TD PHP MVC

Étape 12 : Améliorons MainRepository

Nous allons modifier notre template home/index.view.php pour y créer des liens vers une nouvelle route correspondant au détail d'une catégorie.

template > home > index.view.php

```
4  <ul>
5      <?php
6          foreach ($categories as $category) {
7      ?>
8          <li>
9              <a href="/category/read/<?= $category->id?>"><?= $category->name ?></a>
10         </li>
11     <?php
12         }
13     ?>
14 </ul>
```

Pour que ces routes existent réellement (pour qu'elles fonctionnent et ne soient pas redirigées vers error404), nous devons créer un controller et un template.

Nous créons donc un controller : CategoryController avec une méthode read correspondant à la route (/category/read/id)

controller > category.controller.php > CategoryController > read

```
1   <?php
2
3   class CategoryController extends BaseController{
4
5       function read(){
6
7           $repository = new MainRepository("category");
8           $category = $repository->getOne($this->id);
9
10          $this->entities = ['category' => $category];
11
12          $this->render();
13      }
14
15  }
```

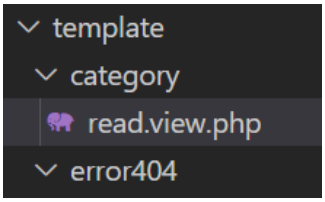
Dans le code ci-dessus, nous instancions la classe MainRepository pour aller chercher des éléments dans la table category (ligne 7)
Puis nous allons utiliser une méthode getOne (que nous allons devoir créer dans la classe MainRepository) à laquelle nous passons l'id contenu dans la route

Nous allons donc modifier la classe MainRepository et lui ajouter une méthode getOne qui servira à récupérer une ligne d'une table grâce à son id.

repository > main.repository.php > MainRepository > getOne

```php
44
45      function getOne($id){
46          $sql = "SELECT * FROM $this->table WHERE id=$id";
47          $resp = $this->connect()->query($sql);
48          $rows = $resp->fetchAll(PDO::FETCH_CLASS, $this->entity);
49          return count($rows) == 1 ? $rows[0] : null;
50      }
51
52  }
```

Il nous reste, pour que les routes « /category/read/id » fonctionnent et ne soient pas redirigées vers error404, a créer le template.

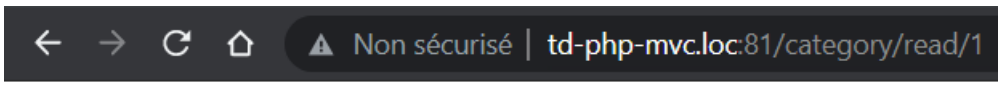


template > category > read.view.php

```php
1   <h1>
2   Détail de la catégorie : <?= $category->name ?>
3   </h1>
```

Nous affichons ici le noms de la catégorie dans un titre h1, en fonction de l'id passé dans l'url (du lien cliqué sur la page /home) nous récupérons la catégorie correspondante :-)

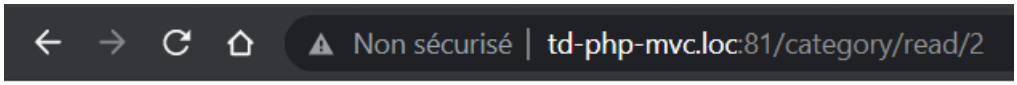
Résultats :



