

2024



UNIVERSITE PROTESTANTE AU CONGO

# **SPRING FRAMEWORK MODULE 01**

VERSION 3.0.0

OMER PITOU NTUMBA KATUALA, CMA, OCP

## CONTENU

<b>1. A PROPOS DU COURS .....</b>	<b>4</b>
1.1. OBJECTIFS DU COURS .....	4
1.2. TECHNOLOGIES/LIBRAIRIES/OUTILS UTILISES .....	4
1.3. PREREQUIS .....	4
1.4. ASSISTANCE/COMMUNICATION .....	4
1.5. ÉVALUATION .....	4
1.6. TRAVAIL PRATIQUE .....	14
<b>2. LANGAGE SQL.....</b>	<b>ERROR! BOOKMARK NOT DEFINED.</b>
<b>3. SPRING FRAMEWORK : CONCEPTS.....</b>	<b>5</b>
3.1. QUID? .....	5
3.2. SON ARCHITECTURE .....	6
3.3. SPRING BOOT .....	7
3.4. ANNOTATIONS .....	8
3.4.1. Références Web.....	8
3.4.2. Spring Core .....	8
3.4.3. Spring Web .....	9
3.4.4. Spring Data.....	11
3.4.5. Spring Scheduling .....	11
3.4.6. Spring Boot.....	11
3.4.7. Spring Persistence .....	11
3.5. ENVIRONNEMENT DE TRAVAIL .....	12
3.6. STRUCTURE D'UN PROJET SPRING .....	13
3.7. TRAVAIL PRATIQUE .....	<b>ERROR! BOOKMARK NOT DEFINED.</b>
<b>4. TECHNOLOGIES/LIBRARIES/OUTILS : APERCU GENERAL.....</b>	<b>15</b>
4.1. SPRING DATA JPA .....	15
4.1.1. Quid ? .....	15
4.2. THYMELEAF .....	15
4.2.1. Quid ? .....	15
4.2.2. Références Web.....	15
4.3. LOMBOK .....	15
4.3.1. Quid ? .....	15
4.3.2. @NonNull .....	15
4.3.3. @Getter & @Setter .....	16
4.3.4. @ToString.....	16
4.3.5. @NoArgsConstructor & @AllArgsConstructor .....	16
4.3.6. @RequiredArgsConstructor .....	17
4.3.7. @Data .....	18
4.3.8. @Log .....	18
4.3.9. @Cleanup.....	19
4.3.10. Autres: @EqualsAndHashCode, @Value, @Builder, @SneakyThrows, @Synchronized, @With... .....	19
4.4. MYSQL.....	19
4.4.1. Références Web.....	19
4.5. MAVEN .....	19
4.5.1. Quid ? .....	19

4.5.2.	<i>Dépôt « Repository » .....</i>	19
4.5.3.	<i>Standards de nomenclature dans le POM (Project Objet Model).....</i>	19
4.6.	<b>SPRING TOOLS SUITE .....</b>	20
4.6.1.	<i>Créer un Projet.....</i>	20
4.6.2.	<i>Importer un Projet existant .....</i>	20
4.6.3.	<i>Restaurer les dépendances.....</i>	20
4.6.4.	<i>Gérer les Conflits de dépendances.....</i>	20
<b>5.</b>	<b>PRATIQUE: APPLICATION WEB .....</b>	<b>21</b>
5.1.	<i>ARCHITECTURE APPLICATION WEB (THREE LAYERS).....</i>	21
5.2.	<i>COMMENT SPRING MVC TRAITE LES REQUETES/REPONSES ? .....</i>	21
5.3.	<i>APP-01 : MA PREMIERE APPLICATION SPRING BOOT .....</i>	23
5.3.1.	<i>Objectifs .....</i>	23
5.3.2.	<i>Créer un Projet.....</i>	23
5.3.3.	<i>Implémenter la couche « Présentation » : @Controller .....</i>	25
5.3.4.	<i>Configurer l'Internationalisation .....</i>	25
5.3.5.	<i>Gérer le changement de langue .....</i>	26
5.3.6.	<i>Passer des variables au fichier index.html.....</i>	27
5.3.7.	<i>Améliorer la présentation du fichier index.html.....</i>	28
5.3.8.	<i>Changer Configurations dans le fichier application.properties .....</i>	29
5.3.9.	<i>Implémenter la couche « Business Logic » : @Service.....</i>	29
5.3.10.	<i>Implémenter la couche « Data » : @Repository.....</i>	30
5.3.11.	<i>Utiliser l'interface JdbcTemplate dans la couche « Data » .....</i>	34
5.3.12.	<i>Injecter la nouvelle @Repository dans le @Service.....</i>	35
5.3.13.	<i>Utiliser l'interface JdbcTemplate avec la librairie externe « simpleflatmapper” .....</i>	35
5.3.14.	<i>Modifier le @Service en incluant la méthode getRecordsNew() .....</i>	37
5.3.15.	<i>Inclure une barre de recherche.....</i>	37
5.3.16.	<i>Inclure les fonctionnalités Ajouter/Éditer/Supprimer Enregistrement dans le « Backend ».....</i>	40
5.3.17.	<i>Inclure les fonctionnalités Ajouter/Éditer/Supprimer Enregistrement dans le « Frontend ».....</i>	49
5.3.18.	<i>Travail Pratique .....</i>	56
5.4.	<i>APP-02 : DEVELOPPER COMME UN PRO .....</i>	57
5.4.1.	<i>Objectifs .....</i>	60
5.4.2.	<i>Créer un Projet Maven/Projet Parent.....</i>	60
5.4.3.	<i>Créer des Modules Maven .....</i>	61
5.4.4.	<i>Configurer le Module Lib .....</i>	63
5.4.5.	<i>Configurer le Module Data .....</i>	64
5.4.6.	<i>Configurer le Module API.....</i>	65
5.4.7.	<i>Configurer le Module WEB .....</i>	66
5.4.8.	<i>Configurer le Module Starter API.....</i>	67
5.4.9.	<i>Configurer le Module Starter WEB .....</i>	68
5.4.10.	<i>Tester le Projet .....</i>	69
5.4.11.	<i>Apprêter les ressources de l'application.....</i>	69
5.4.12.	<i>Créer les Modèles @Entity dans le module Lib.....</i>	73
5.4.13.	<i>Créer les classes DTO (Data Transfer Object) dans le module Lib.....</i>	75
5.4.14.	<i>Créer les Mappers @Component dans le module Lib .....</i>	77
5.4.15.	<i>Implémenter la couche @Repository dans le module Data.....</i>	79
5.4.16.	<i>Implémenter la couche @Service dans le module Data .....</i>	82
5.4.17.	<i>Implémenter la couche @Controller dans le module Web .....</i>	87
5.4.18.	<i>Implémenter le Thymeleaf Layout.....</i>	89
5.4.19.	<i>Mettre à jour les fichiers .html .....</i>	90
5.4.20.	<i>Tester l'application Web.....</i>	98
<b>6.</b>	<b>PRATIQUE: APPLICATION WEB SERVICE (API) .....</b>	<b>99</b>

6.1.	CONCEPTS .....	99
6.1.1.	<i>Généralités : Http .....</i>	99
6.1.2.	<i>Difference entre Application Web &amp; API .....</i>	99
6.1.3.	<i>Méthodes les plus utilisées (POST, PUT, DELETE, GET...) .....</i>	99
6.1.4.	<i>Status de Réponses http (CREATED, OK, CONFLICT, UNAUTHORIZED, BAD REQUEST...) .....</i>	99
6.2.	PRATIQUE .....	99
6.2.1.	<i>Personnaliser la réponse Http .....</i>	99
6.2.2.	<i>Gérer les Exceptions .....</i>	100
6.2.3.	<i>Implémenter la couche @RestController.....</i>	101
6.2.4.	<i>Installer Postman.....</i>	102
6.2.5.	<i>Tester l'API.....</i>	102
6.3.	TRAVAIL PRATIQUE .....	107
<b>7.</b>	<b>SPRING SECURITY .....</b>	<b>107</b>
7.1.	CONCEPTS .....	107
7.1.1.	<i>Terminologie.....</i>	107
7.1.2.	<i>Architecture .....</i>	109
7.1.3.	<i>Spring Security avec JWT ( JSON Web Token).....</i>	109
7.2.	PREPARER L'ENVIRONNEMENT .....	110
7.2.1.	<i>Dépendances .....</i>	110
7.2.2.	<i>Classe User : @Entity, @Repository, @Service.....</i>	110
7.2.3.	<i>Sécuriser les @Services : Formation, Profession, Etudiant, EtudiantFormation .....</i>	117
7.3.	PRATIQUE : PREALABLES .....	118
7.3.1.	<i>Configurer la sécurité dans le module Data .....</i>	118
7.4.	PRATIQUE : APPLICATION WEB.....	122
7.4.1.	<i>Configurer la sécurité dans le module Web.....</i>	122
7.4.2.	<i>Mettre à jour les fichiers .html .....</i>	124
7.4.3.	<i>Pratique .....</i>	127
7.5.	PRATIQUE : APPLICATION API .....	127
7.5.1.	<i>Configurer la sécurité dans le module API .....</i>	127
7.5.2.	<i>Implémenter @RestController et @Service : Authentification .....</i>	129
7.5.3.	<i>Pratique .....</i>	131
<b>8.</b>	<b>DEPLOYER UNE APPLICATION .....</b>	<b>133</b>
8.1.	CONCEPTS .....	133
8.2.	PRATIQUE .....	133
<b>9.</b>	<b>EXTRA .....</b>	<b>133</b>

## 1. A PROPOS DU COURS

### 1.1. Objectifs du Cours

- Comprendre les concepts de base autour de la plateforme Spring Framework
- Développer une application web complète sous Spring Framework
- Développer une application web service (API) sous Spring Framework
- Comprendre et Intégrer le module de données Spring Data JPA
- Comprendre et Intégrer le module de sécurité Spring Security
- Déployer une application sur un serveur web

### 1.2. Technologies/Libraries/Outils utilisés

- Spring Boot 3
- Java 17/21
- Spring Data JPA
- Spring Security 6
- Thymeleaf
- Lombok
- Maven
- Spring Tools Suite/IntelliJ
- MySQL 8.x.x

### 1.3. Prérequis

- Bonne maîtrise du langage de programmation Java
- Bonne maîtrise de technologies Web : HTML, CSS, Javascript
- Bonne maîtrise de Bases de données (Modélisation & Implémentation): MySQL
- Bonne maîtrise du langage SQL

### 1.4. Assistance/Communication

- Un groupe WhatsApp sera créé pour permettre aux étudiants d'interagir entre eux et avec l'instructeur uniquement autour de préoccupations/difficultés qu'ils rencontrent.
- Seront définis comme administrateurs du groupe : Instructeur, Chef de Promotion et Chef de Promotion Adjoint.
- Un code de bonne conduite sera de stricte application.

### 1.5. Évaluation

- Ce cours est essentiellement un cours à caractère pratique.
- La table d'évaluation se présente comme suit :

Participation	05
Test d'évaluation/Travaux Pratiques	05
Projet individuel	10
Total	20

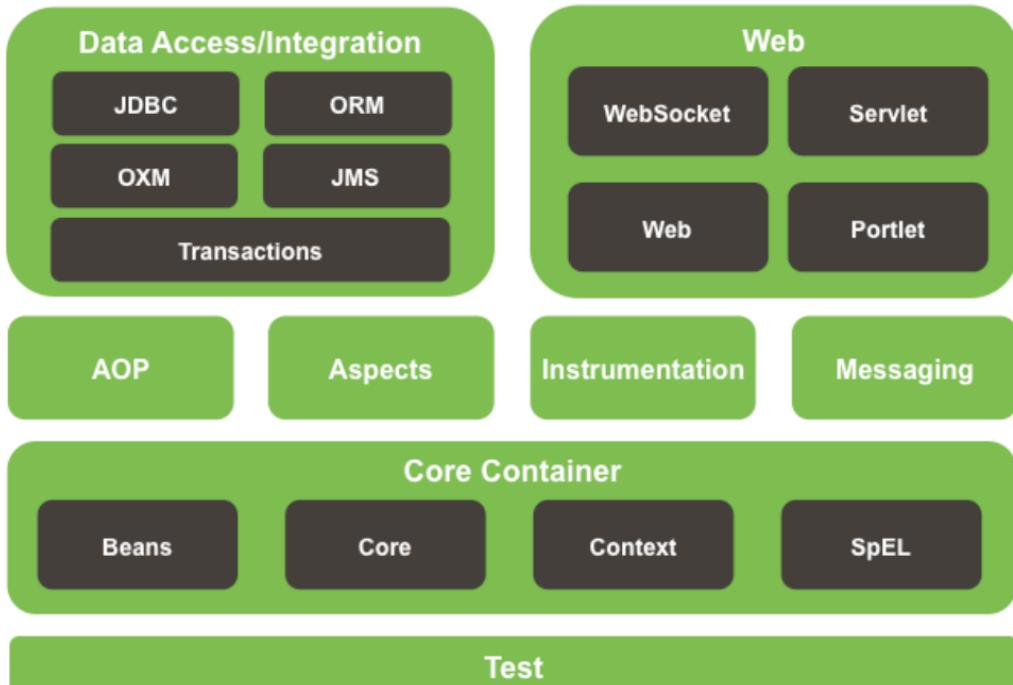
## 2. SPRING FRAMEWORK : CONCEPTS

### 2.1. Quid?

- Spring Framework est une plateforme de développement des applications Java EE (Entreprise Edition), communément appelées « applications d'entreprise ». Elle permet de développer des applications simples, fiables et évolutives.
- Plus utilisée à ce jour, cette plateforme « Open Source » a été initialement développée par Rod Johnson et mise sur le marché sous la licence Apache 2.0 en Juin 2003.
- Cette plateforme offre plusieurs avantages :
  - Programmation Orientée Objet
  - Développement par Modules
  - Intégration avec plusieurs autres frameworks
  - Modélisation MVC (Model-View-Controller)
  - Gestion centralisée des exceptions
  - Gestion de transactions sur les opérations de bases de données
  - Plateforme légère : Consommation minimale de ressources machine (CPU)
  - ...
- Spring Framework tourne autour de 2 principes clés: « Inversion Of Control (IoC) » et « Dependency Injection (DI) ».
  - L'IoC est ce principe faisant que le contrôle de l'application soit assuré par la plateforme elle-même, au travers de configurations, création d'instances de classes et leurs dépendances, alors que dans le modèle traditionnel (procédural), c'est le code inclus dans la fonction principale de l'application qui assure le contrôle (Instance de classes, appel des méthodes...)
  - La DI, aussi appelée « Wiring », est ce principe consistant à injecter une ou plusieurs classes dans une classe X, tout en les gardant indépendantes les unes des autres.
- Appelée plateforme des plateformes (Framework of Frameworks), L'architecture Spring Framework est composée de plusieurs modules: Spring Core, Spring Web, Spring Data Access...

## 2.2. Son architecture

- Aperçu



- Spring Core : est le module principal de SPRING FRAMEWORK qui offre l'environnement où s'applique le principe de IoC décrit ci-haut, et qui est communément appelé « IoC Container ». Cet environnement contient 2 types d'implémentations (interfaces) : Bean Factory et Application Context.
  - Le Bean Factory sert à créer les instances des objets de l'application, faire les configurations nécessaires et rassembler les dépendances nécessaires.
  - L'ApplicationContext offre les fonctionnalités tel que :
    - l'internationalisation (I18N)
    - la validation,
    - la gestion des événements (Event Propagation),
    - la gestion de ressources (Resource Loading). Les ressources gérées dans le IoC Container sont connues sous le nom de Spring Beans.
- Spring Web : est le module qui implémente l'architecture MVC.
- Spring DAO : est le module qui fournit les objets d'accès aux données au travers de technologies tel que : JDBC (Java Database Connectivity), Hibernate, JDO (Java Data Object) ; et implémente aussi la gestion des transactions pendant les opérations sur la base de données.

### 2.3. Spring Boot

- Les modules de l'architecture Spring requièrent un certain nombre de configurations manuelles, complexes...
- Pour remédier à cette complexité, le module **Spring Boot** a été mis en place en 2012. Ce dernier embarque tous les modules de Spring, connus sous le nom de « starters ». Chaque « starter » initialise une série de configurations par défaut, nécessaires à son fonctionnement.
- Spring Boot offre les avantages ci-après :
  - Création des applications web et non-web (« stand-alone »)
  - Embarquement des serveurs web : Tomcat, Jetty
  - Augmentation la productivité
  - Réduction du temps de développement
- Les configurations par défaut sont disponibles sur : <https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html>
- Les valeurs par défaut peuvent être modifiées dans le fichier **application.properties** inclus dans le dossier **Resources** du projet Spring.
  - Plusieurs fichiers application.properties peuvent être créés selon les exigences de votre projet : application-aws.properties, application-dev.properties, application-prod.properties...
    - Le suffixe (aws, dev, prod...) indique donc le profil d'un quelconque environnement
    - La propriété **spring.profiles.active** dans le fichier principal application.properties permet de rendre effectif un environnement
- Quelques Spring Boot starters :
  - spring-boot-starter-web
  - spring-boot-starter-data-jpa
  - spring-boot-starter-security
  - spring-boot-starter-mail
  - spring-boot-starter-thymeleaf
  - spring-boot-starter-batch
  - ...
- Ces starters sont injectés dans un projet Spring sous forme de dépendances, et ce, dans le fichier **pom.xml** (Project Object Model).
  - Ce fichier pom.xml contient non les dépendances, mais aussi les paramètres nécessaires à la configuration du projet Spring.
  - Le gestionnaire de dépendances **Maven** utilise donc ce fichier pour configurer le projet.
- La plateforme Spring s'appuie aussi sur un ensemble d'**annotations** qui servent à configurer les composants du projet.

