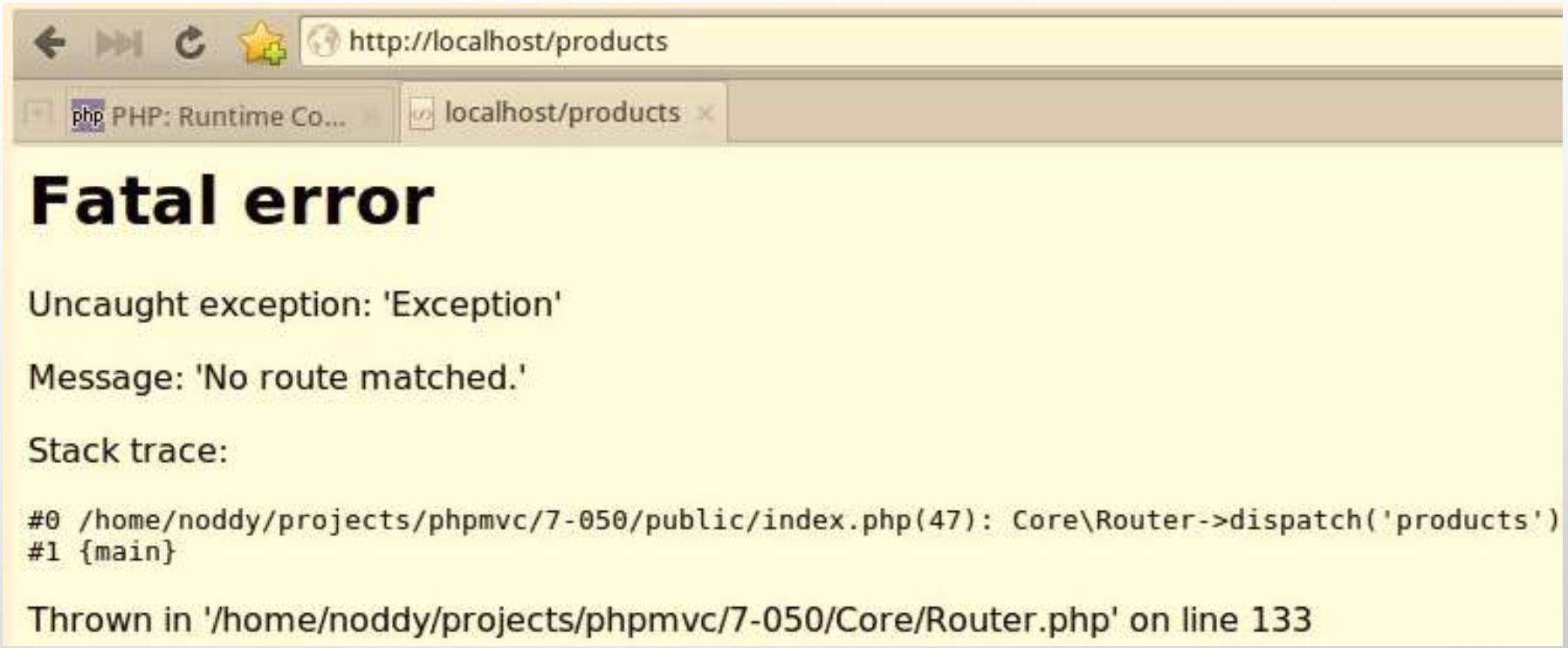

http://php.net/manual/en/errorfunc.configuration.php#ini.error-reporting
PHP: Runtime Co...
Home
php
Downloads
Documentation
Get Involved
Help
error_reporting integer
Set the error reporting level. The parameter is either an integer representing a bit field, or named constants. The error_reporting levels and constants are described in Predefined Constants, and in php.ini. To set at runtime, use the error_reporting() function. See also the display_errors directive.
PHP 5.3 or later, the default value is E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED. This setting does not show E_NOTICE, E_STRICT and E_DEPRECATED level errors. You may want to show them during development. Prior to PHP 5.3.0, the default value is E_ALL & ~E_NOTICE & ~E_STRICT. In PHP 4 the default value is E_ALL & ~E_NOTICE.
Note:
Enabling E_NOTICE during development has some benefits. For debugging purposes: NOTICE messages will warn you about possible bugs in your code. For example, use of unassigned values is warned. It is extremely useful to find typos and to save time for debugging. NOTICE messages will warn you

