


PHP Blog Project

6. MVC : héritage et autoload

Une fois les 2 derniers contrôleurs implémentés (TP5), nous pouvons constater que du code identique (ou presque) se retrouve dans chaque contrôleur.

Nous allons donc mutualiser le code en créant une classe mère dont tous les contrôleurs hériteront : la classe BaseController

```
app > controllers >  BaseController.php >  BaseController
1  <?php namespace Controllers;
2
3  4 references | 4 implementations
4  class BaseController
5  {
6      2 references
7      public $actionName;
8      4 references
9      protected $params;
10     0 references | 0 overrides
11     public function __construct($routeParts)
12     {
13         $this->actionName = array_shift($routeParts) ?? 'index';
14         if (!method_exists(get_called_class(), $this->actionName)) {
15             header('HTTP/1.0 404 Not Found');
16             die();
17         }
18         $this->params = $routeParts;
19     }
20 }
```

L'attribut \$params doit être protected (et plus private) afin d'être visible dans les classes qui hériteront de la classe BaseController

doc :

<https://www.php.net/manual/en/language.oop5.visibility.php>

Le code de nos contrôleurs s'en trouve raccourcis en héritant de BaseController, comme dans l'exemple ci-dessous pour la classe SeriesController

```
app > controllers > SeriesController.php > SeriesController
1  <?php namespace Controllers;
2
3  class SeriesController extends BaseController
4  {
5
6      0 references | 0 implementations
7      public function index(){
8          echo "<br/>Executing ".get_called_class()." -> ".__FUNCTION__."()";
9      }
10
11      0 references | 0 overrides
12      public function articles(){
13          $id = (int)$this->params[0];
14          if($id < 1) {
15              header('HTTP/1.0 404 Not Found');
16              die();
17          }
18          echo "<br/>Executing ".get_called_class()." -> ".__FUNCTION__."() with id=".$id;
19      }
20  }
```

doc :

<https://www.php.net/manual/en/reflection.extending.php>

Si l'on teste notre code en l'état, une erreur se produit

```
← → ↺ 🏠 ⚠ Non sécurisé php-blog-project.loc/series

Controller Name : Series
Action Name: index
Params : array(0) { }
Fatal error: Uncaught Error: Class 'Controllers\BaseController' not found in C:\w
include_once() #1 C:\wamp64\www\php_blog_project\app\index.php(26): Contro
```

Nous n'avons pas importé la classe BaseController avant de l'utiliser dans chaque classe fille !

Deux possibilités s'offrent à nous :

1. Ajouter un include_once avant chaque utilisation de la classe BaseController dans tous nos contrôleurs

```
app > controllers > SeriesController.php > SeriesController
1  <?php namespace Controllers;
2  include_once "controllers/BaseController.php";
3  class SeriesController extends BaseController
4  {
```

2. Créer un fonction autoload et l'ajouter au début de notre index.php

```
app > index.php > autoload()
1  <?php
2      0 references
3      function autoload($className) {
4          $classFilePath = lcfirst("$className.php");
5          if (file_exists($classFilePath)) {
6              require_once $classFilePath;
7          }
8      }
9      spl_autoload_register("autoload");
10     $route = filter_var(trim($_SERVER["REQUEST_URI"],
```

doc :

<https://www.php.net/manual/en/language.oop5.autoload.php>

L'autoload des classes est bien entendu la méthode à privilégier.

Si nous testons à nouveau (avec des points d'arrêt) sur <http://php-blog-project.loc/series>

```
app > controllers > SeriesController.php > SeriesController
1  <?php namespace Controllers;
2
3  0 references | 0 implementations
4  class SeriesController extends BaseController
5  {
```

Lors de l'appel de la classe BaseController dont hérite SeriesController l'autoload est utilisé

```
app > index.php > autoload()
1  <?php
2      0 references
3      function autoload($className) {
4          $classFilePath = lcfirst("$className.php");
5          if (file_exists($classFilePath)) {
6              require_once $classFilePath;
7          }
8      }
9      spl_autoload_register("autoload");
```

```
RUN AND DEBUG Listen for Xdebug
VARIABLES
  Locals
    $classFilePath = "controllers\BaseController.php"
    $className = "Controllers\BaseController"
  Superglobals
```

Plus d'erreur affichée 😊

```
← → ↻ 🏠 ⚠ Non sécurisé php-blog-project.loc/series

Controller Name : Series
Action Name: index
Params : array(0) { }
Executing Controllers\SeriesController -> index()
```

Grâce à l'autoload, l'import du fichier contenant la classe Router n'est plus nécessaire dans index.php

```
var_dump($params);

// include_once "controllers/Router.php";
$route = new Controllers\Router();
```

Nous profitons du **refactoring du code** que nous avons commencé pour améliorer la classe Router et notamment sortir l'exécution de l'action du contrôleur du constructeur de Router. Un constructeur ne doit pas servir à exécuter des actions, il doit initialiser les variables dont nous avons besoin.

Nous commençons par créer une variable pour stocker le contrôleur à appeler dans la classe Router

```
app > controllers > Router.php > Router
1  <?php namespace Controllers;
2
3  1 reference | 0 implementations
   class Router{
4
5     3 references
   private $controllerInstance;
6
7     1 reference | 0 overrides
   public function __construct()
8     {
```

Puis nous commentons (ou supprimons) l'exécution de la méthode du contrôleur (en bas du constructeur de Router)

```
20     }
21     // include_once $controllerFilePath;
22     $this->controllerInstance = new $controllerClassName($routeParts);
23     // $controllerInstance->{$controllerInstance->actionName}();
24
25 }
```

L'import de la classe du contrôleur (ligne 21) n'est plus nécessaire, nous avons créé un autoload, nous pouvons également le supprimer.

Nous créons une méthode start (dans le routeur) responsable d'exécuter l'action du contrôleur

```
1 reference | 0 overrides
27 public function start(){
28     $action = $this->controllerInstance->actionName;
29     $this->controllerInstance->{$action}();
30 }
31
32 }
```

Enfin nous appelons cette méthode start après avoir instancié notre Router dans le fichier index.php

```
21 var_dump($params);
22
23 // include_once "controllers/Router.php";
24 $router = new Controllers\Router();
25 $router->start();
```

git : https://github.com/DWWM-23526/PHP_BLOG_PROJECT/tree/Step06