

PHP Blog Project

16. API Rest : Mise à jour des données (Update)

Méthodes get() dans BaseController

Si nous comparons les méthodes get() déjà écrites dans les 3 contrôleurs nous constatons les différences suivantes

- Le type de retour (ligne 7)
- Le repository instancié (ligne 9)

```
api > Controller > ArticleController.php > ArticleController > get()
5  class ArticleController extends BaseController
    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 }
```

```
api > Controller > SerieController.php > SerieController > get()
5  class SerieController extends BaseController
    0 references | 0 overrides
7  public function get() : array | Serie | null
8  {
9      $serieRepository = new SerieRepository();
10     if($this->id <= 0){
11         $series = $serieRepository->getAll();
12         return $series;
13     }
14     $serie = $serieRepository->getOneById($this->id);
15     return $serie;
16 }
```

```
api > Controller > TechController.php > TechController > get()
5  class TechController extends BaseController
    0 references | 0 overrides
7  public function get() : array | Tech | null
8  {
9      $techRepository = new TechRepository();
10     if($this->id <= 0){
11         $techs = $techRepository->getAll();
12         return $techs;
13     }
14     $tech = $techRepository->getOneById($this->id);
15     return $tech;
16 }
```

Pour le reste, seul le nom des variables qui stockent l'instance du Repository (\$articleRepository, \$serieRepository, \$techRepository) et le nom des entités renvoyées par la méthode (\$article, \$serie, \$tech) varient mais au final elles stockent la même chose.

Le type de retour peut être remplacé par BaseEntity qui est la classe mère d'Article, Serie et Tech

Pour remplacer la classe Repository, nous avons besoin d'un nom de classe dynamique. Nous pouvons nous inspirer de ce que nous avons déjà fait dans BaseRepository avec la méthode getEntityClassName()

Nous créons donc 2 méthodes qui vont nous permettre de récupérer le nom de la classe Repository dynamiquement.

```
api > Controller > BaseController.php > BaseController > getRepositoryClassName()
4 class BaseController

1 reference
21 private function getBaseClassName() : string
22 {
23     $baseClassName = str_replace("Controller", "", get_called_class());
24     return str_replace("\\", "", $baseClassName);
25 }
26

2 references
27 private function getRepositoryClassName() : string
28 {
29     return "Repository\\" . $this->getBaseClassName() . "Repository";
30 }
```

Nous pouvons maintenant écrire notre méthode get() dans BaseController à l'aide de la méthode getRepositoryClassName(). Nous modifions également les noms de variable spécialisés avec des termes plus génériques (\$entities, \$entity)

```
api > Controller > BaseController.php > BaseController > get()
4 class BaseController

0 references | 0 overrides
32 protected function get() : array | BaseEntity | null
33 {
34     $repositoryClassName = $this->getRepositoryClassName();
35     $repository = new $repositoryClassName();
36     if($this->id <= 0){
37         $entities = $repository->getAll();
38         return $entities;
39     }
40     $entity = $repository->getOneById($this->id);
41     return $entity;
42 }
```

Il reste à commenter les méthodes get() dans les classes filles et tester avec Thunder Client

Méthodes post() dans BaseController

Sur le même principe que pour les méthodes get() :

```
api > Controller > BaseController.php > BaseController > post()
4  class BaseController
    0 references | 0 overrides
44  protected function post() : array
45  {
46      $repositoryClassName = $this->getRepositoryClassName();
47      $repository = new $repositoryClassName();
48      $insertedEntity = $repository->insert();
49      return ["result" => $insertedEntity];
50  }
```

Commentez également les méthodes post() des classes filles et testez à Thunder Client.

La méthode update de BaseRepository

```
api > Repository > BaseRepository.php > BaseRepository > update()
8  class BaseRepository
    0 references | 0 overrides
99  public function update(int $id) : BaseEntity | false
100  {
101      $tableName = $this->getTableName();
102      $requestBody = HttpRequest::get(HttpReqAttr::BODY);
103      $entityClassName = $this->getEntityClassName();
104      $entity = new $entityClassName($requestBody);
105      $where = "id_{$tableName} = ?";
106      $entityArray = get_object_vars($entity);
107      unset($entityArray["id_{$tableName}"]);
108      $set = implode(",", array_map(function ($item){ return $item."=?"; }, array_keys($entityArray)));
109      $params = array_values($entityArray);
110      array_push($params, $id);
111      $sql = "UPDATE $tableName SET $set WHERE $where";
112      $resp = $this->preparedQuery($sql, $params);
113      if($resp->result && $resp->statement->rowCount() <= 1){
114          $entity = $this->getOneById($id);
115          return $entity;
116      }
117      return false;
118  }
```

Les premières lignes de la méthode servent à construire les variables \$tableName, \$set, \$where et \$params dont nous aurons besoin pour exécuter la requête préparée (ligne 112)
Pour rappel, structure d'une requête SQL insert :

```
UPDATE $tableName SET $set WHERE $where
```

Si la requête s'est bien passé (ligne 113) nous renvoyons l'entité mise à jour, sinon nous renvoyons false.

La méthode put de BaseControlleur

```
api > Controller > BaseController.php > BaseController > put()
4  class BaseController
    0 references | 0 overrides
52  protected function put() : array
53  {
54      $repositoryClassName = $this->getRepositoryClassName();
55      $repository = new $repositoryClassName();
56      $updatedEntity = $repository->update($this->id);
57      return ["result" => $updatedEntity];
58  }
59
```

Très similaire à la méthode post()

A vous de **coder la suppression de données**, les méthodes delete dans le contrôleur et dans le repository (elles ont le même nom)

git :

https://github.com/DWWM-23526/PHP_BLOG_PROJECT/tree/Step16