## 2.4. Annotations

### 2.4.1. Références Web

<https://springframework.guru/spring-framework-annotations/>

<https://www.digitalocean.com/community/tutorials/spring-annotations#spring-annotations>

### 2.4.2. Spring Core

	Portée	Description
@Configuration	Classe	Indique que la classe contient de définitions de Beans
@Bean	Méthode	Indique que la Méthode est définie comme une Bean
@ComponentScan	Classe	Configure les composants (Components) à scanner lors de l'initialisation des classes @Configuration
@Import	Classe	Permet d'inclure plusieurs autres classes de configuration, dans une classe @Configuration
@ImportResource	Classe	Permet d'inclure les configurations incluses dans un fichier XML, dans une classe de configuration
@PropertySource	Classe	Indique la liste de fichiers de propriétés (.properties) à ajouter à l'environnement Spring
@Component	Classe	Indique que la classe est définie comme Spring Bean
@Controller	Classe	Indique que c'est une classe @Component définie sous la couche « Présentation »/Application Web
@RestController	Classe	Indique que c'est une classe @Component définie sous la couche « Présentation »/API (Web Service)
@Service	Classe	Indique que c'est une classe @Component définie sous la couche « Business Logic »
@Repository	Classe	Indique que c'est une classe @Component définie sous la couche « Data »
@Autowired	Variable/ Méthode	Permet d'injecter une dépendance
@Qualifier	Variable/ Méthode	Permet de spécifier le nom de la ressource à injecter, en cas de plusieurs Beans créées de même type. Elle est toujours associée à l'annotation @Autowired

@Primary	Méthode	Permet de définir une préférence sur une Bean lorsqu'il existe plusieurs de même type
@Lazy	Classe	Permet de signifier que l'initialisation de la classe ne se fera qu'aussitôt qu'elle soit requise, parce que par défaut toutes les classes annotées et leurs dépendances sont initialisées au démarrage de l'application. Cette annotation s'applique sur les classes @Component, @Configuration
@Value	Attribut	Permet d'indiquer une valeur par défaut qui sera assignée à un attribut d'une classe, un paramètre (Constructeur, Méthode)
@Scope	Méthode	Permet de définir le cycle de vie d'une Bean : singleton (Valeur par défaut), prototype (Instance créée à chaque réquisition de l'environnement), request, session, global-session
@Required	Méthode	Permet de rendre obligatoire l'assignation d'une valeur à la méthode Setter d'une Bean
@Profile	Classe/ Méthode	Permet de limiter l'utilisation de certains composants à un profil d'environnement bien déterminé

#### 2.4.3.Spring Web

	Portée	Description
@ModelAttribute	Classe	Permet de mapper un paramètre/valeur renvoyée à un modèle accessible sur le Web
@RequestMapping	Classe/ Méthode	Permet de mapper les requêtes web
@RequestBody	Paramètre	Permet de mapper une requête http à un Objet
@PathVariable	Paramètre	Permet d'extraire le paramètre inclus dans une URI (Unique Resource Identifier) : /employes/18939
@RequestParam	Paramètre	Permet d'extraire le paramètre inclus dans une URL (Unique Resource Location) : /employes/ ?matricule=18939
@RequestPart	Paramètre	Idem comme @RequestParam, mais cette fois ci pour les paramètres de type Fichier (File)
@ResponseBody	Méthode	Permet de mapper l'Objet renvoyé au corps d'une réponse http

@ResponseStatus	Classe/ Méthode	Marque une méthode ou une classe Exception avec le Code Status et le Message à renvoyer au Client
@ExceptionHandler	Méthode	Permet de gérer les exceptions au niveau du « Controller »

#### 2.4.4.Spring Data

- `@EnableJpaRepositories`
- `@EnableTransactionManagement`
- `@Transactional`

#### 2.4.5.Spring Scheduling

- `@EnableScheduling`
- `@Scheduled`

#### 2.4.6.Spring Boot

- `@SpringBootApplication` : permet de marquer la classe qui va servir de classe principale « Main Class » de l'application Spring Boot. Elle embarque les annotations : `@Configuration`, `@EnableAutoConfiguration` et `@ComponentScan` ainsi que leurs attributs par défaut

#### 2.4.7.Spring Persistence

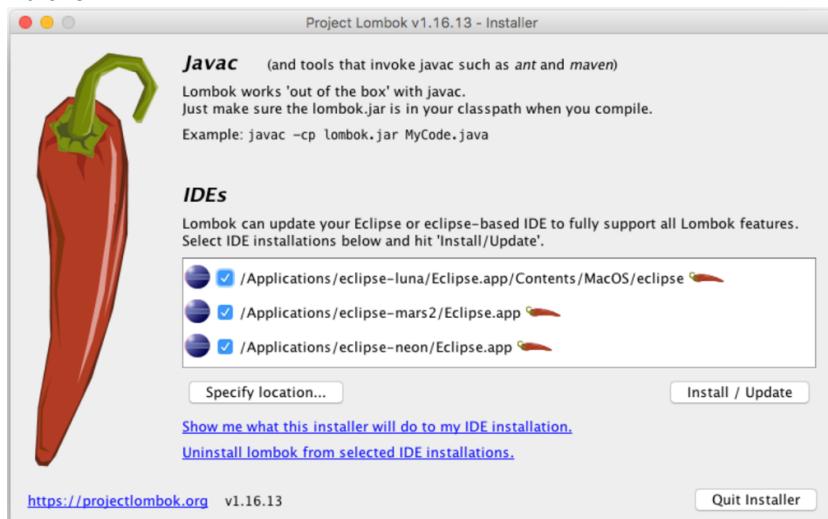
- Général
  - `@Entity`
  - `@Table`
  - `@Column`
  - `@Id`
  - `@GeneratedValue`
  - `@.GenerationType`
  - `@Transient`
  - `@CreatedBy`, `@LastModifiedBy`, `@CreatedDate`, `@LastModifiedDate`
- Validation
  - `@NotEmpty`
  - `@NotNull`
  - `@Min`
  - `@Size`

## 2.5. Environnement de travail

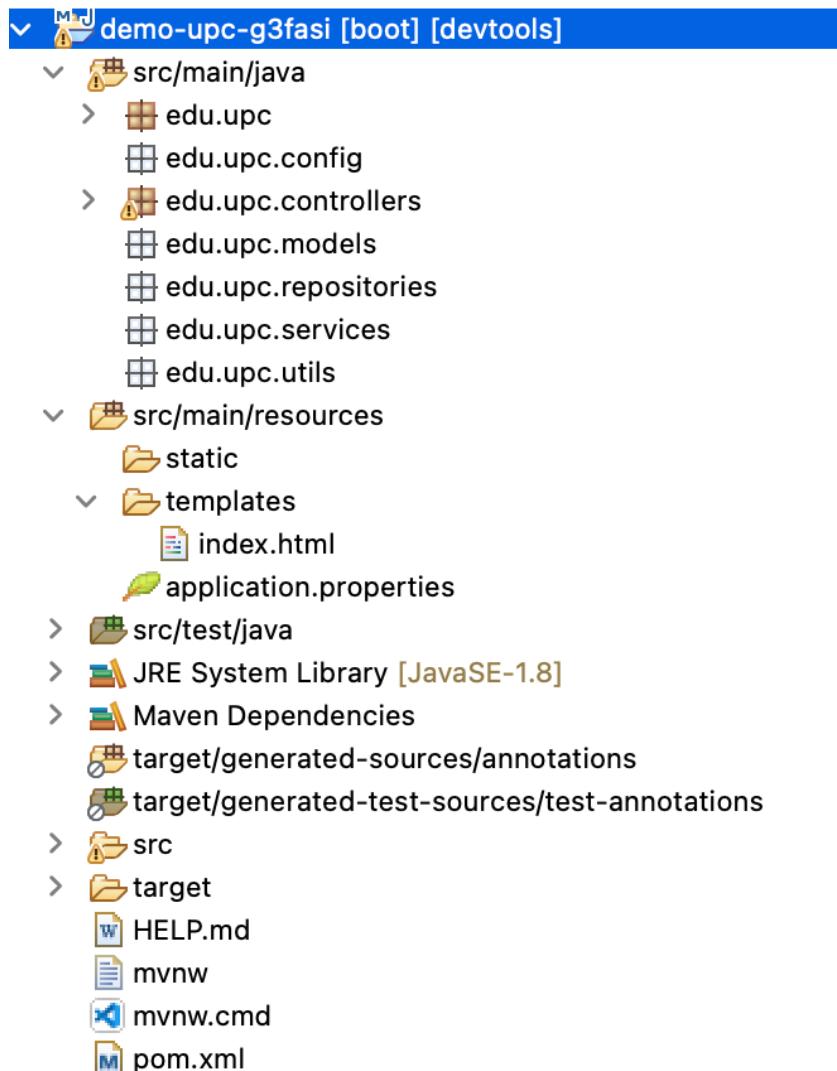
- Télécharger/Installer Spring Tools Suite :  
<https://spring.io/tools>
- Télécharger/Installer MySQL :  
<https://dev.mysql.com/downloads/windows/installer/5.7.html>
- Télécharger/Installer Lombok :
  - Télécharger sur <https://projectlombok.org/download> et lancer le fichier lombok...jar
  - Ou encore Ajouter Lombok...jar dans le POM, le Retrouver dans le dossier « Maven Dependencies » du projet, et l'Exécuter (Run As/Java Application)

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
```

- Installer



## 2.6. Structure d'un Projet Spring



## 2.7. Travail Pratique

- Adhérer au groupe WhatsApp
- Préparer l'environnement de travail
  - Installer MYSQL
  - Installer Java 17/21
  - Spring Tools Suite/IntelliJ
- Former de groupes de travail

### 3. TECHNOLOGIES/LIBRARIES/OUTILS : APERCU GENERAL

#### 3.1. Spring DATA JPA

##### 3.1.1. Quid ?

- Spring Data regroupe de modules qui offrent chacun une possibilité de configurer et d'injecter une implémentation d'un « repository » pour un système de gestion de base de données. Ces modules sont : Spring Data JDBC, Spring Data JPA, Spring Data R2DBC, Spring Data MongoDB, Spring Data REST, Spring Data Apache Cassandra, etc...
- Spring Data JPA est le module qui nous permet d'interagir avec une base de données relationnelles en représentant les objets du domaine métier sous la forme **d'entités JPA**.
- Spring Data JPA fournit l'interface **JpaRepository<T, ID>** qui hérite de **CrudRepository<T, ID>** et qui fournit un ensemble de méthodes plus spécifiquement adaptées pour interagir avec une base de données relationnelle.
- Pour définir un « repository », il suffit de créer une interface qui hérite d'une des interfaces ci-dessus.

#### 3.2. Thymeleaf

##### 3.2.1. Quid ?

- Thymeleaf est un moteur de template (template engine), sous licence Apache 2.0, écrit en Java pouvant générer du XML/XHTML/HTML5.
- Thymeleaf peut être utilisé dans un environnement web ou non web.
- Son but principal est d'être utilisé dans un environnement web pour la génération de vue (views) pour les applications web basées sur le modèle MVC.

##### 3.2.2. Références Web

- <https://www.thymeleaf.org>

#### 3.3. Lombok

##### 3.3.1. Quid ?

- Lombok est une librairie permettant de générer automatiquement du code lors de la compilation des classes JAVA.
- Référence web : <https://medium.com/javarevisited/all-the-16-lombok-annotations-explained-in-a-4-minute-article-926f71934ec6>

##### 3.3.2. @NonNull

- Cette annotation permet de valider un champ.

The image shows two side-by-side code snippets in a dark-themed code editor. Both snippets are titled 'doSomething' and contain the same Java code:

```
void doSomething(Object arg) {  
    if(arg == null) {  
        throw new NullPointerException();  
    }  
    // ...  
}
```

### 3.3.3. @Getter & @Setter

- Cette annotation permet de générer automatiquement les Getters/Setters des attributs d'une classe.

The image shows two side-by-side code snippets in a dark-themed code editor. The left snippet shows a class 'FullName' with manual getter and setter methods:

```
public class FullName {  
    private String firstName;  
    private String lastName;  
  
    public String getFirstName() {  
        return firstName;  
    }  
    public void setFirstName(String firstName) {  
        this.firstName = firstName;  
    }  
    public String getLastName() {  
        return lastName;  
    }  
    public void setLastName(String lastName) {  
        this.lastName = lastName;  
    }  
}
```

The right snippet shows the same class 'FullName' using the @Getter and @Setter annotations:

```
@Getter  
@Setter  
public class FullName {  
    private String firstName;  
    private String lastName;  
}
```

### 3.3.4. @ToString

- Cette annotation permet d'écraser la méthode `toString()` d'un objet.

The image shows two side-by-side code snippets in a dark-themed code editor. The left snippet shows a class 'FullName' with an overridden `toString()` method:

```
public class FullName {  
    private String firstName;  
    private String lastName;  
  
    @Override  
    public String toString() {  
        return "FullName(firstName=" + firstName + ", lastName=" + lastName + ")";  
    }  
}
```

The right snippet shows the same class 'FullName' using the `@ToString` annotation:

```
@ToString  
public class FullName {  
    private String firstName;  
    private String lastName;  
}
```

### 3.3.5. @NoArgsConstructor & @AllArgsConstructor

- Cette annotation permet de générer le constructeur d'une classe.

The image shows two side-by-side code editors. Both have a dark background with red, yellow, and green window control buttons at the top. The left editor contains the following Java code:

```
public class FullName {  
    private String firstName;  
    private String lastName;  
  
    public FullName() {}  
  
    public FullName(String firstName, String lastName) {  
        this.firstName = firstName;  
        this.lastName = lastName;  
    }  
}
```

The right editor contains the following Java code:

```
@NoArgsConstructor  
@AllArgsConstructor  
public class FullName {  
    private String firstName;  
    private String lastName;  
}
```

### 3.3.6. @RequiredArgConstructor

- Cette annotation permet de générer le constructeur d'une classe pour les attributs de type Final.

The image shows two side-by-side code editors. Both have a dark background with red, yellow, and green window control buttons at the top. The left editor contains the following Java code:

```
public class FullName {  
    private final String firstName;  
    private String lastName;  
  
    public FullName(String firstName) {  
        this.firstName = firstName;  
    }  
}
```

The right editor contains the following Java code:

```
@RequiredArgsConstructor  
public class FullName {  
    private final String firstName;  
    private String lastName;  
}
```

### 3.3.7. @Data

- Cette annotation permet de générer automatiquement les Getters/Setters des attributs d'objets, et les annotations @ToString, @EqualsAndHashCode.



```
public class FullName {
    private final String firstName;
    private String lastName;

    public FullName(String firstName) {
        this.firstName = firstName;
    }
    public String getFirstName() {
        return firstName;
    }
    public String getLastname() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    @Override
    public boolean equals(Object o) {
        // ...
    }
    @Override
    public int hashCode() {
        return Objects.hash(firstName, lastName);
    }
}
```

```
@Data
public class FullName {
    private final String firstName;
    private String lastName;
}
```

### 3.3.8. @Log

- Cette annotation permet de créer une instance Log pour une classe.



```
public class FullName {
    private static final org.slf4j.Logger log =
        org.slf4j.LoggerFactory.getLogger(FullName.class);

    private String firstName;
    private String lastName;
}
```

```
@Slf4j
public class FullName {
    private String firstName;
    private String lastName;
}
```

### 3.3.9.@Cleanup

- Cette annotation permet d'implémenter une interface Closable, et est similaire à la combinaison **try-with-ressources**.

