TD PHP MVC

Étape 14 : Ajoutons les relations aux Entity

Il existe 4 types de relations entre les tables d'une base de donnée :

- OneToOne
- OneToMany
- ManyToOne
- ManyToMany

Dans notre application, une catégories possède plusieurs produits (OneToMany) et inversement, plusieurs produits appartiennent à une catégorie (ManyToOne)

Nous allons ajouter à notre application la possibilité de récupérer directement les entités correspondantes à ces relations.
Quand nous irons chercher des produits avec le MainRepository en base de données, il sera possible de stocker directement dans nos objets de type Product la Category à laquelle ils appartiennent.
Inversement, lorsque nous irons chercher des catégories avec le MainRepository en base de données, il sera possible de stocker directement dans nos objets de type Category la liste des Product qui la composent.

1ère étape, nous ajoutons dans nos classes Entity un tableau contenant les relations (sous forme de tableau associatif). Cette attribut sera static car il dépend de la classe et pas de ses instances.

PHP static class attributes → https://www.w3schools.com/php/php_oop_static_properties.asp

Pour la classe Product :

entity > product.entity.php > Product

```php
<?php

class Product{

    static $relations =
        ["Category" => ['type'=>'hasOne',
                        'table'=>'category',
                        'attribute'=>'category',
                        'foreignKey'=>'category_id']
        ];
}

```

Pour la classe Category :

entity > category.entity.php > Category

```php
<?php

class Category{

    static $relations =
        ["Products" => ['type'=>'hasMany',
                        'table'=>'product',
                        'attribute'=>'products',
                        'foreignKey'=>'category_id']
        ];
}
```

Une relation (tableau associatif) aura un ***type*** (hasMany, hasOne), une ***table*** dans laquelle aller chercher les objets, la ***foreignKey*** nécessaire pour faire la relation et le nom de l'***attribute*** qui servira à stocker les données.

Nous allons ajouter la possibilité de sélectionner les données à aller chercher avec le MainRepository, se sera une option, activable ou pas en fonction des besoins, des données utiles pour le template.
Nous créons une fonction with dans la classe MainRepository

repository > main.repository.php > MainRepository > with

```php
    function with($name){
        $relationToAdd = $this->entity::$relations[$name];
        array_push($this->relations, $relationToAdd);
        return $this;
    }
```

Nous allons devoir modifier les fonctions getOne et getAll pour aller chercher les données complémentaires en fonction des relations activées à l'aide de cette fonction.

Pour activer cette fonction :

Par exemple dans ProductController, function read (commenté, le code inutile) et encadré en vert ce que nous devons ajouter pour activer l'option « récuperer la catégorie du produit »

(Voir page suivante …)

controller > product.controller.php > ...

```php
function read(){

    $repository = new MainRepository("product");
    $repository->with("Category");
    $product = $repository->getOne($this->id);

    if($product == null){
        header("Location: /error404");
        die;
    }

    // $repository = new MainRepository("category");
    // $category = $repository->getOne($product->category_id);

    $this->entities = ['product' => $product];//, 'category' => $category];

    $this->render();
}
```

Plus besoin de faire de requête sur la table category dans ce controller. Pour récupérer la catégorie du produit : $product→category

Par exemple dans CategoryController, function read (commenté, le code inutile) et encadré en vert ce que nous devons ajouter pour activer l'option « récuperer les produits de la category »

controller > category.controller.php > CategoryController > index

```php
function read(){

    $repository = new MainRepository("category");
    $repository->with("Products");
    $category = $repository->getOne($this->id);

    if($category == null){
        header("Location: /error404");
        die;
    }

    // $repository = new MainRepository("product");
    // $products = $repository->getAll("category_id = $this->id");

    $this->entities = ['category' => $category];//, 'products' => $products];

    $this->render();
}
```

Plus besoin de faire de requête sur la table product dans ce controller. Pour récupérer la liste des produits de la catégorie : $category→product**s**

Nous pouvons modifier les templates :

template > product > read.view.php

```php
<h1>
Détail du produit : <?= $product->name ?>
</h1>
<h2>
Catégorie du produit : <?= $product->category->name ?>
</h2>
<h4>
Description : <?= $product->description ?>
</h4>
```

template > category > read.view.php

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

Il nous reste à modifier getOne et getAll de la classe MainRepository.

(Voir page suivante …)

repository > main.repository.php > MainRepository > getOne

```php
function getOne($id){
    $sql = "SELECT * FROM $this->table WHERE id=$id";
    $resp = $this->connect()->query($sql);
    $rows = $resp->fetchAll(PDO::FETCH_CLASS, $this->entity);
    $row = count($rows) == 1 ? $rows[0] : null;
    if($row == null){
        return null;
    }

    foreach($this->relations as $relation){
        if($relation['type'] == 'hasMany'){
            $repo = new MainRepository($relation['table']);
            $results = $repo->getAll($relation['foreignKey']." = $row->id");
            $row->{$relation['attribute']} = $results;
        }
        if($relation['type'] == 'hasOne'){
            $repo = new MainRepository($relation['table']);
            $results = $repo->getAll("id = ".$row->{$relation['foreignKey']});
            $row->{$relation['attribute']} = count($results) == 1 ? $results[0] : null;
        }
    }

    return $row;
}
```

Nous pouvons tester nos routes category/read/id et product/read/id pour vérifier que nous n'avons rien cassé.

Idem pour le getAll :

repository > main.repository.php > MainRepository > getAll

```php
function getAll($where = "1"){
    $sql = "SELECT * FROM $this->table WHERE $where";
    $resp = $this->connect()->query($sql);
    $rows = $resp->fetchAll(PDO::FETCH_CLASS, $this->entity);
    if(count($rows) == 0){
        return $rows;
    }
    foreach($this->relations as $relation){
        if($relation['type'] == 'hasMany'){
            $repo = new MainRepository($relation['table']);
            $results = $repo->getAll();
            foreach($rows as $row){
                $currentResults = array_filter($results, function($item) use ($row, $relation){
                    return $item->{$relation['foreignKey']} == $row->id;
                });
                $row->{$relation['attribute']} = $currentResults;
            }
        }
        if($relation['type'] == 'hasOne'){
            $repo = new MainRepository($relation['table']);
            $results = $repo->getAll();
            foreach($rows as $row){
                $currentResults = array_filter($results, function($item) use ($row, $relation){
                    return $item->id == $row->{$relation['foreignKey']};
                });
                $row->{$relation['attribute']} = count($currentResults) == 1 ? $currentResults[0] : null;
            }
        }
    }
    return $rows;
}
```

Pour tester la modification du getAll, modifions la route /product

Le controller

controller > product.controller.php > ...

```php
<?php

class ProductController extends BaseController{

    function index(){

        $repository = new MainRepository("product");
        $repository->with("Category");
        $products = $repository->getAll();

        $this->entities = ['products' => $products];

        $this->render();
    }
```

Le template

template > product > index.view.php

```php
<h1>
    Liste des produits
</h1>
<ul>
    <?php
        foreach ($products as $product) {
    ?>
        <li>
            <a href="/product/read/<?= $product->id?>"><?= $product->name ?></a>
            <span> (<?= $product->category->name ?>) </span>
        </li>
    <?php
        }
    ?>
</ul>
```

Nous affichons ici dans la liste des produits la catégorie à laquelle ils appartiennent (et il n'y a que des pizza en base de données)