PHP Blog Project

13. API Rest : Le routeur et les contrôleurs

La classe Router (1ère partie)

La structure de nos route est la suivante : ***METHOD /table[/:id]***
La route va nous permettre de connaître le contrôleur à instancier en fonction de la table.
La méthode Http employée déterminera quelle action du contrôleur doit être exécutée.
Dans un premier temps nous récupérons ces infos ainsi que l'id s'il est présent dans la route, ce qui nous permettra d'identifier sur quelle ligne en DB nous devons agir pour la lecture, la modification ou la suppression.

api > Controller > Router.php > Router > __construct()

```php
<?php namespace Controller;
    use Core\HttpRequest;
    use Core\HttpReqAttr;

2 references | 0 implementations
class Router{

    1 reference | 0 overrides
    public function __construct()
    {
        $table = HttpRequest::get(HttpReqAttr::ROUTE)[0];
        $method = strtolower(HttpRequest::get(HttpReqAttr::METHOD));
        $id = intval(HttpRequest::get(HttpReqAttr::ROUTE)[1] ?? 0);
        $bp = true;
    }
}
```

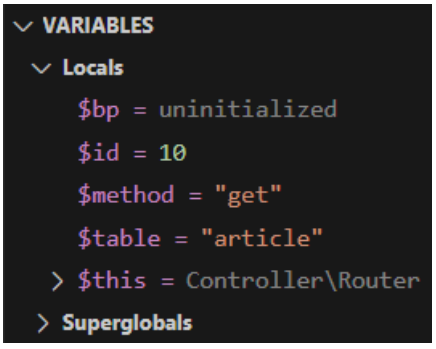
Nous créons une instance de Router dans l'index.php

api > index.php > ...

```php
<?php
use Controller\Router;

0 references
function autoload($className) {
    $classFilePath = "$className.php";
    if (file_exists($classFilePath)) {
        require_once $classFilePath;
    }
}
spl_autoload_register("autoload");

$router = new Router();
```

Nous testons avec ThunderClient

Résultat



Pour pouvoir instancier un contrôleur il faut qu'il existe. Nous allons devoir créer au moins un contrôleur pour poursuivre l'écriture du routeur.

