

Cours de WEBMASTERING

Concepts

- Modéliser un Besoin
- Architecture MVC
- Structuration d'un projet Symfony
- Création d'un Projet Symfony
- ORM:Doctrine
- Entity
- Fixtures
- Contrôleurs
- 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 un design pattern

- En français : Modèle-Vue-Contrôleur
- En anglais : Model-View-Controller

WariCustomAuthenticatorAuthenticator.php

todo: check the credentials inside F:\wamp64\...

Historique

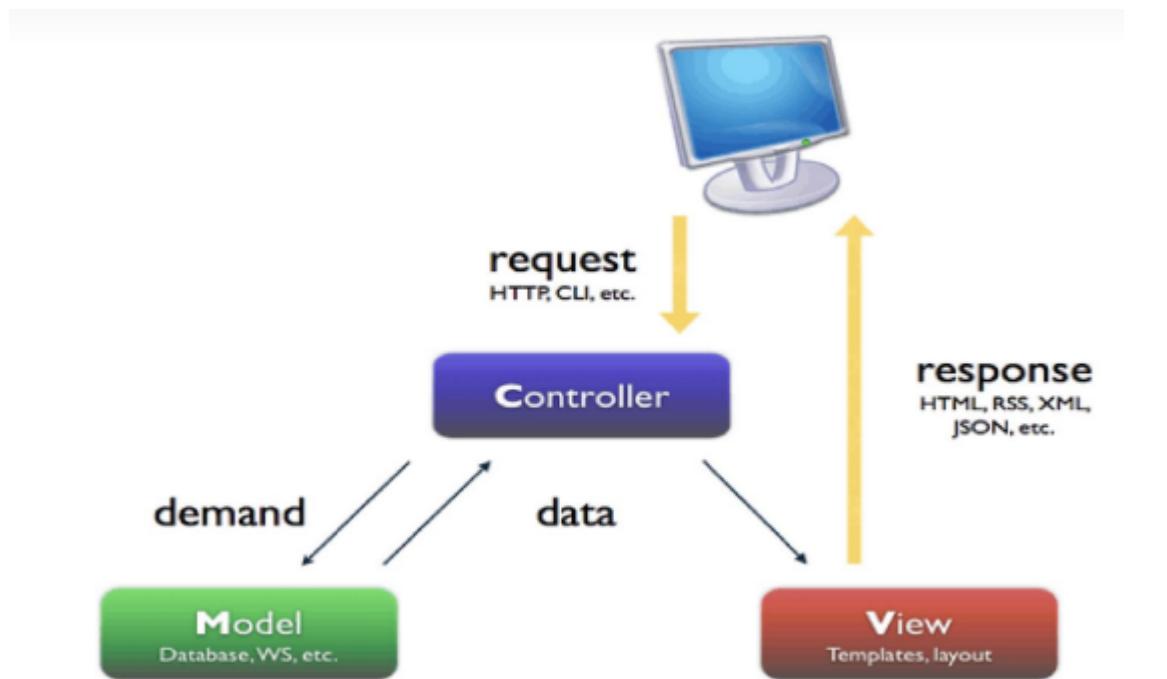
- introduit par Trygve Reenskaug en 1978
- utilisé pour la première fois par **Smalltalk** en 1980

Principe

- permet de bien organiser son code source
- permet de savoir quel rôle joue précisément chaque fichier dans l'application
- consiste à séparer la logique du code en trois parties, modèle (modèle de données), vue (présentation, interface utilisateur) et contrôleur (logique de contrôle, gestion des événements, synchronisation...)

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.