Зміни в фронт контролері



Тестування

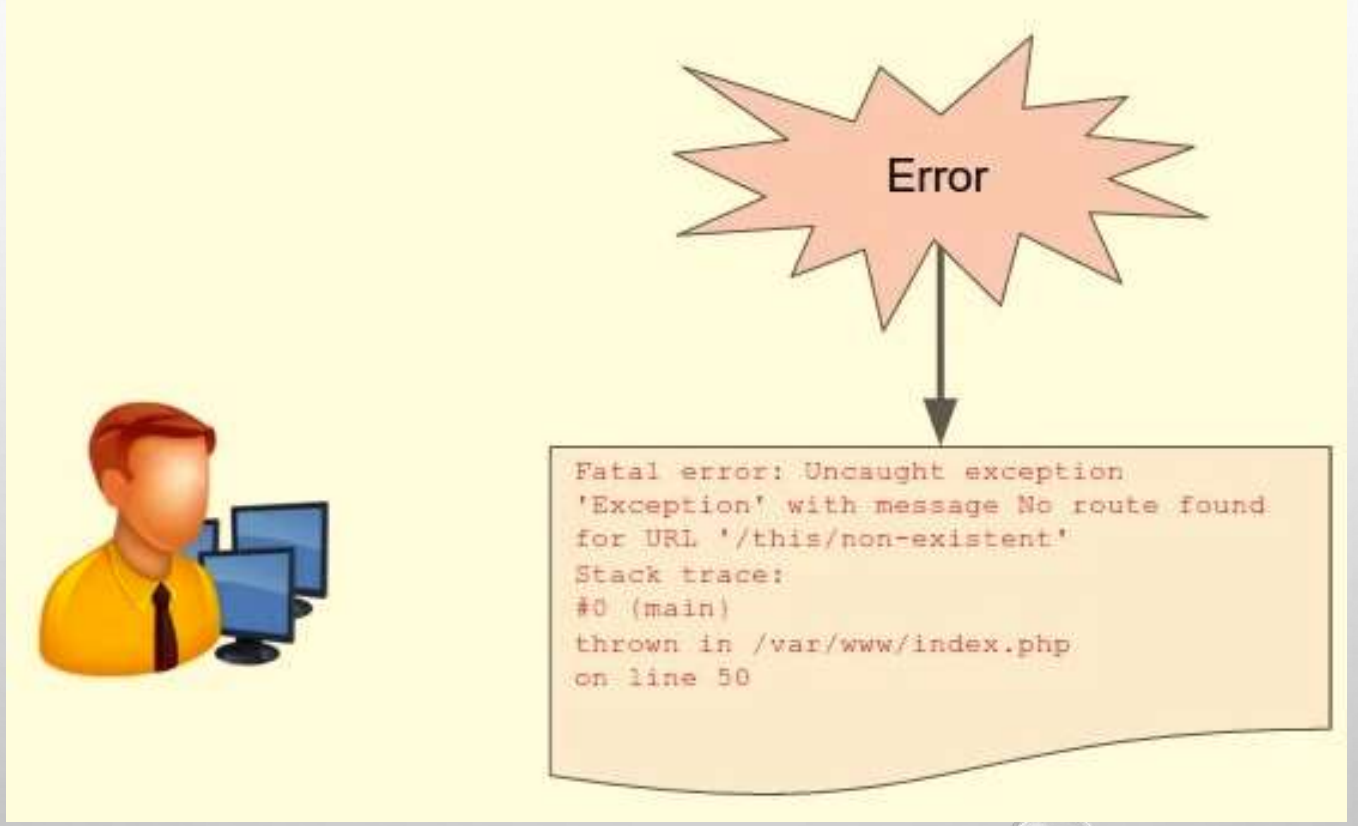

http://localhost/
PHP: Runtime Co...
Home
Home | Posts
Welcome
Hello from a Twig template, Dave!
red
green
blue


http://localhost/products
PHP: Runtime Co...
localhost/products
Fatal error
Uncaught exception: 'Exception'
Message: 'No route matched.'
Stack trace:
#0 /home/noddy/projects/phpmvc/7-050/public/index.php(47): Core\Router->dispatch('products')
#1 {main}
Thrown in '/home/noddy/projects/phpmvc/7-050/Core/Router.php' on line 133

