Université de Poitiers

# Rapport Projet Web : Création d'un site marchand avec le framework Symfony

Auteur(s) : Florian Legendre

# Légendes et Abbréviations utilisées

```
1 Ceci est du code source.
2 Selon les langages, différents mots seront colorés selon
3 si ce sont des mots clefs ou non (comme int, char, etc.).
```

**Listing 1** – Exemple de code source

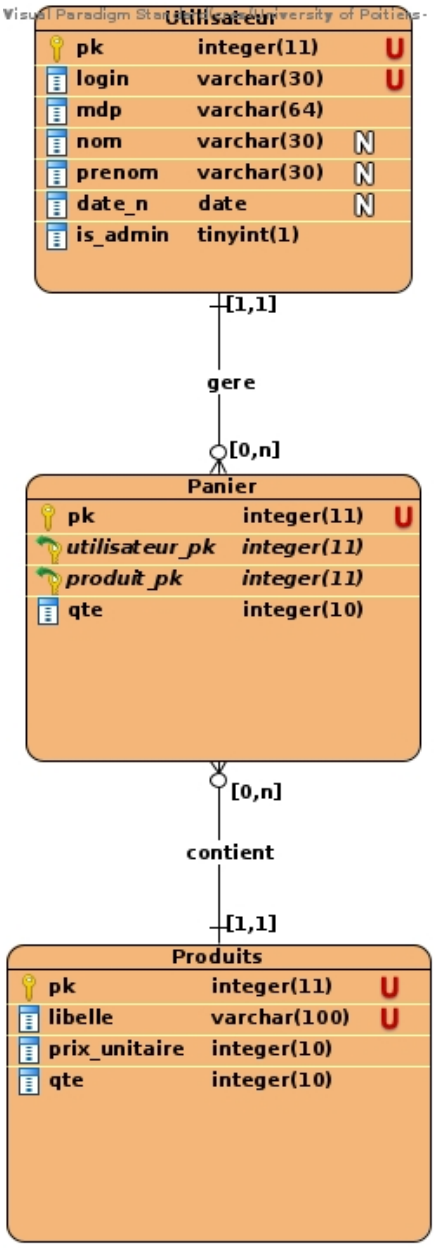
```
Ceci est un formattage automatique Latex d'un texte copié-collé
directement depuis un terminal Bash ayant valeur de capture
d'écran. La coloration correspond à une coloration quelconque
d'un terminal Bash (les chemins étant habituellement coloré et
le nom de l'utilisateur aussi comme crex@crex:~$ ...)
```