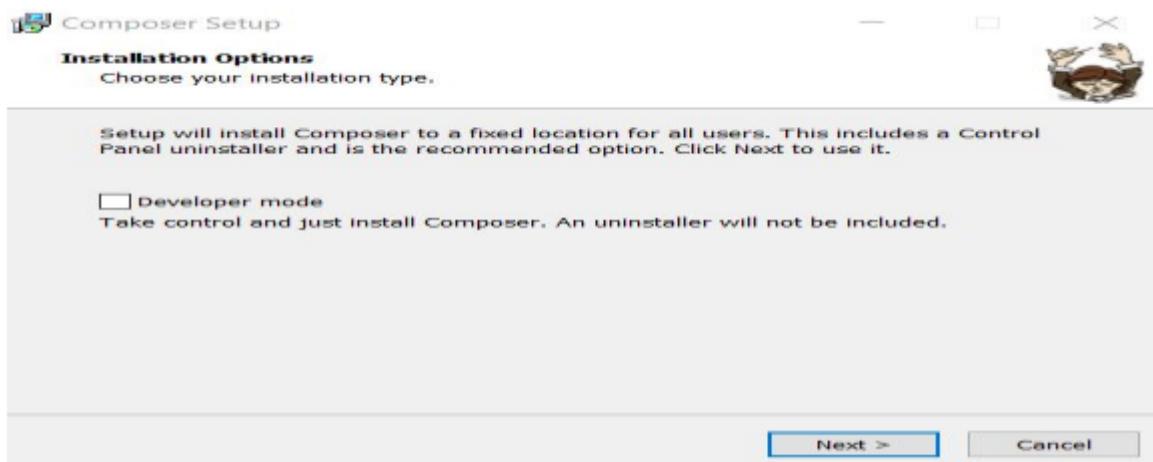
C'est un outil simple et fiable que les développeurs utilisent pour gérer et intégrer des paquets ou des bibliothèques externes dans leurs projets basés sur PHP. Ainsi, ils n'ont pas à créer leurs pages ou applications web à partir de zéro.

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>
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) Cration 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**

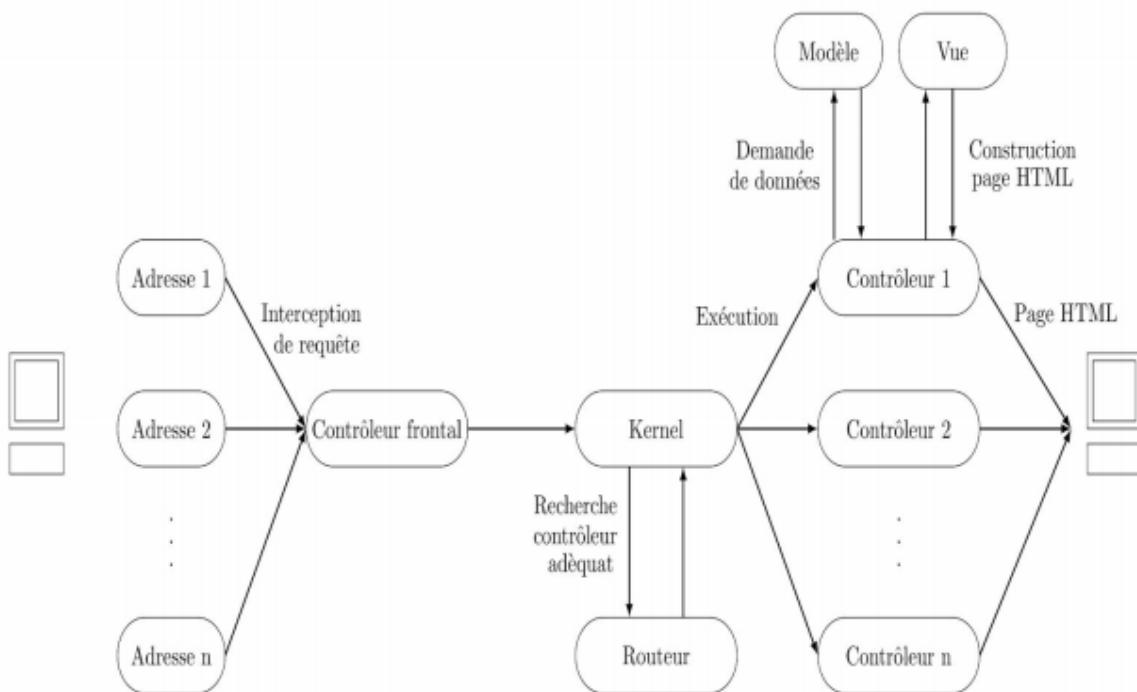
Deux environnements de travail

- **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) Crédation 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) Crédation des entités

`php bin/console make:entity NomEntity`

NB: Pour Valider les Schémas on utilise la commande

`php bin/console doctrine:schema:validate`

(7) Crédation des Entités

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

(8) Crédier 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) Planification Cours**
- (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. Contrôleurs

