SYMFONY 5 commerce part-1

FRAMEWORK PHP

Créez un nouveau projet

- Vous savez comment le faire, créez un projet boutique en ligne du nom de votre choix.
- Créez la base de données
- Modifiez le fichier .env

Intégrer bootstrap au projet

- Créez un dossier assets dans public et dans votre dossier assets créez 2 dossiers : JS et CSS
- Je vous fournis un zip et je vous laisse tout ranger à sa place, n'oubliez pas le base.html

C’est partie !

- Créez la homePage, pour ce faire :
- Créez le contrôleur « home » (vous savez maintenant comment faire)

Asset / homePage

- Pour rajouter des assets il faut faire comme ceci en twig :

```
<link href="{{asset("assets/css/bootstrap.css")}}" rel="stylesheet" />
```

- Je vous laisse remplacer le css avec la fonction asset.
- Vous allez également modifier et laisser ce qui est propre au template dans base.html et mettre le reste dans l'index de home.
- Créez un block pour la balise title.

Création entité User

```
PS C:\wamp64\www\laBoutiqueDeBenjamin> symfony console make:user

 The name of the security user class (e.g. User) [User]:
 > 
```

- Le texte entre [] est le nom de la class par défaut.
- Vous allez du coup tout laisser par défaut et faire entrer à chaque fois.
- Que s'est il passé ?

Doctrine

- Cherchez ce qu'est doctrine
- Qu'est ce qu'une migration en symfony avec doctrine?

Doctrine

- A l'aide de doctrine nous allons créer la table correspondante à l'entité User

Devinette :

- Quel commande pour créer la migration ?
- Quel commande pour la lancer ?

Doctrine

- **make:migration**

Créer la migration

- **doctrine:migrations:migrate**

Execute la migration par rapport à la dernière version disponible

Register

- **Nous allons créer une page pour s'enregistrer.**

Vous savez comment faire ☺

(utilisez la console)

Register

- **Maintenant il nous reste plus qu'à faire notre formulaire.**

On va utiliser encore et toujours la console :

symfony console make:form (register)

Dans src un registerType a été crée.

Il crée le formulaire à partir de l'entité User.

Les "add" correspondent aux Inputs, lequel pouvons nous enlever à votre avis ?

Register

- Pour afficher le formulaire il faut déjà dans le contrôleur y mettre ceci :

```php
$user = new User();
$form = $this->createForm(RegisterType::class, $user);
return $this->render('register/index.html.twig',[
    'form' => $form->createView()
]);
```

- Ensuite il faut l'utiliser dans la vue twig :

```twig
{{form(form)}}
```

formulaires

Super le formulaire s'affiche, mais il est moche …

Il est possible de changer le thème de celui-ci dans config->packages->twig.yaml

```
twig:
    default_path: '%kernel.project_dir%/templates'
    form_themes: ['bootstrap_5_layout.html.twig']
```

Comme par magie notre formulaire s'est transformé en form bootstrap.

Mettre à jour une entité

- Dans notre formulaire il nous manque les champs nom et prénom et ville.

Pour mettre à jour une class existante il suffit de refaire la commande :

Symfony console make:entity

Et de spécifier l'entité existante pour la mettre à jour.

Rajoutez les éléments à l'entité et faite toutes les étapes pour mettre à jour la bdd avec doctrine.

Améliorez le RegisterType

```php
<?php

namespace App\Form;

use App\Entity\User;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\Extension\Core\Type\EmailType;
use Symfony\Component\Form\Extension\Core\Type\PasswordType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;

class RegisterType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('email', EmailType::class, [
                "label" => "Votre E-mail",
                "attr"=>[
                    'placeholder'=> "Saisir votre E-mail"
                ]
            ])
```