**Listing 2** – Exemple d'une pseudo capture d'écran Bash

# Table des matières

# Chapitre 1

# Schémas de la base



# Chapitre 2

# Organisation du code

## Présentation de l'arborescence

```
.
├── codes
│   ├── bin
│   │   └── console
│   ├── composer.json
│   ├── composer.lock
│   ├── config
│   │   ├── bundles.php
│   │   ├── packages
│   │   │   ├── cache.yaml
│   │   │   ├── dev
│   │   │   │   ├── debug.yaml
│   │   │   │   ├── monolog.yaml
│   │   │   │   ├── swiftmailer.yaml
│   │   │   │   └── web_profiler.yaml
│   │   │   ├── doctrine_migrations.yaml
│   │   │   ├── doctrine.yaml
│   │   │   ├── framework.yaml
│   │   │   ├── prod
│   │   │   │   ├── deprecations.yaml
│   │   │   │   ├── doctrine.yaml
│   │   │   │   ├── monolog.yaml
│   │   │   │   └── routing.yaml
│   │   │   ├── routing.yaml
│   │   │   ├── security.yaml
│   │   │   ├── sensio_framework_extra.yaml
│   │   │   ├── swiftmailer.yaml
│   │   │   ├── test
│   │   │   │   ├── doctrine.yaml
│   │   │   │   ├── framework.yaml
│   │   │   │   ├── monolog.yaml
│   │   │   │   ├── swiftmailer.yaml
│   │   │   │   ├── twig.yaml
│   │   │   │   ├── validator.yaml
│   │   │   │   └── web_profiler.yaml
│   │   │   ├── translation.yaml
│   │   │   ├── twig.yaml
│   │   │   └── validator.yaml
```

```
│   │   ├── preload.php
│   │   ├── routes
│   │   │   ├── annotations.yaml
│   │   │   └── dev
│   │   │       ├── framework.yaml
│   │   │       └── web_profiler.yaml
│   │   ├── routes.yaml
│   │   └── services.yaml
│   ├── migrations
│   │   ├── Version20210330093836.php
│   │   ├── Version20210330094836.php
│   │   ├── Version20210330103614.php
│   │   ├── Version20210330104417.php
│   │   └── Version20210330105034.php
│   ├── public
│   │   ├── css
│   │   │   ├── base.css
│   │   │   ├── secondlevel.css
│   │   │   └── welcome.css
│   │   ├── image
│   │   │   ├── admin.png
│   │   │   ├── customers.jpg
│   │   │   ├── logo.png
│   │   │   ├── unicorn.png
│   │   │   └── visitors.jpg
│   │   └── index.php
│   ├── sqlite
│   │   └── data.db
│   ├── src
│   │   ├── Controller
│   │   │   ├── MainController.php
│   │   │   ├── PaniersController.php
│   │   │   ├── ProduitsController.php
│   │   │   └── Utilisateurs
│   │   │       ├── AdminsController.php
│   │   │       ├── ClientsController.php
│   │   │       ├── UtilisateursController.php
│   │   │       └── VisiteursController.php
│   │   ├── DataFixtures
│   │   │   └── AppFixtures.php
│   │   ├── Entity
│   │   │   ├── Panier.php
│   │   │   ├── Produit.php
│   │   │   └── Utilisateur.php
│   │   ├── Form
│   │   │   ├── AddProductType.php
│   │   │   ├── ClientProfilType.php
│   │   │   └── CreateAccountType.php
│   │   ├── Kernel.php
│   │   ├── Repository
│   │   │   ├── PaniersRepository.php
│   │   │   ├── ProduitsRepository.php
│   │   │   └── UtilisateursRepository.php
│   │   └── Service
│   │       ├── GlobalUserService.php
│   │       ├── ProduitsService.php
```

```
│   │   └── SecretService.php
│   ├── symfony.lock
│   ├── templates
│   │   ├── Layouts
│   │   │   ├── base.html.twig
│   │   │   ├── footer.html.twig
│   │   │   ├── header.html.twig
│   │   │   ├── index.html.twig
│   │   │   ├── level2.html.twig
│   │   │   └── menu.html.twig
│   │   ├── Produits
│   │   │   ├── product_list.html.twig
│   │   │   └── product_mail.html.twig
│   │   └── Utilisateurs
│   │       ├── Admins
│   │       │   ├── add_product.html.twig
│   │       │   ├── admin_header.html.twig
│   │       │   ├── admin_menu.html.twig
│   │       │   └── manage_users.html.twig
│   │       ├── Clients
│   │       │   ├── basket.html.twig
│   │       │   ├── customer_header.html.twig
│   │       │   ├── customer_menu.html.twig
│   │       │   └── manage_profil.html.twig
│   │       └── Visiteurs
│   │           ├── create_customer_account.html.twig
│   │           ├── visitor_header.html.twig
│   │           └── visitor_menu.html.twig
│   ├── translations
│   ├── var    [...]
│   └── vendor [...]
├── docs
│   ├── #
│   │   └── documents.tar.gz
│   ├── documents_etu
│   │   ├── allerDansMagasin.png
│   │   ├── clientsSR2.jpg
│   │   ├── clientsSR.jpg
│   │   ├── clientsSR.vpp
│   │   ├── gererPanier.png
│   │   ├── pageAccueil.png
│   │   ├── pageAccueil.svg
│   │   ├── pageAllerDansMagasin.png
│   │   ├── pageAllerDansMagasin.svg
│   │   ├── pageGererPanier.png
│   │   ├── pageGererPanier.svg
│   │   ├── pageSiteStandarde.png
│   │   ├── pageSiteStandarde.svg
│   │   ├── schemaRelationnel_ClientsProduits.png
│   │   └── TODO
│   ├── documents_UP
│   │   ├── OREADME
│   │   ├── im2021_utilisateurs.sql
│   │   ├── liste_produits.html
│   │   ├── ls-R.txt
│   │   ├── panier.html
```

```
│   │   └── verif_archive.sh
│   ├── ls-R.txt
│   └── projet.pdf
├── librairies
│   ├── composer.phar
│   ├── recup_composer.sh
│   ├── recup_symfony.sh
│   ├── set_rights.sh
│   └── symfony
└── README.md

2298 directories, 8461 files
```

## Controller

Le MainController permet de s'occuper des menus et des headers à afficher selon l'utilisateur connecté. La classe PanierController gère, comme son nom l'indique, le panier, sa création. Le ProduitController crée les produits.

