

Cours de WEBMASTERING

Concepts

- Modéliser un Besoin
- Architecture MVC
- Structuration d'un projet Symfony
- Création d'un Projet Symfony
- ORM:Doctrine
- Entity
- Fixtures
- Contrôleurs
- Service
- Route
- Configuration de la Sécurité(authentification et autorisation)
- Twig
- Intégration des Views
- Les Formulaires

Cours I

Durée: 4H Projet Fil Rouge

L'ISM fait appel à vous pour la réalisation d'une application Web de gestion des cours et des absences de l'école.

Chaque début d'année le Responsable Pédagogique(RP) peut créer,lister des classes (libellé,filière) ainsi que les professeurs(nom,prénom,spécialité,grade).

Le RP peut planifier un cours (date,heure début,heure fin, nombre heure,semestre) .Un cours est enseigné par un professeur et devant une ou plusieurs classes.Dans un cours on enseigne qu'un seul module(libellé).

Un RP a la possibilité de lister les cours planifiés et de filtrer par période(date de début,date de fin),lister les classes qui suivent le même cours et accéder à la liste des étudiants d'une classe.

Un professeur peut lister ses cours et marquer les absences d'un cours.

Une absence est caractérisée par sa date , l'étudiant et le cours

Les Attachés de classe font les inscriptions et les réinscriptions des étudiants durant la période d' inscription.Un étudiant peut s'inscrire plusieurs fois mais une seule fois dans une année.

Un Attaché peut lister les étudiants (matricule,nom complet,adresse) inscrits dans une classe ,les absences d'un cours,les absences d'un étudiant.

Un étudiant peut lister ses cours,ses absences .

- Le RP a toutes les fonctionnalités du AC
- Le AC a toute les fonctionnalités du RP

On voudrait avoir les statiques suivantes:

- le nombre de cours par professeur,
- le nombre de cours par classe
- les 5 étudiants les plus absentéisme
- les étudiants qui ont dépassé 25 H dans l'année

I) Modélisation (3H)

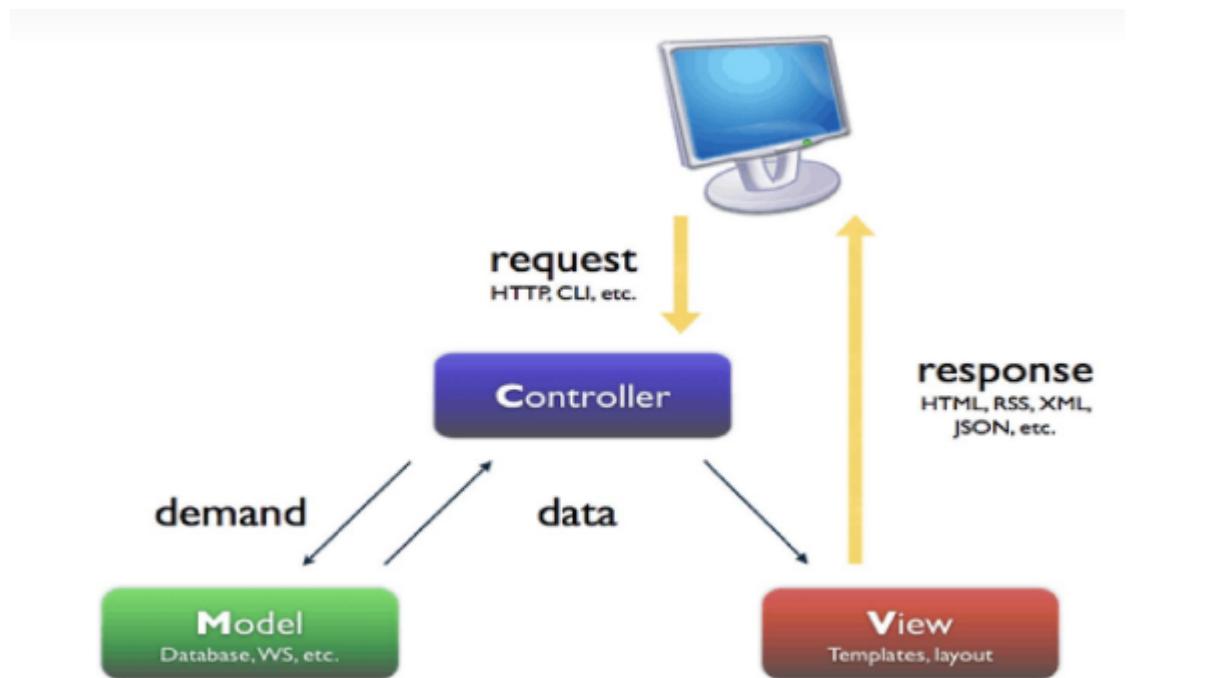
- Diagramme Use Case
- Diagramme de Classe
- MLD
- Créer la Base de donnée

Cours II

II) Architecture MVC(1H)

C'est-à-dire

- Modèle : la partie qui concerne les données (base de données, fichiers...) de notre application (site...). Il dispose de toutes les méthodes d'accès aux données (insert, delete, update, select)
- Vue : la partie qui concerne l'affichage : l'interface avec laquelle l'utilisateur interagit (page HTML...). C'est la vue qui s'occupe de la visualisation des données retournées par le modèle
- Contrôleur : c'est l'intermédiaire entre le modèle et la vue. Il demande les données au modèle, les analyse, et prend ensuite des décisions et renvoie le texte à afficher à la vue.



Et le MVC 2 ?

Exemple

- Le MVC est un peu difficile à mettre en place à cause de la multitude de contrôleurs
- Donc une nouvelle version a été introduite : MVC 2
- Il s'agit du même modèle avec un seul contrôleur pour orienter les requêtes

Frameworks basés sur l'architecture MVC (2)

MVC : explication

Déroulement

- ➊ Le client envoie une requête depuis la vue (son navigateur)
- ➋ Le contrôleur intercepte et analyse la requête du client
- ➌ Le contrôleur détermine quelle partie du modèle est concernée afin d'effectuer les traitements nécessaires
- ➍ Le modèle s'occupe de l'interaction avec les données, applique les règles métier et renvoie les données (dénués de toute présentation) au contrôleur
- ➎ Le contrôleur sélectionne la vue correspondante et lui injecte les données
- ➏ La vue présente les données au client

Et le MVC 2 ?

Exemple

- Le MVC est un peu difficile à mettre en place à cause de la multitude de contrôleurs
- Donc une nouvelle version a été introduite : MVC 2
- Il s'agit du même modèle avec un seul contrôleur pour orienter les requêtes

Exemple

- 1 CakePHP
- 2 FuelPHP
- 3 Jelix
- 4 Laravel
- 5 Symfony
- 6 Zend
- 7 ...

II) Présentation de Symfony(1H)

1) Historique

Symfony

- framework **PHP** sorti en octobre 2005
- français
- conçu et développé par SensioLabs
- open-source
- basé sur l'architecture **MVC**
- utilisant le protocole **HTTP**

2) Notion de Framework

Framework ?

- En français : cadre de travail
- Ensemble de composants logiciels et API facilitant le développement d'applications : pour les développeurs maîtrisant certains concepts informatiques (POO, SQL, MVC...)

Attention

Ne pas confondre framework et

- IDE (en anglais Integrated Development Environment, Environnement de développement intégré)
- CMS (en anglais Content Management System, Système de gestion de contenu) pour les développeurs novices

Pourquoi utiliser un framework ? (Ce n'est pas obligatoire)

- Un code de qualité
- Une meilleure structuration de notre projet
- Conflits entre dépendances gérés par le framework
- Plusieurs composants et API mis à disposition de développeurs

Exemple d'utilisation de **Symfony**

- Dailymotion (depuis 2009)
- Composer
- Covoiturage
- OpenClassroom
- SNCF
- ...

Les différentes versions de **Symfony**

- **Symfony 1** : sorti en octobre 2005
- **Symfony 2** : sorti en août 2011
- **Symfony 3** : sorti en novembre 2015
- **Symfony 4** : sorti en novembre 2017
- **Symfony 5** : sorti en novembre 2019

3) Installation de Symfony

a) Composer

Composer est un gestionnaire de dépendances pour PHP.

C'est un outil simple et fiable que les développeurs utilisent pour gérer et intégrer des paquets ou des bibliothèques externes dans leurs projets basés sur PHP. Ainsi, ils n'ont pas à créer leurs pages ou applications web à partir de zéro.

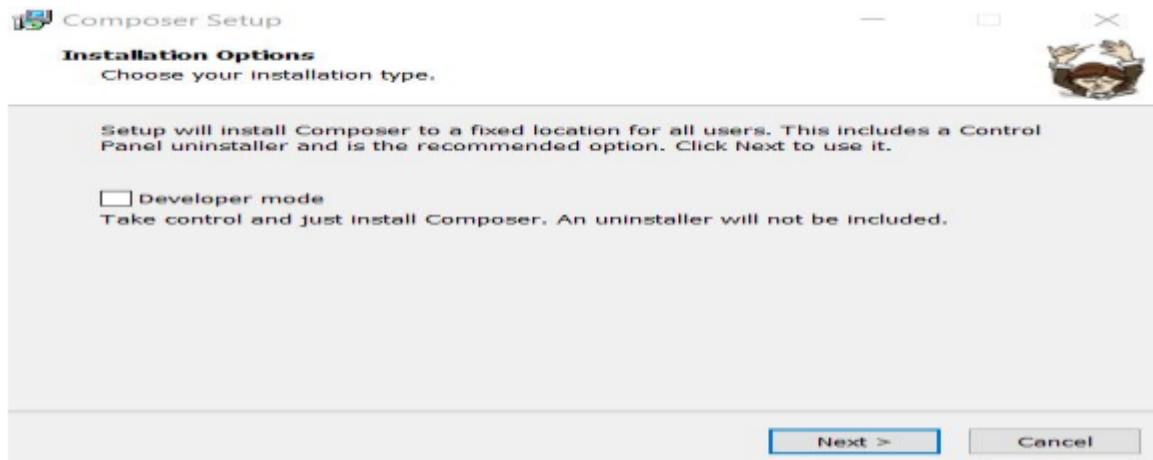
b) Installation de composer

■ Windows

Nous vous recommandons d'utiliser XAMPP ou WAMP à cette fin, car le processus est simple et vous pouvez le terminer en quelques minutes.

Une fois que XAMPP ou WAMP est installé, téléchargez la dernière version de Composer.

Lancez l'assistant d'installation de Composer. Lorsqu'il vous demande d'activer le mode développeur, ignorez-le et poursuivez le processus d'installation.



c) Installation des outils

a) Windows

[Telechargez](https://get.symfony.com/cli/setup.exe) <https://get.symfony.com/cli/setup.exe>
puis installez

b) Mac OS

Tapez la commande :

wget https://get.symfony.com/cli/installer -O - | bash

c) Linux

Tapez la commande :

curl -sS https://get.symfony.com/cli/installer | bash

d) Création du Projet

■ Avec la commande Symfony

Tapez les commandes:

#création d'un projet web application

symfony new --full my_project_name

#création d'un projet micro-service, console application ou Api

symfony new premier my_project_name

■ Avec la commande composer

#création d'un projet web application

composer create-project symfony/website-skeleton:"5.2.x@dev" my_project_name

#création d'un projet micro-service, console application ou Api

composer create-project symfony/skeleton:"5.2.x@dev" my_project_name

NB: Pour Installer les autres versions 4<

composer create-project symfony/website-skeleton:"^4.4" my_project_name

Ou

symfony new my_project_name --version=4.4

e) Lancement du Projet

Se Placer dans le Répertoire du Projet

cd --full premier my_project_name

Tapez la commande:

symfony server:start

4) Structure et Fonctionnement d'un projet Symfony

Structure d'un projet Symfony 4/5

- bin/ : contenant deux exécutables, la **console de Symfony** et **phpunit**
- config/ : contenant les fichiers de configuration (routes, ORM...)
- public/ : seul dossier accessible de l'extérieur (contenant le contrôleur frontal **index.php**)
- src/ : contenant les fichiers sources de l'application (contrôleurs, entités, formulaires, DAO...)
- templates/ : contenant les vues (vue partielle) de l'application
- tests/ : contenant les fichiers permettant de tester l'application
- translations/ : contenant les fichiers de l'internationalisation
- var/ : utilisé par **Symfony** pendant l'exécution, contenant les données de cache, le log et les sessions
- vendor/ : contenant les fichiers nécessaires pour une application **Symfony** (mentionnés dans **composer.json**)

Kernel ?