Show detailed error messages to developers, friendly error messages to users

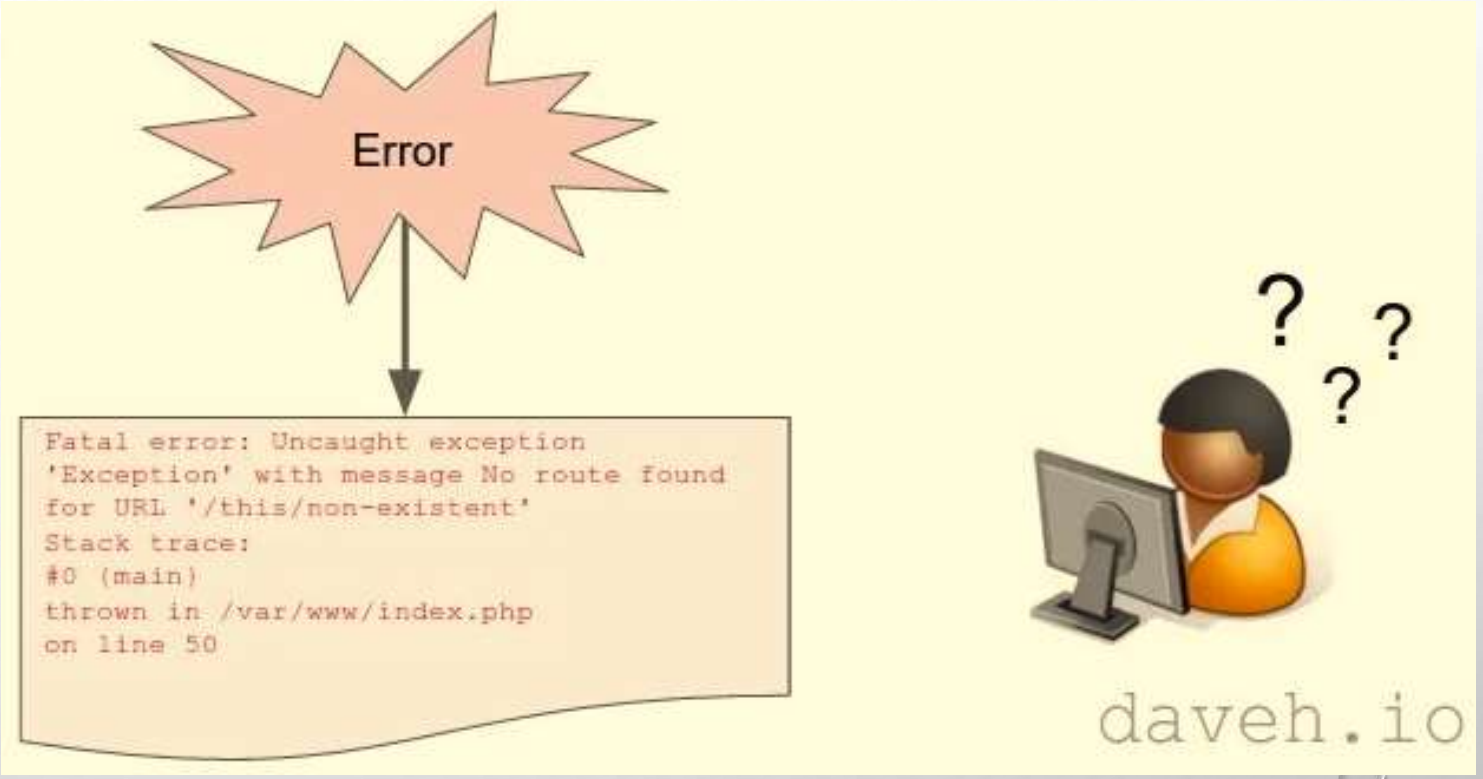
Detailed error messages

- Useful for **developers**, to help to debug errors.



