# **PHP Blog Project**

## 14. API Rest : Lecture des données (Read)

Avant de nous attaquer au développement des méthodes get de nos contrôleurs qui vont servir à la lecture des données, nous allons faire un peu de ménage dans le code.

Nous commençons par renommer correctement les namespaces des classes de type entity et repository que nous avons récupéré du projet MVC.

## **Entities -> Entity**

```
api > Entity > 	 BaseEntity.php > 	 BaseEntity

1 	 ⟨?php namespace Entity;

2
```

#### Repositories -> Repository

Nous avons légèrement changé les règles de nommage des namespaces et devons adapter certaines méthodes à ces nouvelles règles.

Dans le constructeur de BaseEntity

Dans les méthodes getBaseClassName() et getEntityClassName() de la classe BaseRepository

## Méthode get des classes de type Controller

Si l'id est absent de la route nous lui avons affecté la valeur 0 dans le constructeur de la classe Router (ligne 15)

```
$ $method = strtolower(HttpRequest::get(HttpReqAttr::METHOD));
$ $id = intval(HttpRequest::get(HttpReqAttr::ROUTE)[1] ?? 0) $
$ $this->controllerInstance = new $controllerClassName($method, $id);
```

Nous commençons par traiter la route GET/table

Nous remplissons la condition (ligne 8) et pouvons utiliser la méthode getAll() de la classe ArticleRepository pour récupérer tous les articles

Nous devons convertir notre tableau d'articles au format json avant d'envoyer la réponse à la requête Http, nous modifions légèrement la méthode execute() de notre classe BaseController

#### doc:

https://www.php.net/manual/en/function.json-encode.php

Il ne reste plus que la méthode start() du routeur à modifier pour envoyer la réponse en json

Nous testons avec Thunder Client sur la route GET/article

```
Status: 200 OK Size: 26.68 KB Time: 42 ms
Response
         "title": "nulla ac enim in tempor",
         "summary": "Fusce posuere felis sed lacus. Morbi sem mauris
            laoreet ut, rhoncus aliquet, pulvinar sed, nisl. Nunc
             rhoncus dui vel sem. Sed sagittis. Nam congue, risus
             amet eleifend pede libero quis orci. Nullam molestie
             potenti. Cras in purus eu magna vulputate luctus.",
         "img_src": "https://picsum.photos/id/1/400/300",
         "published_at": "2022-05-09 00:00:00",
         "updated_at": "2023-08-26 00:00:00",
         "is_deleted": true,
         "id_appuser": 8,
         "id_serie": 9,
         "id article": 1
         "title": "faucibus orci luctus",
         "summary": "Suspendisse potenti. Nullam porttitor lacus at
```

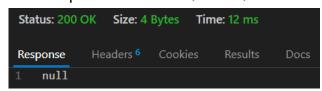
Nous allons maintenant traiter la route GET/table/:id

```
api > Controller > 🐡 ArticleController.php > ધ ArticleController > 🗘 get()
       <?php namespace Controller;</pre>
           use Core\HttpResponse;
  3
           use Entity\Article;
  4
           use Repository\ArticleRepository;
       0 references | 0 implementations
       class ArticleController extends BaseController
           0 references | 0 overrides
           public function get() : array | Article | null
                $articleRepository = new ArticleRepository();
                if($this->id <= 0){
                    $articles = $articleRepository->getAll();
                    return $articles;
                $article = $articleRepository->getOneById($this->id);
                return $article;
 16
```

Si un id est présent dans la route, nous ne rentrons pas dans la condition (ligne 10) et utilisons la méthode getOneById() de la classe ArticleRepository pour récupérer notre article. Pensez à modifier le type de retour de la méthode.

Nous testons avec Thunder Client sur la route GET/article/10

Résultat pour la route GET/article/100



(pas de données avec id=100 dans la table article)

Vous pouvez vous inspirer de la **méthode get** d'ArticleController pour faire celles de **SerieController** et **TechController** (Attention aux copiés/collés  $\bigcirc$  )

Pour préparer le TP 15, essayez de coder par vous même la méthode post() de ces 2 contrôleurs (Serie et Tech)

Pour vous aider, voici la structure d'une requête INSERT MySql en PHP :

```
INSERT INTO $table ($columns) VALUES ($values)
```

Dans BaseRepository, il faut commencer par créer une méthode insert().

Dans cette méthode, il faut (à partir du body de la requête) construire les variables \$columns et \$values et utiliser ensuite la méthode preparedQuery() pour exécuter la requête SQL.

Une fois cela fait, vous pouvez utiliser cette méthode insert() dans la méthode post() du contrôleur pour créer la ligne.

#### git:

https://github.com/DWWM-23526/PHP\_BLOG\_PROJECT/tree/Step14