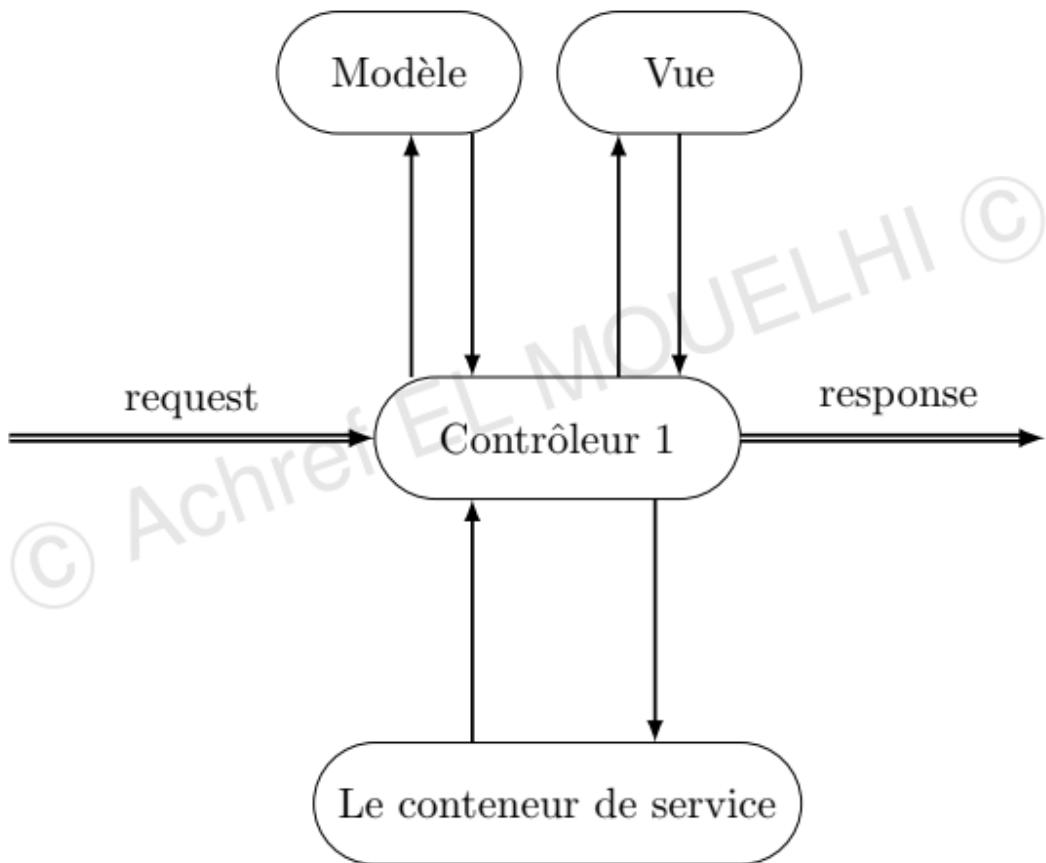
A. Rôle

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. Crédit 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

E. 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

F. 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.

G.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

H.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
```

● Création d'un Service avec Symfony

En plus de proposer un ensemble de services, Symfony donne la possibilité de créer nos propres services.

a) Création du Service

Création du service `SmsGenerate` qui offre la possibilité de générer un Matricule,un NCI.

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

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

b) Utilisation du Service

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.

- **Injection de Dépendance 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**

Fixtures 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 :

Le conteneur prend en charge toute la complexité (gestion de dépendance...)

Question : 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/'
```

● Présentation de Quelques Services de Symfony

1. Le Service Request

2. Le Service Session

a) Utilisation 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') }}`

```
public function index(SessionInterface $session): Response
{
    return $this->render('planification/index.html.twig', [
        'controller_name' => 'PlanificationController',
    ]);
}
```

b) Utilisation des Messages de Sessions

Flash Messages

- Un messages 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 %}
```

3. Le Service Mailer

III) 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:
```

Page 13 / 48 | - | +

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>  
</xliff>
```

```
</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

8. Gestion des autorisations

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

a) 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

b) 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] }
```

C) 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('ROLE_AC') and is_granted('ROLE_RP')")

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
}
```

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 %}
```

d) 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]
```