```
public void doSomething(String file) throws IOException {
    InputStream in = new FileInputStream(file);
    try {
        doSomethingElse(in);
    } finally {
        in.close();
    }
}
```

```
public void doSomething(String file) throws IOException {
    @Cleanup InputStream in = new FileInputStream(file);
    doSomethingElse(in);
}
```

### 3.3.10. Autres: @EqualsAndHashCode, @Value, @Builder, @SneakyThrows, @Synchronized, @With.

## 3.4. MySQL

### 3.4.1. Références Web

- <https://www.mysqltutorial.org>

## 3.5. Maven

### 3.5.1. Quid ?

- Maven est un outil permettant de configurer un projet Java et ses dépendances.
- POM : est le fichier de configuration de Maven. Il contient les informations sur le projet ainsi que les dépendances requises pour ce dernier.
- <https://maven.apache.org>

### 3.5.2. Dépôt « Repository »

- <https://mvnrepository.com>

### 3.5.3. Standards de nomenclature dans le POM (Project Objet Model)

- Référence Web : <https://maven.apache.org/guides/mini/guide-naming-conventions.html>
- Standards

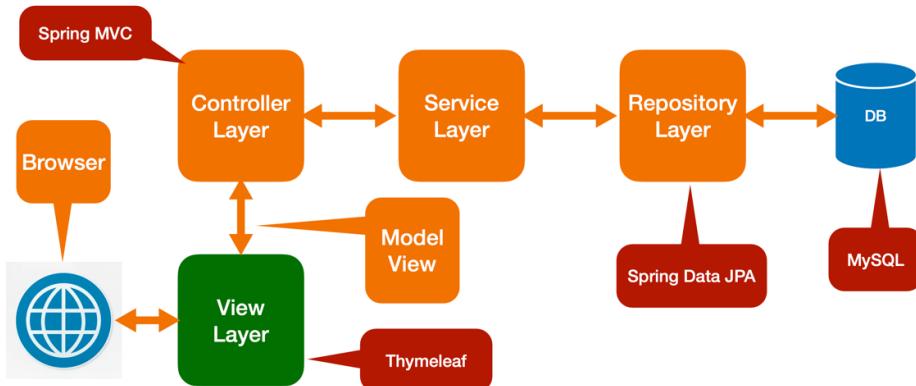
- o Group : identifie de façon unique votre projet dans le répertoire de vos projets, et doit le nom renversé d'un domaine web sous votre contrôle : com.finger.
- o Artifact : est le nom du module de votre projet, et le fichier compilé (jar/war) portera ce nom.
- o Version : indique la version du module (Valeur par défaut 0.0.1-SNAPSHOT)

### 3.6. Spring Tools Suite

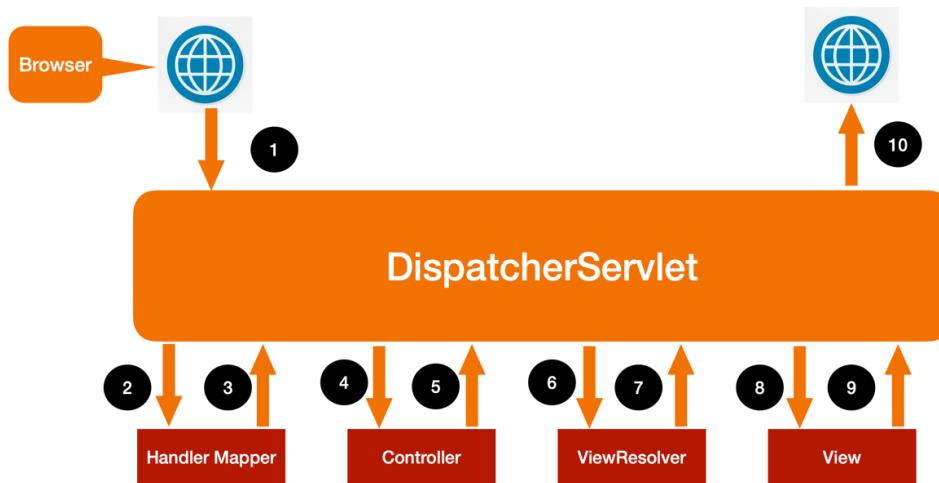
- 3.6.1. Crée un Projet
- 3.6.2. Importer un Projet existant
- 3.6.3. Restaurer les dépendances
- 3.6.4. Gérer les Conflits de dépendances

## 4. PRATIQUE: APPLICATION WEB

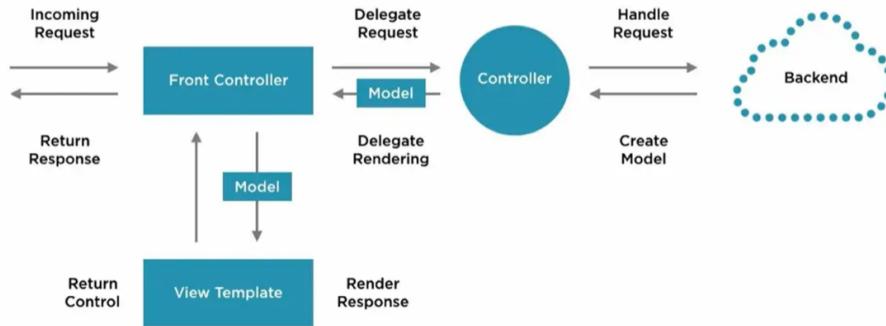
### 4.1. Architecture Application Web (Three Layers)



### 4.2. Comment Spring MVC traite les requêtes/réponses ?



## Request / Response Lifecycle



### Notes

- Une interface appelée « **Dispatcher Servlet** » (Front Controller) reçoit la requête en premier, et la transfère vers le « **Controller** » approprié en tenant compte d'une série de chemins d'accès définis au travers de ce qu'on appelle « **Handler Mapper** »
- Ce « **Controller** » extrait, traite l'information contenue dans la requête, et renvoie le résultat à l'interface « **Dispatcher Servlet** » sous forme d'un objet appelé « **Object Model** »
- L'interface « **Dispatcher Servlet** » renvoie le résultat au Client par le biais d'une interface appelée « **View Resolver** ».

### Concepts

- Model** : encapsule les données de l'application sous forme d'objet.
- View** : se charge de rendre le « **Model** » sous une forme lisible par le Client, généralement sous forme de HTML qui peut être interprété par le navigateur.
- Controller** : traite les requêtes du Client, prépare la réponse sous forme de « **Model** », et la passe à la « **View** » pour affichage.

### Composants Spring MVC

- Dispatcher Servlet
- Controller
- Handler Method
- View Resolver
- View
- Model

#### 4.3. Requête/Réponse

#### 4.4. Structure d'un Projet

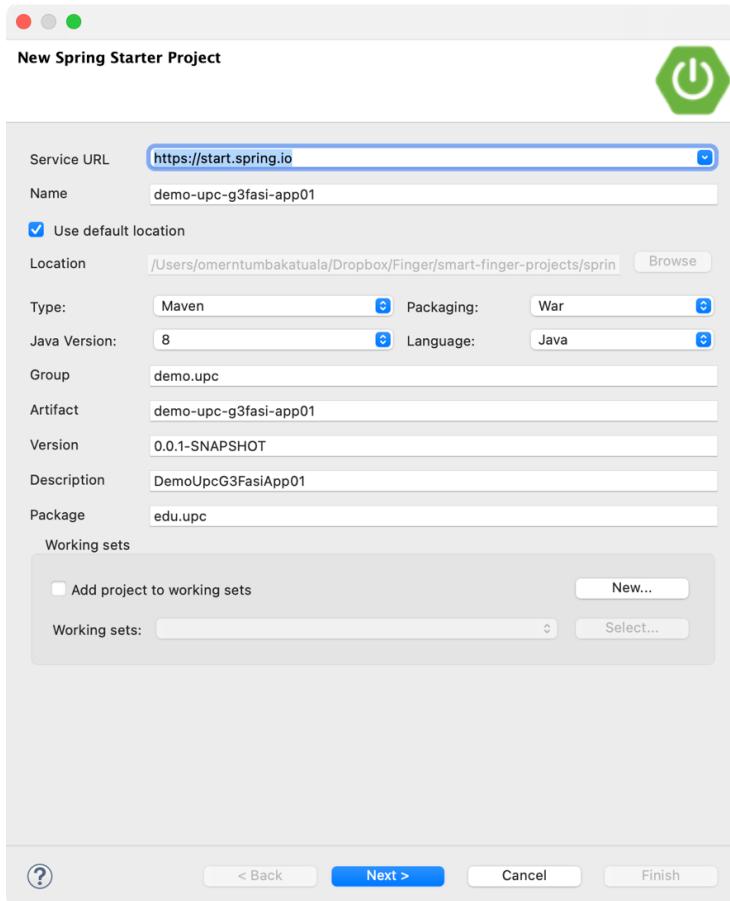
#### 4.5. App-01 : Ma Première Application Spring Boot

##### 4.5.1. Objectifs

- Comprendre la structure d'un projet Spring Boot
- Comprendre la gestion de dépendances dans le POM
- Comprendre et Implémenter les couches d'une application web
- Crée une application CRUD (Create, Read, Update, Delete) : Crée, Lire, Modifier, Supprimer
- Implémenter l'internationalisation
- Gérer les exceptions
- Exécuter une application Spring Boot

##### 4.5.2. Créer un Projet

- Lancer Spring Tools Suite (STS)
- Crée Nouveau Projet : Menu File/New/Spring Starter Project



- Cliquer sur Next/Sélectionner la version 2.7.8
- Sélectionner les dépendances
  - Developer Tools : Spring Boot DevTools, Lombok, Spring Configuration Processor
  - Template Engines : Thymeleaf
  - Web : Spring Web
- Cliquer sur Next/Finish
- Inspecter la structure du Projet
  - demo-upc-g3fasi-app01 [boot] [devtools]
  - > src/main/java
  - > src/main/resources
  - > src/test/java
  - > JRE System Library [JavaSE-1.8]
  - > Maven Dependencies
  - target/generated-sources/annotations
  - target/generated-test-sources/test-annotations
  - > src
  - > target
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml
- Cliquer droit sur le nom du Project/New/Package, et Créer les « packages » :
  - edu.upc.config,
  - edu.upc.controllers,
  - edu.upc.services,
  - edu.upc.repositories,
  - edu.upc.models,
  - edu.upc.utils

#### 4.5.3.Implémenter la couche « Présentation » : @Controller

- Sous le package edu.upc.controllers, Créer une classe @Controller WelcomeController (Cliquer droit sur le package/New/Class)

```
@Controller
public class WelcomeController {

    @RequestMapping("/")
    public String getHome(HttpServletRequest request) { // Handler Method

        return "index"; // View Name
    }

}
```

---

- Sous le dossier Resources/Templates, Créer un fichier index.html et Incrire votre nom complet
- Exécuter le projet : Cliquer droit sur le nom du Projet/Run As/ Spring Boot App
- Taper sur le navigateur : http://localhost:8080

#### 4.5.4.Configurer l'Internationalisation

- Créer deux fichiers .properties sous le dossier Resources (Bundle.properties, Bundle\_fr.properties), et définir les propriétés ci-dessous :
  - Bundle.properties

```
firstName=First Name
lastName=Last Name
gender=Gender
maritalStatus=Marital Status
placeOfBirth=Place Of Birth
dateOfBirth=Date Of Birth
numberOfChildren=#Children
address=Address
```
  - Bundle\_fr.properties

```
firstName=Prenom
lastName=Nom
gender=Genre
maritalStatus=Etat-Civil
placeOfBirth=Lieu de Naissance
dateOfBirth=Date de Naissance
numberOfChildren=Nbre Enfants
address=Adresse
```
- Configurer le @Bean MessageSource et @Bean LocaleResolver dans un fichier @Configuration

```

@Configuration
public class AppConfig {

    @Bean(name = "messageSource")
    public MessageSource messageSource() {
        ReloadableResourceBundleMessageSource resource = new ReloadableResourceBundleMessageSource();
        resource.setBasename("classpath:Bundle");
        resource.setDefaultEncoding("ISO-8859-1"); // UTF-8
        resource.setAlwaysUseMessageFormat(true); // Handle Apostroph & others
        resource.setUseCodeAsDefaultMessage(true);
        return resource;
    }

    @Bean(name = "localeResolver")
    public LocaleResolver localeResolver() { // Identify which locale is used
        return new SessionLocaleResolver();
    }
}

```

- Mettre à jour le fichier index.html

```

<table>
  <thead>
    <tr>
      <th th:text="#{firstName}">firstName</th>
      <th th:text="#{lastName}">lastName</th>
      <th th:text="#{gender}">gender</th>
      <th th:text="#{maritalStatus}">maritalStatus</th>
      <th th:text="#{placeOfBirth}">placeOfBirth</th>
      <th th:text="#{dateOfBirth}">dateOfBirth</th>
      <th th:text="#{numberOfChildren}">numberOfChildren</th>
      <th th:text="#{address}">address</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Jacques</td>
      <td>Kabeya</td>
      <td>M</td>
      <td>M</td>
      <td>Kinshasa</td>
      <td>01/02/1975</td>
      <td>2</td>
      <td>Ngaliema</td>
    </tr>
    <tr>
      <td>Jean</td>
      <td>Kanku</td>
      <td>M</td>
      <td>M</td>
      <td>Kinshasa</td>
      <td>01/02/1974</td>
      <td>5</td>
      <td>Gombe</td>
    </tr>
  </tbody>
</table>

```

#### 4.5.5. Gérer le changement de langue

- Créer une classe @Configuration WebConfig qui va implémenter l'interface WebMvcConfigurer
- Redéfinir la méthode addInterceptors() pour ajouter un intercepteur du paramètre de changement de langue, dans le registre des intercepteurs

```
@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        LocaleChangeInterceptor localeIntcp = new LocaleChangeInterceptor();
        localeIntcp.setParamName("locale");
        registry.addInterceptor(localeIntcp);
    }

}
```

- Taper sur le navigateur : <http://localhost:8080/?locale=fr> ou <http://localhost:8080/?locale=en>

#### 4.5.6. Passer des variables au fichier index.html

- Créer la classe Employe dans le package edu.upc.models

```
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
public class Employe {

    private String prenom;
    private String nom;
    private String genre;
    private String etatCivil;
    private String lieuNais;
    private LocalDate dateNais;
    private int nbreEnfants;
    private String adresse;

}
```

- Injecter l'objet Model dans la Handler Method qui retourne index.html
- Ajouter les propriétés suivantes dans les 2 fichiers Bundle.properties :

- welcome=Bienvenue {0},
- error.noDataFound=Aucune donnée trouvée

- Changer le code sous le @Controller

```
@Controller
public class WelcomeController {

    @RequestMapping("/")
    public String getHome(HttpServletRequest request, Model model) { // Handler Method

        // Data
        List<Employe> data = new ArrayList<>();
        data.add(new Employe("Jacques", "Kabeya", "M", "M", "Kinshasa", LocalDate.of(1975, 3, 1), 2, "Ngaliema"));
        data.add(new Employe("Jean", "Kanku", "M", "M", "Kinshasa", LocalDate.of(1974, 3, 1), 5, "Gombe"));

        // Assign
        model.addAttribute("user", "Omer Pitou Ntumba Katuala");
        model.addAttribute("data", data);

        return "index"; // View Name
    }
}
```

- Mettre à jour le fichier index.html

```

<body>
    <h1 th:text="#{welcome(${user})}"></h1>

    <table>
        <thead>
            <tr>
                <th th:text="#{firstName}">firstName</th>
                <th th:text="#{lastName}">lastName</th>
                <th th:text="#{gender}">gender</th>
                <th th:text="#{maritalStatus}">maritalStatus</th>
                <th th:text="#{placeOfBirth}">placeOfBirth</th>
                <th th:text="#{dateOfBirth}">dateOfBirth</th>
                <th th:text="#{numberOfChildren}">numberOfChildren</th>
                <th th:text="#{address}">numberOfChildren</th>
            </tr>
        </thead>
        <tbody th:if="${data != null && !data.isEmpty()}">
            <tr th:each="item: ${data}">
                <td th:text="${item.prenom}"></td>
                <td th:text="${item.nom}"></td>
                <td th:text="${item.genre}"></td>
                <td th:text="${item.etatCivil}"></td>
                <td th:text="${item.lieuNais}"></td>
                <td th:text="${item.dateNais}"></td>
                <td th:text="${item.nbreEnfants}"></td>
                <td th:text="${item.adresse}"></td>
            </tr>
        </tbody>
        <tbody th:if="${data == null || data.isEmpty()}">
            <tr>
                <td colspan="8" th:text="#{error.noDataFound}"></td>
            </tr>
        </tbody>
    </table>
</body>
</html>

```

---

#### 4.5.7. Améliorer la présentation du fichier index.html

- Télécharger le fichier bootstrap.min.css du site <https://getbootstrap.com>
- Copier ce fichier dans le dossier /static/css
- Ajouter un lien pour assurer le basculement Français/Anglais

```

<p>
    <a th:href="@{?locale=en}" th:text="#{english}"></a> | <a
    th:href="@{?locale=fr}" th:text="#{french}"></a>
</p>

```

- Ajouter les propriétés suivantes dans les 2 fichiers Bundle.properties :

  - o english=Anglais
  - o french=Français

- Mettre à jour le fichier index.html

```

<link th:href="@{/css/bootstrap.min.css}" rel="stylesheet"
      type="text/css" />