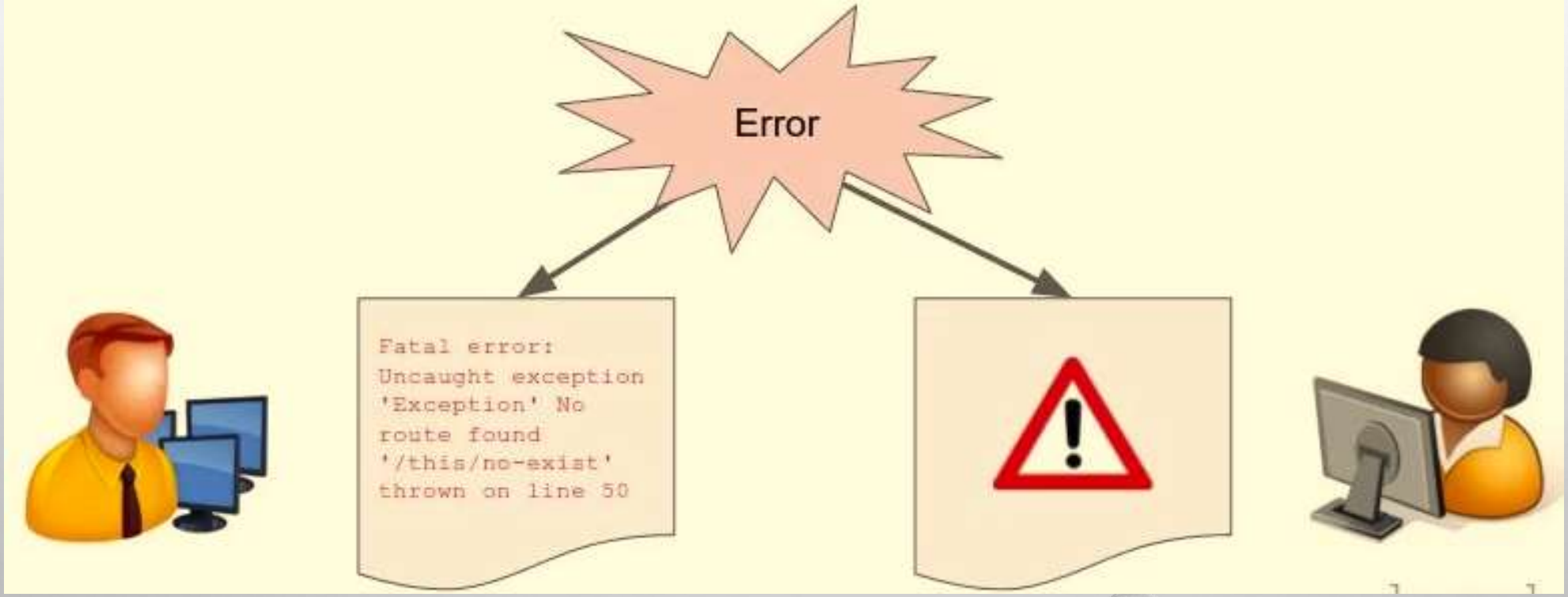
Detailed error messages

- Not useful for **users**:
  - It looks **unprofessional**
  - It can be **unsafe**



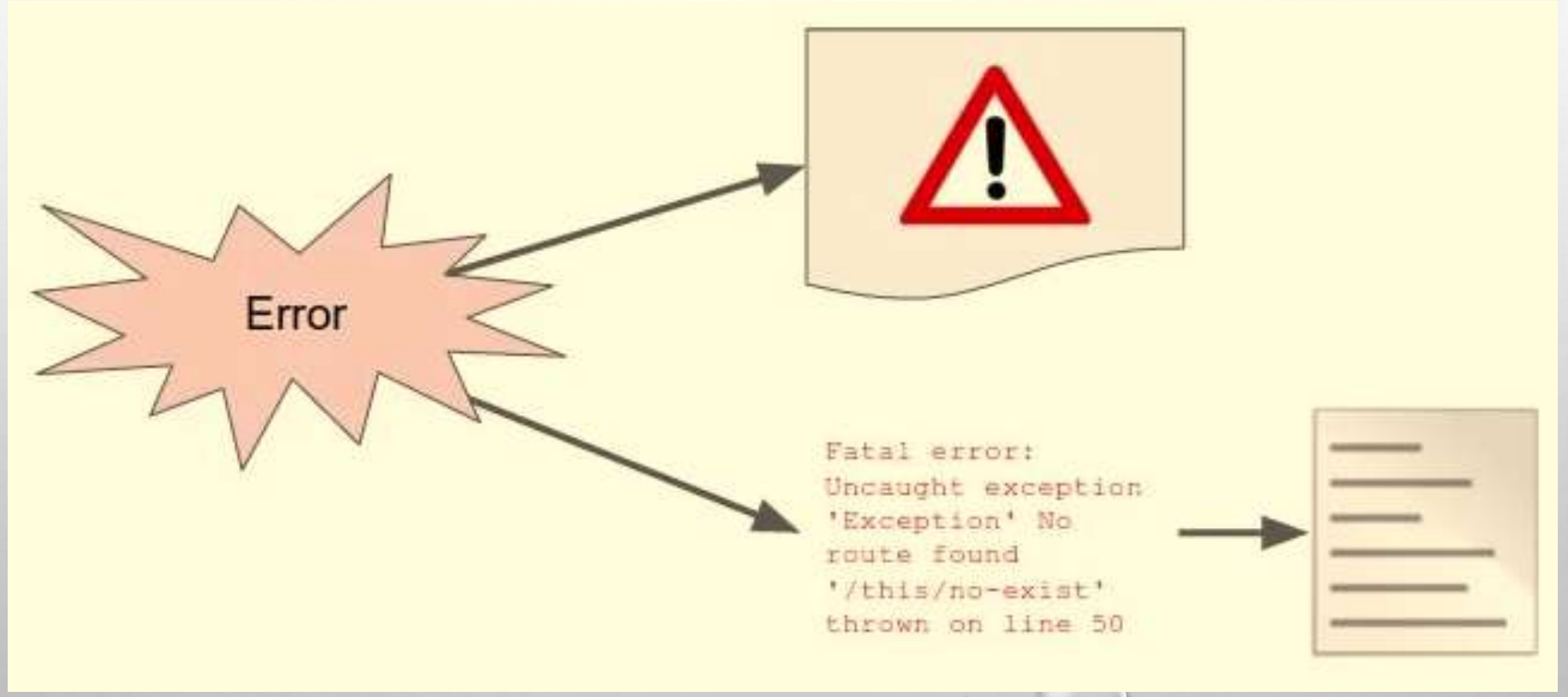
Detail for developers only

- So developers see **detailed** error messages
- Users see a **generic** message