- **noyau de Symfony**
- défini dans **vendor/symfony/http-kernel**
- utilisé par le contrôleur frontal pour désigner le contrôleur adéquat pour répondre à la requête **HTTP** reçue

Contrôleur frontal

- point d'entrée d'une application **Symfony**
- défini dans **public/index.php**

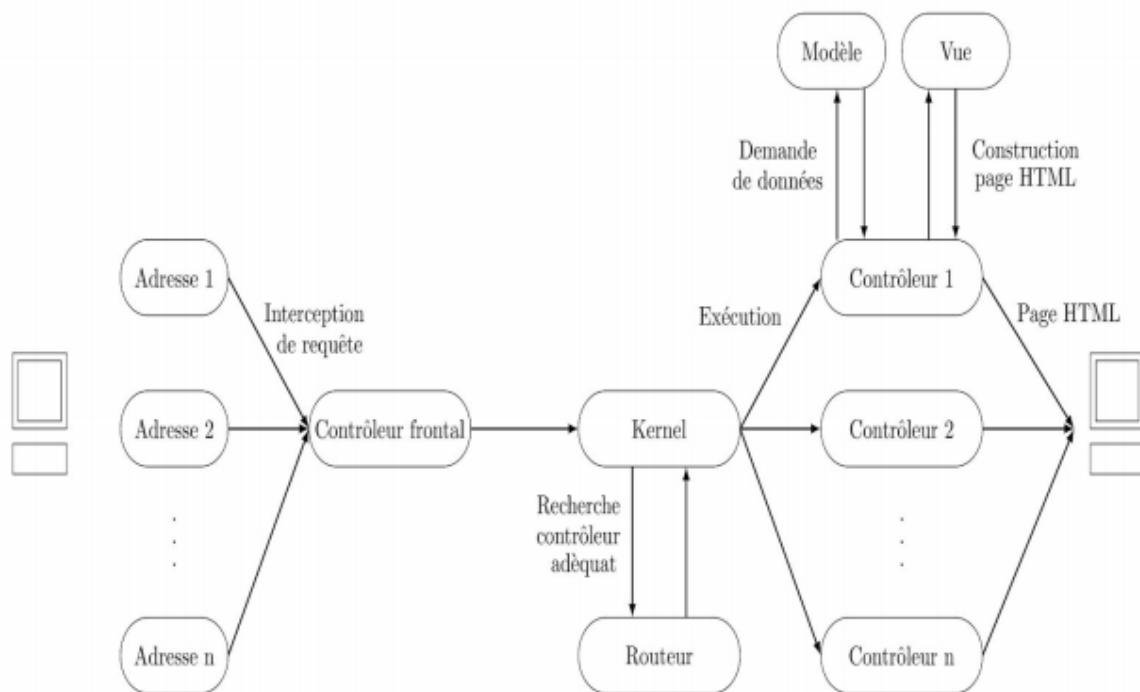
Deux environnements de travail

- **prod** (destiné aux utilisateurs finaux de l'application)
 - montrant l'application telle qu'elle sera visible par les visiteurs
 - rapide à exécuter
 - n'affichant pas les messages d'erreur.
- **dev** (destinés aux développeurs)
 - Plus lent que la version de production
 - Environnement de débogage complet
 - Possibilité d'ajouter des nouvelles fonctionnalités

Remarque

- Par défaut, une application est configuré à l'environnement **dev**
- Pour changer d'environnement, allez dans `.env` et mettez la valeur de `APP_ENV` à `prod`

Symfony2 : schématisation



Déroulement

- L'utilisateur saisit l'adresse d'une page de notre site
- Le contrôleur frontal intercepte la requête et il la transmet au Kernel
- Le Kernel demande au Routeur le contrôleur adéquat à la page demandée
- À la réception d'une réponse, le Kernel exécute le contrôleur
- Le contrôleur communique avec le modèle pour récupérer ou stocker certaines données
- Ensuite il renvoie ces données à la vue pour qu'elle construise la page HTML et la lui retourne.
- Enfin le contrôleur envoie à l'utilisateur la réponse (page HTML).

II) Réalisation Application web

1) Installation des outils (1H)

- a) Composer
- b) Commande de Symfony
- c) Commande maker

2) Couche Modèle(1H)

i) ORM

Object-Relational Mapping (lien objet-relation)

- est une couche d'abstraction à la base de données
- est une classe qui permet à l'utilisateur d'utiliser les tables d'une base de données comme des objets
- consiste à associer :
 - une ou plusieurs classes à chaque table
 - un attribut de classe à chaque colonne de la table

Quel choix pour PHP ?

- Doctrine
- pdoMap
- RedBean
- FoxORM
- ...

(1) Doctrine

Doctrine ?

- un ORM pour **PHP**
- proposé en 2006 par Konsta Vesterinen (2.0 fin 2010)
- utilisé par **Symfony** depuis la version 1.3 (et autres comme Zend Framework, CodeIgniter...)
- inspiré par **Hibernate** : ORM **Java**

Doctrine est composé de (deux) couches

- Doctrine (ORM) qui se base sur Doctrine (DBAL)
- Doctrine (DBAL) (DataBase Abstraction Layer ou couche d'abstraction de base de données) qui se base aussi sur PDO (PHP Data Objets) pour l'abstraction d'accès aux données

Doctrine (DBAL)

- ajoute des fonctionnalités à PDO
- permet de manipuler les bases de données avec des fonctions prédéfinies (pas d'utilisation du concept objet)

Doctrine (ORM)

- définit le lien entre DBAL et le monde objet
- permet de manipuler les éléments d'une base de données comme des objets

(2) Installation des composants

Si on n'a pas choisi la version complète à la création du projet, exécutez

- `composer require symfony/orm-pack`
- `composer require --dev symfony/maker-bundle`

(3) Configuration de la BD (fichier .env)

Préparation de la chaîne de connexion

- Allez dans le fichier .env
- Cherchez la ligne DATABASE_URL=mysql://db_user:
db_password@127.0.0.1:3306/db_name?serverVersion=5.7
- Remplacez la par DATABASE_URL=mysql://root:
@127.0.0.1:3308/courssymfony?serverVersion=5.7 puis enregistrez

(4) Création de la BD

`php bin/console doctrine:database:create`

(5) Les Étapes

3 étapes pour créer ou modifier une table associée à une entité

- créer ou modifier une entité
- créer une migration ⇒ générer le script **SQL**
- appliquer la migration ⇒ exécuter le script

(6) Création des entités

`php bin/console make:entity NomEntity`

NB: Pour Valider les Schémas on utilise la commande

`php bin/console doctrine:schema:validate`

(7) Création des Entités

- **Classe**
- **AnneeScolaire**
- **Module**

(8) Créer de la classe de connexion User

`php bin/console make:user User`

(9) Héritage entre les entités

Trois possibilités avec l'héritage

- SINGLE_TABLE
- JOINED

Pour indiquer comment transformer les classes mère et filles en tables

Il faut utiliser l'annotation `@InheritanceType`

Il faut aussi indiquer la solution choisie pour l'héritage

Dans la classe mère on ajoute

```
@ORM\InheritanceType ("SINGLE_TABLE")
```

i) Single Table

- Classe Mère User

```
/**  
  
 * @ORM\InheritanceType("SINGLE_TABLE")  
 * @ORM\DiscriminatorColumn(name="type", type="string")  
 * @ORM\DiscriminatorMap({"user" = "User", "etudiant" =  
"Etudiant", "professeur" = "Professeur"})  
 */  
class User implements UserInterface,  
PasswordAuthenticatedUserInterface  
{
```

ii) Joined

```
/**  
  
 * @ORM\InheritanceType("JOINED")  
 * @ORM\DiscriminatorColumn(name="type", type="string")  
 * @ORM\DiscriminatorMap({"user" = "User", "etudiant" =  
"Etudiant", "professeur" = "Professeur"})  
 */  
class User implements UserInterface,  
PasswordAuthenticatedUserInterface  
{
```

iii) Classes Filles

```
class Professeur extends User  
{
```

```
class Etudiant extends User  
{
```

NB: Dans la suite de notre cours nous utiliserons la stratégie Joined.

Les différents types de Doctrine

Annotation

@ORM\Entity
@ORM\Table
@ORM\Column
@ORM\Id
@ORM\GeneratedValue
@ORM\OneToOne
@ORM\OneToMany
@ORM\ManyToMany

désignation

marque qu'une classe PHP est une entité
décrit la table d'une entité persistante
définit les caractéristiques d'une colonne
marque l'identifiant de l'entité
utilisée pour générer des identifiants annotés par @Id
entité en relation avec une seule entité
entité en relation avec plusieurs entités
entités en relation avec plusieurs entités

NB:Lors de création d'une entité NomEntity un fichier NomEntityRepository sera aussi généré.Ce fichier sera utilisé pour les requêtes d'interrogation de données.

(10) Migration

Pour régénérer la table dans la base de données, exécutez

- php bin/console make:migration
- php bin/console doctrine:migrations:migrate

Attention Erreur:

```
In MetadataStorageError.php line 13:
```

```
The metadata storage is not up to date, please run the sync-metadata-storage command  
to fix this issue.
```

Correction : fichier .env

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/db_sms_ism_2020?serverVersion=5.7"
```

Remplacer Par la version de Maria DB

- **Voir la Version de Maria DB**

```
C:\Users\USER>cd c:\xampp\mysql\bin  
  
c:\xampp\mysql\bin>mysql.exe -u root  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MariaDB connection id is 356  
Server version: 10.4.20-MariaDB mariadb.org binary distribution  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]>
```

- **Remplacer la valeur de serverVersion =mariadb-10.4.20**

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/db_sms_ism_2020?serverVersion=maria  
db-5.7"
```

(11) Creation du Schema de BD

```
php bin/console doctrine:schema:update --force
```

(12) Suppression de BD

```
"php bin/console doctrine:database:drop --if-exists --force
```

(13) Création des Entités du Projet du projet

- **Module**
- **Année Scolaire**
- **Panification**
- **Cours**
- **Inscription**
- **Absence**

3) Fixtures ou fausses données(1H)

A. Installation des dépendances

```
composer req orm-fixtures --dev
```

B. Crédation des fixtures

1. Commande

```
php bin/console make:fixtures NomFixtures
```

2. Crédation des Fixtures

a) AnnéeScolaire

```
for ($i=2019; $i <2022 ; $i++) {  
    $data=new AnnéeScolaire();  
    $annee= $i."-".($i+1);  
    $data->setLibelle($annee)  
        ->setEtat(false);  
    $manager->persist($data);  
    $this->addReference("AnnéeScolaire".$i, $data);
```

b) Module

```
public function load(ObjectManager $manager)  
{  
    for ($i = 1; $i <=10; $i++) {  
        $module = new Module();  
        $module->setLibelle('Module '.$i);  
        $manager->persist($module);  
        $this->addReference("Module".$i, $module);  
    }  
  
    $manager->flush();  
}
```

C) Classe

```
public function load(ObjectManager $manager)
{
    $niveaux=["L1","L2","L3"];
    $filieres=[ "MAE", "IAGE", "GLRS"] ;
    for ($i = 1; $i <=10; $i++) {
        $pos= rand(0,2);
        $classe = new Classe();
        $classe->setLibelle($niveaux[$pos] ."".$filieres[$pos]);
        $manager->persist($classe);

        $this->addReference("Classe".$i, $classe);
    }

    $manager->flush();
}
```

d) User

```
private $encoder;
public function __construct(UserPasswordHasherInterface $encoder) {
    $this->encoder=$encoder;
}
public function load(ObjectManager $manager)
{
    $roles=[ "ROLE_USER", "ROLE_RP", "ROLE_AC"] ;
    $plainPassword = 'passer@123';
    for ($i = 1; $i <=10; $i++) {
        $user = new User();
        $pos= rand(0,2);
        $user->setNomComplet('Nom et Prenom ' .$i);
        $user->setEmail(strtolower($roles[$pos]) . "@gmail.com" . $i);
        $encoded = $this->encoder->hashPassword($user,
$plainPassword);
        $user->setPassword($encoded);
        $user->setRoles([$roles[$pos]]);
        $manager->persist($user);
        $this->addReference("User".$i, $user);
    }

    $manager->flush();
}
```

```
}
```

e) Professeur

```
class ProfesseurFixtures extends Fixture implements  
DependentFixtureInterface  
{  
  
    private $encoder;  
    public function __construct(UserPasswordHasherInterface $encoder) {  
        $this->encoder=$encoder;  
    }  
    public function load(ObjectManager $manager)  
{  
        $grade=[ "MASTER" , "INGENIEUR" , "DOCTEUR" ] ;  
        $plainPassword = 'passer@123' ;  
        for ($i = 1; $i <=10; $i++) {  
            $user = new Professeur();  
            $pos=round(0,2);  
            $user->setNomComplet('Nom et Prenom ' . $i);  
            $user->setEmail('professeur' . $i . '@gmail.com');  
            $encoded = $this->encoder->hashPassword($user,  
$plainPassword);  
            $user->setPassword($encoded);  
            $user->setRoles(["ROLE_PROFESSEUR"]);  
            $user->setNci("1 619 1986 005" . $i);  
            $user->setGrade($grade[$pos]);  
            $user->addModule($this->getReference("Module" . $i));  
            $manager->persist($user);  
            $this->addReference("Professeur" . $i, $user);  
        }  
  
        $manager->flush();  
    }  
  
    public function getDependencies()  
    {  
        return array(  
            ModuleFixtures::class,  
        );  
    }  
}
```

```
}
```

f) Etudiant

```
class EtudiantFixtures extends Fixture
{
    private $encoder;
    public function __construct(UserPasswordEncoderInterface
$encoder)
{
    $this->encoder=$encoder;
}
public function load(ObjectManager $manager)
{
    $plainPassword = 'passer@123';
    for ($i = 1; $i <=10; $i++) {
        $user = new Etudiant();
        $user->setNomComplet('Nom et Prenom ' . $i);
        $user->setLogin('login_etu' . $i);
        $encoded = $this->encoder->encodePassword($user,
$plainPassword);
        $user->setPassword($encoded);
        $user->setRoles(['ROLE_ETUDIANT']);
        $user->setTuteur("Tuteur " . $i);
        $user->setMatricule("0000" + $i);
        $manager->persist($user);
        $this->addReference("Etudiant" . $i, $user);
    }

    $manager->flush();
}

}
```

g) Inscription

```
class InscriptionFixtures extends Fixture
{
    public function load(ObjectManager $manager)
    {
        for ($i = 1; $i <=10; $i++) {
            $annee=rand(2019,2020);

            $data
            ->setAnneeScolaire($this->getReference("AnneeScolaire".$annee));
            $data ->setClasse($this->getReference("Classe".$i));
            $data
            ->setEtudiant($this->getReference("Etudiant".$i));
            ->setEtudiant($this->getReference("Etudiant".$i));
            $manager->persist($data);
            $this->addReference("Inscription".$i,$data);
        }
        $manager->flush();
    }

    public function getDependencies()
    {
        return array(
            EtudiantFixtures::class,
            ClasseFixtures::class,
        );
    }
}
```

Evaluation I: Réaliser ces fixtures

- (a) Séance
- (b) Cours
- (c) Absences

3. Chargement des fixtures

```
php bin/console doctrine:fixtures:load -n
```

NB: Automation des commandes dans le fichier composer.json

```
"scripts": {  
    "prepare": [  
        "php bin/console doctrine:database:drop --if-exists  
--force",  
        "php bin/console doctrine:database:create",  
        "php bin/console doctrine:schema:update --force",  
        "php bin/console doctrine:fixtures:load -n"]
```

Cours III

I) Configuration de Security(2H)

But de la sécurité

Interdire, à un utilisateur, l'accès à une ressource à laquelle il n'a pas droit

Deux étapes

- Qui veut accéder à la ressource ?
- A t-il le droit d'y accéder ?