9. 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 }
```

Cours IV

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

III. Réalisation des Fonctionnalités

A. Menu : Paramétrage

1. Option Classe

■ 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

■ liste.html.twig

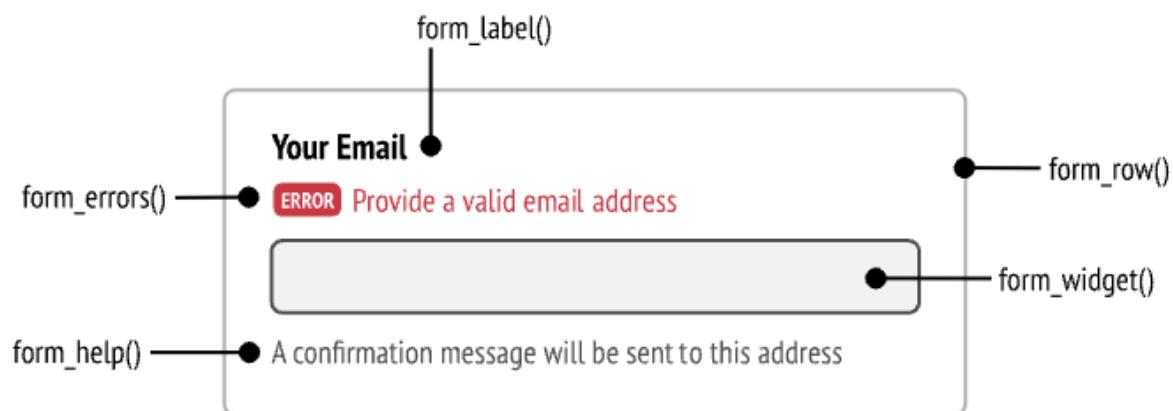
- Lister les Modules

- 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) }}



### Liste des Modules



| ID                  | Libelle                  | Actions                                                                                                                                                                                                                                                                                                                                            |
|---------------------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| &nbsp;{{module.id}} | &nbsp;{{module.libelle}} | <a class="btn btn-primary btn-warning btn-xs" href="{{path('module_edit',{id:module.id})}}" role="button">&lt;i class="fas fa-edit" &gt;&lt;/i&gt; Modifier</a> <a class="btn btn-primary btn-danger btn-xs" href="{{path('module_delete',{id:module.id})}}" role="button">&lt;i class="fa fa-trash" aria-hidden="true"&gt;&lt;/i&gt; Modifier</a> |


```

- **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;
```

3. Menu Inscription: AC

- Lister les Inscrits(Filtre de Classe)
 - Modifier la méthode **index()** dans le controller **Inscription**
 - Charger le Select Annee Scolaire
 - lister les inscrits de l'année en cours
 - filtrer les inscrits par date ou classe
 - **Code**
 - **Controller**

```
# [Route('/AC/inscription/liste', name:  
'inscription_liste', methods: ["GET", "POST"])]  
  
    public function index(InscriptionRepository $repo,  
                          SessionInterface $session,  
                          ClasseRepository $repoClasse,  
                          Request $request,  
                          AnneeScolaireRepository $repoAnnee): Response
```

```

{
    $classes=$repoClasse->findAll();
    //Lorsqu'on change AnneeScolaire encours
    if($request->request->has('btn_annee')) {
        $idAnneeEncours=$request->request->get('annee');
        $anneeEncours= $repoAnnee->find($idAnneeEncours);
        $session->set("annee_encours",$anneeEncours);
    }
    $params=[];
    //Lorsqu'on clique sur le Bouton filtre
    if($request->request->has('btn_annee_filtre')){
        $params=[ "anneeScolaire"=>
$session->get("annee_encours")];
        if(!empty($request->request->get('classe'))){
            $idClasse= $request->request->get('classe');
            $params[ 'classe']=$repoClasse->find($idClasse);
        }
        if(!empty($request->request->get('date'))){

$date=\DateTime::createFromFormat("Y-m-d", $request->request->get('date'));
};

        $params[ 'date']=$date;
    }

} else{
    $params=[ "anneeScolaire"=> $session->get("annee_encours")];
}

$inscrits=$repo->findBy( $params);
return $this->render('inscription/liste.html.twig', [
    "inscrits"=>$inscrits,
    "classes"=>$classes,
]);
}

```

■ Views

```
{% block page_content %}
<div class="container-fluid">
```

```

<a href="{{path('inscription_add')}}" class="btn btn-primary
btn-sm float-right" role="button"><i class="fa fa-plus"
aria-hidden="true"></i> Nouvelle</a>