Logging the error

- If an error occurs in production, we still want to know the **details**
- **Save** the error message so we can still access it



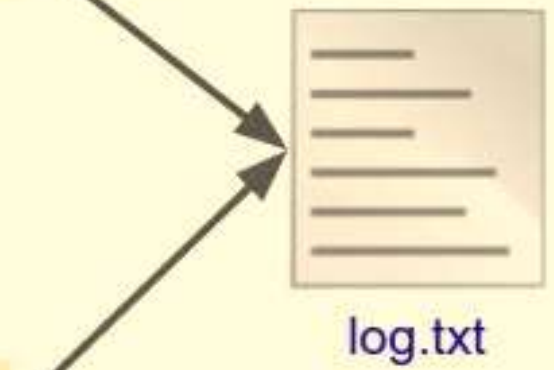
Logging error messages

- Set the **location** of the log file:

```
ini_set('error_log', 'log.txt');
```

- **Write** a message to the log file:

```
error_log('An error occurred');
```



Конфігурування застосунку

- Файл App\Config.php

  Додайте рядок: `const SHOW_ERRORS = false;`

- Файл Core\Error.php

```
public static function exceptionHandler($exception)  {
  if (\App\Config::SHOW_ERRORS) {
    echo "<h1>Fatal error</h1>";
    echo "<p>Uncaught exception: '" . get_class($exception) . "'</p>";
    echo "<p>Message: '" . $exception->getMessage() . "'</p>";
    echo "<p>Stack trace:<pre>" . $exception->getTraceAsString() .
"</pre></p>";
    echo "<p>Thrown in '" . $exception->getFile() . "' on line " . $exception-
>getLine() . "</p>";
```

```
} else {
        $log = dirname(__DIR__) . '/logs/' . date('Y-m-d') . '.txt';
         ini_set('error_log', $log);
        $message = "Uncaught exception: '" . get_class($exception) . "'";
        $message .= " with message '" . $exception->getMessage() . "'";
        $message .= "\nStack trace: " . $exception->getTraceAsString();
        $message .= "\nThrown in '" . $exception->getFile() . "' on line " .
                                        $exception->getLine();
        error_log($message);
        echo "<h1>An error occurred</h1>";
 }
```