Il faut importer le composant correspondant pour l'utiliser

Paramétrer un champ

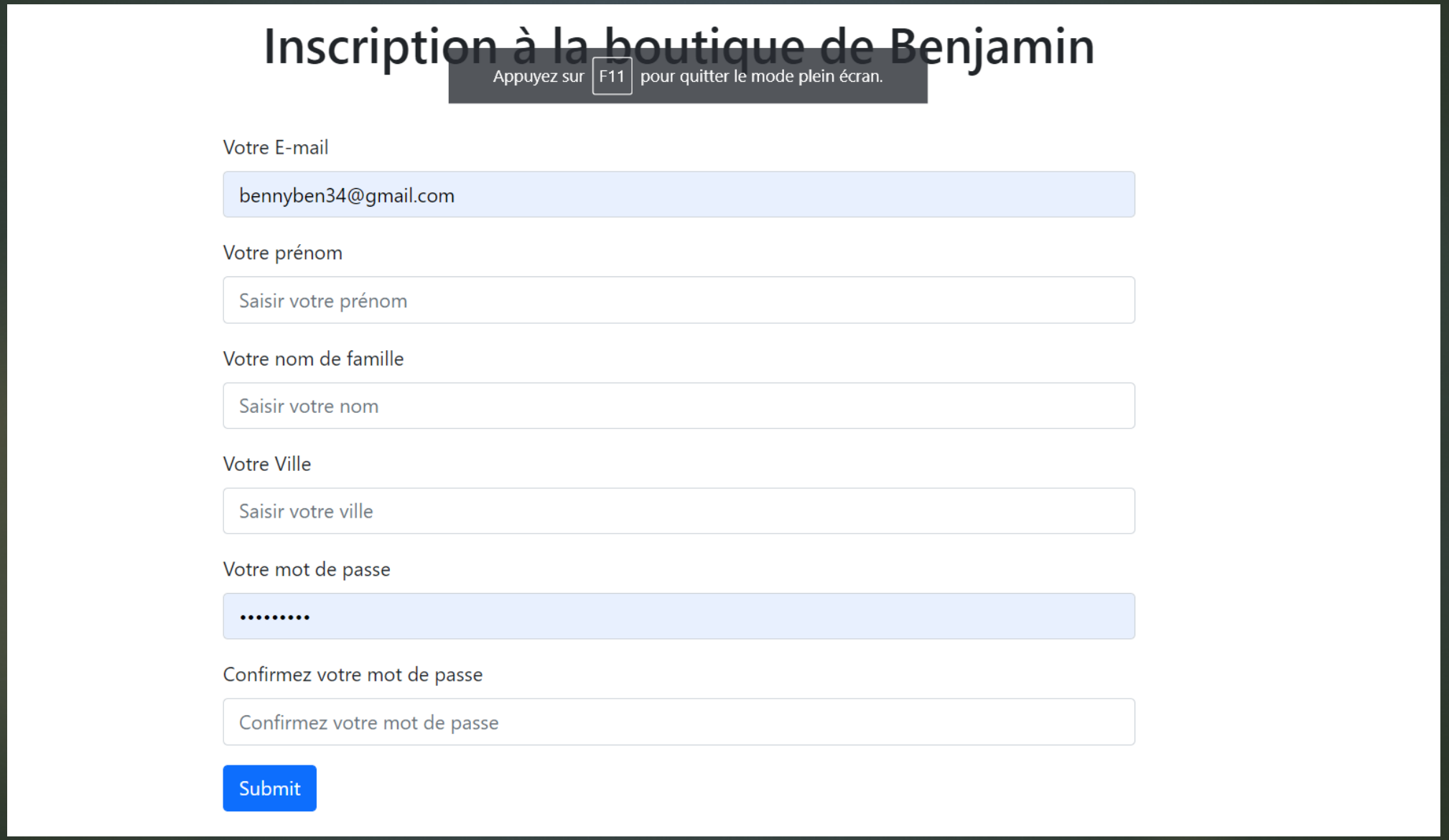
Améliorez le RegisterType

```
->add("password_confirm", PasswordType::class,[
    "label"=>"Confirmez votre mot de passe",
    "mapped"=>false,
    'attr'=>[
        "placeholder"=>"Confirmez votre mot de passe"
    ]
])
-> add("submit", SubmitType::class)
```

Pensez à rajouter tout les champs nécessaires avec les bon paramètres :

- Password confirm
- Submit
- City
- Lastname
- Firstname

Résultat final



Améliorer le registerType pour le password confirm

Il y a un type en particulier possible pour ajouter la confirmation de mot de passe, je vous laisse le chercher et changer le register type en conséquence.