<h3 class="my-2">Liste des Inscrits</h3>
<div class="container">
    <form method="post" action="{{path('inscription_liste')}}">
        <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>
<table class="table table-bordered w-100">
    <thead class="thead-inverse">
        <tr class="w-100">
            <th>Matricule</th>
            <th>Nom et Prenom</th>
            <th>Classe</th>
            <th>Email</th>
            <th>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>{{inscrit.etudiant.email}}</td>
        <td>
            <a href="#" class="btn btn-primary btn-info
btn-xs" role="button"><i class="fas fa-eye" ></i> Absences</a>
        </td>
    </tr>
    {% endfor %}

</tbody>
</table>
</div>
{% endblock %}

```

- Faire une Inscription
 - Générer le **EtdiantType**
 - Générer le **InscriptionType**
 - Créer la méthode **add()** dans le controller **Inscription**
 - Ajouter le Service **SmsGenerate** qui génère le Matricule
 - Créer la view **add.html.twig** dans **templates/inscription**
 - Faire les Validations dans les Entités
 - User
 - Inscription
 - Etudiant
- **Code**
 - **FormType**

EtudiantType

```

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

        ->add('email', EmailType::class, [
            "required"=>false,
            "constraints"=>[
                new NotBlank([
                    "message"=>"Email est Obligatoire"

```

```
        ]),

    ])

->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' =>'Le Email Existe Déja'
                ]
            )
        ]
    ]);
}
```

```

        ])
    ]

]);
}

```

InscriptionType

```

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

```

■ Controller

```

#[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 Inscription();
    //Creation de l'Objet Formulaire et Mappage avec l'Objet
Inscription
    $form = $this->createForm(InscriptionType::class,
$inscription);
    //Recuperation des Donnees du Formulaire
    $form->handleRequest($request);
    //Validation des Donnees de l'entité Inscription
    if ($form->isSubmitted() && $form->isValid()) {

```

```

        $etudiant=$inscription->getEtudiant();
        //Génération 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 AnnéeEncours
        $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(),
    ]);
}

```

■ Views

```

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

```

```
{% endblock %}

{% block page_content %}


{{ form_start(form) }}


<label for="" class="col-form-label col-sm-2">Nom et  
Prenom</label>


{{ form_widget(form.etudiant.nomComplet) }}



{{ form_errors(form.etudiant.nomComplet) }}


</div>


<label for="" class="col-form-label col-sm-2">Tuteur</label>


{{ form_widget(form.etudiant.tuteur) }}



{{ form_errors(form.etudiant.tuteur) }}


</div>


<label for="" class="col-form-label col-sm-2">Email</label>


{{ form_widget(form.etudiant.email) }}



{{ form_errors(form.etudiant.email) }}


</div>


<label for="" class="col-form-label col-sm-2">Password</label>


{{ form_widget(form.etudiant.password) }}



{{ 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) }}
{%
    endblock %}

```

TAF Par les Étudiants

- Faire la RéInscription

4. Menu Planification:

a) Liste Professeur

- (1) Modifier la méthode **index()** dans le controller **Professeur**
- (2) lister les Professeur

Code

contrôleur

```

#[Route('/RP/professeur/liste', name: 'professeur_liste')]
public function index(ProfesseurRepository $repo,
                      Request $request): Response
{
    $professeurs=$repo->findAll();
    //dd($professeurs);
    return $this->render('professeur/liste.html.twig', [
        'professeurs' => $professeurs,
    ]);
}

```

- (3) paginer la liste

Code

contrôleur

```

//Recuperation du Parametre Get et Initialisation à 1
    $page=(int)$request->query->get("page",1) ;
//Recuperation du nbre Element
    $count=(int)$repo->countWithCreateQuery();
//Fixe le Nombre Element par Page
    $limit=6;
$professeurs=$repo->findPaginate($page,$limit);
//dd( $professeurs);
return $this->render('professeur/liste.html.twig', [
    'professeurs' => $professeurs,
    'count' => $count,
    'limit' => $limit,
    'page' => $page,
]);

```

View

```

<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">&laquo;</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">&raquo;</span>
                    <span class="sr-only">Next</span>
                </a>
            </li>
    </ul>
</nav>

```

- b) Affectation de classe et de Module à un Professeur

Code
ProfesseurType
controller
view

- c) Lister les Classes et les Modules

Code
controller

```
public function showModuleByProfesseur(Professeur $professeur) : Response
{
    return $this->render('professeur/liste.modules.html.twig', [
        'professeur' => $professeur,
    ]);
}
```

view

```
{% extends 'base.html.twig' %}
{% block page_title %}    {% 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">
    <div class="row">
        <div class="col-md-6">
            <div class="card text-white bg-light">