</head>
<body>
    <div class="container">
        <div class="row float-right">
            <p>
                <a th:href="@{?locale=en}" th:text="#{english}"></a> | <a
                    th:href="@{?locale=fr}" th:text="#{french}"></a>
            </p>
        </div>
        <div class="row">
            <h1 th:text="#{welcome(${user})}"></h1>
        </div>
        <div class="row table-responsive">
            <table class="table">
                <thead>
                    <tr>
                        <th scope="col">#</th>
                        <th scope="col" th:text="#{firstName}">firstName</th>
                        <th scope="col" th:text="#{lastName}">lastName</th>
                        <th scope="col" th:text="#{gender}">gender</th>
                        <th scope="col" th:text="#{maritalStatus}">maritalStatus</th>
                        <th scope="col" th:text="#{placeOfBirth}">placeOfBirth</th>
                        <th scope="col" th:text="#{dateOfBirth}">dateOfBirth</th>
                        <th scope="col" th:text="#{numberOfChildren}">numberOfChildren</th>
                        <th scope="col" th:text="#{address}">address</th>
                    </tr>
                </thead>
                <tbody th:if="${data != null && !data.isEmpty()}">
                    <tr th:each="item, iter: ${data}"
                        th:class="${item.genre == 'F' ? 'text-success' : 'text-primary'}">
                        <td scope="row" th:text="${iter.index + 1}"></td>
                        <td th:text="${item.prenom}"></td>
                        <td th:text="${item.nom}"></td>
                        <td th:text="${item.genre}"></td>
                        <td th:text="${item.etatCivil}"></td>
                        <td th:text="${item.lieuNais}"></td>
                        <td th:text="${temporals.format(item.dateNais, 'dd-MM-yyyy')}"></td>
                        <td th:text="${item.nbreEnfants}"></td>
                        <td th:text="${item.adresse}"></td>
                    </tr>
                </tbody>
                <tbody th:if="${data == null || data.isEmpty()}">
                    <tr>
                        <td colspan="9" th:text="#{error.noDataFound}"></td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>

```

#### 4.5.8. Changer Configurations dans le fichier application.properties

- Changer de port de server : server.port=8081
- Taper sur le navigateur : http://localhost:8081

#### 4.5.9. Implémenter la couche « Business Logic » : @Service

- Créer une interface EmployeService et Inclure la méthode getRecords()

```

public interface EmployeService {
    List<Employe> getRecords();
}

```

- Créer une classe @Service EmployeServiceImpl qui implémente cette interface

```
@Service
public class EmployeServiceImpl implements EmployeService {

    @Override
    public List<Employe> getRecords() {

        final List<Employe> data = new ArrayList<>();
        data.add(new Employe("Jacques", "Kabeya", "M", "M", "Kinshasa", LocalDate.of(1975, 3, 1), 2, "Ngaliema"));
        data.add(new Employe("Jean", "Kanku", "M", "M", "Kinshasa", LocalDate.of(1974, 3, 1), 5, "Gombe"));
        data.add(new Employe("Patrick", "Konde", "M", "M", "Kinshasa", LocalDate.of(1970, 3, 1), 5, "Gombe"));
        data.add(new Employe("Rose", "Mande", "F", "M", "Kinshasa", LocalDate.of(1970, 3, 1), 5, "Gombe"));

        return data;
    }
}
```

- Injecter ce @Service dans le @Controller

```
@Controller
public class WelcomeController {

    @Autowired
    private EmployeService service;

    @RequestMapping("/")
    public String getHome(HttpServletRequest request, Model model) { // Handler Method
        // Data
        final List<Employe> data = service.getRecords();

        // Assign
        model.addAttribute("user", "Omer Pitou Ntumba Katuala");
        model.addAttribute("data", data);

        return "index"; // View Name
    }
}
```

#### 4.5.10. Implémenter la couche « Data » : @Repository

- Créer une base de données MySQL : demo\_upc\_app01

```
CREATE SCHEMA `demo_upc_app01` DEFAULT CHARACTER SET utf8 ;
```

- Créer une table employes

```
CREATE TABLE `employes` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `prenom` varchar(45) NOT NULL,
    `nom` varchar(45) NOT NULL,
    `genre` char(1) NOT NULL,
    `etatCivil` char(1) NOT NULL,
    `lieuNais` varchar(45) NOT NULL,
    `dateNais` date NOT NULL,
    `nbreEnfants` int(11) NOT NULL DEFAULT '0',
    `adresse` text,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

- Ajouter des enregistrements à la table

```
INSERT INTO `demo_upc_app01`.`employes`  
(`prenom`, `nom`, `genre`, `etatCivil`, `lieuNais`, `dateNais`, `nbreEnfants`, `adresse`)  
VALUES  
('Jacques', 'Kabeya', 'M', 'M', 'Kinshasa', '19750301', '2', 'Kinshasa/Ngaliema'),  
('John', 'Kabongo', 'M', 'M', 'Kinshasa', '19700501', '1', 'Kinshasa/Gombe'),  
('Marc', 'Nsele', 'M', 'M', 'Kinshasa', '19750301', '2', 'Kinshasa/Kalamu'),  
('Sylvie', 'Muntu', 'F', 'C', 'Kisangani', '19770715', '0', 'Kinshasa/Limete'),  
('Jacky', 'Ngalula', 'F', 'M', 'Matadi', '19791212', '2', 'Kinshasa/Ndjili');
```

- Ajouter ces dépendances dans le POM : Spring Data JPA, Validation, Driver MySQL
- ```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>  
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-validation</artifactId>  
</dependency>  
<dependency>  
    <groupId>com.mysql</groupId>  
    <artifactId>mysql-connector-j</artifactId>  
</dependency>
```
- Annoter la classe Employe comme une Entité JPA.  
Notes :
  - Un champ « id » est ajouté pour servir d'identifiant unique, et rencontrer l'exigence JPA en la matière
  - Le champ dateNais devient de type java.sql.Date au lieu de java.time.LocalDate, comme l'entité sera désormais mappée avec une table de base de données)

```
@Entity
@Table(name = "employes")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class Employe implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @NotEmpty
    @Size(min = 1, max = 45)
    private String prenom;

    @NotEmpty
    @Size(min = 1, max = 45)
    private String nom;

    @NotEmpty
    @Size(min = 1, max = 1)
    private String genre;

    @NotEmpty
    @Size(min = 1, max = 1)
    private String etatCivil;

    @NotEmpty
    @Size(min = 1, max = 45)
    private String lieuNais;

    @NotNull
    private Date dateNais;

    @NotNull
    @Min(0)
    private int nbreEnfants;

    private String adresse;

}
```

- Créer une interface EmployeRepo qui héritera de l'interface JpaRepository dans le package Repositories

```
@Repository
public interface EmployeRepo extends JpaRepository<Employe, Integer> {
}
```

- Injecter cette @Repository dans le @Service

```
@Service
public class EmployeServiceImpl implements EmployeService {

    @Autowired
    private EmployeRepo repo;

    @Override
    public List<Employe> getRecords() {
        return repo.findAll();
    }
}
```

- Exécuter le projet et faire face à l'erreur due à la non-configuration de la base de données

```
*****
APPLICATION FAILED TO START
*****

Description:
Failed to configure a DataSource: 'url' attribute is not specified and no embedded datasource could be configured.
Reason: Failed to determine a suitable driver class

Action:
Consider the following:
If you want an embedded database (H2, HSQL or Derby), please put it on the classpath.
If you have database settings to be loaded from a particular profile you may need to activate it (no profiles are currently active)
```

- Configurer la base de données dans le fichier application.properties

```
#JDBC Driver
spring.datasource.url=jdbc:mysql://localhost:3306/demo_upc_app01?useSSL=false
spring.datasource.username=root
spring.datasource.password=finger4

spring.jpa.hibernate.ddl-auto=none
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
spring.jpa.properties.hibernate.dialect=org.springframework.orm.hibernate.dialect.MySQLInnoDBDialect
```

- Modifier le fichier index.html :

- Inclure le « id » venant de la table de données
- Reformatter le champ dateNais, comme il s'agit maintenant d'un champ de type java.sql.Date, et non de type java.time.LocalDate

```
<tbody th:if="${data != null && !data.isEmpty()}">
    <tr th:each="item, iter: ${data}"
        th:class="${item.genre == 'F' ? 'text-success' : 'text-primary'}">

        <!-- <td scope="row" th:text="${iter.index + 1}"></td> -->
        <td scope="row" th:text="${item.id}"></td>
        <td th:text="${item.prenom}"></td>
        <td th:text="${item.nom}"></td>
        <td th:text="${item.genre}"></td>
        <td th:text="${item.etatCivil}"></td>
        <td th:text="${item.lieuNais}"></td>
        <td th:text="#dates.format(item.dateNais, 'dd-MM-yyyy')}></td>
        <td th:text="${item.nbreEnfants}"></td>
        <td th:text="${item.adresse}"></td>

    </tr>
</tbody>
```

#### 4.5.11. Utiliser l'interface JdbcTemplate dans la couche « Data »

- Cette pratique est très recommandée quand il s'agit des opérations complexes de base des données (Jointure de tables, Appel de Procédures stockées/Fonctions...)
- Créer une interface EmployeRepoCustom, et une classe EmployeRepoImpl qui l'implémente

```
public interface EmployeRepoCustom {

    List<Employe> getRecords();

}
```

```
@Repository
public class EmployeRepoImpl implements EmployeRepoCustom {

    @Autowired
    private JdbcTemplate templ;

    @Override
    public List<Employe> getRecords() {
        final String sql = "SELECT * FROM employes ORDER BY id";

        return templ.query(sql, new RowMapper<Employe>() {
            public Employe mapRow(ResultSet rs, int rowNum) throws SQLException {
                Employe row = new Employe ();
                row.setId(rs.getInt("id"));
                row.setPrenom(rs.getString("prenom"));
                row.setNom(rs.getString("nom"));
                row.setGenre(rs.getString("genre"));
                row.setEtatCivil(rs.getString("etatCivil"));
                row.setLieuNais(rs.getString("lieuNais"));
                row.setDateNais(rs.getDate("dateNais"));
                row.setNbreEnfants(rs.getInt("nbreEnfants"));
                row.setAdresse(rs.getString("adresse"));
                return row;
            }
        });
    }
}
```

#### 4.5.12. Injecter la nouvelle @Repository dans le @Service

```
@Service
public class EmployeServiceImpl implements EmployeService {

    @Autowired
    private EmployeRepo repo;

    @Autowired
    private EmployeRepoCustom repoC;

    @Override
    public List<Employe> getRecords() {
        //return repo.findAll();
        return repoC.getRecords();
    }
}
```

#### 4.5.13. Utiliser l'interface JdbcTemplate avec la librairie externe « simpleflatmapper »

- Référence Web : <https://simpleflatmapper.org>
- Accéder au site, copier la référence de la dépendance et l'insérer dans le POM

```
<dependency>
    <groupId>org.simpleflatmapper</groupId>
    <artifactId>sfm-springjdbc</artifactId>
    <version>8.2.3</version>
</dependency>
```

- Ajouter la méthode getRecordsNew() dans l'interface EmployeRepoCustom.

```
public interface EmployeRepoCustom {

    List<Employe> getRecords();

    List<Employe> getRecordsNew();

}
```

- Dans la classe @Repository EmployeRepoImpl, Créer une instance de l'extracteur de données ResultSetExtractorImpl, et Mettre à jour la méthode implémentée getRecordsNew()

```
@Repository
public class EmployeRepoImpl implements EmployeRepoCustom {

    @Autowired
    private JdbcTemplate templ;

    private static final ResultSetExtractor<Employe> rsExtr = JdbcTemplateMapperFactory.newInstance()
        .addKeys("id")
        .ignorePropertyNotFound()
        .newResultSetExtractor(Employe.class);

    @Override
    public List<Employe> getRecords() {
        final String sql = "SELECT * FROM employes ORDER BY id";

        return templ.query(sql, new RowMapper<Employe>() {
            public Employe mapRow(ResultSet rs, int rowNum) throws SQLException {
                Employe row = new Employe();
                row.setId(rs.getInt("id"));
                row.setPrenom(rs.getString("prenom"));
                row.setNom(rs.getString("nom"));
                row.setGenre(rs.getString("genre"));
                row.setEtatCivil(rs.getString("etatCivil"));
                row.setLieuNais(rs.getString("lieuNais"));
                row.setDateNais(rs.getDate("dateNais"));
                row.setNbreEnfants(rs.getInt("nbreEnfants"));
                row.setAdresse(rs.getString("adresse"));
                return row;
            }
        });
    }

    @Override
    public List<Employe> getRecordsNew() {
        final String sql = "SELECT * FROM employes ORDER BY id";

        final Object[] params = new Object[] {};
        return templ.query(sql, rsExtr, params);
    }
}
```

4.5.14. Modifier le @Service en incluant la méthode getRecordsNew()

```
@Service
public class EmployeServiceImpl implements EmployeService {

    @Autowired
    private EmployeRepo repo;

    @Autowired
    private EmployeRepoCustom repoC;

    @Override
    public List<Employe> getRecords() {
        //return repo.findAll();
        //return repoC.getRecords();
        return repoC.getRecordsNew();
    }

}
```

4.5.15. Inclure une barre de recherche

- Ajouter un fichier app.css dans le dossier static/css

```
.search {
    position: relative;
}

.search input {
    text-indent: 25px;
    border: 2px solid #d6d4d4;
}

.search input:focus {
    box-shadow: none;
    border: 1px solid grey;
}

.search .fa-search {
    position: absolute;
    top: 12px;
    left: 10px;
}

.search button {
    position: absolute;
    top: 0px;
    right: 0px;
    width: 125px;
}
```

- Modifier le fichier index.html : Inclure <link.../>, la barre de recherche

```

<link rel="stylesheet" type="text/css"
      th:href="@{/css/bootstrap.min.css}" />
<link rel="stylesheet" type="text/css" th:href="@{/css/app.css}" />

</head>
<body>
    <div class="container">
        <div class="row float-right">
            <p>
                <a th:href="@{?locale=en}" th:text="#{english}"></a> | <a
                th:href="@{?locale=fr}" th:text="#{french}"></a>
            </p>
        </div>
        <div class="row">
            <h1 th:text="#{welcome(${user})}"></h1>
        </div>

        <div class="my-3">
            <form th:action="@{/}">
                <div class="row d-flex">
                    <div class="col-md-6 mt-2">
                        <div class="search">
                            <i class="fa fa-search"></i> <input id="keyword" type="search"
                                name="keyword" th:value="${keyword}"
                                class="form-control" th:placeholder="#{keyword}">
                            <button type="submit" class="btn btn-primary" th:text="#{search}"></button>
                        </div>
                    </div>
                </div>
            </form>
        </div>
    </div>

```

- Ajouter les propriétés suivantes dans les 2 fichiers Bundle.properties :
  - keyword=Mot Clé
  - search=Rechercher
- Inclure le paramètre « keyword » dans la couche @Repository/Méthode getRecordsNew()

```

public interface EmployeRepoCustom {

    List<Employe> getRecords();

    List<Employe> getRecordsNew(String keyword);

}

```

```

@Repository
public class EmployeRepoImpl implements EmployeRepoCustom {

    @Autowired
    private JdbcTemplate templ;

    private static final ResultSetExtractor<Employe> rsExtr = JdbcTemplateMapperFactory.newInstance()
        .addKeys("id")
        .ignorePropertyNotFound()
        .newResultSetExtractor(Employe.class);

    /**
     * @Override
     * public List<Employe> getRecords() {
     *     final String sql = "SELECT * FROM employes ORDER BY id";
     *
     *     return templ.query(sql, new RowMapper<Employe>() {
     *         public Employe mapRow(ResultSet rs, int rowNum) throws SQLException {
     *             Employe row = new Employe();
     *             row.setId(rs.getInt("id"));
     *             row.setPrenom(rs.getString("prenom"));
     *             row.setNom(rs.getString("nom"));
     *             row.setGenre(rs.getString("genre"));
     *             row.setEtatCivil(rs.getString("etatCivil"));
     *             row.setLieuNais(rs.getString("lieuNais"));
     *             row.setDateNais(rs.getDate("dateNais"));
     *             row.setNbreEnfants(rs.getInt("nbreEnfants"));
     *             row.setAdresse(rs.getString("adresse"));
     *             return row;
     *         }
     *     });
     *
     *     @Override
     *     public List<Employe> getRecordsNew(String keyword) {
     *         final String sql = "SELECT * FROM employes "
     *             + "WHERE id LIKE ? OR prenom LIKE ? OR nom LIKE ? "
     *             + "ORDER BY id";
     *
     *         final String _keyword = StringUtils.isEmpty(keyword) ? "%" : "%".concat(keyword).concat("%");
     *         final Object[] params = new Object[] { _keyword, _keyword, _keyword };
     *
     *         return templ.query(sql, rsExtr, params);
     *     }
     * }

```

- Inclure le paramètre « keyword » dans la couche @Service/Méthode getRecords()

```

public interface EmployeService {

    List<Employe> getRecords(String keyword);
}

```

```
@Service
public class EmployeServiceImpl implements EmployeService {

    @Autowired
    private EmployeRepo repo;

    @Autowired
    private EmployeRepoCustom repoC;

    @Override
    public List<Employe> getRecords(String keyword) {
        //return repo.findAll();
        //return repoC.getRecords();
        return repoC.getRecordsNew(keyword);
    }

}
```