Тестування


http://localhost/products
localhost/products
An error occurred

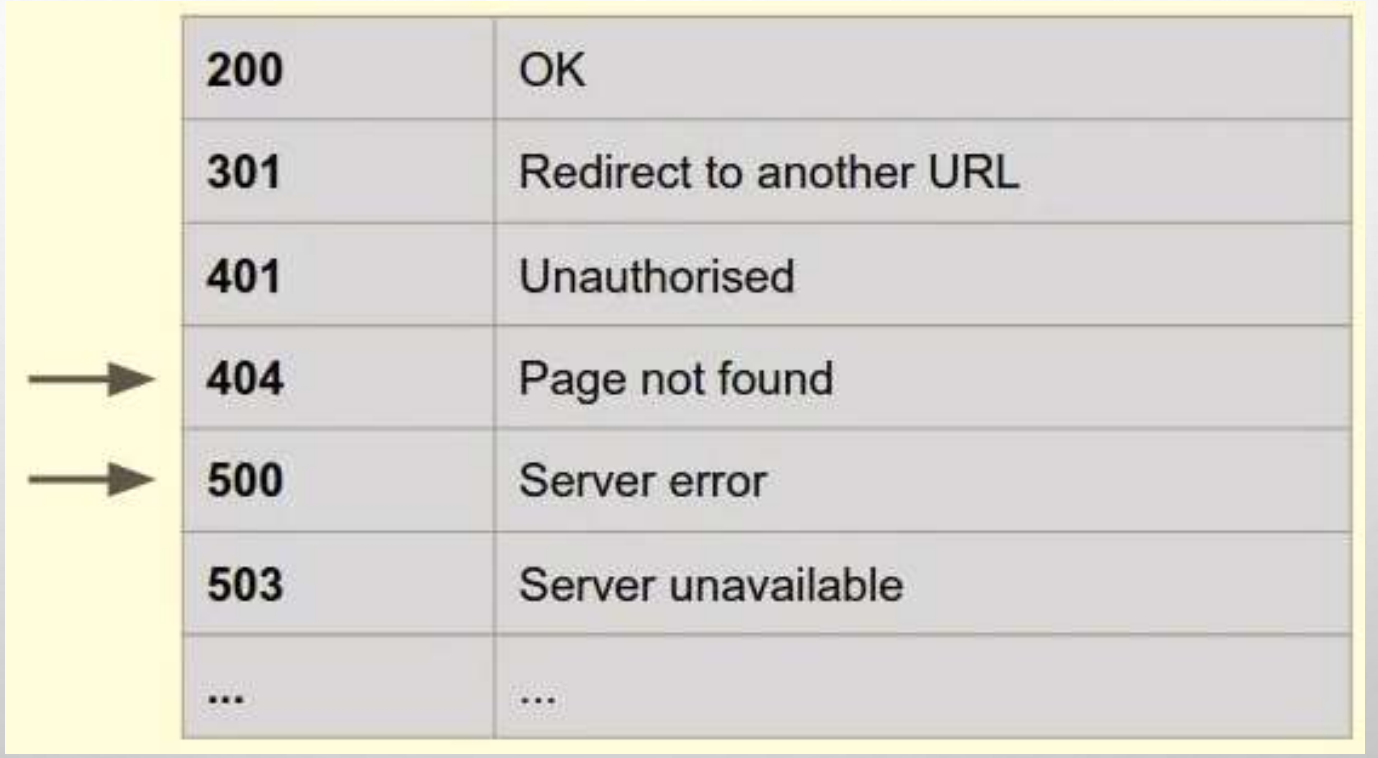

File Browser
www
App
Controllers
Models
Views
Config.php
Core
Controller.php
Error.php
Model.php
Router.php
View.php
logs
2016-01-29.txt
public
Config.php
Error.php
2016-01-29.txt
[29-Jan-2016 09:51:32 Europe/Berlin] Uncaught exception: 'Exception' with message 'No route matched.'
Stack trace: #0 /home/noddy/projects/phpmvc/7-060/public/index.php(47): Core\Router->dispatch('products')
#1 {main}
Thrown in '/home/noddy/projects/phpmvc/7-060/Core/Router.php' on line 133

Categorise different types of error using HTTP status codes

The status of an HTTP request

HTTP status codes

- HTTP has many different status codes that are sent **from the server to the browser**.



200	OK
301	Redirect to another URL
401	Unauthorised
404	Page not found
500	Server error
503	Server unavailable
...	...

Sending status codes with exceptions

- We can throw exceptions **with codes**:

```
throw new Exception("Route not found", 404);
```

- **Get the code** from the exception:

```
$code = $exception->getCode();
```

- **Set** the HTTP response code:

```
http_response_code(404);
```

1) Файл Core\Router.php, метод dispatch

```
        } else {
            throw new \Exception('No route matched.', 404)
        }
```

2) Файл Core\Error.php

```
    public static function exceptionHandler($exception)    {
        // Code is 404 (not found) or 500 (general error)
        $code = $exception->getCode();
        if ($code != 404) {
            $code = 500;
        }
        http_response_code($code);
```

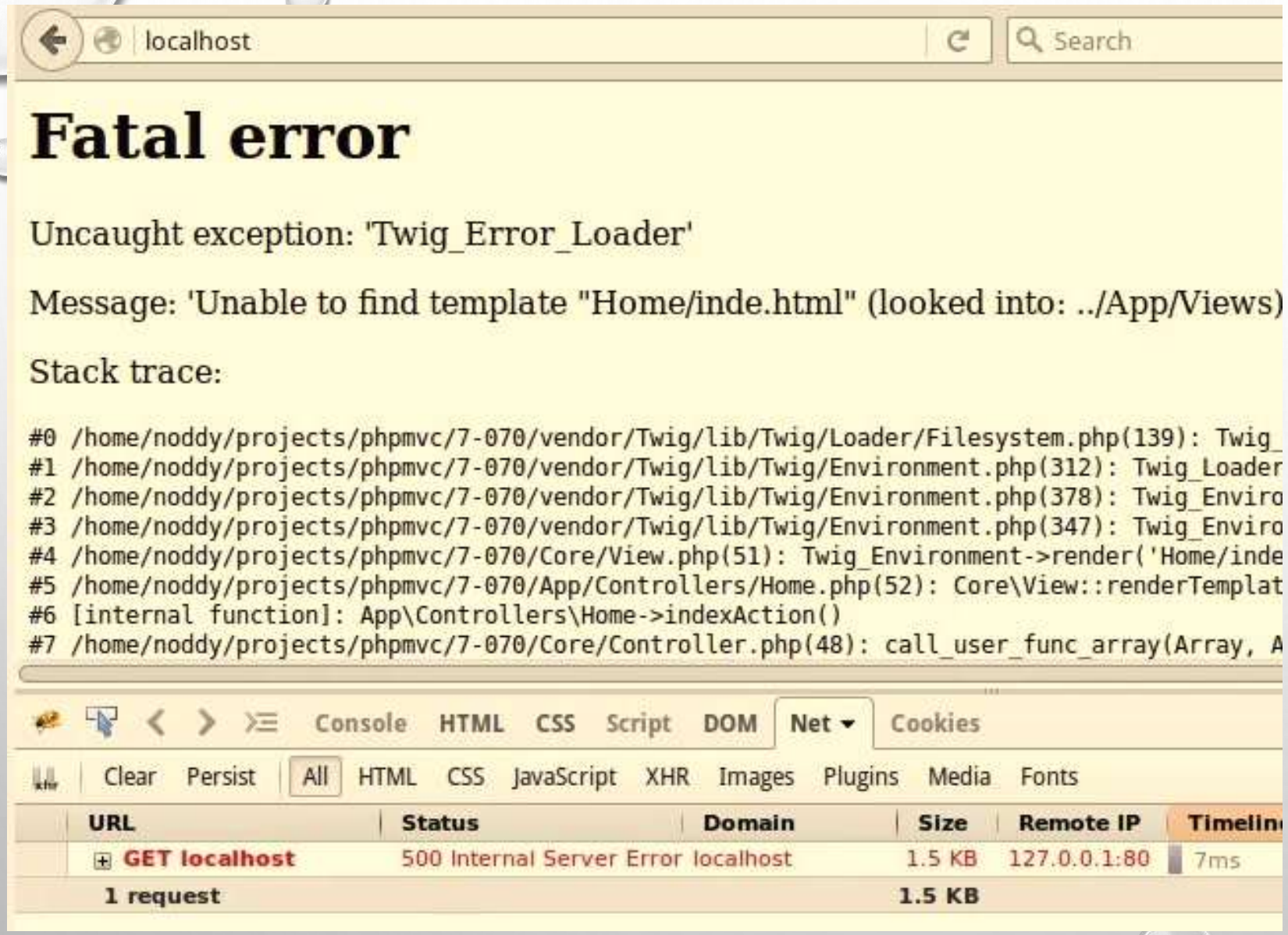
3) Файл Core\Error.php

```
error_log($message);
//echo "<h1>An error occurred</h1>";
if ($code == 404) {
    echo "<h1>Page not found</h1>";
} else {
    echo "<h1>An error occurred</h1>";
}
```