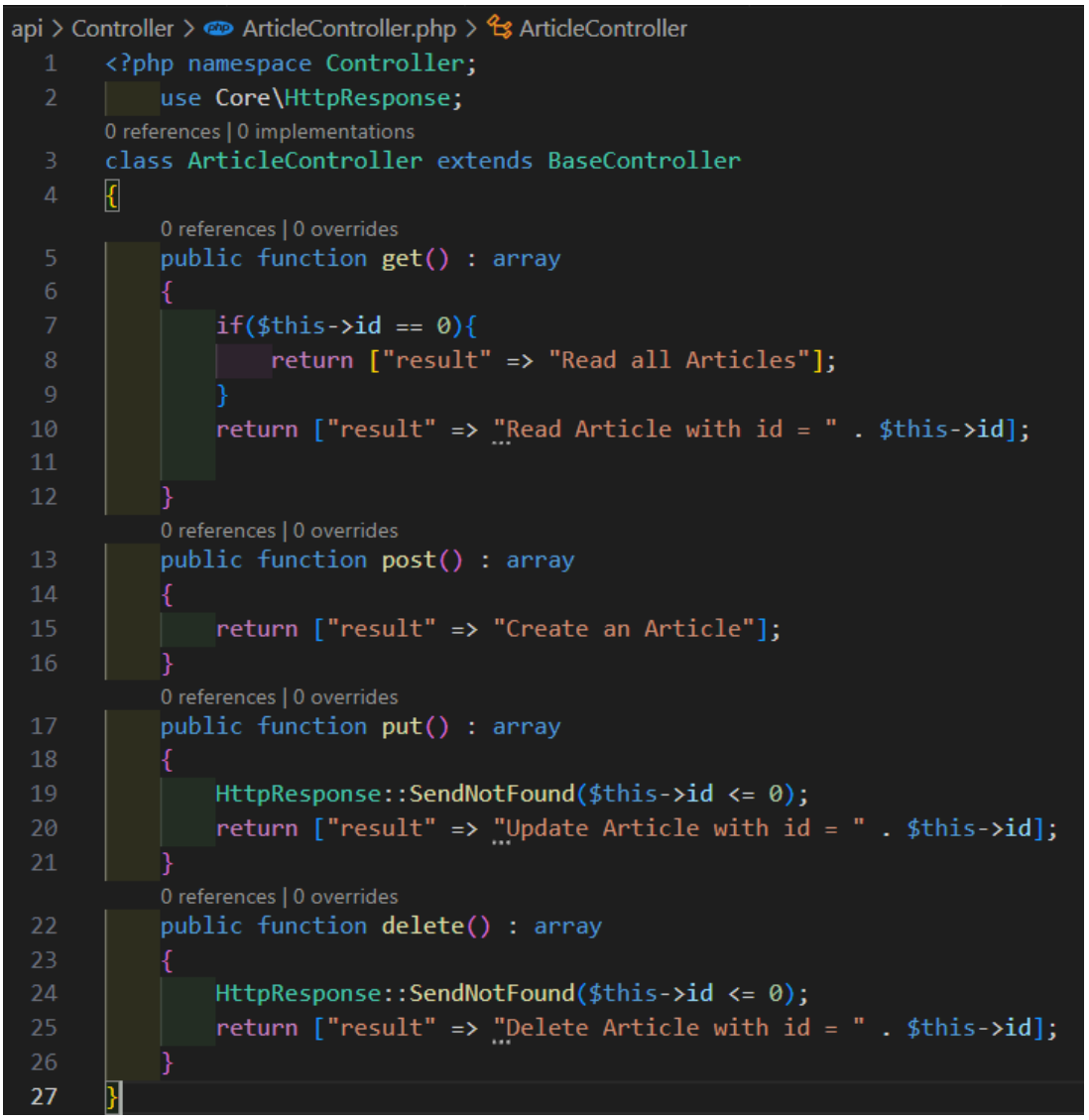
Les classes BaseController et ArticleController

api > Controller > BaseController.php > BaseController

```php
<?php namespace Controller;
    use Core\HttpResponse;
3 references | 2 implementations
class BaseController
{
    3 references
    private string $method;
    7 references
    protected int $id;
    0 references | 0 overrides
    public function __construct($method, $id)
    {
        $this->method = $method;
        $methodNotExists = !method_exists(get_called_class(), $this->method);
        HttpResponse::SendNotFound($methodNotExists);
        $this->id = $id;
    }
    1 reference | 0 overrides
    public function execute() : void
    {
        $result = $this->{$this->method}();
        var_dump($result);
    }
}
```

Le constructeur va récupérer la méthode et vérifier qu'elle est présente dans le contrôleur appelé (ligne 9 à 11)
Nous stockons également l'id dans une propriété protected afin qu'elle soit accessible dans les classes filles (les contrôleurs)
La méthode execute() sera chargée d'appeler la méthode de la classe fille.

Nous créons ensuite la classe ArticleController qui hérite de BaseController avec les 4 méthodes que nous avons définies initialement pour notre API, afin de pouvoir les appeler.



La méthode get servira à la lecture d'une ligne de la table (si l'id est précisé) ou de toutes les lignes de la table.
La méthode post permettra de créer une nouvelle ligne dans la table.
La méthode put permettre de modifier une ligne de la table en fonction de son id.
La méthode delete permettra de supprimer une ligne de la table en fonction de son id.
Pour l'instant nous renvoyons un tableau avec un message.