Configuration de la sécurité

- En utilisant des données statiques (en mémoire)
- En utilisant des données dynamiques (stockées dans une base de données)

Pour cela

On va utiliser un bundle **Symfony** à savoir security-bundle

Configuration de la sécurité

- En utilisant les annotations
- Et en définissant quelques règles dans config/packages/security.yml
- Mais on peut aussi utiliser :
 - le format XML
 - les tableaux imbriqués de PHP

NB: On utilisera le fichier security.yml

A. Configuration Security.yml

a. Installation de dépendances

Si on n'a pas choisi la version complète à la création du projet, exécutez

- composer require symfony/security-bundle

`composer req symfony/security-bundle`

b. Contenu du fichier Security.yml

Contenu de security.yaml

```
security:
    # https://symfony.com/doc/current/security.html#where-do-users-come
    # -from-user-providers
    providers:
        users_in_memory: { memory: null }
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            anonymous: lazy
            provider: users_in_memory
    access_control:
        # - { path: ^/admin, roles: ROLE_ADMIN }
        # - { path: ^/profile, roles: ROLE_USER }
```

C. Configuration de l'authentification

```
php bin/console make:auth
```

Préparation de l'authentification

À partir du terminal, exécutez la commande suivante

```
php bin/console make:auth

What style of authentication do you want? [Empty authenticator]:
[0] Empty authenticator
[1] Login form authenticator
> 1

The class name of the authenticator to create (e.g.
    AppCustomAuthenticator):
> LoginFormAuthenticator

Choose a name for the controller class (e.g. SecurityController) [
    SecurityController]:
> SecurityController

Do you want to generate a '/logout' URL? (yes/no) [yes]:
> yes
```

d. Nouveau contenu du fichier Security.yml

Nouveau contenu de security.yaml

```
security:
    encoders:
        App\Entity\User:
            algorithm: auto

    # https://symfony.com/doc/current/security.html#where-do-users-come
    # -from-user-providers
    providers:
        # used to reload user from session & other features (e.g.
        # switch_user)
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            anonymous: lazy
            provider: app_user_provider
    access_control:
```

Pour tester, allez sur la route /login

- essayez de vous connecter avec un email inexistant
- ensuite essayez de vous connecter avec un email existant et un mot de passe incorrect
- enfin connectez-vous avec wick@wick.us et wick

Remarque

Problème de redirection après la connexion

Pour résoudre ce problème, il faut modifier la méthode onAuthenticationSuccess définie dans security/LoginFormAuthenticator pour rediriger vers la

```
public function onAuthenticationSuccess(Request $request,
TokenInterface $token, string $firewallName): ?Response
{
    if ($targetPath = $this->getTargetPath($request->getSession(), $firewallName)) {
        return new RedirectResponse($targetPath);
    }
    return new
RedirectResponse($this->urlGenerator->generate('app_login'));
}
```

4. Traduction des Erreurs

Pour modifier les messages d'erreurs de la page d'accueil, créez un fichier security.en.xlf dans translations avec le contenu suivant

Extrait de Code

```
<?xml version="1.0"?>
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2">
    <file source-language="en" datatype="plaintext"
original="file.ext">
        <body>
            <trans-unit id="Invalid credentials.">
                <source>Invalid credentials.</source>
                <target>Le mot de passe est invalide</target>
            </trans-unit>
            <trans-unit id="Login could not be found.">
                <source>Login could not be found.</source>
                <target>Login non-trouvé</target>
            </trans-unit>
        </body>
    </file>
</xliff>
```

5. Déconnexion

Pour se déconnecter

essayez la route /logout

6. Redirection après connexion

Allez à la section logout de security.yaml

```
logout:
    path: app_logout
    # where to redirect after logout
    # target: app_any_route
```

Décommentez la clé target et ajoutez la route

```
logout:
    path: app_logout
    # where to redirect after logout
    target: app_login
```

7. Mise en Forme du Formulaire de Connexion

a) Intégration de bootstrap

templates > base.html.twig

```
{% block stylesheets %}  
    <link rel="stylesheet" type="text/css"  
 href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.  
min.css">  
    {{ encore_entry_link_tags('app') }}#}
```

b) Form Thèmes ou Mise en forme des formulaires

config > packages > twig.yaml

```
twig:  
    default_path: '%kernel.project_dir%/templates'  
    form_themes: ['bootstrap_4_horizontal_layout.html.twig']
```

c) Intégration du Formulaire de Connexion

The screenshot shows a light gray rounded rectangular form. At the top center, it says "Formulaire de Connexion". Below that is a "Email" label with a blue input field containing "douwane852@gmail.com". Underneath is a "Password" label with a light blue input field containing ".....". In the bottom right corner of the form is a blue button labeled "Connexion".

d) Code de View connexion

```
{% block body %}  
<form method="post">  
    {% if error %}  
        <div class="alert alert-danger">{{  
            error.messageKey|trans(error.messageData, 'security') }}</div>  
    {% endif %}
```

```
<div class="card w-50 position-relative" style="top:  
25vh; margin: auto;">  
  
    <div class="card-body">  
        <h4 class="card-title"> <h1 class="h3 mb-3 font-weight-normal  
text-center">Formulaire de Connexion</h1></h4>  
        <div class="form-group">  
            <label for="inputEmail">Email</label>  
            <input type="email" value="{{ last_username }}"  
name="email" id="inputEmail" class="form-control" autocomplete="email"  
required autofocus>  
        </div>  
        <div class="form-group">  
            <label for="inputPassword">Password</label>  
            <input type="password" name="password"  
id="inputPassword" class="form-control" autocomplete="current-password"  
required>  
        </div>  
        <div class="form-group">  
            <input type="hidden" name="_csrf_token"  
value="{{ csrf_token('authenticate') }}">  
  
            <button class="btn btn-lg btn-primary float-right mr-2"  
type="submit">  
                Connexion  
            </button>  
        </div>  
    </div>  
  
</form>  
  
{% endblock %}
```

Cours IV

I. Contrôleurs

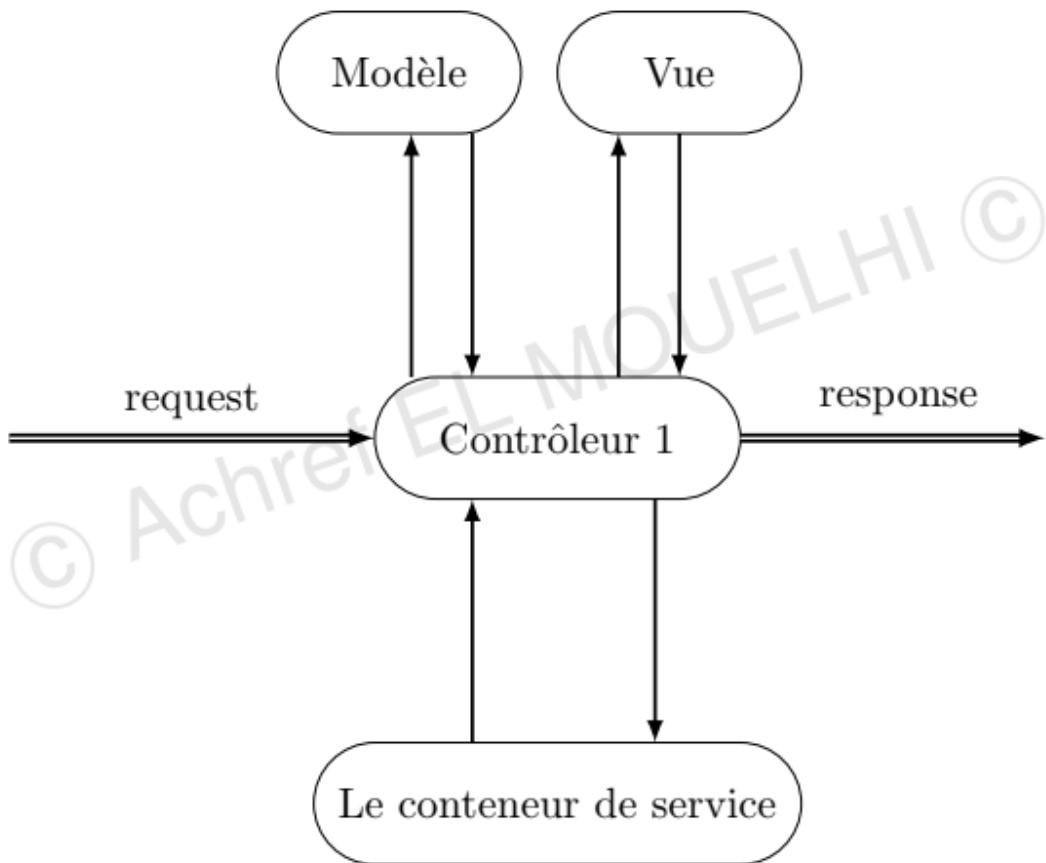
A. Rôle

Rôle

- Un élément indispensable de l'architecture MVC
- Il reçoit une requête et il interagit avec les différents composants d'une application **Symfony** :
 - les vues
 - les services
 - les modèles
 - les constructeurs de formulaires
 - ...
- pour retourner une réponse

Techniquement

- Un contrôleur est une classe **PHP** qui hérite d'`AbstractControl`
- Chaque méthode (action) de contrôleur est associée à une route
- Dans un contrôleur, il n'y a que du code **PHP** (pas de **HTML** ni **CSS** ni **JS**)



Explication

- **request et response** sont deux objets
- **request** contient les données concernant la requête utilisateur
- **response** correspond à la réponse préparée puis retourner par le contrôleur
- Les services, les modèles... vont nous permettre de réaliser tout le travail nécessaire pour préparer le contenu de la réponse.

B.Génération d'un contrôleur

```
php bin/console make:controller NomController
```

Pour générer un contrôleur nommé HomeController

```
php bin/console make:controller HomeController
```

Le résultat est

```
created: src/Controller/HomeController.php
created: templates/home/index.html.twig
```

Constats

- HomeController.php : un contrôleur généré dans src/controller
- index.html.twig : une vue générée dans templates/home
- home : un répertoire créé pour le contrôleur HomeController qui contiendra toutes ses vues. Par défaut, **Symfony** cherchera les vues dans ce répertoire.

NB: on peut générer un contrôleur sans template

```
php bin/console make:controller NomController --no-template
```

C.Création des contrôleurs du projet

- a) Home
- b) Année Scolaire
- c) Classe
- d) Professeur
- e) Etudiant
- f) User
- g) Module
- h) Panification
- i) Cours
- j) Inscription
- k) Absence

D.Notion de Route

Plusieurs modes de routage avec **Symfony**

- par annotation (par défaut en **Symfony**)
- dans un fichier **YAML**
- dans un fichier **XML**
- dans un fichier **PHP**

NB: Dans ce cours nous utiliserons les routes sous forme d'annotation

```
@Route('/classe', name: 'classe')
public function index(): Response
{
    return $this->render('classe/index.html.twig', [
        'controller_name' => 'ClasseController',
    ]);
}
```

Remarque : Depuis la version 8 de PHP , on peut utiliser la syntaxe ci dessous pour définir une route.La `@Route()` est remplacé par [`@Route()`]

```
#[Route('/classe', name: 'classe')]
public function index(): Response
{
    return $this->render('classe/index.html.twig', [
        'controller_name' => 'ClasseController',
    ]);
}
```

Explication

- Tous les contrôleurs sont définis dans un namespace App\Controller
- La méthode index retourne la vue home/index.html.twig et lui envoie un paramètre controller_name avec comme valeur le nom du contrôleur HomeController
- La méthode index est annotée par @Route qui définit le chemin qui permettra d'exécuter cette méthode
- L'annotation @Route permet d'associer un nom à la route pour qu'on puisse l'appeler

Syntaxe Générale d'une route

```
/**  
 * @Route(  
 *     "/uri",  
 *     name="name_route",  
 *     methods={"GET", "HEAD"}  
 *     requirements={"page"="\d+"},  
 *     condition="expression de test"  
 * )  
 */
```

Explication

- "/uri" : représente le lien d'appel de la fonction associer à cette route
 - /classe
 - /classe/{id} ,uri avec paramètre obligatoire
 - /classe/{id?},uri avec paramètre non obligatoire

NB :

Si la route à un paramètre on peut avoir l'attribut requirements={"page"="\d+"} qui fixe une condition(expression régulière) sur une route

- name="name_route", identificateur unique de la route
- methods={"GET", "POST"}, les méthodes de la route
- condition : condition que devrait satisfaire la route

II. Les Services en Symfony

A. Notion de Service

Service

- Classe **PHP**
- Singleton
- Réalise une et une seule fonctionnalité
 - Envoi de mail
 - Manipulation d'une base de données...
- Accessible de partout dans notre code
- Injectable dans les classes où on a besoin de l'utiliser
- Pouvant utiliser un ou plusieurs autres services
- Ayant un identifiant = nom de la classe

B. Notion de Conteneur de Service

Un conteneur de service

- Classe **PHP**
- Gestionnaire de services
 - Instanciation de services
 - Renvoie de services

NB :

Pour accéder à un service, il faut passer par le conteneur de services.