- Injecter le paramètre @RequestParam « keyword » dans la Handler Method « / » du @Controller, et inclure aussi dans la méthode getRecords() du @Service injecté
- ```
@Controller
public class WelcomeController {

    @Autowired
    private EmployeService service;

    @RequestMapping("/")
    public String getHome(@RequestParam(value = "keyword", required = false, defaultValue = "") String keyword,
                          HttpServletRequest request, Model model) { // Handler Method
        // Data
        final List<Employe> data = service.getRecords(keyword);

        // Assign
        model.addAttribute("user", "Omer Pitou Ntumba Katuala");
        model.addAttribute("data", data);
        model.addAttribute("keyword", keyword);

        return "index"; // View Name
    }
}
```

#### 4.5.16. Inclure les fonctionnalités Ajouter/Éditer/Supprimer Enregistrement dans le « Backend »

- Créer un package « edu.upc.utils.beans » et Créer une classe POJO (Plain Old Java Object) dans ce package, qui permettra de mapper les données du formulaire .html, la bonne pratique exigeant que les structures de classes @Entity ne soient pas exposées au niveau « Frontend ».

```
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Getter
@Setter
public class FormEmployeBean implements Serializable {

    private static final long serialVersionUID = 1L;

    @NotNull
    private int id;

    @NotEmpty(message = "{error.firstName}")
    @Size(min = 1, max = 45)
    private String prenom;

    @NotEmpty(message = "{error.name}")
    @Size(min = 1, max = 45)
    private String nom;

    @NotEmpty(message = "{error.gender}")
    @Size(min = 1, max = 1)
    private String genre;

    @NotEmpty(message = "{error.maritalStatus}")
    @Size(min = 1, max = 1)
    private String etatCivil;

    @NotEmpty(message = "{error.placeOfBirth}")
    @Size(min = 1, max = 45)
    private String lieuNais;

    @NotNull(message = "{error.dateOfBirth}")
    private String dateNais;

    @NotNull
    @Min(0)
    private int nbreEnfants;

    private String adresse;

}
```

- L'attribut « message » ci-dessus fait référence au message d'erreur compris dans le fichier Bundle, qui sera renvoyé en cas d'erreur de validation.
- Mettre à jour le fichier Bundle\_fr

```
format.date=dd/MM/yyyy
format.date1=dd/mm/yyyy

english=Anglais
french=Francais
welcome=Bienvenue {0}
keyword=Mot Clé
search=Rechercher
new=Nouveau

firstName=Prenom
lastName=Nom
gender=Genre
maritalStatus=Etat-Civil
placeOfBirth=Lieu de Naissance
dateOfBirth=Date de Naissance
numberofChildren=Nbre Enfants
address=Adresse

error.noDataFound=Aucune donnée trouvée

select=Sélectionner
actions=Actions
edit=Editer
delete=Supprimer
save=Enregistrer
cancel=Annuler
error=Erreur
no=Non
yes=Oui
question=Question

error.firstName=Prenom invalide
error.name=Nom invalide
error.gender=Genre invalide
error.maritalStatus=Etat civil invalide
error.placeOfBirth=Lieu de naissance invalide
error.dateOfBirth=Date de naissance invalide

message.taskSuccessfullyCompleted=Tache exécutée avec succès

question.save=Voulez-vous valider cet enregistrement?
question.delete=Voulez-vous supprimer cet enregistrement?
```

- Mettre à jour le fichier Bundle

```
format.date=yyyy/MM/dd
format.date1=yyyy/mm/dd

english=English
french=French
welcome>Welcome {0}
keyword=Keyword
search=Search
new>New

firstName=First Name
lastName=Last Name
gender=Gender
maritalStatus=Marital Status
placeOfBirth=Place Of Birth
dateOfBirth=Date Of Birth
numberOfChildren=#Children
address=Address

error.noDataFound=No Data Found

select>Select
actions=Actions
edit>Edit
delete>Delete
save=Save
cancel=Cancel
error=Error
no>No
yes=Yes
question=Question

error.firstName=Invalid first name
error.name=Invalid name
error.gender=Invalid gender
error.maritalStatus=Invalid marital status
error.placeOfBirth=Invalid place of birth
error.dateOfBirth=Invalid date of birth

message.taskSuccessfullyCompleted=Task successfully completed

question.save=Would you like to save this record?
question.delete=Would you like to delete this record?
```

- Ajouter l'annotation @Builder à la classe « Employe » en vue de faciliter le transfert de données avec la classe « FormEmployeBean »

```
@Entity
@Table(name = "employes")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class Employe implements Serializable {

    private static final long serialVersionUID = 1L;
```

- Inclure la dépendance org.apache.commons dans le fichier POM, pour être en mesure d'utiliser la librairie StringUtils.

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
</dependency>
```

- Créer un @Component MyDateUtils dans le package « edu.upc.utils » pour convertir le type « String » au type « Date » et vice-versa.

```
@Component
public class MyDateUtils {

    public java.sql.Date getSqlDate(String date, String format) {
        java.sql.Date rdate = null;
        try {
            if (StringUtils.isNotEmpty(date) && StringUtils.isNotEmpty(format)) {
                SimpleDateFormat sdf = new SimpleDateFormat(format);
                java.util.Date xdate = sdf.parse(date);

                rdate = new java.sql.Date(xdate.getTime());
            }
        } catch (ParseException e) {
            e.printStackTrace();
        }
        return rdate;
    }

    public String getStrDate(java.sql.Date date, String format) {
        return date != null && StringUtils.isNotEmpty(format) ? new SimpleDateFormat(format).format(date) : null;
    }
}
```

- Créer un @ControllerAdvice dans le package « edu.upc.utils », pour disponibiliser certaines données au lancement de l'application, lesquelles données seront utilisées dans le formulaire .html

```
@ControllerAdvice
public class MyControllerAdvice {

    @ModelAttribute("dataGender")
    public List<String> getDataGender(HttpServletRequest request) {
        final List<String> data = new ArrayList<>();
        data.add("M");
        data.add("F");
        return data;
    }

    @ModelAttribute("dataMaritalStatus")
    public List<String> getDataMaritalStatus(HttpServletRequest request) {
        final List<String> data = new ArrayList<>();
        data.add("C");
        data.add("M");
        data.add("D");
        data.add("V");
        return data;
    }
}
```

- Créer un package « edu.upc.utils.mappers » et Créer un @Component « EmployeMapper » devant permettre de transférer les données de « FormEmployeBean » à « Employe » et vice-versa.

```

@Component
public class EmployeMapper {

    @Autowired
    private MyDateUtils utils;

    @Autowired
    private MessageSource messageSource;

    @Autowired
    private LocaleResolver resolver;

    @Autowired
    private HttpServletRequest request;

    public FormEmployeBean mapToFormBean(Employe entity) {
        return FormEmployeBean.builder()
            .id(entity.getId())
            .prenom(entity.getPrenom())
            .nom(entity.getNom())
            .genre(entity.getGenre())
            .etatCivil(entity.getEtatCivil())
            .lieuNaissance(entity.getLieuNaissance())
            .dateNaissance(utils.getDateNaissance(),
                messageSource.getMessage("format.date", null, resolver.resolveLocale(request)))
            .nbreEnfants(entity.getNbreEnfants())
            .adresse(entity.getAdresse()).build();
    }

    public Employe mapToEntity(FormEmployeBean bean) {
        return Employe.builder()
            .id(bean.getId())
            .prenom(bean.getPrenom())
            .nom(bean.getNom())
            .genre(bean.getGenre())
            .etatCivil(bean.getEtatCivil())
            .lieuNaissance(bean.getLieuNaissance())
            .dateNaissance(utils.getDateNaissance(),
                messageSource.getMessage("format.date", null, resolver.resolveLocale(request)))
            .nbreEnfants(bean.getNbreEnfants())
            .adresse(bean.getAdresse()).build();
    }
}

```

- Aucune modification au niveau de la couche @Repository vu que les méthodes de l'interface EmployeRepo qui hérite de JpaRepository serviront à cette fin.

```

@Repository
public interface EmployeRepo extends JpaRepository<Employe, Integer> {
}

```

- Faire les modifications suivantes au niveau de la couche @Service

```

public interface EmployeService {

    Employe saveRecord(FormEmployeBean bean, Locale locale);

    List<Employe> deleteRecord(int id, Locale locale);

    Employe getRecord(int id);

    List<Employe> getRecords(String keyword);

}

```

```
@Service
public class EmployeServiceImpl implements EmployeService {

    @Autowired
    private EmployeRepo repo;

    @Autowired
    private EmployeRepoCustom repoc;

    @Autowired
    private EmployeMapper mapper;

    @Autowired
    private MessageSource messageSource;

    @Transactional(rollbackFor=Exception.class)
    @Override
    public Employe saveRecord(FormEmployeBean bean, Locale locale) {
        if (StringUtils.isEmpty(bean.getDateNaissance())) {
            throw new RuntimeException(messageSource.getMessage("error.dateOfBirth", null, locale));
        }

        return repo.saveAndFlush(mapper.mapToEntity(bean));
    }

    @Transactional(rollbackFor=Exception.class)
    @Override
    public List<Employe> deleteRecord(int id, Locale locale) {
        repo.deleteById(id);
        return this.getRecords(null);
    }

    @Transactional(readOnly=true)
    @Override
    public Employe getRecord(int id) {
        Optional<Employe> row = repo.findById(id);
        return row.isPresent() ? row.get() : null;
    }

    @Transactional(readOnly=true)
    @Override
    public List<Employe> getRecords(String keyword) {
        //return repo.findAll();
        //return repoc.getRecords();
        return repoc.getRecordsNew(keyword);
    }
}
```

- Faire les modifications suivantes au niveau de la couche @Controller : « Handlers Mappings » « doCreate » pour renvoyer un formulaire vide, et « doEdit » pour éditer un enregistrement existant.

```
@Controller
@RequestMapping({ "/", "/employes" })
public class EmployeController {

    @Autowired
    private EmployeService service;

    @Autowired
    private EmployeMapper mapper;

    @Autowired
    private MessageSource messageSource;

    @Autowired
    private LocaleResolver resolver;

    // Helper
    private String getFormTitle(FormEmployeBean entity, HttpServletRequest request) {
        final Locale locale = resolver.resolveLocale(request);
        final int id = entity != null ? entity.getId() : 0;

        return id != 0 ? messageSource.getMessage("edit", null, locale).concat("# " + id)
                      : messageSource.getMessage("new", null, locale);
    }

    // Form
    @GetMapping("/create")
    public String doCreate(Model model, HttpServletRequest request) {
        model.addAttribute("title", this.getFormTitle(new FormEmployeBean(), request));
        model.addAttribute("entryBean", new FormEmployeBean());

        return "employe-form";
    }

    @GetMapping("/edit/{id}")
    public String doEdit(@PathVariable("id") int id, Model model, HttpServletRequest request) {
        final Employe row = service.getRecord(id);
        final FormEmployeBean bean = mapper.mapToFormBean(row);

        model.addAttribute("title", this.getFormTitle(bean, request));
        model.addAttribute("entryBean", bean);

        return "employe-form";
    }
}
```

- Poursuivre avec les modifications au niveau de la couche @Controller : « Handlers Mappings » « doSave » pour valider l'enregistrement, et « doDelete » pour supprimer un enregistrement.

```

// Tasks
@PostMapping("/save")
public String doSave(@Valid @ModelAttribute("entryBean") FormEmployeBean bean, BindingResult result, Model model,
    RedirectAttributes redAttr, HttpServletRequest request) {
    final Locale locale = resolver.resolveLocale(request);

    // Test
    if (result.hasErrors()) {
        model.addAttribute("title", this.getFormTitle(bean, request));
        model.addAttribute("entryBean", bean);
        return "employe-form";
    }

    // Process
    try {
        long id = bean != null ? bean.getId() : 0;
        service.saveRecord(bean, locale);
        if (id == 0) { // New
            model.addAttribute("title", this.getFormTitle(new FormEmployeBean(), request));
            model.addAttribute("entryBean", new FormEmployeBean());
            model.addAttribute("message",
                messageSource.getMessage("message.taskSuccessfullyCompleted", null, locale));
            return "employe-form";
        } else {
            redAttr.addFlashAttribute("message",
                messageSource.getMessage("message.taskSuccessfullyCompleted", null, locale));
            return "redirect:/employees";
        }
    } catch (Exception ex) {
        model.addAttribute("title", this.getFormTitle(bean, request));
        result.reject(ex.getLocalizedMessage());
        return "employe-form";
    }
}

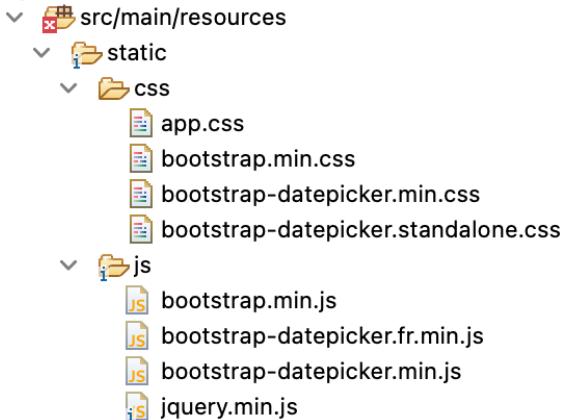
@RequestMapping("/delete/{id}")
public String doDelete(@PathVariable("id") int id, Model model, HttpServletRequest request) {
    final Locale locale = resolver.resolveLocale(request);

    service.deleteRecord(id, locale);
    return "redirect:/employees";
}

```

#### 4.5.17. Inclure les fonctionnalités Ajouter/Éditer/Supprimer Enregistrement dans le « Frontend »

- Ajouter les ressources manquantes



- Dans le fichier app.css, ajouter cette classe pour mettre en évidence les champs à erreur sur le formulaire .html.

```
.has-error {
    border-color: red;
}
```

- Modifier le fichier index.html, section <head></head>

```
<head>
<meta charset="UTF-8">
<title>Home</title>

<link rel="stylesheet" type="text/css" th:href="@{/css/app.css}" />
<link rel="stylesheet" type="text/css"
      th:href="@{/css/bootstrap.min.css}" />

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.min.css"
      integrity="sha512-xh60/CkQoPOWDdYTDqeRdPCVd1SpvCA9XXcUzZ5FmJNp1coAFzvtCN9BbamE+E4aHK8yyUHUSCcJHgXloTyT2A=="
      crossorigin="anonymous" referrerPolicy="no-referrer" />

</head>
```

- Modifier le fichier index.html, section <body></body> : Inclure le bouton « Nouveau » et une « div » pour les messages de confirmation.

```
<div class="my-3">
    <div class="row d-flex">
        <div class="col-md-6 mt-2">
            <a id="btnNew" class="btn btn-primary btn-sm"
               th:href="@{/employees/create}" th:text="#{new}"></a>
        </div>
        <div class="col-md-6 mt-2">
            <form th:action="@{/employees}">
                <div class="search">
                    <i class="fa fa-search"></i> <input id="keyword" type="search"
                       name="keyword" th:value="${keyword}"
                       class="form-control form-control-sm" th:placeholder="#{keyword}">
                    <button type="submit" class="btn btn-primary btn-sm"
                           th:text="#{search}"></button>
                </div>
            </form>
        </div>
    </div>
<div th:if="${message != null}">
    <div class="row d-flex col-md-4 alert-success alert-dismissible fade show text-center"
         role="alert">
        [[${message}]]
        <button type="button" class="close" data-dismiss="alert"
               aria-label="Close">
            <span aria-hidden="true">&times;</span>
        </button>
    </div>
</div>
```

- Modifier le fichier index.html, section <body></body> : Inclure une colonne Actions (Éditer/Supprimer) dans la table.

```

<div class="row table-responsive" th:with="formatDate=#{format.date}">
    <table class="table">
        <thead>
            <tr>
                <th scope="col">#</th>
                <th scope="col" th:text="#{firstName}">firstName</th>
                <th scope="col" th:text="#{lastName}">lastName</th>
                <th scope="col" th:text="#{gender}">gender</th>
                <th scope="col" th:text="#{maritalStatus}">maritalStatus</th>
                <th scope="col" th:text="#{placeOfBirth}">placeOfBirth</th>
                <th scope="col" th:text="#{dateOfBirth}">dateOfBirth</th>
                <th scope="col" th:text="#{numberOfChildren}">numberOfChildren</th>
                <th scope="col" th:text="#{address}">address</th>
                <th scope="col" th:text="#{actions}">actions</th>
            </tr>
        </thead>
        <tbody th:if="${data != null && !data.isEmpty()}">
            <tr th:each="item, iter: ${data}" th:class="${item.genre == 'F' ? 'text-success' : 'text-primary'}">
                <!-- <td scope="row" th:text="${iter.index + 1}"></td> -->
                <td scope="row" th:text="${item.id}"></td>
                <td th:text="${item.prenom}"></td>
                <td th:text="${item.nom}"></td>
                <td th:text="${item.genre}"></td>
                <td th:text="${item.etatCivil}"></td>
                <td th:text="${item.lieuNais}"></td>
                <td th:text="${#dates.format(item.dateNais, formatDate)}"></td>
                <td th:text="${item.nbreEnfants}"></td>
                <td th:text="${item.adresse}"></td>
                <td><a th:href="@{/employes/edit/{id}(id=${item.id})}" class="btn btn-secondary btn-sm"><i class="fa fa-pen-to-square"></i> </a> <a th:href="@{/employes/delete/{id}(id=${item.id})}" th:itemId="${item.id}" class="btn btn-danger btn-sm btn-delete"><i class="fa fa-trash" aria-hidden="true"></i> </a></td>
            </tr>
        </tbody>
        <tbody th:if="${data == null || data.isEmpty()}">
            <tr>
                <td colspan="9" th:text="#{error.noDataFound}"></td>
            </tr>
        </tbody>
    </table>
</div>

```

- Modifier le fichier index.html, section <body></body> : Inclure la « div » pour la boite de dialogue et les liens « Scripts ».

