PROJET DE TECHNOLOGIES DU WEB 2

ANDRIANARIVONY Henintsoa
GOUBEAU Samuel

Rapport de projet :
Création d'un site web marchand

Stud'eCook

L3 Informatique
2020-2021

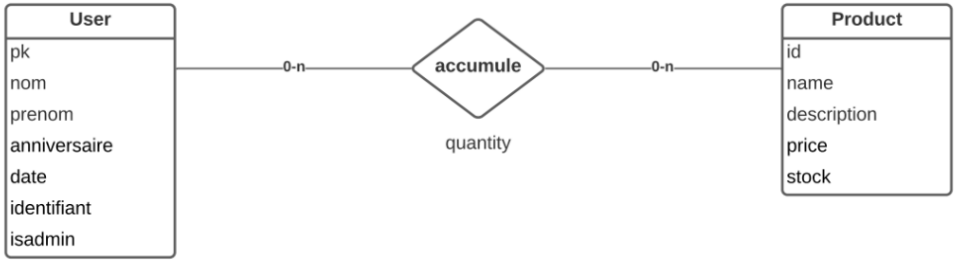
Table des matières

Schéma de la base de données

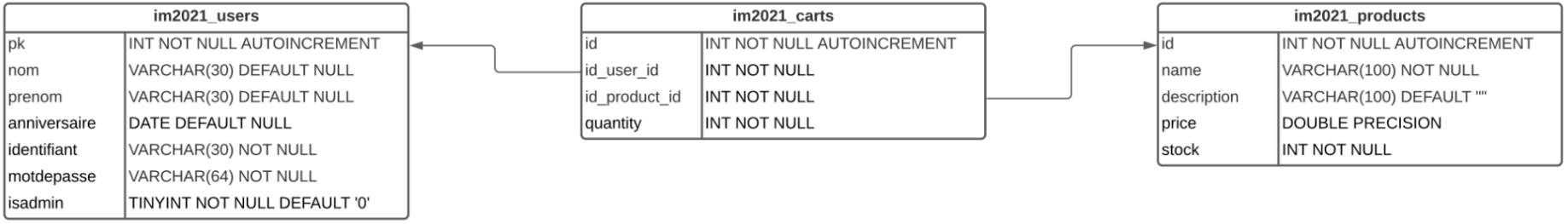
[Projet L3 - WEB] Base de données

Andrianarivony Henintsoa & Goubeau Samuel | April 21, 2021

SEA :



SR :



Organisation du code

Notre code est hiérarchisé selon l'architecture de base d'un site Symfony.
Dans le fichier '**/src**' se trouve tous les fichiers sources du site :

- **/Controller** : Regroupe tous les contrôleurs du site
- **/Entity** : Regroupe tous fichiers .php qui gèrent les entités de la base de données
- **/Form** : Regroupe tous les formulaires du site
- **/Repository** : Regroupe les fichiers .php qui gèrent les répertoires de chaque entités
- **/Service** : Regroupe tous les services du site

Dans le dossier '**/templates**' se trouvent chacune de nos vues. Et chacune d'entre elle est rangée dans un dossier correspondant au type de page qu'elle affichera (sauf pour base.html.twig, afin de respecter l'architecture à 3 niveaux) :

- **/account** : Chacune des vues qui référence une page en rapport avec le compte utilisateur (connexion, inscription, etc.)
- **/cart** : Unique vue affichant le panier d'un utilisateur
- **/common** : Unique vue affichant la page d'accueil
- **/main** : Chacune des vues gérant l'aspect global du site (bandeaux, menu, etc)
- **/product** : Chacune des vues gérant la table Products (Ajout et affichage des produits disponibles)

Puis dans le dossier '**/public**' se trouvent les fichiers de styles graphiques pour le site :

- **/css** : Regroupe toutes les feuilles de styles du sites
- **/img** : Regroupe toutes les images présentes sur le site

Liste des classes :

- **UtilityController** : Classe mère de tous les autres contrôleurs. C'est cette dernière qui hérite d'**AbstractController**. Cette conception nous permet de simplifier notre code, et de limiter un maximum les répétitions grâce à la création de méthode :
  - **AccountController**
  - **BannerTopController**
  - **CartController**
  - **ConnectionController**
  - **HomeController**
  - **MenuController**
  - **ProductController**