localhost/products
Search
Fatal error
Uncaught exception: 'Exception'
Message: 'No route matched.'
Stack trace:
#0 /home/noddy/projects/phpmvc/7-070/public/index.php(47): Core\Router->dispatch('products')
#1 {main}
Thrown in '/home/noddy/projects/phpmvc/7-070/Core/Router.php' on line 130
Console HTML CSS Script DOM Net Cookies
Clear Persist All HTML CSS JavaScript XHR Images Plugins Media Fonts
URL Status Domain Size Remote IP Timeline
GET products 404 Not Found localhost 229 B 127.0.0.1:80 5ms
1 request 229 B

Якщо в Home контролері зробимо помилку

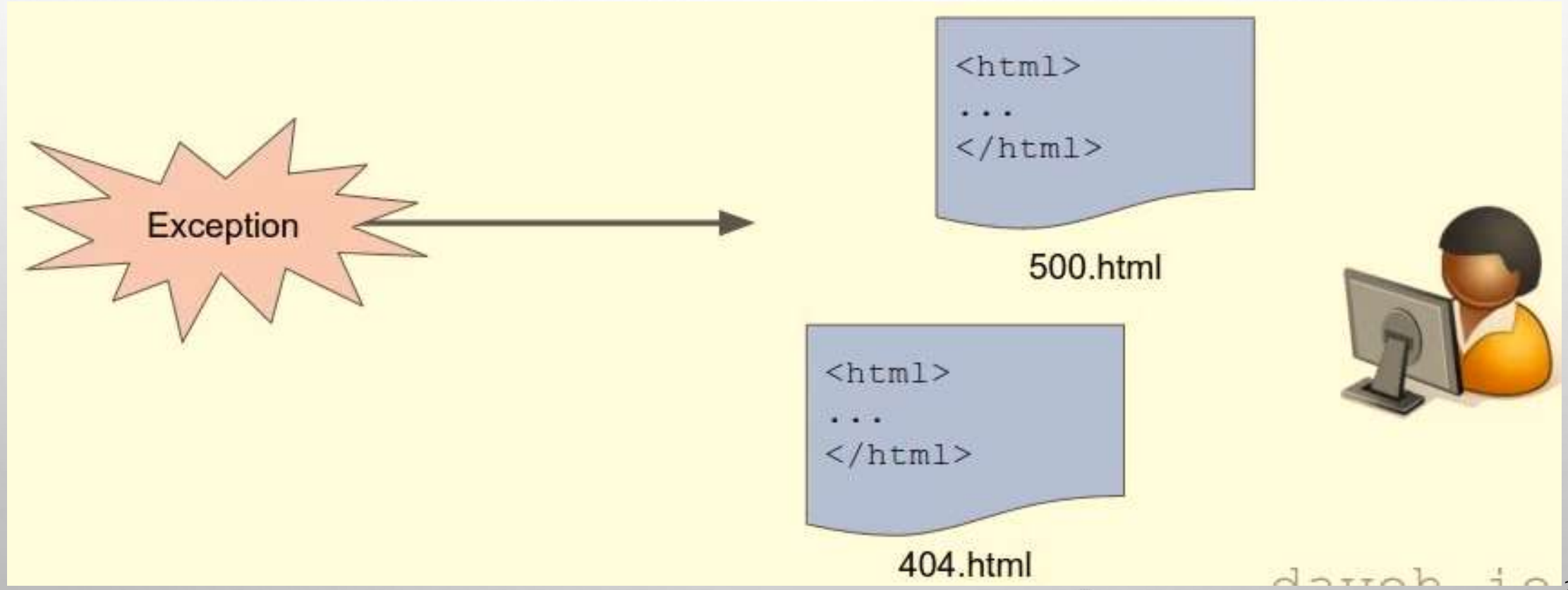
```
public function indexAction()
{
    /*
    View::render('Home/index.php', [
        'name'    => 'Dave',
        'colours' => ['red', 'green', 'blue']
    ]);
    */
    View::renderTemplate('Home/inde.html', [
        'name'    => 'Dave',
        'colours' => ['red', 'green', 'blue']
    ]);
}
```