```

<!-- DIALOG -->
<div class="modal fade text-center" id="confirmModal">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" th:text="#{question}"></h5>
                <button type="button" class="close" data-dismiss="modal">
                    <span aria-hidden="true">&times;</span>
                </button>
            </div>

            <div class="modal-body">
                <span id="confirmText"></span>
            </div>

            <div class="modal-footer">
                <a type="button" id="yesBtn" class="btn btn-danger" th:text="#{yes}"></a>
                <button type="button" class="btn btn-secondary"
                        data-dismiss="modal" th:text="#{no}"></button>
            </div>
        </div>
    </div>
</div>

<!-- SCRIPTS -->
<script type="text/javascript" th:src="@{/js/jquery.min.js}"></script>
<script type="text/javascript" th:src="@{/js/bootstrap.min.js}"></script>

<script type="text/javascript" th:inline="javascript">
/*<! [CDATA[*/
$(document).ready(function() {
    $(".btn-delete").on("click", function (e) {
        e.preventDefault();

        var link = $(this);
        var itemId = link.attr("itemId");
        var text = /*[[#{question.delete}]]*/;

        $("#yesBtn").attr("href", link.attr("href"));
        $("#confirmText").html(text + " #" + itemId);
        $("#confirmModal").modal();
    });
});

/*]]>*/
</script>

```

- Créer le fichier employe-form.html : Section <head></head>

```

<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="viewport"
      content="width=device-width,initial-scale=1.0,minimum-scale=1.0" />

<title>Employee Form</title>

<link rel="stylesheet" type="text/css"
      th:href="@{/css/bootstrap.min.css}" />
<link rel="stylesheet" type="text/css"
      th:href="@{/css/bootstrap-datepicker.min.css}" />
<link rel="stylesheet"
      th:href="@{/css/bootstrap-datepicker.standalone.css}" />

<link rel="stylesheet"
      href="//cdnjs.cloudflare.com/ajax/libs/jquery-confirm/3.3.2/jquery-confirm.min.css">

<link rel="stylesheet" type="text/css" th:href="@{/css/app.css}" />

</head>

```

- Modifier le fichier employe-form.html : Section <body></body>

```

<body>
<div class="container">
    <div class="my-3 col-md-6">
        <form id="myform" th:action="@{/employees/save}" method="post"
              enctype="multipart/form-data" th:object="${entryBean}">

            <div class="p-3">
                <h2 class="text-left" th:text="${title}"></h2>
                <hr />

                <!-- ENTRY FIELDS -->
                <div class="row">
                    <div class="form-group col-sm">
                        <label for="prenom" th:text="#{firstName}"></label> <input
                            id="prenom" type="text" class="form-control form-control-sm"
                            th:field="*{prenom}" th:errorclass="has-error" />
                    </div>
                    <div class="form-group col-sm">
                        <label for="nom" th:text="#{lastName}"></label> <input id="nom"
                            type="text" class="form-control form-control-sm"
                            th:field="*{nom}" th:errorclass="has-error" />
                    </div>
                </div>
                <div class="row">
                    <div class="form-group col-sm">
                        <label for="genre" th:text="#{gender}"></label> <select
                            id="genre" class="form-control form-control-sm"
                            th:field="*{genre}" th:errorclass="has-error">
                            <option value="" th:text="#{select}"></option>
                            <option th:each="item : ${dataGender}" th:value="${item}"
                                   th:text="${item}"></option>
                        </select>
                    </div>
                    <div class="form-group col-sm">
                        <label for="etatCivil" th:text="#{maritalStatus}"></label> <select
                            id="etatCivil" class="form-control form-control-sm"
                            th:field="*{etatCivil}" th:errorclass="has-error">
                            <option value="" th:text="#{select}"></option>
                            <option th:each="item : ${dataMaritalStatus}" th:value="${item}"
                                   th:text="${item}"></option>
                        </select>
                    </div>
                </div>
            </div>
        </form>
    </div>
</div>

```

```

<div class="row">
    <div class="form-group col-sm">
        <label for="lieuNais" th:text="#{placeOfBirth}"></label> <input
            id="lieuNais" type="text" class="form-control form-control-sm"
            th:field="*{lieuNais}" th:errorclass="has-error" />
    </div>
    <div class="form-group col-sm">
        <label for="dateNais" th:text="#{dateOfBirth}"></label> <input
            id="dateNais" type="text" class="form-control form-control-sm"
            th:field="*{dateNais}" th:errorclass="has-error" />
    </div>
    <div class="form-group">
        <label for="nbreEnfants" th:text="#{numberOfChildren}"></label> <input
            id="nbreEnfants" type="number"
            class="form-control form-control-sm" th:field="*{nbreEnfants}"
            th:errorclass="has-error" />
    </div>
    <div class="form-group">
        <label for="adresse" th:text="#{address}"></label>
        <textarea id="adresse" class="form-control form-control-sm"
            data-parsley-trigger="keyup"
            data-parsley-validation-threshold="10" th:field="*{adresse}"
            th:errorclass="has-error"></textarea>
    </div>
    <hr />
    <div class="form-group">
        <button id="btnSave" type="submit" class="btn btn-primary btn-sm"
            th:text="#{save}">Save</button>
        <a id="btnCancel" type="button" class="btn btn-danger btn-sm"
            th:href="@{/employes}" th:text="#{cancel}">Cancel</a>
        <!--
        <button id="btnCancel" type="button" class="btn btn-primary btn-sm"
            th:text="#{save}">Save</button>
        -->
    </div>

    <!-- ERROR -->
    <div th:if="${#fields.hasErrors('*')}">
        <div class="alert alert-danger alert-dismissible fade show"
            role="alert">
            <strong th:text="#{error}">Error</strong>
            <button type="button" class="close" data-dismiss="alert"
                aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
            <ul th:if="${#fields.hasErrors('*')}">
                <li th:each="err : ${#fields.errors('*')}" th:text="${err}"></li>
            </ul>
        </div>
    </div>

    <!-- MESSAGE -->
    <div th:if="${message != null}">
        <div class="alert alert-success alert-dismissible fade show"
            role="alert">
            [[${message}]]
            <button type="button" class="close" data-dismiss="alert"
                aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
        </div>
    </div>

    <!-- HIDDEN FIELDS -->
    <input id="id" type="hidden" th:field="*{id}" />
</div>
<!-- <p th:if="${#fields.hasErrors('prenom')}" th:errorclass="has-error" th:errors="*{prenom}" /> -->
</form>
</div>
</div>

```

- Modifier le fichier employe-form.html : Section Scripts

```
<!-- SCRIPTS -->
<script type="text/javascript" th:src="@{/js/jquery.min.js}"></script>
<script type="text/javascript" th:src="@{/js/bootstrap.min.js}"></script>
<script type="text/javascript"
    th:src="@{/js/bootstrap-datepicker.min.js}"></script>
<script type="text/javascript"
    th:src="@{/js/bootstrap-datepicker.fr.min.js}"></script>
<script
    src="//cdnjs.cloudflare.com/ajax/libs/jquery-confirm/3.3.2/jquery-confirm.min.js"></script>

<script type="text/javascript" th:inline="javascript">
/*<![CDATA[*/
$(document).ready(function() {
    $('#dateNais').datepicker({
        format : /*[[#{format.date1}]]*/'dd/mm/yyyy',
        orientation : 'bottom right',
        todayHighlight : true,
        language : /*[$[#locale.language]]*/'fr'
    });

    $("#btnSave").confirm({
        title: /*[[#{question}]]*/,
        content: /*[[#{question.save}]]*/,
        type: 'blue',
        buttons: {
            confirm: {
                text: /*[[#{yes}]]*/,
                btnClass: 'btn-secondary',
                action: function(){
                    $('#myform').submit();
                }
            },
            cancel: {
                text: /*[[#{no}]]*/,
                btnClass: 'btn-danger',
                action: function(){
                }
            }
        }
    });
});

/*]]]&gt;*
&lt;/script&gt;</pre>
```

#### 4.5.18. Travail Pratique

Anglais | Français

## Bienvenue Omer Pitou Ntumba Katuala

Nouveau		Mot Clé							Rechercher	
#	Prenom	Nom	Genre	Etat-Civil	Lieu de Naissance	Date de Naissance	Nbre Enfants	Adresse	Actions	
1	Jacques	Kabeya	M	M	Kinshasa	05/10/1929	2	Kinshasa/Ngaliema		
2	John	Kabongo	M	M	Kinshasa	16/03/2023	1	Kinshasa/Gombe		
4	Sylvie	Muntu	F	C	Kisangani	08/07/1977	0	Kinshasa/Limete		
5	Jacky	Ngalula	F	M	Matadi	12/12/1979	2	Kinshasa/Ndjili		
11	ISAAC	NZAU	M	M	KINSHASA	16/03/2023	1	KINSHASA		

### Editer# 1

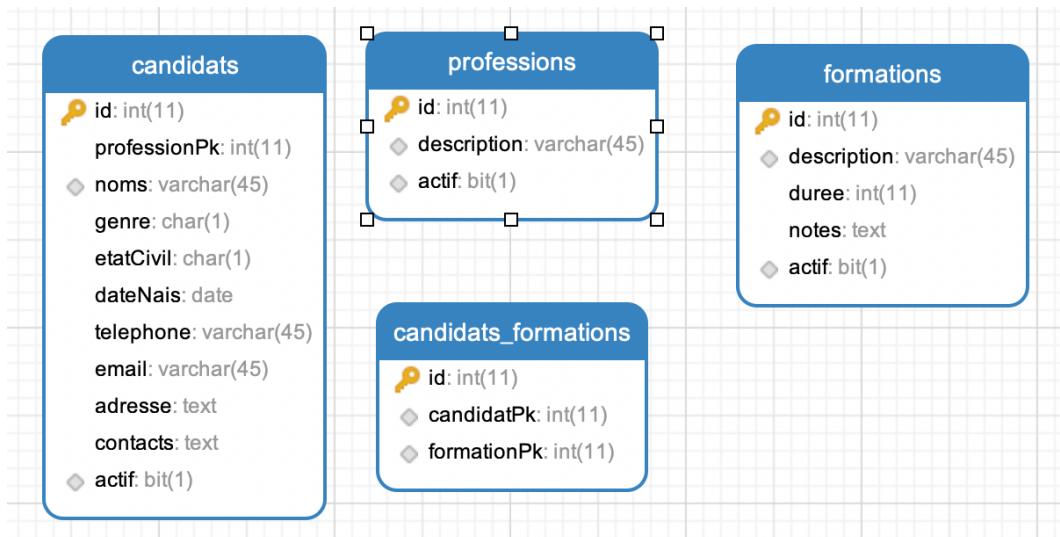
Prenom	Nom
<input type="text" value="Jacques"/>	<input type="text" value="Kabeya"/>
Genre	Etat-Civil
<input type="text" value="M"/>	<input type="text" value="M"/>
Lieu de Naissance	Date de Naissance
<input type="text" value="Kinshasa"/>	<input type="text" value="05/10/1929"/>
Nbre Enfants	
<input type="text" value="2"/>	
Adresse	
<input type="text" value="Kinshasa/Ngaliema"/>	
<input type="button" value="Enregistrer"/>	<input type="button" value="Annuler"/>

## 4.6. App-02 : Développer par étapes

### 4.6.1. Objectifs

- Développer par étapes
- Rencontrer le maximum d'exigences pour une application Spring en production
- Explorer les bonnes pratiques

### 4.6.2. Analyser le diagramme de la base données



### 4.6.3. Créer la base de données en MySQL

- Utiliser n'importe quel « Client » MySQL pour créer la base de données.
- Créer les différents index.

### 4.6.4. Créer un projet Spring

### 4.6.5. Créer les packages de base

### 4.6.6. Configurer l'internationalisation

- Créer deux fichiers .properties sous le dossier Resources (Bundle.properties, Bundle\_fr.properties), et définir des propriétés si possible:
  - Bundle.properties
  - Bundle\_fr.properties
- Configurer le @Bean MessageSource et @Bean LocaleResolver dans un fichier @Configuration

```

@Configuration
public class AppConfig {

    @Bean(name = "messageSource")
    public MessageSource messageSource() {
        ReloadableResourceBundleMessageSource resource = new ReloadableResourceBundleMessageSource();
        resource.setBasenames("classpath:Bundle");
        resource.setDefaultEncoding("ISO-8859-1"); // UTF-8
        resource.setAlwaysUseMessageFormat(true); // Handle Apostroph & others
        resource.setUseCodeAsDefaultMessage(true);
        return resource;
    }

    @Bean(name = "localeResolver")
    public LocaleResolver localeResolver() { // Identify which locale is used
        return new SessionLocaleResolver();
    }

}

```

- Créer une classe « WebConfig » qui va implémenter l'interface WebMvcConfigurer
- Redéfinir la méthode « addInterceptors() » pour ajouter un intercepteur du paramètre de changement de langue, dans le registre des intercepteurs

```

@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        LocaleChangeInterceptor localeIntcp = new LocaleChangeInterceptor();
        localeIntcp.setParamName("locale");
        registry.addInterceptor(localeIntcp);
    }

}

```

#### 4.6.7. Préparer l'environnement

- Créer deux autres fichiers « application.properties »
  - Fichier : application\_prod.properties
  - Fichier : application\_dev.properties
- Définir dans le fichier application.properties :
  - Le profiling de l'application
  - Les paramètres de connexion de la base de données
  - Le port du serveur

#### 4.6.8. Créer les modèles et leurs « DTOs »

#### 4.6.9. Implémenter la couche « Repository »

#### 4.6.10. Implémenter la couche « Service »

#### 4.6.11. Implémenter la couche « Controller »

#### 4.6.12. Préparer les « views »

#### 4.6.13. Exécuter le projet



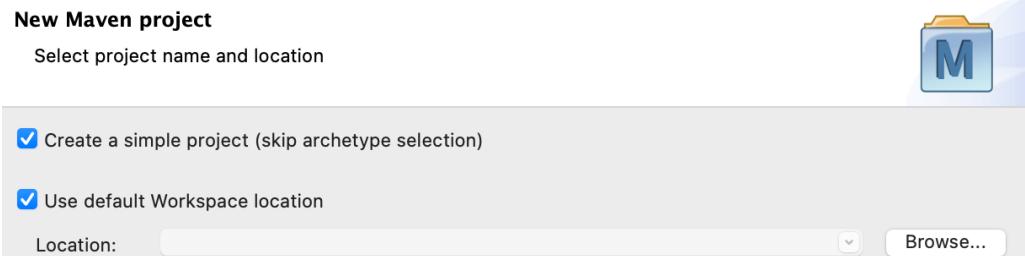
## 4.7. App-03 : Développer comme un Pro

### 4.7.1. Objectifs

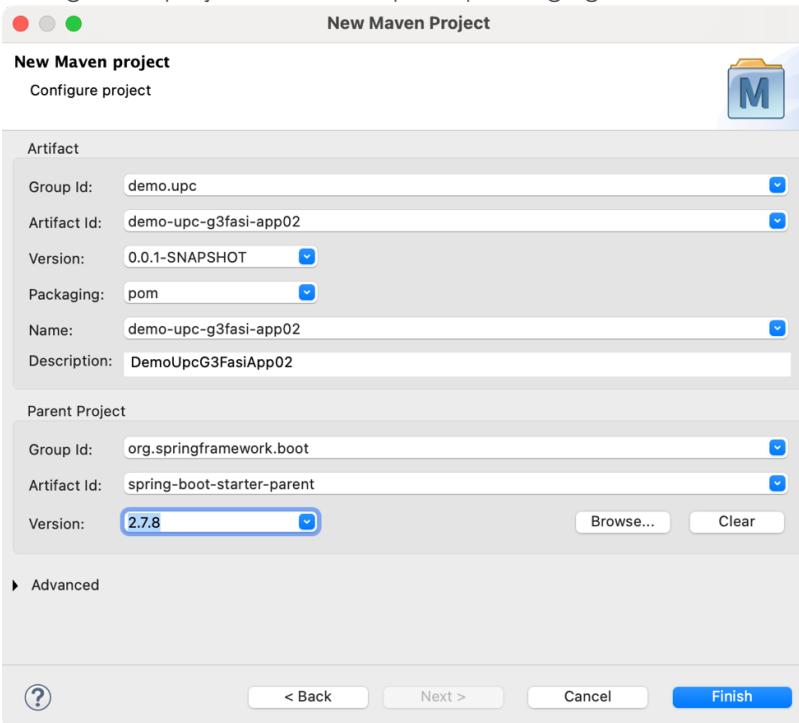
- Comprendre le développement par module
- Définir le « Thymeleaf Layout » partant d'un « template » existant
- Utiliser JdbcTemplate pour les requêtes complexes

### 4.7.2. Créer un Projet Maven/Projet Parent

- Lancer Spring Tools Suite (STS)
- Créer Nouveau Projet : Menu File/New/Project/Maven/Maven Project
- Cocher la case « Create a simple project (skip archetype selection) », et Cliquer sur Next



- Configurer le projet, se rassurer que le packaging soit POM, et Cliquer sur Finish



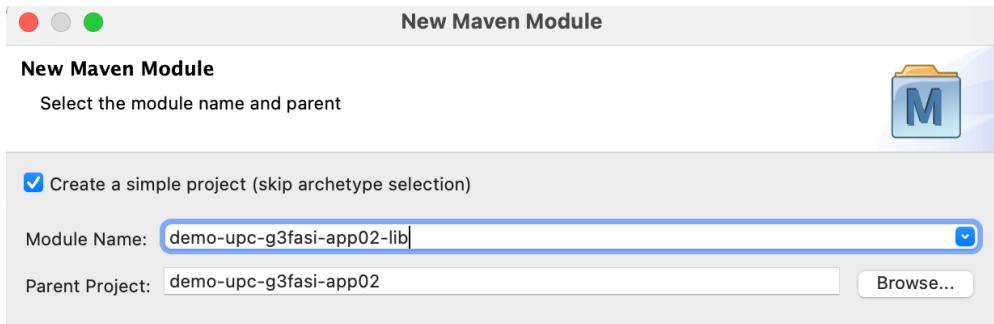
- Ouvrir le POM du projet, et ajouter la propriété <java.version> et les dépendances ci-après (Note : Ces dépendances sont ajoutées à ce niveau parce qu'elles seront utilisées dans tous les modules)