C. Déroulement

Déroulement

- Le contrôleur demande un service au conteneur de services
- Le conteneur de service vérifie si ce service a déjà été instancié
- Si c'est le cas, il renvoie l'objet au contrôleur
- Sinon, il instancie ce service
- Ensuite il l'enregistre
- Enfin il le renvoie

D. Injection de Dépendance

On utilise un service par **injection de dépendance**, c'est-à-dire il est passé en paramètre de fonctions dans un contrôleur ou comme paramètre dans un constructeur lorsqu'il est utilisé dans une fixtures ou un autre services.

E. Les Services de Symfony

Symfony propose un ensemble de Services tels que:

- `request`
- `logger`
- `session`
- `mailer`
- `etc...`

Remarques:

Pour consulter la liste des services disponibles

```
php bin/console debug:container
```

OU

Consultation de la liste des services avec alias

```
php bin/console debug:autowiring
```

Ou

Consultation de la liste des services avec ou sans alias

```
php bin/console debug:autowiring --all
```

1. Présentation de Quelques Services de Symfony

a) Le Service Request

La Classe **Request** de Symfony enveloppe les variables superglobales (`$_GET`, `$_POST`, etc) auxquelles vous êtes peut-être habitué.

- **`$_GET`**

Exemple : <http://test.com/?user=modou&age=33>

Récupération :

- PHP :

- `$_GET['user'] //modou`
- `$_GET['age'] //33`

- Symfony

- `$request->query->get('user');`
- `$request->query->get('age');`

NB :

- `$request->query->all()` : récupère tout le contenu du `$_GET`
- `request->query->has('myKey')` : vérifie l'existence de la clé `myKey` dans le `$_GET`

- **`$_POST` : body de la requête**

Équivalent `$_POST` en Symfony `$request->request`

NB :

- `$request->request->all()` : récupere tout le contenu du body de la requête
- `$request->request->get('myname')` : récupération du champ dont le contenu est `myname`
- `request->request->has('myKey')` : vérifie l'existence de la clé `myKey` dans le `$_POST`

b) Utilisation des Messages de Sessions

Flash Messages

- Un message flash est une variable de session qui ne dure que le temps d'une seule page
- Une fois le message flash est affiché, il sera détruit de la session (donc il disparaîtra après actualisation ou changement de vue)

- **Dans le Contrôleur**

```
$this->addFlash(  
    'version',  
    'Symfony 5'  
) ;
```

- **Dans la vue**

```
{% for message in app.flashes('version') %}  
    {{ message }}  
{% endfor %}
```

c) Le Service Mailer

I. Notion de twig

Twig

- moteur de templates pour **PHP**
- apparu en 2009
- syntaxe inspirée par Jinja (moteur de template du framework Django de Python)
- issu et utilisé par **Symfony**
- supporté par plusieurs IDE : NetBeans, PhpStorm, Eclipse, Visual Studio Code...
- supporté par plusieurs éditeurs de texte : Sublime text, notepad++, vim...
- **Symfony 5** utilise la version 3 de **Twig**

Twig, Pourquoi ?

- permet de séparer le code **PHP** du code html (lisibilité, maintenabilité)
- offre la possibilité de modifier un fichier sans influencer le deuxième
- facilite le travail d'équipe

Inconvénients

- ralentir le chargement de page
- Un langage (de template) de plus à étudier
- La gestion d'erreurs est plus compliquée

Autres moteurs de template

- Smarty
- Liquid
- Mustache
- Plates
- Talus'TPL
- ...

Trois types de balises

- { % ... % } : pour exécuter une action
- {# ... #} : pour définir un commentaire
- {{ ... }} : pour afficher

A. Affichage

1. Variable

`{{ var }}`

- permet de récupérer et afficher la valeur d'une variable `var` envoyée par le contrôleur
- est l'équivalent de `<?php echo $var; ?>`

2. Tableau

`{{ tableau['idColonne'] }}`

- affiche le contenu d'un élément du tableau
- est l'équivalent de `<?php echo $tableau['idColonne']; ?>`

3. Passage de Variable entre le contrôleur et une vue

Exemple : contenu de la méthode index de HomeController

```
$stab = [2, 3, 8];
return $this->render('home/index.html.twig', [
    'controller_name' => 'HomeController',
    'tableau' => $stab
]);
```

Pour afficher le tableau dans index.html.twig : trois écritures correctes

```
<ul>
    <li>      {{ tableau[0] }}</li>
    <li>      {{ tableau['1'] }}</li>
    <li>      {{ tableau["2"] }}</li>
</ul>
```

4. Affichage d'un Objet

`{{ objet.attribut }}`

- affiche, logiquement, la valeur de \$_attribut de \$objet
- est l'équivalent de `<?php echo $objet->attribut(); ?>`

Réellement `{{ objet.attribut }}`

- affiche \$objet['attribut'] si \$objet est un tableau
- affiche \$objet->_attribut si \$objet est un objet et \$_attribut est public
- affiche \$objet->attribut() si \$objet est un objet et attribut() est une méthode public
- affiche \$objet->getAttribut() si \$objet est un objet et getAttribut() est une méthode public
- affiche \$objet->isAttribut() si \$objet est un objet et isAttribut() est une méthode public
- n'affiche rien et retourne null sinon.

5. Concaténation

```
{ { variable1 ~ " " ~ variable2 } }
```

- affiche le résultat de la concaténation de variable1 et variable2
- est l'équivalent de
`<?php echo $variable1 . ' ' . $variable2 ; ?>`

B. Opérateurs de Twig

Autres opérations

- Arithmétiques : +, -, *, /, %, ** (puissance) et // (division entière)
- Logiques : and, or et not
- Comparaisons : ==, !=, <, >, >=, <=, ===, starts with, ends with, matches
- Autres : is, in, [], ., . . . , ??, ?:

C. Condition

Exemple avec if ... endif

```
{% if tableau[0] > 0 %}
    {{ tableau[0] }} est positif
{% endif %}
```

Exemple avec if ... else ... endif

```
{{ tableau[0] }} est
{% if tableau[0] > 0 %}
    positif
{% else %}
    négatif
{% endif %}
```

Exemple avec if ... elseif ... else ... endif

```
 {{ tableau[0] }}  
 {%- if tableau[0] > 0 %}  
     positif  
 {%- elseif tableau[0] < 0 %}  
     négatif  
 {%- else %}  
     nul  
 {%- endif %}
```

Tester l'existence d'une variable

```
{%- if nom is defined %}  
    {{ nom }}  
{%- else %}  
    doe  
{%- endif %}
```

D. Boucle

Structure itérative : for

```
{%- for i in tableau %}  
    {{ i }} <br>  
{%- endfor %}
```

Le résultat

```
2  
3  
8
```

Structure itérative : `for (clé, valeur)`

```
{% for cle, valeur in tableau %}  
    {{ cle ~ ' : ' ~ valeur }} <br>  
{% endfor %}
```

E. filtre

Filtre

- Permettant de formater et modifier l'affichage d'une donnée
- Pouvant prendre un ou plusieurs paramètres
- Syntaxe : `{{ variable | fonction_filtre[paramètres] }}`
- Possibilité d'appliquer un filtre sur le résultat d'un autre filtre
- Liste complète :
<https://twig.symfony.com/doc/3.x/filters/index.html>

Quelques exemples

- `upper` : convertit les lettres en majuscules comme `strtoupper()` en PHP (lower est la réciproque)
- `length` : calcule le nombre d'éléments d'un tableau ou le nombre de caractères d'une chaîne
- `sort` : trie les éléments d'un tableau
- `trim` : supprime les caractères spéciaux indiqués du début et de la fin d'une chaîne de caractères
- `striptags` : supprime les balises HTML
- ...

Exemples avec les chaînes de caractères

```
## personne.prenom | capitalize ~ " " ~ personne.nom | upper ##  
## affiche John WICK ##  
  
## personne.prenom | length ##  
## affiche 4 ##  
  
## ' john wick! ' | trim ##  
## affiche 'john wick!' ##  
  
## ' john wick!' | trim('!') ##  
## affiche ' john wick' ##  
  
## ' john wick! ' | trim(side='left') ##  
## affiche 'john wick! ' ##  
  
## ' john wick! ' | trim(' ', 'right') ##  
## affiche ' john wick!' ##
```

Pour appliquer un filtre à une portion du code

```
{% apply upper %}  
    Bonjour {{ personne.prenom }}  
{% endapply %}  
## affiche BONJOUR JOHN ##
```

Exemples avec les tableaux (reduce)

```
## tableau | reduce((somme, valeur) => somme + valeur) ##  
## affiche 13 ##
```

reduce accepte aussi une valeur initiale

```
## tableau | reduce((somme, valeur) => somme + valeur, 5) ##  
## affiche 18 ##
```

Exemple avec map et reduce

```
## tableau | map(elt => elt + 2) | reduce((somme, valeur) =>  
    somme + valeur) ##  
## affiche 19 ##
```

Remarques

- Par défaut, **Twig** protège les variables en appliquant un filtre pour les protéger de balises **HTML**
- Pour désactiver le filtre, on peut utiliser le filtre `raw`
Par exemple `{{ variable | raw }}`
- Pour les chaînes de caractères non-définies dans une variable, on utilise `e` pour échapper les balises **HTML**
Par exemple `{{ 'texte
' | e }}`

F. fonctions

1. **asset()**

Caractéristiques

- Elle permet de faire référence au répertoire `web` du projet **Symfony** depuis les vues
- Elle permet donc de référencer des fichiers de ressource (CSS, JavaScript, images ...) définis dans `public`

Exemples

```
<link href="{{ asset('css/style.css') }}" rel="stylesheet" />
<script src="{{ asset('js/jquery-1.11.3.js') }}></script>
<script src="{{ asset('js/bootstrap.js') }}></script>
<script src="{{ asset('js/script.js') }}></script>
```

2. path() ou url()

path et url

- Elles permettent de référencer une route enregistrée dans notre routeur
- path génère une URL relative.
- url génère une URL absolue.

Exemple

```
<a href="{{ path('vehicule_route') }}">Véhicule</a>
<a href="{{ url('vehicule_route') }}">Véhicule</a>
```

Code HTML équivalent

```
<a href="/vehicule">Accueil</a>
<a href="http://localhost:8000/vehicule">Véhicule</a>
```

On peut aussi construire une route avec paramètres

```
<a href="{{ path('vehicule_route', {'id': 'value'}) }}"
}>Véhicule</a>
<a href="{{ url('vehicule_route') }}">Véhicule</a>
```

3. Variables globales

Les variables globales

- app.request : la requête d'un contrôleur
- app.session : service session
- app.user : pour récupérer l'utilisateur courant
- app.debug : True si le mode debug est activé, False sinon.
- app.environment : l'environnement courant dev ou prod

Exemple

```
{% set id = app.request.get('personne').nom %}
```

4. Inclusion de Page

Méthode 1

Pour inclure le menu dans la vue associée à HomeController

```
{% include 'menu.twig' %}
```

Inclusion avec ignorance d'erreur si page inexistante

```
{% include 'page.twig' ignore missing %}
```

Notion d'inclusion conditionnelle

```
{% include condition ? 'menu-admin.twig' : 'menu-user.twig' ignore missing %}}
```

Méthode 2

Pour inclure le menu, on peut créer une méthode dans HomeController et lui associer une route

```
/**
 * @Route("/menu", name="menu_route")
 */
public function menu()
{
    return $this->render('menu.twig', []);
}
```

Pour exécuter cette méthode dans la vue, on utilise la fonction render

```
{{ render("menu") }}
```

G. Block

Notion de block : zone réservée

```
{% block nom_block %}  
    ...  
{% endblock %}
```

H. Layout ou Base Template

Exemple

```
{# base.html.twig #}  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta charset="UTF-8">  
        <title>  
            {% block title %}Welcome!{% endblock %}  
        </title>  
        {% block stylesheets %}{% endblock %}  
        <link href="{{ asset('css/style.css') }}" rel="stylesheet"/>  
    </head>  
    <body>  
        {% block body %}{% endblock %}  
        {% block javascripts %}{% endblock %}  
    </body>  
</html>
```

I. Héritage de blocs

Héritage entre block

```
{% extends 'base.html.twig' %}

{% block title %}
    Home
    {%# ce contenu sera inséré dans le bloc title de base.
        html.twig #}
{% endblock %}

{% block body %}
    {%# contenu précédent #}
    {%# ce contenu sera inséré dans le bloc body de base.
        html.twig #}
{% endblock %}
```