Nous pouvons maintenant compléter notre routeur

La classe Router (2ème partie)

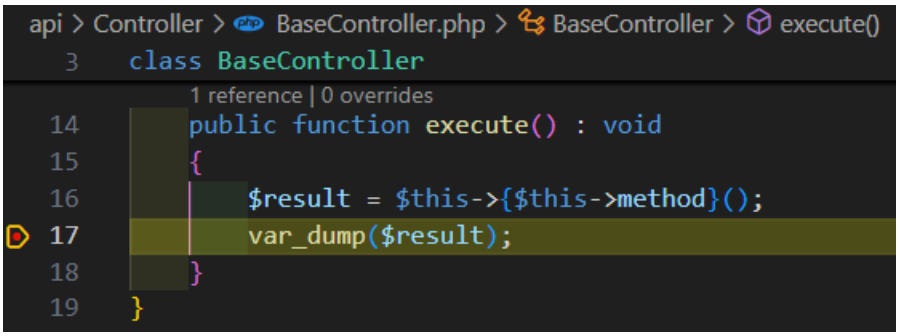
```php
api > Controller > Router.php > Router
<?php namespace Controller;
    use Core\HttpRequest;
    use Core\HttpReqAttr;
    use Core\HttpResponse;

2 references | 0 implementations
class Router{
    3 references
    private BaseController $controllerInstance;
    1 reference | 0 overrides
    public function __construct()
    {
        $table = HttpRequest::get(HttpReqAttr::ROUTE)[0];
        $controllerClassName = "Controller\\".ucfirst(empty($table) ? "Home" : $table)."Controller";
        $controllerFilePath = "$controllerClassName.php";
        HttpResponse::SendNotFound(!file_exists($controllerFilePath));
        $method = strtolower(HttpRequest::get(HttpReqAttr::METHOD));
        $id = intval(HttpRequest::get(HttpReqAttr::ROUTE)[1] ?? 0);
        $this->controllerInstance = new $controllerClassName($method, $id);
    }
    1 reference | 0 overrides
    public function start() : void
    {
        $this->controllerInstance->execute();
    }
}
```

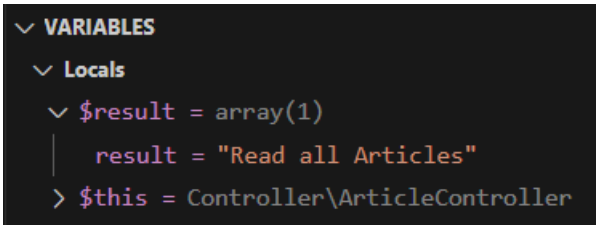
Dans le constructeur, nous vérifions que le fichier correspondant au contrôleur existe (lignes 11 à 13) puis nous créons une instance (ligne 16) que nous stockons dans une variable (ligne 7)
La méthode start permet d'exécuter l'action du contrôleur (voir méthode execute de BaseController)

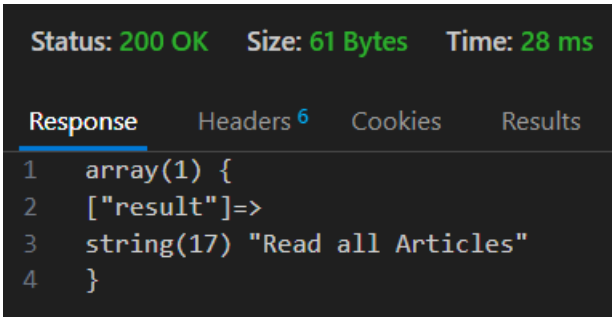
Testons avec Thunder Client (en mode debug)

