

PHP Blog Project

11. API Rest : Structure du projet et classe HttpRequest

Qu'est ce qu'une API REST(full) ?

<https://blog.hubspot.fr/website/api-rest>

<https://openclassrooms.com/fr/courses/6573181-adoptez-les-api-rest-pour-vos-projets-web>

Première définition des routes

Le routeur va devoir exécuter des actions en fonction de la route mais également de la méthode Http utilisée.

GET /table -> récupérer toutes les lignes d'une table

GET /table/:id -> récupéré la ligne ayant pour id :id dans une table


POST /table -> créer une ligne dans une table

PUT /table/:id -> modifier la ligne ayant pour id :id dans une table

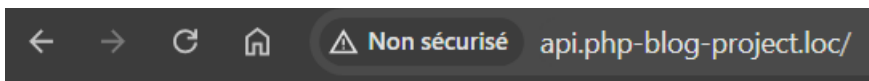
DELETE /table/:id -> supprimer la ligne ayant pour id :id dans une table

Initialisation du projet

Nous commençons par créer un répertoire api ainsi qu'un virtual host pointant vers celui-ci. Nous y ajoutons le fichier .htaccess ainsi qu'un fichier index.php pour tester le virtual host.

```
api >  index.php > ...  
1 <?php  
2  
3 echo 'ok';  
4
```

Résultat



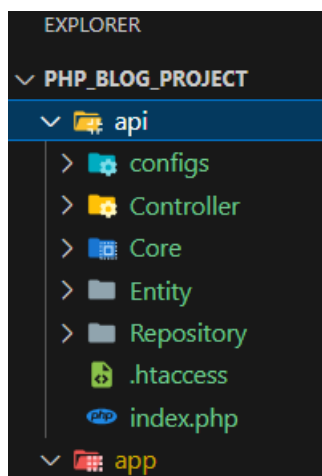
ok

Nous pouvons récupérer les répertoires core, entities et repositories du projet MVC avec leurs classes et les ajouter à notre nouveau projet.

Afin d'améliorer l'autoload (créé dans le projet MVC) nous renommons les répertoires comme suit :

- entities -> Entity
- repositories -> Repository
- core -> Core

Puis nous ajoutons un répertoire Controller ainsi que le dossier configs, à ce stade, voici la structure de notre API :



Nous ajoutons l'autoload (légèrement modifié) dans le fichier racine index.php et vérifions qu'il fonctionne correctement en créant une instance d'une classe existante dans le projet

```
api > index.php > ...
1  <?php
2  use Core\HttpResponse;
   0 references
3  function autoload($className) {
4      $classFilePath = "$className.php";
5      if (file_exists($classFilePath)) {
6          require_once $classFilePath;
7      }
8  }
9  spl_autoload_register("autoload");
10
11  HttpResponse::SendOK("ok");
12
```

Avant d'écrire notre routeur et nos contrôleurs, nous allons créer une classe HttpRequest qui (comme son nom l'indique 😊) sera chargée de stocker les informations relatives à la requête Http.

La classe HttpRequest

Vous allez devoir écrire cette classe dont voici les caractéristiques.

La classe HttpRequest doit permettre de récupérer :

- La méthode Http

ex : GET, POST, etc ...

- Les différentes parties de la route

ex : <http://api.php-blog-project.loc/articles/details/123> -> ["articles", "details", 123]

- Les paramètres de l'url en GET s'ils existent

ex : http://api.php-blog-project.loc/articles?sort=published_at&order=desc

-> ["sort"=>"published_at", "order"=>"desc"]

- Le body contenu dans la requête s'il existe

ex : lors de l'envoi d'un formulaire de connexion le body de la requête correspond aux différentes valeurs du formulaire -> ["login"=>"adress@email.fr", "password"=>"Azerty123"]

La classe doit être un **singleton** car il est inutile de créer plusieurs instances pour une seule et même requête.

Un plus serait de coder une méthode statique get permettant de récupérer les différentes infos stockées dans la classe comme la méthode, la route, etc ...

Vous pouvez tester et déboguer la classe durant son développement sur le fichier index.php.

A vous de coder 😊

git :

https://github.com/DWWM-23526/PHP_BLOG_PROJECT/tree/Step11