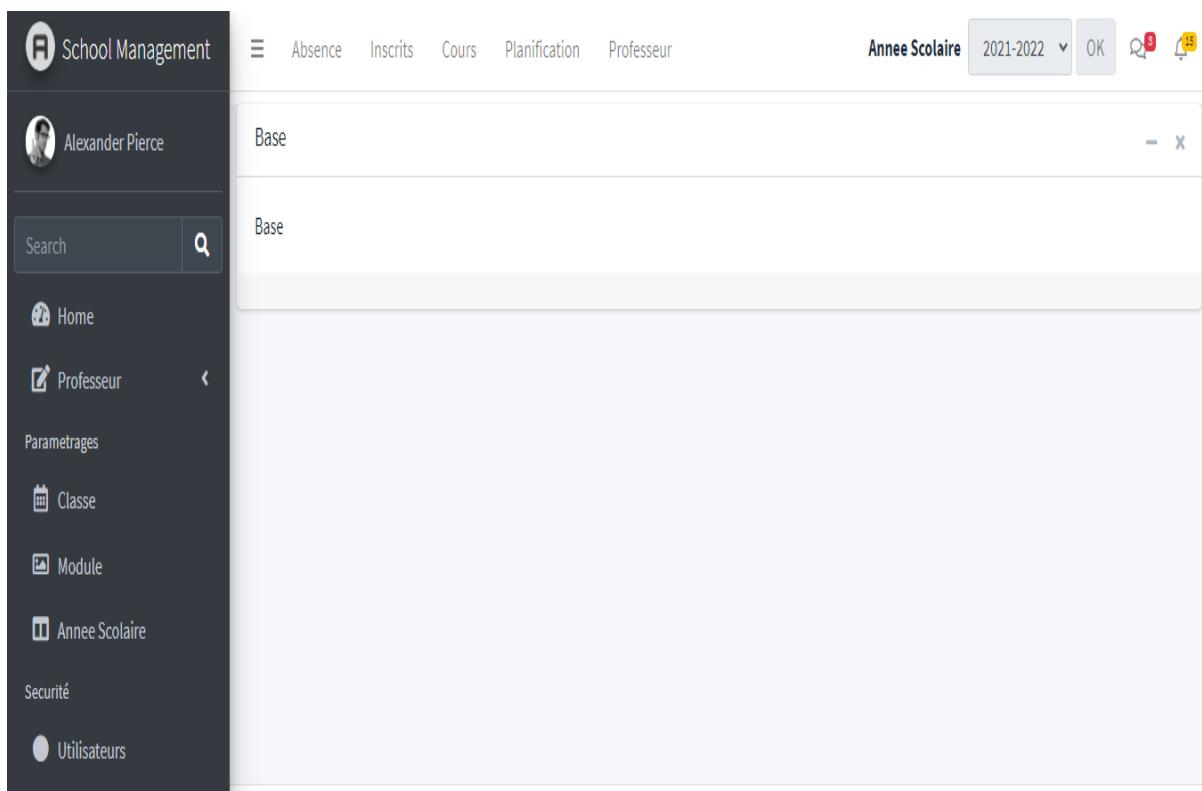
Remarques

- L'héritage sert à créer un template parent (avec un ou plusieurs blocks) qui contient le design de base de notre site pour que les templates enfants puissent l'utiliser
- Si le template enfant ne redéfinit pas un block hérité, il aura la valeur définie par le père pour ce block
- {{ parent() }} permet de récupérer le contenu du block côté père
- On peut faire des include pour ajouter entièrement un template

II. Intégration des Layouts

A. Intégration du Layout de Base

- **template:** <https://github.com/ColorlibHQ/AdminLTE>
- Ajouter dans public les fichiers (js,css et Images)
- Ajout des link css et Js
- Intégration de la page de Base
- Redirection vers une vue après connexion
- Déconnexion
- Définition des Menu
- Layout de Base



III. Réalisation des Fonctionnalités

A. Menu : Paramétrage

1. Option Classe
 - Page

Classe		
Libelle		+ Enregister
Liste des Classes		
ID	Libelle	Actions
1	L2IAGE	[<input checked="" type="checkbox"/> Modifier] [<input type="button" value="Supprimer"/>
2	L2IAGE	[<input checked="" type="checkbox"/> Modifier] [<input type="button" value="Supprimer"/>
3	L1MAE	[<input checked="" type="checkbox"/> Modifier] [<input type="button" value="Supprimer"/>
4	L1MAE	[<input checked="" type="checkbox"/> Modifier] [<input type="button" value="Supprimer"/>
5	L3GLRS	[<input checked="" type="checkbox"/> Modifier] [<input type="button" value="Supprimer"/>
6	L3GLRS	[<input checked="" type="checkbox"/> Modifier] [<input type="button" value="Supprimer"/>

■ liste.html.twig

- form.html.twig
 - Validation des champs

■ Code :

- Controllers

```
# [Route('RP/classe/liste', name: 'classe_liste')]
public function index(ClasseRepository $repo): Response
{
    $classes=$repo->findAll();
    return $this->render('classe/liste.html.twig', [
        "classes"=> $classes
    ]);
}

#[Route('RP/classe/save', name: 'classe_save')]
public function save(Request $request,ClasseRepository
$repo,EntityManagerInterface $manager): Response
{
    $classes=$repo->findAll();
    if($request->request->has("btn_save")){
        $libelle=$request->request->get("libelle");
        if(empty($libelle)){
            $error_libelle="Libelle est Obligatoire";
        }
    }
    return $this->render('classe/liste.html.twig', [
        "error_libelle"=> $error_libelle,
        "classes"=> $classes
    ]);
}
```

```

        ]);

    }

    $idClasse=(int)$request->request->get("id_classe");

    if($idClasse==0){
        //Ajout
        $newClasse=new Classe;
    }else{
        //Mofification
        $newClasse=$repo->find($idClasse);
    }

    $newClasse->setLibelle($libelle);
    $manager->persist( $newClasse);
    $manager->flush();
}

return $this->redirectToRoute("classe_liste");
}

#[Route('RP/classe/edit/{id}', name: 'classe_edit')]
public function edit(Classe $classe, Request
$request,ClasseRepository $repo,EntityManagerInterface $manager): Response
{
    $classes=$repo->findAll();

    return $this->render('classe/liste.html.twig',[

        "classe_selected"=> $classe,
        "classes"=> $classes
    ]);
}

#[Route('RP/classe/delete/{id}', name: 'classe_delete')]
public function delete($id,Request $request,ClasseRepository
$repo,EntityManagerInterface $manager): Response
{
    $classes=$repo->findAll();
    $classeSelected=$repo->find($id);
    $manager->remove($classeSelected);
    $manager->flush();
    return $this->redirectToRoute("classe_liste");
}

```

```
}
```

● Vues

```
{% extends 'base.html.twig' %}

{% block page_title %} Classe  {% endblock %}

{% block page_content %}

<div class="container-fluid">

    <form method="post"
action="{{ path('classe_save') }}>

        <input type="hidden" class="form-control"
name="id_classe" id="inputName" placeholder="" value=" {{ if
classe_selected is defined  %} {{ classe_selected.id}} {%
else %} 0 {%
endif %}}>

        <div class="form-inline w-100">
            <label for=""
class="mr-3">Libelle</label>
            <input type="text" name="libelle"
id="" class="form-control w-75" placeholder=""
aria-describedby="helpId" value=" {{ if classe_selected is defined  %}
{{ classe_selected.libelle}} {%
endif %}}>
            <button type="submit" name="btn_save"
id="" class="btn btn-primary btn-sm ml-3"><i class="fa fa-plus"
aria-hidden="true"></i> Enregistrer </button>
            {{ if  error_libelle is defined  %}
                <small id="helpId"
class="text-danger mt-2"
style="margin-left:7%">{{ error_libelle }}</small>
            {{ endif %}}
        </div>

    </form>

<h3 class="my-2">Liste des Classes</h3>
    <table class="table  table-bordered w-100">
        <thead class="thead-inverse">
```

```

<tr class="w-100">
    <th class=" w-25">ID</th>
    <th class=" w-25">Libelle</th>
    <th class=" w-25">Actions</th>
</tr>
</thead>
<tbody>
    {%
        for classe in classes %}
        <tr>
            <td>{{classe.id}}</td>
            <td>{{classe.libelle}}</td>
            <td>
                <a
                    href="{{path('classe_edit',{id:classe.id})}}"
                    class="btn btn-primary
                    btn-warning btn-xs"
                    role="button"><i class="fas fa-edit" style="font-size: 1.5em;"></i>
                    Modifier</a>
                <a
                    href="{{path('classe_delete',{id:classe.id})}}"
                    class="btn btn-primary
                    btn-danger btn-xs"
                    role="button"><i class="fa fa-trash" style="font-size: 1.5em;"></i>
                    Modifier</a>
                </td>
            </tr>
        {% endfor %}
    </tbody>
</table>
</div>

{%
    endblock %
}

```

TAF Par les Étudiants

- **Année Scolaire(A faire Par les Étudiants)**
 - **liste.html.twig**
 - **form.html.twig**
 - **Validation des champs**

2. Option Module

■ Page

Modules		
Libelle		+ Enregistrer
Liste des Modules		
ID	Libelle	Actions
1	Module 1	
2	Module 2	
3	Module 3	
4	Module 4	
5	Module 5	
6	Module 6	

■ liste.html.twig

● Lister les Modules

- Modifier la méthode `index()` dans le controller **Module**
- Injecter le `ModuleRepository`
- Récupérer les modules
- Afficher les modules dans la vue `module/index.html.php`

● Code

● Créer les Modules

Notion de Formulaire en symfony



○ Affichage du Formulaire

- Créer la méthode `add()` dans le controller **Module**
- Créer le Formulaire à partir de la méthode `createFormBuilder`
- Afficher le formulaire dans la vue

- Validation des de l'entité **Module**
- Traitement du Formulaire
 - Récupération des données soumises
 - valider les données
 - Enregistrer les Données
- Modifier les Modules
- Supprimer les Modules
- **Controllers**

```
# [Route('RP/module/liste', name: 'module_liste')]
public function index(Request $request, ModuleRepository
$repo, EntityManagerInterface $manager): Response
{
    //Liste des Modules
    $modules=$repo->findAll();
    //Creation du Formulaire
    $module = new Module();
    $form = $this->createFormModule($module);
    //Clique du Button Submit
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $manager->persist($module);
        $manager->flush();
    }
    return $this->render('module/liste.html.twig', [
        "modules"=>$modules,
        "form"=>$form->createView()
    ]);
}

#[Route('RP/module/save', name: 'module_save')]
public function save(): Response
{
    return $this->render('module/liste.html.twig', [
    ]);
}
```

```

#[Route('RP/module/edit/{id}', name: 'module_edit')]
public function edit(Module $module, Request $request,
ModuleRepository $repo, EntityManagerInterface $manager): Response
{
    $form = $this->createFormModule($module);
    //Liste des Modules
    $modules=$repo->findAll();
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $manager->persist($module);
        $manager->flush();
    }
    return $this->redirectToRoute('module_liste');
}

#[Route('RP/module/delete/{id}', name: 'module_delete')]
public function delete(Module $module, EntityManagerInterface
$manager): Response
{
    $manager->remove($module);
    $manager->flush();
    return $this->redirectToRoute('module_liste');
}

private function createFormModule(Module $module){
    return $this->createFormBuilder($module)
        ->add('libelle', TextType::class,[
            'required'=>false,
            "attr"=>[
                "class"=>"w-75"
            ]
        ])
        ->getForm();
}

```

○ Vues

```

{%- extends 'base.html.twig' %}

{%- block page_title %} Modules {%- endblock %}

{%- block stylesheets %}
{{ parent() }}

```

```

<style style="text/css">
    .btn-save{
        position: absolute;
        right: 100px;
        top: 70px;
    }
    span.invalid-feedback{
        width: 400px;
        margin-left: 170px;
        margin-top: 10px;
    }
</style>

{%
    endblock
    %}
{%
    block page_content
    %}

<div class="container-fluid">

    {{ form_start(form) }}
    {{ form_row(form.libelle) }}
    <button type="submit" name="btn_save" id="" class="btn
btn-primary btn-sm btn-save"><i class="fa fa-plus"
aria-hidden="true"></i> Enregistrer </button>
    {{ form_end(form) }}


<h3 class="my-2">Liste des Modules</h3>
<table class="table table-bordered w-100">
    <thead class="thead-inverse">
        <tr class="w-100">
            <th class=" w-25">ID</th>
            <th class=" w-25">Libelle</th>
            <th class=" w-25">Actions</th>
        </tr>
    </thead>
    <tbody>
        {% for module in modules %}
        <tr>
            <td >{{module.id}}</td>
            <td>{{module.libelle}}</td>
            <td>
                <a
                    href="{{path('module_edit',{id:module.id})}}" class="btn btn-primary

```

```

btn-warning btn-xs" role="button">><i class="fas fa-edit" "></i>
Modifier</a>
                <a
href="{{ path('module_delete', {id:module.id}) }}" class="btn btn-primary
btn-danger btn-xs" role="button">><i class="fa fa-trash"
aria-hidden="true"></i> Modifier</a>
            </td>
        </tr>
    {% endfor %}

    </tbody>
</table>
</div>

{% endblock %}

```

○ Entité

```

use Symfony\Component\Validator\Constraints as Assert;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
/**
 * @ORM\Entity(repositoryClass=ModuleRepository::class)
 * @UniqueEntity(
 *     fields={"libelle"} ,
 *     message="ce libelle existe déjà en Base de Donée"
 * )
 */
class Module
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255)
     * @Assert\NotBlank(message="le Libelle est Obligatoire")
     */
    private $libelle;
}

```

Cours V

I. Translation en symfony

A. Configuration de la langue

```
# config/service.yaml
```

```
parameters:  
    locale: 'fr'  
    app_locales: en|fr
```

```
# config/packages/translation.yaml
```

```
framework:  
    default_locale: '%locale%'  
    translator:  
        default_path: '%kernel.project_dir%/translations'  
        fallbacks:  
            - '%locale%'
```

```
# src/translations/messages.en.yaml
```

```
module :  
    blank : The Field is required  
    unique : This field must be unique
```

```
# src/translations/messages.fr.yaml
```

```
module :  
    blank : Ce champ est obligatoire  
    unique : Ce champ doit etre unique
```

```
# src/entity/Module.php
```

```
/**  
 * @ORM\Entity(repositoryClass=ModuleRepository::class)  
 * @UniqueEntity(  
 *     fields={"libelle"},  
 *     message="module.blank.unique"  
 * )  
  
 */  
class Module  
{
```

```

/**
 * @ORM\Column(type="string", length=255)
 * @Assert\NotBlank(message="module.blank")
 */
private $libelle;

