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 récupérées du projet MVC.

Entities -> Entity

```
api > Entity > BaseEntity.php > BaseEntity
1   <?php namespace Entity;
2
```

Repositories -> Repository

```
api > Repository > BaseRepository.php > BaseRepository
1   <?php namespace Repository;
2       use PDO;
```

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

Dans le constructeur de BaseEntity

```
api > Entity > BaseEntity.php > BaseEntity > __construct()
4   class BaseEntity
        0 references | 0 overrides
6       function __construct($fields = []){
7           $pk = "id_" . lcfirst(str_replace("Entity\\", "", get_called_class()));
8           $this->{$pk} = 0;
9           foreach($fields as $k => $v){
10              if(property_exists($this, $k)){
11                  $this->{$k} = $v;
12              }
13          }
14      }
```

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

```
api > Repository > BaseRepository.php > BaseRepository
5   class BaseRepository
        2 references
38      private function getBaseClassName(){
39          $baseClassName = str_replace("Repository", "", get_called_class());
40          return str_replace("\\", "", $baseClassName);
41      }
        2 references
42      private function getTableName(){
43          return lcfirst($this->getBaseClassName());
44      }
45
        2 references
46      private function getEntityClassName(){
47          return "Entity\\" . $this->getBaseClassName();
48      }
49
```

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)

```
14          $method = strtolower(HttpRequest::get(HttpReqAttr::METHOD));
15          $id = intval(HttpRequest::get(HttpReqAttr::ROUTE)[1] ?? 0);
16          $this->controllerInstance = new $controllerClassName($method, $id);
17      }
```

Nous commençons par traiter la route GET/table

```
api > Controller > ArticleController.php > ArticleController > get()
1   <?php namespace Controller;
2       use Core\HttpResponse;
3       use Repository\ArticleRepository;
        0 references | 0 implementations
4   class ArticleController extends BaseController
5   {
            0 references | 0 overrides
6       public function get() : array
7       {
8           if($this->id <= 0){
9               $articleRepository = new ArticleRepository();
10              $articles = $articleRepository->getAll();
11              return $articles;
12          }
13          return ["result" => "Read Article with id = " . $this->id];
14      }
```

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

```
api > Controller > BaseController.php > BaseController
3   class BaseController
        1 reference | 0 overrides
14      public function execute() : string
15      {
16          $result = $this->{$this->method}();
17          return json_encode($result);
18      }
19  }
```

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

```php
api > Controller > Router.php > Router > start()
6   class Router{
        1 reference | 0 overrides
18      public function start() : void
19      {
20          HttpResponse::SendOK($this->controllerInstance->execute());
21      }
22  }
```

Nous testons avec Thunder Client sur la route GET/article

Status: 200 OK Size: 26.68 KB Time: 42 ms

Response Headers 6 Cookies Results Docs { }

Copy

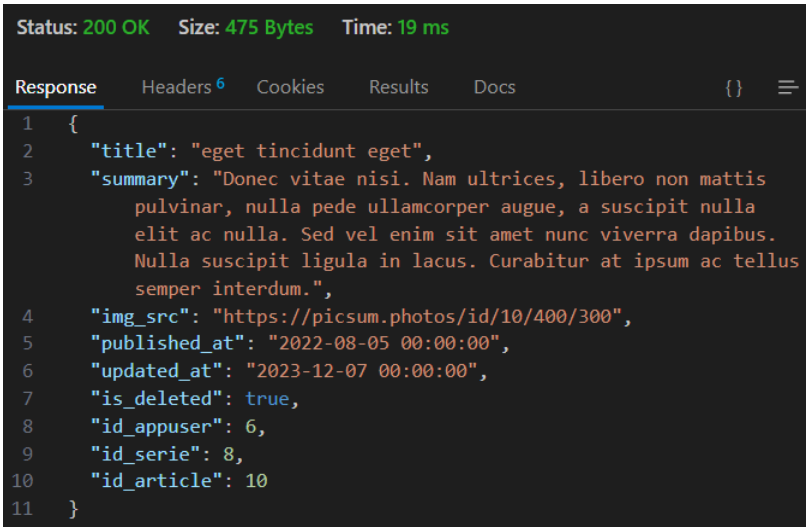

```json
[
  {
    "title": "nulla ac enim in tempor",
    "summary": "Fusce posuere felis sed lacus. Morbi sem mauris laoreet ut, rhoncus aliquet, pulvinar sed, nisl. Nunc rhoncus dui vel sem. Sed sagittis. Nam congue, risus semper porta volutpat, quam pede lobortis ligula, sit amet eleifend pede libero quis orci. Nullam molestie nibh in lectus. Pellentesque at nulla. Suspendisse 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

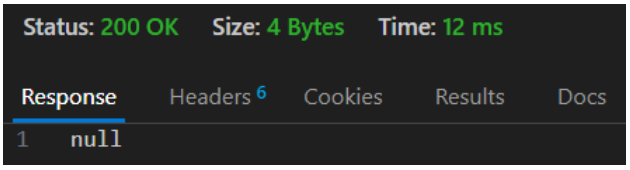
```php
api > Controller > ArticleController.php > ArticleController > get()
1   <?php namespace Controller;
2       use Core\HttpResponse;
3       use Entity\Article;
4       use Repository\ArticleRepository;
    0 references | 0 implementations
5   class ArticleController extends BaseController
6   {
        0 references | 0 overrides
7       public function get() : array | Article | null
8       {
9           $articleRepository = new ArticleRepository();
10          if($this->id <= 0){
11              $articles = $articleRepository->getAll();
12              return $articles;
13          }
14          $article = $articleRepository->getOneById($this->id);
15          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 get.

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 🙂)

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 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.

Cela étant fait, vous pouvez utiliser cette méthode insert() dans la méthode post() du contrôleur pour créer la ligne.

Vous pouvez commencer par écrire la méthode insert() dans les repository fille (Serie, Tech) avant de généraliser et de faire remonter la méthode dans la classe mère BaseRepository

git :
https://github.com/DWWM-23526/PHP_BLOG_PROJECT/tree/Step14