HEADER

Détail de la catégorie : Les Pizza

FOOTER



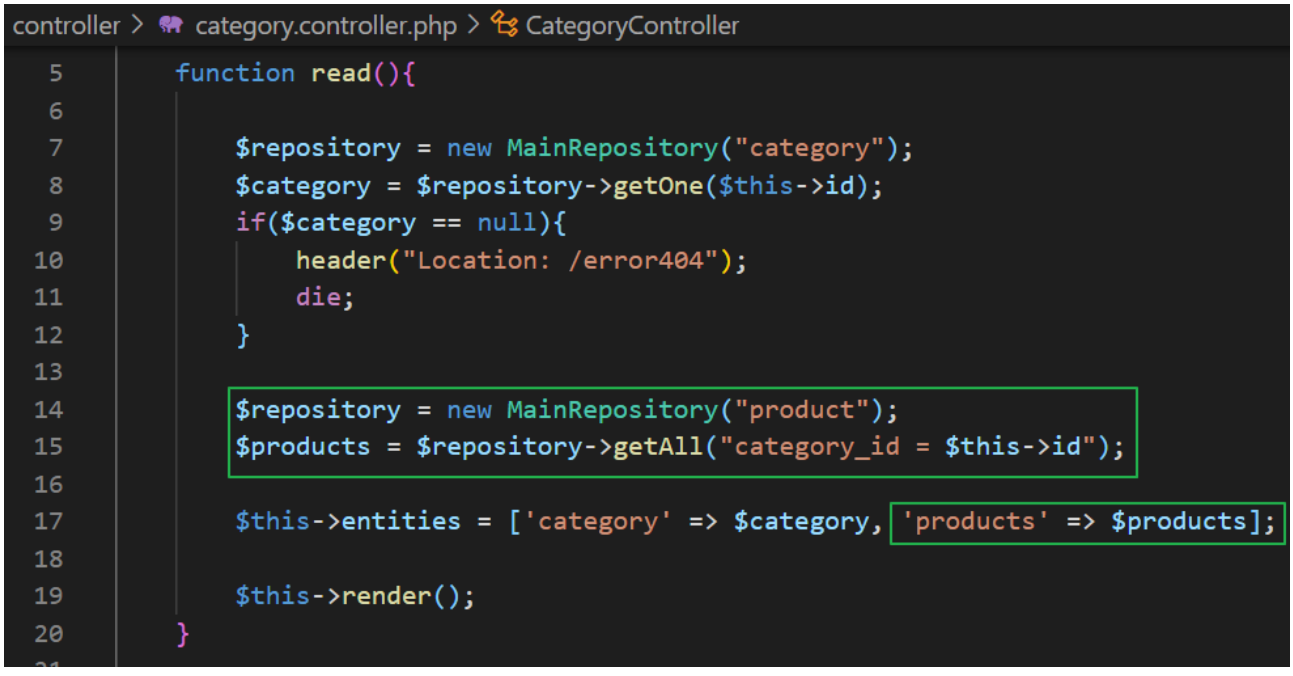
HEADER

Détail de la catégorie : Les Boissons

FOOTER

Si aucune categroy ne correspond à l'id passé dans l'url (par exemple 4), nous avons une belle erreur PHP. Nous modifions donc notre méthode read dans le controller pour faire une redirection vers /error404 dans ce cas.

Pour chaque catégorie, nous allons afficher la liste des produits qui la composent. Nous modifions donc le controller pour récupérer les produits.



Ne pas oublier d'ajouter la liste des produits récupérés aux entities (ligne 17)

Nous avons besoin de modifier la méthode getAll de MainRepository pour ajouter une condition SQL where en paramètre.

repository > main.repository.php > MainRepository > getAll

```php
function getAll($where = "1"){
    $sql = "SELECT * FROM $this->table WHERE $where";
    $resp = $this->connect()->query($sql);
    return $resp->fetchAll(PDO::FETCH_CLASS, $this->entity);
}
```

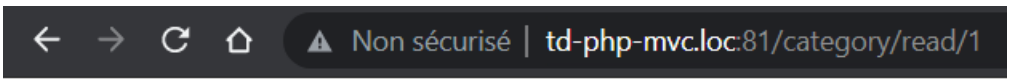
Enfin nous modifions notre template.

template > category > read.view.php

```php
<h1>
Détail de la catégorie : <?= $category->name ?>
</h1>
<h2>
Produits de cette catégorie :
</h2>
<ul>
    <?php
        foreach ($products as $product) {
    ?>
        <li>
            <?= $product->name ?>
        </li>
    <?php
        }
    ?>
</ul>
```

Résultat pour la catégorie des pizza (la seule ayant des produits en base de données) :

(Voir page suivante …)



HEADER

Détail de la catégorie : Les Pizza

Produits de cette catégorie :

- 4 Fromages
- Reine
- Margarita
- Napolitaine
- Regina
- Orientale
- Chèvre-Miel
- Tartiflette
- Ch'ti

FOOTER