```

```

# src/controllers
public function index(TranslatorInterface $translator): Response{
    return $this->render('path_views', [
        'message'=> $translator->trans('message.keys')
    ]);
}

```

```

# templates/views
{{ message|trans }}

```

```

# form/formType , Traduction des messages erreurs

```

```

->add('libelle', TextType::class, [
    'required'=>false,
    "constraints"=>[
        new NotBlank([
            'message'=>$this->translator->trans('module.blank')
        ])
    ]
]

```

```

# form/formType , Traduction des messages erreurs unique

```

```

$resolver->setDefaults([
    'data_class' => Module::class,
    'constraints' => [
        new UniqueEntity(
            [
                'fields' => ['libelle'],
                'message'
            ]=>$this->translator->trans('module.unique')
        )
    ]
]);

```

**NB: Pour que le chargement soit automatique ,
il faudra créer un Subscriber**

II. Service Session : SessionInterface

Le service session

- Les outils de session sont également intégrés dans un service (SessionInterface).
- On utilise la méthode `set` pour ajouter une variable dans la session et `get` pour récupérer.
- Dans les vues : `{{ app.session.get('nom_variable') }}`

III. Menu Inscription: AC

A.Chargement des Année Scolaire

1. Changer la Redirection après connexion vers **app_login**
2. Récupérer la liste des Année Scolaire
3. Utiliser le Service **ServiceInterface**
4. Stocker les année Scolaire dans la Session
5. Afficher les année Scolaire dans le Layout Base

src/security/LoginAuthenticatorAuthenticator

```
public function onAuthenticationSuccess(Request $request, TokenInterface $token,
string $firewallName): ?Response
{
    if ($targetPath = $this->getTargetPath($request->getSession(), $firewallName)) {
        return new RedirectResponse($targetPath);
    }

    //Connexion Success
    return new
RedirectResponse($this->urlGenerator->generate('app_login'));
}
```

src/controllers/SecurityController

```
public function login(AuthenticationUtils $authenticationUtils,
                     AnneeScolaireRepository $repo,
                     SessionInterface $session): Response
{
    if ($this->getUser()) {
        $annee = $repo->findAll();
        $session->set("annees", $annee);
        return $this->redirectToRoute('classe_show');
    }
}
```

templates/base.html.twig

```
<select class="form-control ml-3" name="" id="" style="width:150px">

    { % for annee in app.session.get('annees') %}
        <option if(annee.isStatut==true) 'selected'
value="{{annee.id}}">{{annee.libelle}}</option>
    { % endfor %}

</select>
```

B.Gestion Inscription

1. Maquette

Liste des Inscrits			
Classe	Nom & Prenom	Date	Actions
L1MAE	Etudiant17	jj/mm/aaaa	 Absences  Annuler
L2MAE	Etudiant14		 Absences  Annuler
L3GLRS	Etudiant7		 Absences  Annuler
	...		

2. Lister les Inscrits(Sans Filtre ,Sans Pagination)

src/Controller/inscriptionController.php

```
class InscriptionController extends AbstractController
{
    #[Route('/AC/inscription', name: 'inscription_show')]
    public function index(InscriptionRepository $repoIns,
                         AnneeScolaireRepository $repoAnn,
                         ClasseRepository $repoClasse,
                         ) : Response
    {

        $anneeEncours = $repoAnn->findOneBy(
            [
                'isStatut'=>1
            ]
        );


        $inscrits = $repoIns->findBy([
            'anneeScolaire'=>$anneeEncours
        ]);
        $classes = $repoClasse->findAll();

        return $this->render('inscription/index.html.twig', [
            'inscrits' => $inscrits ,
            "classes"=>$classes,
        ]);
    }
}
```

templates/inscription/index.php

```
{% extends 'base.html.twig' %}

{% block title %}Inscription {% endblock %}
{% block card_title %} Liste des Inscrits{% endblock %}
{% block stylesheets %}
{{ parent() }}
<style style="text/css">

    span.invalid-feedback{
        width: 400px;
        margin-left: 170px;
        margin-top: 10px;
    }


```

```

</style>
{%
    endblock
}

{%
    block card_body
%}

<div class="container-fluid">
    <a name="" id="" class="btn btn-info btn-sm float-right" href="#" role="button"> <i class="fa fa-plus" aria-hidden="true"></i> Nouveau</a><br>
    {# Recherche #}
<div class="container">
    <form method="post" action="{{path('inscription_show')}}">
        <div class="form-inline">
            <label for="" class="mx-4 my-2">Classe</label>
            <select class="form-control w-25 my-2" name="classe" id="">
                <option value="">Classe</option>
                {% for classe in classes %}
                    <option
                        value="{{classe.id}}>{{classe.libelle}}</option>
                {% endfor %}
            </select>
            <label for="" class="mx-4 my-2">Date</label>

            <input type="date"
                    class="form-control mx-4 w-25 my-2" name="date" id=""
                    aria-describedby="helpId" placeholder="">

            <button type="submit" name="btn_annee_filtre" class="btn btn-primary btn-sm mx-4"><i class="fa fa-search" aria-hidden="true"></i>
            Rechercher</button>
        </div>
    </form>
</div>
{# Fin Recherche #}

<table class="table table-bordered w-100 mt-3">
    <thead class="thead-inverse">
        <tr>
            <th class="">Matricule</th>
            <th class="w-50">Nom & Prenom</th>
            <th class="">Classe</th>
            <th class="">Actions</th>
        </tr>
    </thead>

```

```

<tbody>
    {%
        for inscrit in inscrits %
    <tr>
        <td>{{inscrit.etudiant.matricule}}</td>
        <td>{{inscrit.etudiant.nomComplet}}</td>
        <td>{{inscrit.classe.libelle}}</td>
        <td>
            <a name="" id="" class="btn btn-info btn-sm"
href="#" role="button"><i class="fa fa-edit" ></i> Absences</a>
            <a name="" id="" class="btn btn-warning
btn-sm" href="#" role="button"><i class="fa fa-trash" aria-hidden="true"></i>
Annuler</a>
        </td>
    </tr>
    {% endfor %}

```



```

</tbody>
</table>

</div>
{%
    endblock %
}
```

3. Changer Année Scolaire En Cours

templates/base.layout.php

#Fonctions Ajax pour le chargement Asynchrone de la Page

```

<script>

$(function() {
    $(document).on('change', '#select-annee', function() {
        const path = $(this).find(':selected').data('path');
        $.ajax({
            url: path,
            type: "GET",
            dataType: 'JSON', //or html or whatever you want
            success:function(data) {
                window.location.href=data
            }
        });
    })
})
```

```
})
```

```
</script>
```

```
#Fonction JavaScript en utilisant fetch
```

```
const select =document.querySelector('#select-annee');

select.addEventListener("change",function(event){
    const option=event.target.options[select.selectedIndex];
    const path=option.getAttribute('data-path')
    fetch(path,{
        method: 'GET',
        headers: {
            'Content-Type': 'application/json'
        }
    }).then(response => response.json())
    .then(promesse => {
        window.location.href=promesse;
    })
    .catch(err => console.log(err))
})
```

```
#Select Annee Scolaire
```

```
<select class="form-control ml-3" name="" id="select-annee" style="width:150px">
    {% for annee in app.session.get('annees') %}
        <option data-path="{{ path('annee_scolaire_change',{id:annee.id}) }}"
value="{{annee.id}}"
{% if annee.isStatut==true %} selected=selected {% endif %} > {{annee.libelle}}
</option>
    {% endfor %}

</select>
```

src/Controller/AnneeScolaireController.php

```
#Fonctions de Changement des Année Scolaire appelé par Ajax
```

```
#[Route('RP/annee/change', name: 'annee_scolaire_change')]
public function changeAnneeEncours(Request $request,
                                    AnneeScolaireRepository $repo,
                                    SessionInterface $session): Response
{
    if($request->isXmlHttpRequest()) {
```

```

        $id =(int) $request->query->get('id');
        $anneesInSession= $session->get("annees");
        $anneeEncours = $repo->find($id );
        $session->set('anneeEncours', $anneeEncours);
        $this->changeAnneeInSession($anneesInSession, $id );

    }

    return new JsonResponse($this->generateUrl('inscription_show'));

}

```

#Fonctions de Mis à Jour de la session

```

private function changeAnneeInSession(array $anneesInSession,int $id):void{
    foreach ($anneesInSession as $key=> $annee) {
        if($annee->getId() ==$id){
            $anneesInSession[$key]->setIsStatut(true);
        }else{
            $anneesInSession[$key]->setIsStatut(false);
        }
    }
}

```

src/Controller/SecuriteController.php

#Mis à Jour la fonction login, Sauvegarde de l'année En cours dans la session

```

-----
if ($this->getUser()) {
    //Les Annescolaire
    $annees = $repo->findAll();
    $session->set('annees',$annees);
    $session->set('anneeEncours',$repo->findOneBy([
        'isStatut'=>1
    ]));
    return $this->redirectToRoute('classe_show');
}

```

src/Controller/InscriptionController.php

```
#Modification : Recuperation de l'année en cours dans la Session  
,plutôt que dans la base de Données  
  
$anneeEncours = $session->get("anneeEncours");
```

4. Lister les Inscrits(Avec Filtre ,Sans Pagination)

templates/inscription/index.layout.php

```
#Select Classe
```

```
<select class="form-control w-25 my-2" name="classe" id="">  
    <option value="">Classe</option>  
    {% for classe in classes %}  
        <option  
data-path="{{ path('inscription_filtre_classe',{id:classe.id})}}"  
value="{{classe.id}}">{{classe.libelle}}  
</option>  
    {% endfor %}  
</select>
```

```
#Fonction Ajax de chargement des classes Filtrées
```

```
#Fonction Jquery
```

```
<script>  
  
$(function(){  
    $(document).on('change', '#select-inscription-classe', function(){  
        const path = $(this).find(':selected').data('path');  
        $.ajax({  
            url: path,  
            type: "GET",  
            dataType: 'JSON', //or html or whatever you want  
            success:function(data) {  
                window.location.href=data  
            }  
        });  
    })  
})
```

```
})
```

```
</script>
```

src/Controller/InscriptionController.php

#Définition de la Méthode de Filtre de Classe

```
#[Route('/AC/inscription/classe', name: 'inscription_filtre_classe')]
public function showInscriptionByClasse(
    InscriptionRepository $repoIns,
    ClasseRepository $repoClasse,
    SessionInterface $session,
    Request $request ): Response
{
    if($request->isXmlHttpRequest()) {
        $id =(int) $request->query->get('id');

        $classe = $repoClasse->find($id );
        $anneeEncours = $session->get("anneeEncours");
        $inscrits = $repoIns->findBy([
            'classe'=>$classe,
            'anneeScolaire'=>$anneeEncours
        ]);

        $session->set("inscrits", $inscrits);
        $session->set("classeSelected", $classe);
    }

    return new JsonResponse($this->generateUrl('inscription_show'));
}
```

#Ajout dans la Méthode Index pour le chargement des inscrits filtrés par classe

```
if($session->has('classeSelected')){
    $classeSelected= $session->get('classeSelected');
    $inscrits=$repoIns->findBy(
        [
            'classe'=>$classeSelected,
            'anneeScolaire'=>$anneeEncours,
        ]
    );
}
```

```
        ]
    );
    $session->remove('classeSelected');
    return $this->render('inscription/index.html.twig', [
        'inscrits' => $inscrits ,
        "classes"=>$classes,
        "classeSelected"=>$classeSelected,
    ]);
}
```

templates/inscription/index.layout.php

#Garder la Classe Sélectionnée

```
<option
    {% if classeSelected is defined %}
        {% if classe.id==classeSelected.id %}
            selected=selected
        {% endif %}
    {% endif %}

    data-path="{{ path('inscription_filtre_classe',{id:classe.id}) }}"
value="{{classe.id}}">{{classe.libelle}}
</option>
```

5. Lister les Inscrits(Avec Filtre ,Avec Pagination)

a) **Repository**

Repository

- une classe PHP
- contenant les méthodes de récupération de données relatives à nos entités
- pouvant être utilisé pour
 - définir des nouvelles méthodes
 - personnaliser des méthodes existantes

NB: Dans nos Repository, on peut utiliser des méthodes magiques

Les méthodes magiques

- `findByAttribut ($valeur)` : retourne un tableau de tous les tuples dont Attribut a comme valeur \$valeur)
- `findOneByAttribut ($valeur)` : retourne un seul tuple dont Attribut a comme valeur \$valeur

Exemple Dans notre Entité User on peut utiliser les méthodes:

- `findByNomComplet()` ou `findOneByNomComplet()`
- `findByEmail()` ou `findOneByEmail()`

b) **Notion de QueryBuilder**

Query Builder

- Dans la classe PersonneRepository, on définit une méthode qui
 - utilise un objet `QueryBuilder` : on l'obtient avec la méthode `createQueryBuilder ()` de l'`EntityManager` et on l'utilise pour construire la requête
 - récupère l'objet `Query` de `QueryBuilder`
 - récupère les résultats de la `Query`
- retourne le résultat

#src/Repository/InscriptionRepository

```
public function findPaginate($page,$limit,$anneeEncours)
{
    return $this->createQueryBuilder('i')
        ->leftJoin('i.anneeScolaire','a')
        ->andWhere('a.id=:id')
        ->setParameter('id', $anneeEncours->getId())
        ->orderBy('i.dateAt', 'desc')
        ->setFirstResult($page)
        ->setMaxResults($limit)
        ->getQuery()
        ->getResult()
    ;
}
```

```
public function countInscriptions($anneeEncours)
{
    return $this->createQueryBuilder('i')
        ->select('count(i.id) as count')
        ->leftJoin('i.anneeScolaire', 'a')
        ->where('a.id = :id')
        ->setParameter('id', $anneeEncours->getId())
        ->getQuery()
        ->getSingleScalarResult();
}
```