- Des classes servant d'entité pour les tables correspondantes dans la base de données :

    - **Carts**
    - **Products**
    - **Users**

- Des classes héritant d'**AbstractType** pour pouvoir construire les formulaires :
    - **ProductType**
    - **UserType**

- Une classe pour construire un service :
    - **CountProducts**

Explication/Tutoriel : création d'un service sous Symfony

Pour commencer, il faut créer un dossier '/src/Service'. C'est ici que tous nos services vont résider. Dans ce tutoriel, nous allons créer un service qui permet de compter le nombre de produits total contenu dans la base de données.

Créez le fichier 'CountProducts.php'. En début de fichier, mettez la ligne :

```
namespace App\Service ;
```

Puis créez votre classe CountProducts, avec la méthode countProducts($products) qui va se charger de faire ce qu'on lui ordonne, soit de compter le nombre de produits. Mais libre à vous de rajouter ou non d'autres méthodes, mais ce n'est pas le propos ici.

Et voilà ! Vous venez de créer votre service ! Il ne vous reste plus qu'à l'utiliser dans vos contrôleurs. Il existe pour ça deux méthodes, soit vous appelez directement le service dans votre contrôleur, soit vous passez par un alias.

Pour la première c'est simple, il suffit d'ajouter le 'use' correspondant au service (**use App\Service\CountProducts**) en début de votre contrôleur, puis de créer une instance de notre service dans la méthode où l'on veut appeler notre service.

Cette manière de faire est pratique si vous n'avez que peu de méthodes utilisant le service. Mais si il est plus souvent sollicité, cela peut très vite devenir contraignant, voire problématique. Et c'est pourquoi la seconde méthode existe !

Pour créer un alias correspondant à votre service, il faut passer par le fichier 'service.yaml' se trouvant dans le dossier '/config'. Puis descendre jusqu'à trouver la ligne '**services:**'.

C'est dans cette section que vous allez devoir rajouter ces lignes :

```
services:
```

```
  #...
  App\Service\CountProducts:    # On précise le chemin du service
     public: false                 # On rend le service privé, et donc il ne peut plus
être instancié dans d'autres classes

  app.countproducts :                  # On donne un nom à notre alias
    alias: App\Service\CountProducts  # On précise ce que référence notre alias
    public: true                       # On rend l'alias public, par conséquent
l'appelle de ce dernier est possible
```

Maintenant que notre alias a été créé, pour l'appeler il suffit d'ajouter dans notre contrôleur la ligne :

```
$countProducts = $this->get('app.countproducts') ;
```

Et vous avez votre instance vous permettant d'appeler la méthode countProducts($products) autant de fois que vous le souhaitez.

Points Particuliers

- **UtilityController** :

  Pour une question de simplification et de pureté de code, nous avons décidé de créer une classe intermédiaire entre **AbstractController** et les autres contrôleurs. Le but de cette conception est, comme dit dans la partie de l'organisation du code, de limiter un maximum la répétition de code. Une méthode qui peut illustrer ça est *setRestriction($type)*. Cette dernière permet de lever une erreur lorsqu'un utilisateur se retrouve dans une page dans laquelle il ne doit pas être. Par conséquent, il faut l'appeler pour chaque action qui affiche une page et rajouter 28 lignes d'un même code plusieurs fois. Cela est inutile.

  Nous avons pensé à utiliser un service pour effectuer ceci, cependant cette méthode requiert l'utilisateur courant. Or, on ne peut faire cela qu'avec une instance d'**AbstractController**, ce qui empêche de passer par cette méthode.

- **Problèmes** :
  - **Alias d'un service** :

    Pour simplifier l'appel d'un service dans un contrôleur, nous voulions utiliser un alias plutôt que de créer une instance brute de celui-ci. Cependant, et même en suivant la documentation Symfony en ligne sur le sujet (cf tutoriel p.5), symfony lève une erreur et ne reconnaît pas l'alias utilisé.