Route GET /article

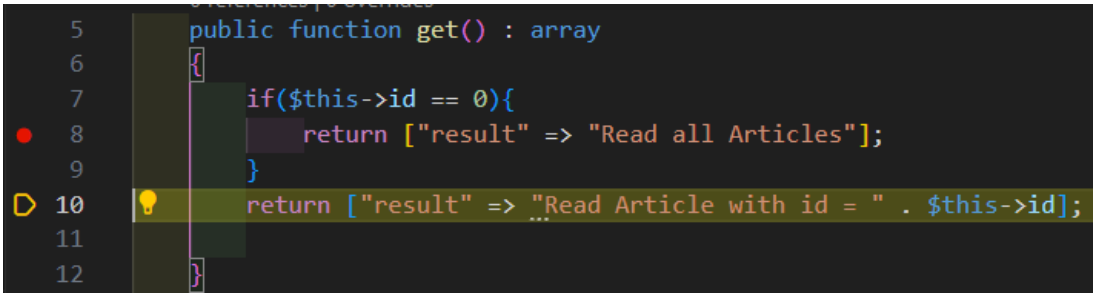

```php
class ArticleController extends BaseController
{
    0 references | 0 overrides
    public function get() : array
    {
        if($this->id == 0){
            return ["result" => "Read all Articles"];
        }
        return ["result" => "Read Article with id = " . $this->id];

    }
```

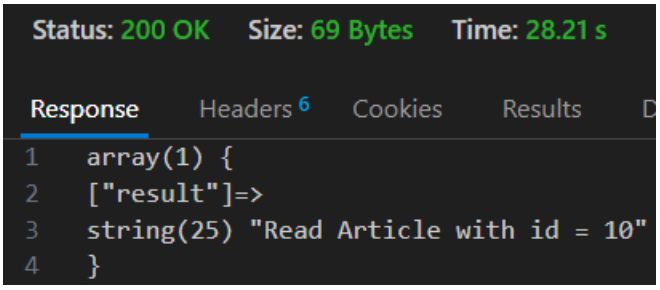






Route GET /article/:id





Route POST /article


```
0 references | 0 overrides
13  public function post() : array
14  {
15      return ["result" => "Create an Article"];
16  }
```

Status: 200 OK Size: 61 Bytes Time: 44 ms

Response Headers 6 Cookies Results

```
1  array(1) {
2  ["result"]=>
3  string(17) "Create an Article"
4  }
```

Route PUT /article/:id


```
17  public function put() : array
18  {
19      HttpResponse::SendNotFound($this->id <= 0);
20      return ["result" => "Update Article with id = " . $this->id];
21  }
```

Status: 200 OK Size: 71 Bytes Time: 21 ms

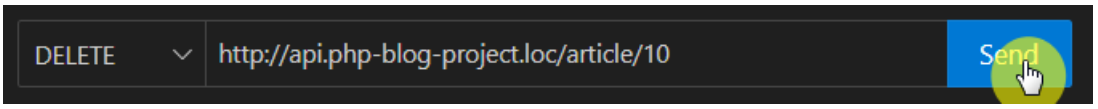
Response Headers 6 Cookies Results Docs

```
1  array(1) {
2  ["result"]=>
3  string(27) "Update Article with id = 10"
4  }
```

Sans id dans la route, nous obtenons une répons 404 Not Found

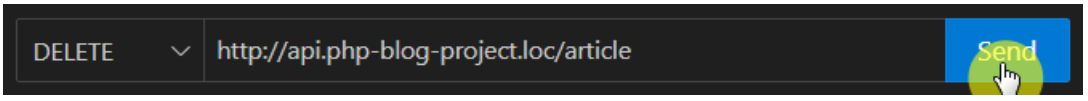
Status: 404 Not Found Size: 0 Bytes Time: 39 ms

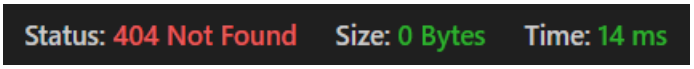
Route DELETE /article/:id




```php
22    public function delete() : array
23    {
24        HttpResponse::SendNotFound($this->id <= 0);
25        return ["result" => "Delete Article with id = " . $this->id];
26    }
```

Sans id dans la route, nous obtenons une répons 404 Not Found





Vous pouvez **créer les contrôleurs pour** les tables **Serie** et **Tech**

git :
https://github.com/DWWM-23526/PHP_BLOG_PROJECT/tree/Step13