Autres méthodes de Query

- `getResult()` : Exécute la requête et retourne le résultat sous forme d'un tableau d'objets (même quand il s'agit d'un seul objet)
- `getArrayResult()` : Exécute la requête et retourne le résultat sous forme d'un tableau de tableaux
- `getScalarResult()` : Exécute la requête et retourne le résultat sous forme d'une valeur (à utiliser lorsque la requête retourne une unique valeur)
- `getOneOrNullResult()` : Exécute la requête et retourne un seul objet ou une valeur null
- Plusieurs autres : `getSingleResult()`, `getSingleScalarResult()` ...
- `execute()` : à utiliser pour exécuter des requêtes `insert`, `update`, `delete` ou `select`

src/Controller/inscriptionController.php

#Pagination Liste des Inscrits

```
//Définition du nombre de valeurs par page
private const LIMIT=3;
public function index(InscriptionRepository $repoIns,
                      ClasseRepository $repoClasse,
                      SessionInterface $session,
                      Request $request) : Response
{
    $anneeEncours = $session->get("anneeEncours");
    $classes = $repoClasse->findAll();

    //Recuperation du Parametre Get et Initiamisation à 1
    $page=(int)$request->query->get("page",1);
    //Recuperation du nbre Element
    $count=(int)$repoIns->countInscriptions($anneeEncours);
    $inscrits=$repoIns->findPaginate($page,$limit,$anneeEncours);
    return $this->render('inscription/index.html.twig', [
        'inscrits' => $inscrits ,
        "classes"=>$classes,
        'count' => $count,
        'limit' => self::LIMIT,
        'page' => $page,
    ]);
}
```

templates/inscription/index.html.php

```
<nav aria-label="Page navigation " >
    <ul class="pagination justify-content-center">
        <li class="page-item ">
            <a class="page-link" href="?page={{ (page>1) ? page-1 : '1' }}"
aria-label="Previous">
                <span aria-hidden="true">&lquo;</span>
                <span class="sr-only">Previous</span>
            </a>
        </li>
        {% set nbre_page = (count/limit)|round(0,"ceil") %}
        {% for row in 1..(nbre_page) %}
            <li class="page-item {{ (page==row) ? 'active' : '' }}"><a
class="page-link" href="?page={{row}}">{{row}}</a></li>
        {% endfor %}

        <li class="page-item">
            <a class="page-link" href="?page={{ (page<nbre_page) ? (page+1)
:nbre_page }}" aria-label="Next">
                <span aria-hidden="true">&raqo;</span>
                <span class="sr-only">Next</span>
            </a>
        </li>
    </ul>
</nav>
```

#Pagination Liste des Inscrits après Filtre

#Dans InscriptionRepository

```
public function  
findInscriptionsByPageAndAnneeScolaireAndClasse  
($page,$limit,$anneeEncours,$classeSelected){  
}
```

```
public function  
countInscriptionByAnneeScolaireAndClasse ($anneeEncours,$classes  
selected)  
{  
}
```

#Dans Controller Inscription ,Fonction index,Filtre

```
if($session->has('classeSelected')){  
    $classeSelected= $session->get('classeSelected');  
  
    $count=(int)$repoIns->countInscriptionByAnneeScolaireAndClasse ($anneeEncours,  
$classeSelected);  
    $inscrits=  
    $repoIns->countInscriptionByAnneeScolaireAndClasse ($page, self::LIMIT,$anneeEncours,  
$classeSelected);  
    $session->remove('classeSelected');  
    return $this->render('inscription/index.html.twig', [  
        'inscrits' => $inscrits ,  
        "classes"=>$classes,  
        "classeSelected"=>$classeSelected,  
        'count' => $count,  
        'limit' => self::LIMIT,  
        'page' => $page,  
  
    ]);  
}
```

```

#Fonction showInscriptionByClasse

public function showInscriptionByClasse(
    ClasseRepository $repoClasse,
    SessionInterface $session,
    Request $request ): Response
{

    if($request->isXmlHttpRequest()) {
        $id =(int) $request->query->get('id');
        $classe = $repoClasse->find($id );
        $session->set("classeSelected", $classe);
    }

    return new JsonResponse($this->generateUrl('inscription_show'));
}

```

NB: Pour définir des constantes ou paramètres globaux dans notre application, on peut :

- soit définir le paramètre ou constant global dans le fichier config/service.yaml

```

# config/service.yaml
#Définition de la constante PAGE LIMIT(Nombre d'éléments à afficher
par page)
parameters:
    PAGE_LIMIT: 3

```

src/Controller/InscriptionController

```

#Utilisation dans le controller
$this->getParameters('PAGE_LIMIT')

```

- soit définir un fichier de paramétrage et importer dans fichier

- **Définition du fichier de Paramètres**

- **config/services/parametres.yaml**

```
parameters :  
    PAGE_LIMIT: 3
```

- importation du fichier config/services/parametres.yaml dans config/service.yaml

```
imports :  
  
    - { resource: services/parameters.yaml }  
    # If you want to import a whole directory:  
    - { resource: services/ }
```

- **src/Controller/InscriptionController**

```
$this->getParameters('PAGE_LIMIT')
```

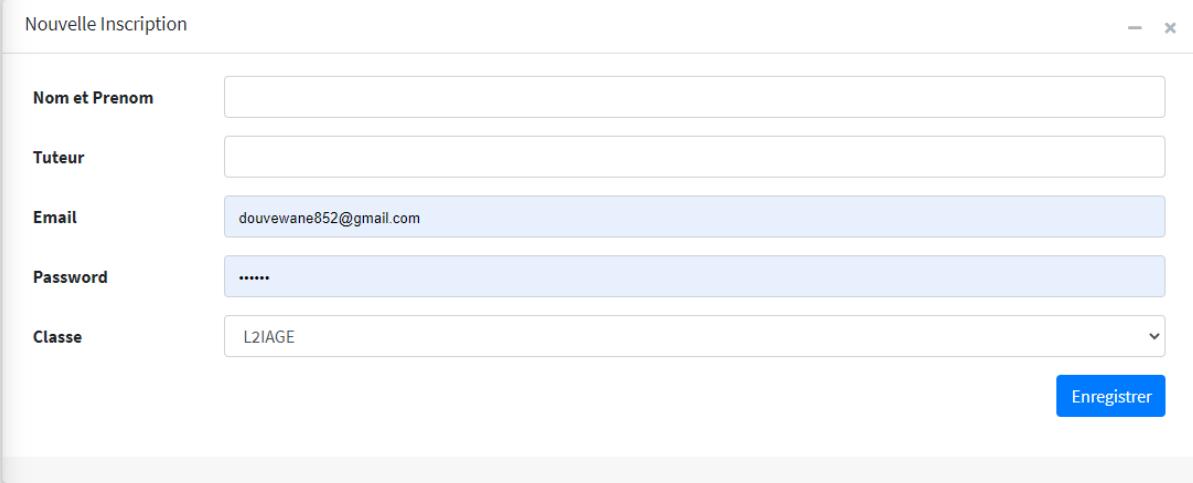
Remarque : On peut définir une classe de Pagination

6. Faire une Inscription et Réinscription

Nouvelle Inscription

Nom et Prenom	<input type="text"/>
Tuteur	<input type="text"/>
Email	douvewane852@gmail.com
Password
Classe	L2IAGE

Enregister



#Création d'un Service avec Symfony

En plus de proposer un ensemble de services, Symfony donne la possibilité de créer nos propres services. Dans cette partie nous créerons le service `SmsGenerate` qui offre la possibilité de générer un Matricule, un NCI.

- 1) Créer un dossier Service dans src
- 2) Créer dans le dossier Service un Fichier `SmsGenerate.php`

```
<?php  
namespace App\Service;  
class SmsGenerate {  
    public function generateMatricule(){  
        return uniqid();  
    }  
}
```

3) Utilisation du Service

Un service est utilisé par injection de dépendance, dans une fonction d'un controller, d'une fixtures ou d'un autre service

#Injection de Dépendance du Service dans un Controller

```
#[Route('/inscription', name: 'inscription')]  
public function inscription(SmsGenerate $smsGenerate): Response  
{  
    return $this->render('inscription/index.html.twig', [  
        'controller_name' => 'InscriptionController',  
    ]);  
}
```

#Injection de Dépendance dans une Fixtures

#Dans la Fixture de Étudiant

```
private $encoder;  
private $smsGenerate;  
public function __construct(UserPasswordEncoderInterface  
$encoder, SmsGenerate $smsGenerate){  
    $this->encoder=$encoder;
```

```
        $this->smsGenerate=$smsGenerate;  
    }  
}
```

```
$user->setMatricule($this->smsGenerate->generateMatricule());
```

NB: Pourquoi Symfony a considéré SmsGenerate comme un service

La réponse est donnée par cette partie de config/services.yaml

```
services:
  # default configuration for services in *this* file
  _defaults:
    autowire: true          # Automatically injects dependencies in
                             # your services.
    autoconfigure: true      # Automatically registers your services as
                             # commands, event subscribers, etc.

  # makes classes in src/ available to be used as services
  # this creates a service per class whose id is the fully-qualified
  # class name
App\:
  resource: '../src/'
```

Faire la Translation de Erreurs d' Incription

```
# src/translations/messages.en.yaml
```

```
module :  
  blank  : The Field is required  
  unique : This field must be unique  
  
inscription :  
  email   :  
    blank : The Field is required  
    unique : This email must be unique
```

src/translations/messages.fr.yaml

```
module :  
    blank  : Ce champ est obligatoire  
    unique : Ce champ doit etre unique  
  
inscription :  
    email   :  
        blank : Ce champ est obligatoire  
        unique : L'email doit etre unique
```

#Générer le form EtudiantType + Validation + Translation

```
public function buildForm(FormBuilderInterface $builder, array
$options): void
{
    $builder

        ->add('email', EmailType::class, [
            "required"=>false,
            "constraints"=>[
                new NotBlank([
                    "message"=>$this->translator->trans('inscription.email.blank')
                ]),
                [
                    ]
                ])
        ->add('password', PasswordType::class, [
            "required"=>false,
            "constraints"=>[
                new NotBlank([
                    "message"=>"Le Mot de Passe est Obligatoire"
                ])
            ]
        ])

        ->add('nomComplet', TextType::class, [
            "required"=>false,
            "constraints"=>[
                new NotBlank([
                    "message"=>"Le Nom et Prenom sont Obligatoires"
                ])
            ]
        ])
        ->add('tuteur', TextType::class, [
            "required"=>false,
            "constraints"=>[
                new NotBlank([
                    "message"=>"Le Tuteur est Obligatoire"
                ])
            ]
        ])
}
```

```

        ])
    ;
}

public function configureOptions(OptionsResolver $resolver): void
{
    $resolver->setDefaults([
        'data_class' => Etudiant::class,
        'constraints' => [
            new UniqueEntity(
                [
                    'fields' => ['email'],
                    'message' => $this->translator->trans('inscription.email.unique')
                ]
            )
        ]
    );
}

```

#Générer le form InscriptionType

InscriptionType

```

$builder
    ->add('etudiant', EtudiantType::class, [
        'label'=>false,
        'attr'=>[
            ...
        ]
    ])
    ->add('classe')
    ->add('save', SubmitType::class, [
        'label' => 'Enregistrer',
        'attr'=>[
            'class'=>'btn btn-primary float-right'
        ]
    ])
;

```

#Méthode add dans InscriptController

```
#[Route('/AC/inscription/add', name:  
'inscription_add',methods:[ "GET", "POST" ])]  
public function add( Request $request,  
                    SessionInterface $session,  
                    EntityManagerInterface $manager,  
                    UserPasswordHasherInterface $encoder,  
                    AnneeScolaireRepository $repoAnnee,  
                    ValidatorInterface $validator,  
                    SmsGenerate $genarator):Response{  
  
    $inscription=new Incription();  
    $form = $this->createForm(InscriptionType::class,  
$inscription);  
    //Recuperation des Donnees du Formulaire  
    $form->handleRequest($request);  
    //Validation des Donnees de l'entité Incription  
    if ($form->isSubmitted() && $form->isValid()) {  
  
        $etudiant=$inscription->getEtudiant();  
        //Generation du Matricule à partir du Service  
  
        $etudiant->setMatricule($genarator->generateMatricule());  
        //Validation des Champs de l'entité etudiant  
        $errorEtudiants = $validator->validate($etudiant);  
        if(count( $errorEtudiants)>0){  
            return $this->render('inscription/add.html.twig', [  
                'form' => $form->createView(),  
                "errors"=>$errorEtudiants  
            ]);  
        }  
        //Encodage du Mot de Passe  
        $encoded =  
$encoder->hashPassword($etudiant,$etudiant->getPassword());  
        $etudiant->setPassword($encoded);  
        $inscription->setEtudiant( $etudiant);  
        //Recuperation de l'Objet AnneeEncours  
        $annee= $repoAnnee->find(  
            $session->get("annee_encours")  
            ->getId());  
  
        $inscription->setAnneeScolaire($annee);  
        //Enregistrement de l'inscription
```

```

    $manager->persist($inscription);
    $manager->flush();
    //Redirection vers la Vue Liste apres Incription
    return $this->redirectToRoute('inscription_liste');
}
return $this->render('inscription/add.html.twig', [
    'form' => $form->createView(),
]);
}