```
<properties>
    <java.version>1.8</java.version>
</properties>

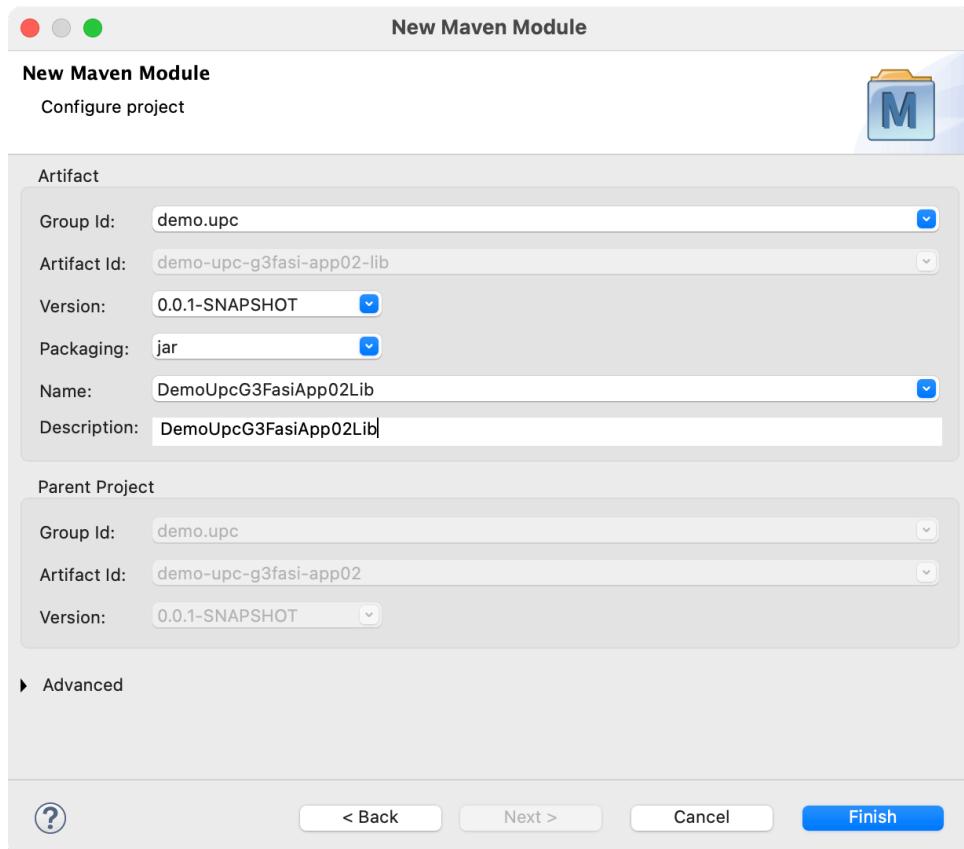
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
```

#### 4.7.3. Créer des Modules Maven

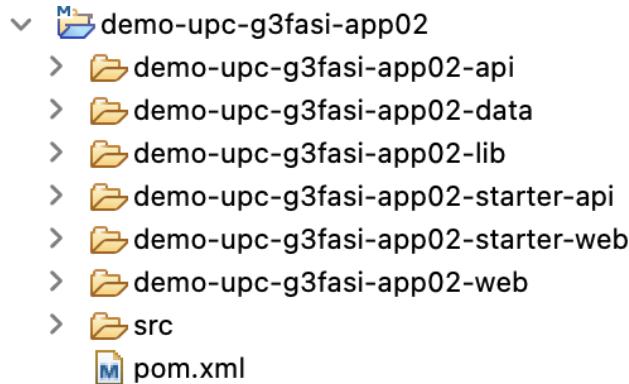
- Cliquer droit sur le Projet créé
- Cliquer sur New/Project/Maven/Maven Module
- Cocher la case « Create a simple project (skip archetype selection) », Cliquer sur Next



- Configurer le module « DemoUpcG3FasiApp02Lib », se rassurer que le packaging soit JAR, et Cliquer sur Finish



- Reprendre la même procédure pour les autres modules opérationnels suivants, et le packaging devra être JAR :
  - DemoUpcG3FasiApp02Data
  - DemoUpcG3FasiApp02Api
  - DemoUpcG3FasiApp02Web
- Procéder de la façon pour les autres modules de lancement suivants, et le packing devra être WAR (vu que le déploiement sera sur un serveur WEB):
  - DemoUpcG3FasiApp02StarterApi
  - DemoUpcG3FasiApp02StarterWeb
- Se rassurer que la structure du projet ressemble à ce qui suit :



- Se rassurer que le POM du projet ressemble à ce qui suit :

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.8</version>
  </parent>
  <groupId>demo.upc</groupId>
  <artifactId>demo-upc-g3fasi-app02</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>demo-upc-g3fasi-app02</name>
  <description>DemoUpcG3FasiApp02</description>

  <properties>
    <java.version>1.8</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
    </dependency>
  </dependencies>

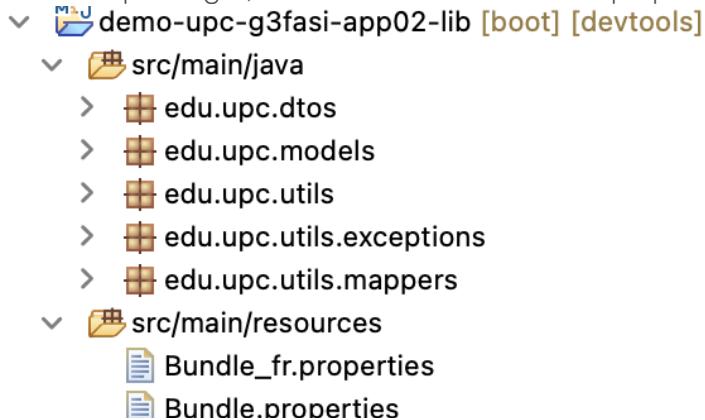
  <modules>
    <module>demo-upc-g3fasi-app02-lib</module>
    <module>demo-upc-g3fasi-app02-data</module>
    <module>demo-upc-g3fasi-app02-api</module>
    <module>demo-upc-g3fasi-app02-web</module>
    <module>demo-upc-g3fasi-app02-starter-api</module>
    <module>demo-upc-g3fasi-app02-starter-web</module>
  </modules>
</project>
  
```

#### 4.7.4. Configurer le Module Lib

- Ajouter ces dépendances dans le POM du module « demo-upc-g3fasi-app02-lib »

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
</dependencies>
```

- Créer ces packages, et inclure ces fichiers Bundle.properties



#### 4.7.5. Configurer le Module Data

- Ajouter ces dépendances dans le POM du module « demo-upc-g3fasi-app02-data »

```
<dependencies>
    <!-- CORE -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>com.mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
    </dependency>
    <dependency>
        <groupId>org.simpleflatmapper</groupId>
        <artifactId>sfm-springjdbc</artifactId>
        <version>8.2.3</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-configuration-processor</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <!-- APP -->
    <dependency>
        <groupId>demo.upc</groupId>
        <artifactId>demo-upc-g3fasi-app02-lib</artifactId>
        <version>${project.version}</version>
    </dependency>
</dependencies>
```

- Créer ces packages
  -  **demo-upc-g3fasi-app02-data [boot] [devtools]**
  -  **src/main/java**
    - ◻ **edu.upc.repositories**
    - ◻ **edu.upc.services**
- Ajouter le fichier « application.properties » (Copier du projet App01)
  -  **src/main/resources**
    - leaf **application.properties**

#### 4.7.6. Configurer le Module API

- Ajouter ces dépendances dans le POM du module « demo-upc-g3fasi-app02-api »

```
<dependencies>
    <!-- CORE -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>

    <!-- APP -->
    <dependency>
        <groupId>demo.upc</groupId>
        <artifactId>demo-upc-g3fasi-app02-data</artifactId>
        <version>${project.version}</version>
    </dependency>

</dependencies>
```

- Créer ces packages
  - ▽  demo-upc-g3fasi-app02-api [boot] [devtools]
    - ▽  src/main/java
      - edu.upc.config.api
      - edu.upc.controllers.api

#### 4.7.7. Configurer le Module WEB

- Ajouter ces dépendances dans le POM du module « demo-upc-g3fasi-app02-web »

```
<dependencies>
    <!-- CORE -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>

    <!-- APP -->
    <dependency>
        <groupId>demo.upc</groupId>
        <artifactId>demo-upc-g3fasi-app02-data</artifactId>
        <version>${project.version}</version>
    </dependency>

</dependencies>
```

- Créer ces packages



#### 4.7.8. Configurer le Module Starter API

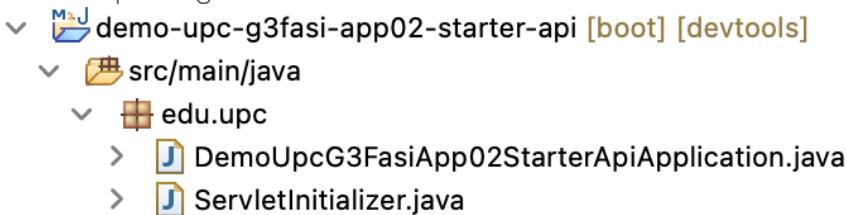
- Ajouter ces dépendances dans le POM du module « demo-upc-g3fasi-app02-starter-api »

```
<dependencies>
    <!-- CORE -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
    </dependency>

    <!-- APP -->
    <dependency>
        <groupId>demo.upc</groupId>
        <artifactId>demo-upc-g3fasi-app02-api</artifactId>
        <version>${project.version}</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
    <finalName>demo-upc-g3fasi-app02-api</finalName>
</build>
```

- Créer ce package



- Classe « DemoUpcG3FasiApp02StarterApiApplication.java »

```
@SpringBootApplication
public class DemoUpcG3FasiApp02StarterApiApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoUpcG3FasiApp02StarterApiApplication.class, args);
    }
}
```

- Classe « ServletInitializer.java »

```
public class ServletInitializer extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(DemoUpcG3FasiApp02StarterApiApplication.class);
    }

}
```

#### 4.7.9. Configurer le Module Starter WEB

- Ajouter ces dépendances dans le POM du module « demo-upc-g3fasi-app02-starter-web »

```
<dependencies>
    <!-- CORE -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
    </dependency>

    <!-- APP -->
    <dependency>
        <groupId>demo.upc</groupId>
        <artifactId>demo-upc-g3fasi-app02-web</artifactId>
        <version>${project.version}</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
    <finalName>demo-upc-g3fasi-app02-web</finalName>
</build>
```

- Créer ce package

```
demo-upc-g3fasi-app02-starter-web [boot] [devtools]
  src/main/java
    edu.upc
      DemoUpcG3FasiApp02StarterWebApplication.java
      ServletInitializer.java
```

- Classe « DemoUpcG3FasiApp02StarterWebApplication.java »

```
@SpringBootApplication
public class DemoUpcG3FasiApp02StarterWebApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoUpcG3FasiApp02StarterWebApplication.class, args);
    }
}
```

- Classe « ServletInitializer.java »

```
public class ServletInitializer extends SpringBootServletInitializer {  
  
    @Override  
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {  
        return application.sources(DemoUpcG3FasiApp02StarterWebApplication.class);  
    }  
}
```

#### 4.7.10. Tester le Projet

- Exécuter le module DemoUpcG3FasiApp02StarterWeb

#### 4.7.11. Apprêter les ressources de l'application

- Créer des classes d'exceptions personnalisées dans le module Lib/package edu.upc.utils.exceptions

```
@ResponseStatus(HttpStatus.CONFLICT)  
public class MyDuplicatedEntryException extends RuntimeException {  
  
    private static final long serialVersionUID = 1L;  
  
    public MyDuplicatedEntryException() {  
        super();  
    }  
  
    public MyDuplicatedEntryException(String message, Throwable cause, boolean enableSuppression,  
        boolean writableStackTrace) {  
        super(message, cause, enableSuppression, writableStackTrace);  
    }  
  
    public MyDuplicatedEntryException(String message, Throwable cause) {  
        super(message, cause);  
    }  
  
    public MyDuplicatedEntryException(String message) {  
        super(message);  
    }  
  
    public MyDuplicatedEntryException(Throwable cause) {  
        super(cause);  
    }  
}
```

```
@ResponseStatus(HttpStatus.CONFLICT)
public class MyInvalidEntryException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public MyInvalidEntryException() {
        super();
    }

    public MyInvalidEntryException(String message, Throwable cause, boolean enableSuppression,
                                   boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }

    public MyInvalidEntryException(String message, Throwable cause) {
        super(message, cause);
    }

    public MyInvalidEntryException(String message) {
        super(message);
    }

    public MyInvalidEntryException(Throwable cause) {
        super(cause);
    }

}

@ResponseStatus(HttpStatus.NOT_FOUND)
public class MyNotFoundException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public MyNotFoundException() {
        super();
    }

    public MyNotFoundException(String message, Throwable cause, boolean enableSuppression,
                             boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }

    public MyNotFoundException(String message, Throwable cause) {
        super(message, cause);
    }

    public MyNotFoundException(String message) {
        super(message);
    }

    public MyNotFoundException(Throwable cause) {
        super(cause);
    }

}
```

```
@ResponseStatus(HttpStatus.UNAUTHORIZED)
public class MyInvalidCredsException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public MyInvalidCredsException() {
        super();
    }

    public MyInvalidCredsException(String message, Throwable cause, boolean enableSuppression,
                                   boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }

    public MyInvalidCredsException(String message, Throwable cause) {
        super(message, cause);
    }

    public MyInvalidCredsException(String message) {
        super(message);
    }

    public MyInvalidCredsException(Throwable cause) {
        super(cause);
    }
}

@ResponseStatus(HttpStatus.UNAUTHORIZED)
public class MyAccessDeniedException extends RuntimeException {

    private static final long serialVersionUID = 1L;

    public MyAccessDeniedException() {
        super();
    }

    public MyAccessDeniedException(String message, Throwable cause, boolean enableSuppression,
                                   boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }

    public MyAccessDeniedException(String message, Throwable cause) {
        super(message, cause);
    }

    public MyAccessDeniedException(String message) {
        super(message);
    }

    public MyAccessDeniedException(Throwable cause) {
        super(cause);
    }
}
```

- Créer le @Component AppUtils dans le module Lib/package edu.upc.utils