                <div class="card-body">
                    <h4 class="card-title text-align-center">
                        IDENTIFICATION</h4>
                    <p class="card-text">
                        <span
                            class="font-weight-bold">NCI:</span>{{professeur.nci}}<br>
                        <span class="font-weight-bold"> Nom et Prenom :</span>{{professeur.nomComplet}}<br>
                        <span class="font-weight-bold"> Grade :</span>{{professeur.grade}}<br>
                    </p>
                </div>
            </div>
        </div>
        <div class="col-md-6">
            <div class="card text-white bg-light">

                <div class="card-body">
                    <h4 class="card-title text-align-center">
                        CONTACT</h4>
                    <p class="card-text">
                        <span class="font-weight-bold">
                            Email:</span>{{professeur.email}}<br>
                        <br>
                        <br>
                    </p>
                </div>
            </div>
        
```

```

        </div>
    </div>
<div class="row">
    <div class="col-md-6">
        <h3 class="my-2"> 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>

                </tr>
            </thead>
            <tbody>
                {%
                    for module in professeur.modules %
                <tr>
                    <td >{{module.id}}</td>
                    <td>{{module.libelle}}</td>

                </tr>
                {% endfor %}
            
```



```

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

        <div class="col-md-6">
            <h3 class="my-2"> 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>

                    </tr>
                </thead>
                <tbody>
                    {%
                        for affectation in professeur.affectations %
                    <tr>
                        <td >{{affectation.classe.id}}</td>
                        <td>{{affectation.classe.libelle}}</td>

```

```

        </tr>
    {%
        endfor
    %}

    </tbody>
</table>
</div>
</div>
</div>
{%
    endblock
%}

```

- d) Lister les Planification et filtrer par professeur et par module

Code controller view

- e) Ajouter les cours d'une Planification

Code Planification Type controller view

5. Gestion des Autorisations

- a) RP
- b) AC
- c) Professeur
- d) Etudiant

NB:Le RP hérite de toutes les fonctionnalités du AC

6. Menu : Professeur

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

- **Menu Cours:AC**

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

- **Menu Etudiant :**

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

- **Menu : Sécurité => RP**

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

I. Réalisation des Fonctionnalités

- **Réalisation des fonctionnalités**

- Lister les Classes

- création de la fonction listerClasse()
 - Intégration du lien dans Base
 - Intégration de la table des JS(DataTable)
 - Mise en forme de la Table
 - Autorisation

```
# [Route('/classe', name: 'parametres.classes')]  
/**  
 * @IsGranted("ROLE_AC", statusCode=403, message="Vous n'avez pas  
les autorisations nécessaires pour Accéder à cette page")  
 * @param ClasseRepository $repo  
 * @return Response
```

```

    */
public function listerClasse(ClasseRepository $repo): Response
{
//Recuperation des Données
$classes=$repo->findAll();
return $this->render('resposable_pedagogique/classe.html.twig',
[
'menu1' => 'Parametres',
'menu2' => 'Classe',
'datas' => $classes
]);
}

```

- JS pour la mise en forme de la Table

```

$(function () {

    $('#example2').DataTable({
        "language": {
            "url": "//cdn.datatables.net/plug-ins/9dcbeecd42ad/i18n/French.json"
        },
        "paging": true,
        "lengthChange": true,
        "searching": true,
        "ordering": true,
        "info": true,
        "autoWidth": false,
        "responsive": true,
        "lengthMenu": [[5, 10, 20, -1], [5, 10, 20, "All"]]

    });
});
</script>

```

- Ajouter

- Dans la fonction `listerClasse()`
- Création du Formulaire

```
$form = $this->createFormBuilder($classe)
```

```

        ->add('libelle', TextType::class)
        ->add('save', SubmitType::class, ['label' =>
'Enregistrer'])
    ->getForm();

```

- Chargement du Formulaire

```

return $this->render('resposable_pedagogique/classe.html.twig', [
    'form' => $form->createView(),
]) ;

```

- Affichage du Formulaire

```

{{ form_start(form, {"attr":{"class":"form-inline mt-3 "}}) }}
    <div class="form-group col-md-8">
        {{ form_label(form.libelle) }}
        {{ form_widget(form.libelle, {"attr":{"class":"col-sm-7 mr-2"}}) }}
        <div class="form-error">
            {{ form_errors(form.libelle) }}
        </div>
        {{ form_widget(form.save) }}
    </div>
{{ form_end(form) }}

```

**NB: Pour afficher un formulaire ,on peut aussi utiliser
{{form(nom_form)}}**

- Validation des champs au niveau des Entités