localhost
Search
Fatal error
Uncaught exception: 'Twig_Error_Loader'
Message: 'Unable to find template "Home/inde.html" (looked into: ../App/Views)
Stack trace:
#0 /home/noddy/projects/phpmvc/7-070/vendor/Twig/lib/Twig/Loader/Filesystem.php(139): Twig_
#1 /home/noddy/projects/phpmvc/7-070/vendor/Twig/lib/Twig/Environment.php(312): Twig_Loader
#2 /home/noddy/projects/phpmvc/7-070/vendor/Twig/lib/Twig/Environment.php(378): Twig_Enviro
#3 /home/noddy/projects/phpmvc/7-070/vendor/Twig/lib/Twig/Environment.php(347): Twig_Enviro
#4 /home/noddy/projects/phpmvc/7-070/Core/View.php(51): Twig_Environment->render('Home/inde
#5 /home/noddy/projects/phpmvc/7-070/App/Controllers/Home.php(52): Core\View::renderTemplat
#6 [internal function]: App\Controllers\Home->indexAction()
#7 /home/noddy/projects/phpmvc/7-070/Core/Controller.php(48): call_user_func_array(Array, A
Console HTML CSS Script DOM Net Cookies
Clear Persist All HTML CSS JavaScript XHR Images Plugins Media Fonts
URL Status Domain Size Remote IP Timelin
GET localhost 500 Internal Server Error localhost 1.5 KB 127.0.0.1:80 7ms
1 request 1.5 KB

Add views to make error pages look nicer in production

Errors in production

- Custom view for **errors in production**



Налаштування для production

1) Файл App\Config.php

   Перевірте, щоб: `const SHOW_ERRORS = false;`

2) Додайте два view templates в каталог View:

   404.html

   500.html

3) Внесіть зміни в файл Error.php

404.html

```
{% extends "base.html" %}
{% block title %}Page not found{% endblock %}
{% block body %}
    <h1>Page not found</h1>
    <p>Sorry, that page doesn't exist.</p>
{% endblock %}
```

500.html

```
{% extends "base.html" %}
{% block title %}Error{% endblock %}
{% block body %}
    <h1>An error occurred</h1>
    <p>Sorry, an error has occurred.</p>
{% endblock %}
```

Зміни в файлі Error.php

```
error_log($message);
//echo "<h1>An error occurred</h1>";
/*          if ($code == 404) {
                echo "<h1>Page not found</h1>";
            } else {
                echo "<h1>An error occurred</h1>";
            }
*/
View::renderTemplate("$code.html");
```

Тестування


localhost
Home | Posts
Welcome
Hello from a Twig template, Dave!
• red
• green
• blue


localhost/products
Home | Posts
Page not found
Sorry, that page doesn't exist.


localhost/posts/nee
Home | Posts
An error occurred
Sorry, an error has occurred.

Conclusion

What you've achieved

- Code organised into **Models**, **Views** and **Controllers**
- Advanced **router**
- Classes in **namespaces** with **autoloading**
- **Controllers** with **action filters**
- **Views** including a **template engine**
- **Models** with **resource-friendly database connectivity**
- Environment-specific **configuration** and **error handling**
- **Understand** how MVC frameworks work

Використані джерела

WRITE PHP LIKE A PRO: BUILD A PHP MVC FRAMEWORK FROM SCRATCH

https://www.udemy.com/php-mvc-from-scratch/