Les controllers concernant les utilisateurs sont séparés du reste pour une meilleure lisibilité. L'UtilisateursController met en place la déconnexion et la création en dur d'un utilisateur. VisteursController n'est utilisé que pour se connecter (de façon artificielle) ou créer un compte, ce qui le fera devenir un client. Avec ClientsController, il est possible de modifier son profil mais aussi de gérer son panier. La particularité de l'AdminController est qu'il peut supprimer un client de la base.

## Form

Les formulaires permettent d'afficher les données de la base pour les enregistrer\modifier et les injecter dans la base. C'est pour cela que nous les utilisons pour ajouter un nouveau produit dans la base, modifier le profil d'un client ou le supprimer. Ce sera au controlleur d'appliquer ces modifications si le formulaire est correctement remplis.

## Service

Les services sont des objets. Ils servent à générer des méthodes utilisables par tous les controllers. GlobalUser nous sert pour régler des problèmes de sécurité comme le cas où un utilisateur souhaite accéder à des fonctionnalités qui lui sont interdites. La classe SecretService affichera un message inversé choisi aléatoirement. Quant à la classe ProduitsService, elle permet de récupérer tous les produits et d'en connaître le nombre total.

## Template

A la base du site, nous avons base.html.twig et level2.html.twig. Toutes les vues en découleront. Dans le fichier Layouts, nous avons aussi le footer ainsi que le menu et le header qui, selon l'utilisateur connecté, afficheront un menu et une bannière différente. Ces deux vues utilisent un if\then\else classique sur l'utilisateur inscrit dans services.yaml avec parameters pour déterminer ses autorisations.

Les vues sont réparties en différents dossiers. Il y a les vues des utilisateurs (admin, client, visiteur) et la liste des produits. Les utilisateurs ont chacun leurs vues spécifiques et celles qui permettent une visulation des formulaires.

# Chapitre 3

# Tutoriel de création d'un service sous Symfony

## Création d'une nouvelle classe

Il est posible d'implanter un service directement dans le controlleur mais cela empêche la réutilisation du code. C'est pour cela qu'il est préférable de créer une nouvelle classe dans le fichier src/Service. Vous pourrez ensuite, à l'intérieur de cette classe, écrire toutes vos méthodes.

## Appel du service dans le controlleur

Une fois votre service créé, il ne reste plus qu'à en faire usage dans le controlleur de votre choix. En procédant comme ceci :

1. On importe le service dans le contrôleur : use App\Service\Nom_du_service
2. Dans les paramètres de l'action du contrôleur où on souhaite utiliser une des méthodes du service, injectez votre dépendance. Exemple : public function monAction(MonService $identifiantDeMonServiceDansAction)
3. Enfin, vous pouvez l'appeler dans votre contrôleur : $identifiantDeMonServiceDansAction->methode_du_service().

# Chapitre 4

# Commentaires Divers

## Action d'édition du profil client : gestion des violations de contraintes

Florian Legendre : Lorsqu'une contrainte d'intégrité qui n'est pas un Callback() est violée cela entraîne une exception dans le navigateur. Comme je voulais que cette exception soit gérée par un message d'erreur du formulaire j'ai donc écrit ces lignes :

```
1  //On gère le formulaire:
2          $form->handleRequest($request);
3          if ($form->isSubmitted() && $form->isValid())
4          {
5              try {
6                  $this->em->flush();
7                  $this->addFlash('info', "Your profile has been edited!");
8                  return $this->redirectToRoute('produits_liste');
9              }
10
11             //TODO : Si plusieurs champs pouvant être uniques? Si plusieurs
       types d'exceptions?
12             catch(\Exception $e)
13             {$form->addError(new FormError("Sorry! This login already
       exists."));}
14         }
```

Dans le commentaire TODO j'indique ne pas savoir comment mieux gérer ces exceptions. J'aimerais ainsi pouvoir : identifier quel attribut a entraîné la levée de l'exception pouvoir avoir une gestion d'erreur différenciée.