```

use Symfony\Component\Validator\Constraints as Assert;
/**
 * @ORM\Column(type="string", length=255)
 * @Assert\NotBlank(message="Le libelle est Obligatoire")
 */
private $libelle;

```

```

/**
 * @ORM\Entity(repositoryClass=ClasseRepository::class)

```

```

*  @UniqueEntity(
*      fields={"libelle"},
*      errorPath="libelle",
*      message="Ce Libelle existe déjà."
*  )
*/
class Classe

```

- Récupération et enregistrement des Données

```

//Hydratation de l'Objet classe à partir de la request
$form->handleRequest($request);
//Verification de la soumission et de la Validation
if ($form->isSubmitted() && $form->isValid()) {
    $classe = $form->getData();
    $manager->persist($classe);
    $manager->flush();

    return $this->redirectToRoute('parametres.classes');
}

```

- Modifier de Classe

- Chargement des données au clique du lien edit

```

if(!is_null($id)){
    $classe=$repo->find($id);
} else{
    $classe=new Classe();
}

```

- Voir les détails d'une Classe

NB: on aura une vue Détail qui inclut les vues liste des inscrits de la classe et liste des cours de la journées.

1. Liste des Inscrits

- Créer la méthode dans le controller
- récupérer les inscrits de la classe sélectionnées
- Afficher ces inscrits dans une vue

```

#[Route('/inscription/annee/{id}', name: 'inscription.classes')]
/**/

```

```

    * @IsGranted("ROLE_AC", statusCode=403, message="Vous n'avez pas
les autorisations nécessaires pour Accéder à cette page")
    * @param ClasseRepository $repo
    * @return Response
    */
    public function inscritsClasse(Classe
$classe, InscriptionRepository $repo): Response
    {

$inscriptions=$repo->findBy(["classe"=>$classe, "annee"=>'2019-2020']
);
    return
$this->render('resposable_pedagogique/inscrit.html.twig', [
        'menu1' => 'Inscription',
        'menu2' => 'liste',
        'datas' => $inscriptions,
    ]);
}

```

2. Liste des Cours de Journée

- Fixtures

```

public function load(ObjectManager $manager)
{
    for ($i = 1; $i <=10; $i++) {
        $data = new PlanificationCours();
        $data ->setAnnee('2019-2020');
        $data ->setNbreHeure(round(20,30));
        $data ->setSemestre($i%2==0?"S1":"S2");
        $data ->setModule($this->getReference("Module".$i));
        $data
->setProfesseur($this->getReference("Professeur".$i));
        $n= $i%2==0?1:2;
        for ($j = 1; $j <=$n; $j++) {
            $data
->addClasse($this->getReference("Classe".$j));
        }

        $manager->persist($data);
        $this->addReference("PlanificationCours".$i,$data);
    }
    $manager->flush();
}

```

```

    }

    public function getDependencies()
    {
        return array(
            ProfesseurFixtures::class,
            ModuleFixtures::class,
            ClasseFixtures::class,
        );
    }
}

```

- Définir la requête dans le repository de Cours
- Récupération des Données dans le contrôleur
- Charger les données dans la vue

Cours V

- **Réalisation des fonctionnalités**
 - **Faire une Inscription**
 - Validation des Entités
 - Générer le Formulaire
 - commande
 - php bin/console make:form**
 - Etudiant

```

$builder
    ->add('nom_complet', TextType::class, [
        'required'=>false
    ])
    ->add('matricule', TextType::class, [
        'required'=>false
    ])
    ->add('tuteur', TextType::class, [
        'required'=>false
    ])
;

```

- **Inscription**

```
$builder
    ->add('annee',ChoiceType::class, [
        'choices' => [
            '2019-2020' => '2019-2020',
            '2020-2021' => '2020-2021',
        ],
    ])
    ->add('etudiant',EtudiantType::class)
    ->add('classe')
    ->add('save', SubmitType::class, ['label' =>
'Enregistrer'])
;
```

- Afficher le Formulaire
- Récupérer et Valider les données
- Enregistrer les l'inscription
- **Faire une Réinscription**
 - Rechercher un etudiant à travers son matricule
 - Charger les données dans le formulaire
 - Enregistrer la Réinscription