```

#View add dans add.html.twig dans templates/inscription

```

{{ parent() }}
<style style="text/css">
.m-l-error-message{
    margin-left: 185px;
}
</style>

{%- endblock %}
{%- block page_content %}


{{ form_start(form) }}
<div class="form-group row">
    <label for="" class="col-form-label col-sm-2">Nom et
Prenom</label>
    <div class="col-sm-10">
        {{ form_widget(form.etudiant.nomComplet) }}<br/>
    </div>
    <div class="m-l-error-message">
        {{ form_errors(form.etudiant.nomComplet) }}<br/>
    </div>
</div>
<div class="form-group row">
    <label for="" class="col-form-label
col-sm-2">Tuteur</label>
    <div class="col-sm-10">
        {{ form_widget(form.etudiant.tuteur) }}<br/>
    </div>
    <div class="m-l-error-message">
        {{ form_errors(form.etudiant.tuteur) }}<br/>
    </div>
</div>


```

```
</div>
</div>
<div class="form-group row">
    <label for="" class="col-form-label col-sm-2">Email</label>
    <div class="col-sm-10">
        {{ form_widget(form.etudiant.email) }}
    </div>
    <div class="m-l-error-message">
        {{ form_errors(form.etudiant.email) }}
    </div>
</div>
<div class="form-group row">
    <label for="" class="col-form-label
col-sm-2">Password</label>
    <div class="col-sm-10">
        {{ form_widget(form.etudiant.password) }}
    </div>
    <div class="m-l-error-message">
        {{ form_errors(form.etudiant.password) }}
    </div>
</div>

    <div class="form-group row">
        {{ form_label(form.classe, null, { 'attr': { 'class': 'col-form-label col-sm-2' } }) }}
        <div class="col-sm-10">
            {{ form_widget(form.classe) }}
        </div>
        {{ form_errors(form.classe) }}
    </div>
    {{ form_end(form) }}
</div>
{% endblock %}
```

#Faire un Upload Image

```
# Ajout du champ File Avatar dans Entity User
#src/Entity/User.php
class User{
    /**
     * @ORM\Column(type="string")
     *
     * @Assert\NotBlank(message=". ")
     * @Assert\File(mimeTypes={"image/jpeg","image/png"})
     */
    private $avatar;

}

# Ajout du champ FileType Avatar dans le form UserType
#src/Form/UserType.php
class UserType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder,
array $options)
{
    $builder
        // ...
        ->add('avatar', FileType::class,
            ['label' => 'Avatar'])
        // ...
    ;
}
}

# Ajout du champ File vatar dans la view
{# templates/inscription/add.html.twig #}

{{ form_start(form) }}

{{ form_row(form.avatar) }}
{{ form_end(form) }}
```

```

# Configuration du dossier Upload
# config/services.yaml

parameters:
    images_avatars: '%kernel.project_dir%/public/uploads/avatars'

# Dans la fonction add() de InscriptionController
#src/Controller/InscriptionController.php

    $etudiant = new Etudiant();
    $form = $this->createForm(EtudiantType::class,
    $etudiant);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {

        $file = $etudiant->getAvatar();
        //Renommer le Fichier
        $fileName= md5(uniqid()).'.'.$file->guessExtension();

        // Déplace le Fichier dans le répertoire définie dans service
        try {
            $file->move(
                $this->getParameter('images_avatars'),
                $fileName
            );
        } catch (FileException $e) {

        }

        $etudiant->setAvatar($fileName);

        // ... persist the $product variable or any other work

# Affichage de l'image dans la vue
#templates/inscription/details.html.twig

<a href="{{ asset('uploads/avatars/' ~ user.avatar) }}>View</a>

#Chargement de l'image d'un utilisateur + Avatar

use Symfony\Component\HttpFoundation\File\File;
$etudiant->setAvatar(new
File($this->getParameter('images_avatars').'/'. $etudiant->getAvatar())
);

```

NB:Il est possible de créer un service pour la Gestion des upload d'image

#Création d'un Service FileUploader

#src/Service/FileUploader.php

```
namespace App\Service;

class FileUploader{

    private $targetDirectory;

    public function upload(UploadedFile $file)
    {
        $fileName = md5(uniqid()).'.'.$file->guessExtension();

        try {
            $file->move($this->getTargetDirectory(), $fileName);
        } catch (FileException $e) {
            dd($e);
        }

        return $fileName;
    }

    public function getTargetDirectory()
    {
        return $this->targetDirectory;
    }

    public function setTargetDirectory(string $targetDirectory):void{
        $this->targetDirectory=$targetDirectory;
    }
}
```

#Gestion des autorisations en Symfony

Pour interdire l'accès à une page : deux solutions possibles

- soit en configurant la section `access_control` dans `security.yaml`
- soit dans le contrôleur
- soit en utilisant la fonction `is_granted()` dans la vue

e) Security.yml

Pour interdire l'accès à tout utilisateur non-authentifié

```
access_control:  
    - { path: '^/login', roles: IS_AUTHENTICATED_ANONYMOUSLY }  
    - { path: '^/*', roles: [IS_AUTHENTICATED_FULLY] }
```

Pour autoriser les utilisateurs qui ont le rôle admin (`ROLE_ADMIN`)

```
access_control:  
    - { path: '^/login', roles: IS_AUTHENTICATED_ANONYMOUSLY }  
    - { path: '^/*', roles: [ROLE_ADMIN] }
```

Remarques

- La clé `path` accepte les expressions régulières
- Le nom d'un rôle doit être écrit en majuscule
- Les mots composants le nom d'un rôle doivent être séparés par un underscore.
- La clé `roles` accepte une valeur ou un tableau de valeurs

f) Restreindre les routes d'un contrôleur

Pour restreindre l'accès aux routes du contrôleur PersonneController aux utilisateurs ayant le rôle ROLE_ADMIN ou ROLE_USER

```
access_control:  
    - { path: '^/personne', roles: [ROLE_USER, ROLE_ADMIN] }
```

g) Restreindre l'accès à une méthode du controller

Pour restreindre l'accès à une méthode de PersonneController aux utilisateurs authentifiés

```
class PersonneController extends AbstractController  
{  
    /**  
     * @Route("/personne/add", name="personne_add")  
     */  
    public function addForm(EntityManagerInterface  
        $entityManager, Request $request)  
    {  
        $this->denyAccessUnlessGranted('  
            IS_AUTHENTICATED_FULLY');  
        // le reste du contenu  
    }  
}
```

Pour restreindre l'accès à une méthode de PersonneController aux utilisateurs ayant le rôle ROLE_ADMIN

```
class PersonneController extends AbstractController  
{  
    /**  
     * @IsGranted("ROLE_ADMIN")  
     * @Route("/personne/add", name="personne_add")  
     */  
    public function addForm(EntityManagerInterface  
        $entityManager, Request $request)  
    {  
        // le reste du contenu  
    }  
}
```

NB: On peut aussi utiliser

```
/**  
 * @Security("is_granted()", statusCode=404, message="Resource not  
found.")  
*/
```

Pour restreindre l'accès à toutes les méthodes de PersonneController aux utilisateurs ayant le rôle ROLE_ADMIN

```
/**  
 *  
 * @IsGranted("ROLE_ADMIN")  
 */  
class PersonneController extends AbstractController  
{  
    // le contenu  
}
```

h) Restreindre l'accès d'une Partie d'une vue

Pour restreindre une partie de la vue aux utilisateurs ayant le rôle ROLE_ADMIN (contenu à ajouter dans home/index.html.twig)

```
{% if is_granted('ROLE_ADMIN') %}  
    <a href="{{ url('personne_add') }}">  
        Ajouter une personne  
    </a>  
{% endif %}
```

i) Restriction des contrôleurs de notre projet

```
access_control:  
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY}  
    - { path: ^/RP, roles: ROLE_RP }  
    - { path: ^/AC, roles: [ROLE_RP,ROLE_AC] }
```

```
- { path: ^/, roles: [ROLE_RP  
, ROLE_AC, ROLE_ETUDIANT, ROLE_PROFESSEUR] }  
    - { path: ^/profile, roles: ROLE_USER }
```

j) Hiérarchie des rôles

Dans security.yaml

```
security:  
    # ...  
  
    role_hierarchy:  
        ROLE_ADMIN:          ROLE_USER  
        ROLE_SUPER_ADMIN:   [ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]
```

Remarques

- L'utilisateur ayant le rôle ROLE_ADMIN a aussi le rôle ROLE_USER
- L'utilisateur ayant le rôle ROLE_SUPER_ADMIN a aussi les rôle ROLE_ADMIN, ROLE_USER et ROLE_ALLOWED_TO_SWITCH

```
security:  
    role_hierarchy:  
        ROLE_RP:  ROLE_AC  
        ROLE_AC:  [ROLE_ETUDIANT, ROLE_PROFESSEUR]
```

8. Contenu Global du Fichier Security.yml

```
security:  
    role_hierarchy:  
        ROLE_RP:  ROLE_AC  
        ROLE_AC:  [ROLE_ETUDIANT, ROLE_PROFESSEUR]  
    #  
https://symfony.com/doc/current/security/experimental\_authenticators.html
```

```

enable_authenticator_manager: true
# https://symfony.com/doc/current/security.html#c-hashing-passwords
password_hashers:

Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface
: 'auto'

    App\Entity\User:
        algorithm: auto

    #

https://symfony.com/doc/current/security.html#where-do-users-come-from-
user-providers
providers:
    # used to reload user from session & other features (e.g.
switch_user)
    app_user_provider:
        entity:
            class: App\Entity\User
            property: login

firewalls:
    dev:
        pattern: ^/(_profiler|wdt)|css|images|js/
        security: false
    main:
        lazy: true
        provider: app_user_provider
        custom_authenticator:
App\Security\LoginFormAuthenticatorAuthenticator
    logout:
        path: app_logout
        # where to redirect after logout
        target: app_login

        # activate different ways to authenticate
        #

https://symfony.com/doc/current/security.html#firewalls-authentication

    #

https://symfony.com/doc/current/security/impersonating_user.html
    # switch_user: true

    # Easy way to control access for large sections of your site
    # Note: Only the *first* access control that matches will be used

```

```

access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY}
    - { path: ^/profile, roles: ROLE_USER }

```

Les autorisations du Projet

NB:Le RP hérite de toutes les fonctionnalités du AC

a) Menu : Professeur

- o Mes Cours (filtre par date et Classe)
- o Marquer Absence

b) Menu Cours:AC

- o Voir les cours (filtre par date et Classe)
- o Marquer Absences
- o Voir les Absences d'un cours

c) Menu Etudiant :

- o Mes Cours (filtre par date)
- o Voir ses Absences

d) Menu : Sécurité => RP

- o Gérer les admins(RP et AC)
- o admin.html.twig

Faire une Réinscription

```

#Ajout du Champ Matricule dans EtudiantFormType
#Ajout de la Fonction Ajax dans templates/inscription/new.html.twig
    • Cette Fonction permet de charger un étudiant à partir de son
       matricule lorsque l'événement keyup se réalise
#Modification de la fonction add() InscriptionController

```

7. Lister Absences d'un Étudiant

src/Controllers/InscriptionController

```

#[Route('/AC/inscription/absense/{id}', name: 'inscription_absence')]
public function showInscriptionAbsence(Inscription $inscription): Response{
    return $this->render('inscription/inscription.absence.html.twig', [
        'inscrit'=>$inscription
    ]);
}

```

templates/inscription/inscription.absence.html.php

8. Annuler inscription d'un Étudiant

TAF Par les Étudiants

- **Lister les Professeurs**
- **Modifier ses Module**
- **Enregistrement d'un professeur avec ses Modules**

- a) Lister les Planification Planification et filtrer par professeur et par module
 - (1) Page

Liste Des Cours de 2021-2022						
Liste des Cours						
Professeur		Professeur	Modules	Modules	Rechercher	+ Nouvelle
Module	Professeur	Classes	Nbre Heure	Heures Faites	Heures Restantes	Actions
Module 1	Nom et Prenom 1	L2IAGE	40	6	34	+ Seances

Code controller view

b) Ajouter 'une Planification
(1) Page

Nouvelle Planification

Nombre HeureFin	<input type="text"/>
Professeur	<input type="text"/> Nom et Prenom 1
Module	<input type="text"/>
Classes	<input type="checkbox"/> L2IAGE <input type="checkbox"/> L2IAGE <input type="checkbox"/> L1MAE <input type="checkbox"/> L1MAE <input type="checkbox"/> L3GLRS <input type="checkbox"/> L3GLRS <input type="checkbox"/> L2IAGE <input type="checkbox"/> L2IAGE <input type="checkbox"/> L3GLRS <input type="checkbox"/> L2IAGE
<input type="button" value="Enregistrer"/>	

c) Ajouter une séance à Planification

PROFESSEUR

NCI: 1 619 1986 0051

Nom et Prenom : Nom et Prenom 1

Grade : MASTER

PLANIFICATION

Classes: L2IAGE

Module : Module 1

Heures : 40 Heures Faite : 6 Heures Restantes : 34

Les Seances
Nouvelle Seance

Date
HeureDebut
HeureFin

18/10/2021
18:00:00
20:00:00
 Fait

18/10/2021
16:00:00
20:00:00
 Fait

Date
Heure Debut
Heure Fin
+ Enregistrer

28/10/2021
--:--
--:--

d) Valider une séance

PROFESSEUR

NCI: 1 619 1986 0051

Nom et Prenom : Nom et Prenom 1

Grade : MASTER

PLANIFICATION

Classes: L2IAGE

Module : Module 1

Heures : 40 Heures Faite : 6 Heures Restantes : 34

Les Seances
Nouvelle Seance

Date
HeureDebut
HeureFin

18/10/2021
18:00:00
20:00:00
 Fait

18/10/2021
16:00:00
20:00:00
 Fait

28/10/2021
15:00:00
17:00:00
 Planifier

Date
Heure Debut
Heure Fin
Etat
+ Enregistrer

28/10/2021
15:00
17:00
Fait

Code
Planification Type
controller
view