Il n'y aura plus deux « add » avec password et password confirm mais bien un seul suite à ce changement.

Améliorer le registerType pour le password confirm

```
->add('password', RepeatedType::class, [
    "type"=>PasswordType::class,
    "invalid_message"=> "Le mot de passe et sa confirmation doivent être identique",
    "first_options"=>[
        "label"=>"Mot de passe",
        "attr"=>[
            "placeholder"=>"Merci de saisir votre mot de passe."
        ]
    ],
    "second_options"=>[
        "label"=>"Confirmez votre mot de passe",
        "attr"=>[
            "placeholder"=>"Merci de confirmer votre mot de passe."
        ]
    ]
])
```

Sauvegarder les données

- Dès que le formulaire est soumis
- Je veux que l'information soit traitée
- Vérification (formulaire valide ?)
- Enregistrement BDD

A votre avis ou devons nous réaliser ces étapes ?

Sauvegarder les données

L'injection de dépendances :

Pour pouvoir avancer sur l'enregistrement des données il va falloir préparer notre contrôleur :

Il va falloir utiliser, utiliser plusieurs composants pour sauvegarder les données (EntityManagerInterface, Request, UserPasswordHasherInterface).

Les faire passer en paramètres de fonction comme ceci (voir screen prochaine slide) s'appelle de l'injection de dépendance, cela les instancie automatiquement dans une variable à l'appel de la fonction.

Sauvegarder les données – injection de dépendances

```php
private $entityManager;

public function __construct(EntityManagerInterface $entityManager)
{
    $this->entityManager = $entityManager;
}
/**
 * @Route("/inscription", name="register")
 */
public function index(Request $request, UserPasswordHasherInterface $hasher): Response
{
```

Sauvegarder les données

```php
$form->handleRequest($request);

if ($form->isSubmitted() && $form->isValid()){
    $user = $form->getData();
    $this->entityManager->persist($user);
    $this->entityManager->flush();

}
```

Checkez la doc et google pour comprendre ce que ce code fait et intégrez le à votre contrôleur.

Que constatez vous quand vous enregistrez un utilisateur ?

Sauvegarder les données – avec le hasher de mot de passes

```php
$form->handleRequest($request);

if ($form->isSubmitted() && $form->isValid()){
    $user = $form->getData();
    $password = $hasher->hashPassword($user,$user->getPassword());
    $user->setPassword($password);
    $this->entityManager->persist($user);
    $this->entityManager->flush();

}
```

Form validation – bien lire la doc

Il y a un composant qui permet de checker nos inputs et renvoyer un message d'erreur selon certains cas. (validation)

Je vous laisse rajouter les contraintes suivantes :

Le prenom et le nom font minimum 2 lettres et maximum 30.

Si l'email est vide un message apparaitra pour gérer ce cas (sans ça si quelqu'un enlève le required par la console le site plantera avec une erreur)

Les messages d'erreurs sont en anglais pour les changer il faut aller dans :

translation.yaml et mettre default_locale en fr

Form validation – bien lire la doc

Il y a un composant qui permet de checker nos inputs et renvoyer un message d'erreur selon certains cas. (validation)

Je vous laisse rajouter les contraintes suivantes :

Le prenom et le nom font minimum 2 lettres et maximum 30.

Si l'email est vide un message apparaitra pour gérer ce cas (sans ça si quelqu'un enlève le required par la console le site plantera avec une erreur)

Les messages d'erreurs sont en anglais pour les changer il faut aller dans :

translation.yaml et mettre default_locale en fr