```

@Component
public class AppUtils {

    @Autowired
    private MessageSource messageSource;

    public String getFilterKeyword(String keyword) {
        return StringUtils.isEmpty(keyword) ? "%" : "%" + keyword + "%";
    }

    public <T> T getObjectFromList(List<T> data) {
        return !CollectionUtils.isEmpty(data) ? data.get(0) : null;
    }

    public String getErrors(BindingResult result) {
        final StringJoiner joiner = new StringJoiner(",");
        final List<ObjectError> errors = result.getAllErrors();
        if (!CollectionUtils.isEmpty(errors)) {
            for (ObjectError item : errors) {
                joiner.add(item.getDefaultMessage());
            }
        }
        return String.valueOf(joiner);
    }

    public String getErrorMsg(Exception ex, Locale locale) {
        final String error;
        if (ex instanceof MyNotFoundException) {
            error = messageSource.getMessage("error.noDataFound", null, locale);
        } else if (ex instanceof MyInvalidEntryException) {
            error = messageSource.getMessage("error.invalidEntry", null, locale);
        } else if (ex instanceof MyDuplicatedEntryException) {
            error = messageSource.getMessage("error.duplicatedEntry1", null, locale);
        } else if (ex instanceof MyAccessDeniedException) {
            error = messageSource.getMessage("error.accessDenied", null, locale);
        } else if (ex instanceof MyInvalidCredsException) {
            error = messageSource.getMessage("error.invalidCredentials", null, locale);
        } else {
            error = messageSource.getMessage("error.system", null, locale);
        }
        final String error_details = ex.getLocalizedMessage();
        return error.concat(StringUtils.isNotEmpty(error_details) ? ":" + error_details : ". ");
    }
}

```

- Créer le @Component AppWebUrlsUtils dans le module Web/package edu.upc.utils

```

@Component(value="webUrls")
public class AppWebUrlsUtils { /* DON'T CHANGE Method Names USED IN HTML */

    public final String home = "/";

    public final String professions_view = "/professions";
    public final String professions_form = "/professions/get-form/{id}";

    public final String formations_view = "/formations";
    public final String formations_form = "/formations/get-form/{id}";

    public final String etudiants_view = "/etudiants";
    public final String etudiants_form = "/etudiants/get-form/{id}";

    public final String etudiants_formations_view = "/etudiants/formations";
    public final String etudiants_formations_form = "/etudiants/formations/get-form/{id}";

}

```

#### 4.7.12. Créer les Modèles @Entity dans le module Lib

- @Entity Profession

```
@Entity
@Table(name = "professions")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class Profession implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @NotEmpty
    @Size(min = 1, max = 45)
    private String description;

    @NotNull
    @Builder.Default
    private boolean actif = true;

}
```

- @Entity Formation

```
@Entity
@Table(name = "formations")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class Formation implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @NotEmpty
    @Size(min = 1, max = 45)
    private String description;

    @NotNull
    @Min(1)
    private int duree;

    private String notes;

    @NotNull
    @Builder.Default
    private boolean actif = true;

}
```

- @Entity Etudiant

```

@Entity
@Table(name = "etudiants")
@NoArgsConstructor
@Builder
@Getter
@Setter
@ToString
public class Etudiant implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @NotNull
    @Min(1)
    private int professionPk;

    @NotEmpty
    @Size(min = 1, max = 15)
    private String prenom, nom, postnom;

    @NotEmpty
    @Size(min = 1, max = 1)
    private String genre, etatCivil;

    @NotEmpty
    @Email
    @Size(min = 1, max = 45)
    private String telephone, email;

    @NotNull
    @Min(1900)
    private int anneeNais;

    private String contacts, adresse;

    @NotNull
    @Builder.Default
    private boolean actif = true;

    @Transient
    private Profession profession;

    @Transient
    private List<Formation> formations;
}

```
- @Entity EtudiantFormation

```

@Entity
@Table(name = "etudiants_formations")
@NoArgsConstructor
@Builder
@Getter
@Setter
@ToString
public class EtudiantFormation implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @NotNull
    @Min(1)
    private int etudiantPk, formationPk;

    @Transient
    private Etudiant etudiant;

    @Transient
    private Formation formation;
}

```

#### 4.7.13. Créer les classes DTO (Data Transfer Object) dans le module Lib

- ProfessionDTO

```
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class ProfessionDTO implements Serializable {

    private static final long serialVersionUID = 1L;

    @NotNull(message = "{form.id}")
    @Min(value = 0, message = "{form.id}")
    private int id;

    @NotEmpty(message = "{form.description}")
    @Size(min = 1, max = 45, message = "{form.description}")
    private String description;

    @NotNull
    @Builder.Default
    private boolean actif = true;

}
```

- FormationDTO

```
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class FormationDTO implements Serializable {

    private static final long serialVersionUID = 1L;

    @NotNull(message = "{form.id}")
    @Min(value = 0, message = "{form.id}")
    private int id;

    @NotEmpty(message = "{form.description}")
    @Size(min = 1, max = 45, message = "{form.description}")
    private String description;

    @NotNull
    @Min(value=1, message = "{form.duration}")
    private int duree;

    private String notes;

    @NotNull
    @Builder.Default
    private boolean actif = true;

}
```

- EtudiantDTO

---

```
public class EtudiantDTO implements Serializable {

    private static final long serialVersionUID = 1L;

    @NotNull(message = "{form.id}")
    @Min(value = 0, message = "{form.id}")
    private int id;

    @NotNull(message = "{form.occupation}")
    @Min(value = 1, message = "{form.occupation}")
    private int professionPk;

    @NotEmpty(message = "{form.firstName}")
    @Size(min = 1, max = 15, message = "{form.firstName}")
    private String prenom;

    @NotEmpty(message = "{form.name}")
    @Size(min = 1, max = 15, message = "{form.name}")
    private String nom;

    @NotEmpty(message = "{form.lastName}")
    @Size(min = 1, max = 15, message = "{form.lastName}")
    private String postnom;

    @NotEmpty(message = "{form.gender}")
    @Size(min = 1, max = 1, message = "{form.gender}")
    private String genre;

    @NotEmpty(message = "{form.maritalStatus}")
    @Size(min = 1, max = 1, message = "{form.maritalStatus}")
    private String etatCivil;

    @NotEmpty(message = "{form.phone}")
    @Size(min = 1, max = 45, message = "{form.phone}")
    private String telephone;

    @NotEmpty(message = "{form.email}")
    @Email(message = "{form.email}")
    @Size(min = 1, max = 45, message = "{form.email}")
    private String email;

    @NotNull(message = "{form.yearOfBirth}")
    @Min(value = 1900, message = "{form.yearOfBirth}")
    private int anneeNais;

    private String contacts, adresse;

    @NotNull
    @Builder.Default
    private boolean actif = true;
```

- EtudiantFormationDTO
- ```
@NoArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@Builder
public class EtudiantFormationDTO implements Serializable {

    private static final long serialVersionUID = 1L;

    @NotNull(message = "{form.id}")
    @Min(value = 0, message = "{form.id}")
    private int id;

    @NotNull(message = "{form.student}")
    @Min(value = 1, message = "{form.student}")
    private int etudiantPk;

    @NotNull(message = "{form.training}")
    @Min(value = 1, message = "{form.training}")
    private int formationPk;

}
```

#### 4.7.14. Créer les Mappers @Component dans le module Lib

- @Component Profession
- ```
@Component
public class ProfessionMapper {

    public ProfessionDTO mapToDTO(Profession entity) {
        return ProfessionDTO.builder()
            .id(entity.getId())
            .description(entity.getDescription())
            .actif(entity.isActif())
            .build();
    }

    public Profession mapToEntity(ProfessionDTO dto) {
        return Profession.builder()
            .id(dto.getId())
            .description(dto.getDescription())
            .actif(dto.isActif())
            .build();
    }
}
```
- @Component Formation

```

@Component
public class FormationMapper {

    public FormationDTO mapToDTO(Formation entity) {
        return FormationDTO.builder()
            .id(entity.getId())
            .description(entity.getDescription())
            .duree(entity.getDuree())
            .notes(entity.getNotes())
            .actif(entity.isActif())
            .build();
    }

    public Formation mapToEntity(FormationDTO dto) {
        return Formation.builder()
            .id(dto.getId())
            .description(dto.getDescription())
            .duree(dto.getDuree())
            .notes(dto.getNotes())
            .actif(dto.isActif())
            .build();
    }
}

```

- @Component Etudiant

```

@Component
public class EtudiantMapper {

    public EtudiantDTO mapToFormBean(Etudiant entity) {
        return EtudiantDTO.builder()
            .id(entity.getId())
            .professionPk(entity.getProfessionPk())
            .prenom(entity.getPrenom())
            .nom(entity.getNom())
            .postnom(entity.getPostnom())
            .genre(entity.getGenre())
            .etatCivil(entity.getEtatCivil())
            .telephone(entity.getTelephone())
            .email(entity.getEmail())
            .anneeNais(entity.getAnneeNais())
            .contacts(entity.getContacts())
            .adresse(entity.getAdresse())
            .actif(entity.isActif())
            .build();
    }

    public Etudiant mapToEntity(EtudiantDTO dto) {
        return Etudiant.builder()
            .id(dto.getId())
            .professionPk(dto.getProfessionPk())
            .prenom(dto.getPrenom())
            .nom(dto.getNom())
            .postnom(dto.getPostnom())
            .genre(dto.getGenre())
            .etatCivil(dto.getEtatCivil())
            .telephone(dto.getTelephone())
            .email(dto.getEmail())
            .anneeNais(dto.getAnneeNais())
            .contacts(dto.getContacts())
            .adresse(dto.getAdresse())
            .actif(dto.isActif())
            .build();
    }
}

```

- @Component EtudiantFormation

```
@Component
public class EtudiantFormationMapper {

    public EtudiantFormationDTO mapToDTO(EtudiantFormation entity) {
        return EtudiantFormationDTO.builder()
            .id(entity.getId())
            .etudiantPk(entity.getEtudiantPk())
            .formationPk(entity.getFormationPk())
            .build();
    }

    public EtudiantFormation mapToEntity(EtudiantFormationDTO dto) {
        return EtudiantFormation.builder()
            .id(dto.getId())
            .etudiantPk(dto.getEtudiantPk())
            .formationPk(dto.getFormationPk())
            .build();
    }
}
```

#### 4.7.15. Implémenter la couche @Repository dans le module Data

- @Repository Profession

- Repo

```
@Repository
public interface ProfessionRepo extends JpaRepository<Profession, Integer> {
}



- RepoCustom


public interface ProfessionRepoCustom {

    boolean isUnique(int id, String description);

    Profession getRecordById(int id);

    List<Profession> getRecords(boolean actif, String keyword);
}
```

- Repolmpl

```

@Repository
public class ProfessionRepoImpl implements ProfessionRepoCustom {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Autowired
    private AppUtils utils;

    private static final ResultSetExtractor<Profession> rsExtractor = JdbcTemplateMapperFactory.newInstance()
        .addKeys("id")
        .ignorePropertyNotFound()
        .newResultSetExtractor(Profession.class);

    /**
     * @Override
     * public boolean isUnique(int id, String description) {
     *     final String sql = "SELECT NOT EXISTS("
     *         + "SELECT id FROM professions WHERE id=>? AND description=? LIMIT 1"
     *         + ")";
     *
     *     final Object[] params = new Object[] { id, description };
     *
     *     return jdbcTemplate.queryForObject(sql, Boolean.class, params);
     * }
     *
     * @Override
     * public Profession getRecordById(int id) {
     *     final String sql = "SELECT * FROM professions WHERE id=?";
     *
     *     final Object[] params = new Object[] { id };
     *
     *     return utils.getObjectFromList(jdbcTemplate.query(sql, rsExtractor, params));
     * }
     *
     * @Override
     * public List<Profession> getRecords(boolean actif, String keyword) {
     *     final String _stmt = "SELECT * FROM professions ";
     *
     *     final String _order = " ORDER BY id";
     *
     *     String sql = _stmt.concat("WHERE actif=? AND description LIKE ?").concat(_order);
     *
     *     final String _keyword = utils.getFilterKeyword(keyword);
     *
     *     Object[] params = new Object[] { actif, _keyword };
     *
     *     if (StringUtils.isEmpty(keyword)) {
     *         sql = _stmt.concat("WHERE actif=?").concat(_order);
     *
     *         params = new Object[] { actif };
     *     }
     *
     *     return jdbcTemplate.query(sql, rsExtractor, params);
     * }
}

```

- @Repository Formation
  - Repo
  - @Repository
  - public interface FormationRepo extends JpaRepository<Formation, Integer> { }
  - RepoCustom
  - public interface FormationRepoCustom {
  - boolean isUnique(int id, String description);
  - Formation getRecordById(int id);
  - List<Formation> getRecords(boolean actif, String keyword);
  - }
  - Repolmpl

```

@Repository
public class FormationRepoImpl implements FormationRepoCustom {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Autowired
    private AppUtils utils;

    private static final ResultSetExtractorImpl<Formation> rsExtractor = JdbcTemplateMapperFactory.newInstance()
        .addKeys("id")
        .ignorePropertyNotFound()
        .newResultSetExtractor(Formation.class);

    /**
     * @Override
     * public boolean isUnique(int id, String description) {
     *     final String sql = "SELECT NOT EXISTS("
     *         + "SELECT id FROM formations WHERE id=>? AND description=? LIMIT 1"
     *         + ")";
     *
     *     final Object[] params = new Object[] { id, description };
     *
     *     return jdbcTemplate.queryForObject(sql, Boolean.class, params);
     * }

     * @Override
     * public Formation getRecordById(int id) {
     *     final String sql = "SELECT * FROM formations WHERE id=?";
     *
     *     final Object[] params = new Object[] { id };
     *
     *     return utils.getObjectFromList(jdbcTemplate.query(sql, rsExtractor, params));
     * }

     * @Override
     * public List<Formation> getRecords(boolean actif, String keyword) {
     *     final String _stmt = "SELECT * FROM formations ";
     *
     *     final String _order = " ORDER BY id";
     *
     *     String sql = _stmt.concat("WHERE actif=? AND description LIKE ?").concat(_order);
     *
     *     final String _keyword = utils.getFilterKeyword(keyword);
     *
     *     Object[] params = new Object[] { actif, _keyword };
     *
     *     if (StringUtils.isEmpty(keyword)) {
     *         sql = _stmt.concat("WHERE actif=?").concat(_order);
     *
     *         params = new Object[] { actif };
     *     }
     *
     *     return jdbcTemplate.query(sql, rsExtractor, params);
     * }
}

```

- @Repository Etudiant
  - Repo
- @Repository
 

```

public interface EtudiantRepo extends JpaRepository<Etudiant, Integer> {
}
```

  - RepoCustom
- public interface EtudiantRepoCustom {

 boolean isUnique(int id, String prenom, String nom, String postnom);

 Etudiant getRecordById(int id);

 List<Etudiant> getRecords(boolean actif, String keyword);
 }
- Repolmpl

```

@Repository
public class EtudiantRepoImpl implements EtudiantRepoCustom {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Autowired
    private AppUtils utils;

    private static final ResultSetExtractorImpl<Etudiant> rsExtractor = JdbcTemplateMapperFactory.newInstance()
        .addKeys("id", "formations_id")
        .ignorePropertyNotFound()
        .newResultSetExtractor(Etudiant.class);

    /**
     * @Override
     * public boolean isUnique(int id, String prenom, String nom, String postnom) {
     *     final String sql = "SELECT NOT EXISTS("
     *         + "SELECT id FROM etudiants WHERE id=>? AND prenom=? AND nom=? AND postnom=? LIMIT 1"
     *         + ")";
     *
     *     final Object[] params = new Object[] { id, prenom, nom, postnom };
     *     return jdbcTemplate.queryForObject(sql, Boolean.class, params);
     * }
     *
     * @Override
     * public Etudiant getRecordById(int id) {
     *     final String sql = "SELECT x1.*," +
     *         + "x2.id AS profession_id, x2.description AS profession_description, " +
     *         + "x4.id AS formations_id, " +
     *         + "x4.description AS formations_description, x4.duree AS formations_duree, x4.actif AS formations_actif " +
     *         + "FROM etudiants x1 " +
     *         + "INNER JOIN professions x2 ON x1.professionPk=x2.id " +
     *         + "LEFT JOIN etudiants_formations x3 ON x3.etudiantPk=x1.id " +
     *         + "LEFT JOIN formations x4 ON x3.formationPk=x4.id " +
     *         + "WHERE x1.id=?";
     *
     *     final Object[] params = new Object[] { id };
     *     return utils.getObjectFromList(jdbcTemplate.query(sql, rsExtractor, params));
     * }
     *
     * @Override
     * public List<Etudiant> getRecords(boolean actif, String keyword) {
     *     final String _sql = "SELECT x1.*," +
     *         + "x2.id AS profession_id, x2.description AS profession_description " +
     *         + "FROM etudiants x1 " +
     *         + "INNER JOIN professions x2 ON x1.professionPk=x2.id ";
     *
     *     final String _order = " ORDER BY id";
     *
     *     String sql = _sql.concat("WHERE x1.actif=? AND (prenom LIKE ? OR nom LIKE ? OR postnom LIKE ?)").concat(_order);
     *     final String _keyword = utils.getFilterKeyword(keyword);
     *
     *     Object[] params = new Object[] { actif, _keyword, _keyword, _keyword };
     *
     *     if (StringUtils.isEmpty(keyword)) {
     *         sql = _sql.concat("WHERE x1.actif=?").concat(_order);
     *         params = new Object[] { actif };
     *     }
     *
     *     return jdbcTemplate.query(sql, rsExtractor, params);
     * }
     *
    ■ @Repository EtudiantFormation
      ○ Repo
      @Repository
      public interface EtudiantFormationRepo extends JpaRepository<EtudiantFormation, Integer> {
    }
  
```

4.7.16. Implémenter la couche @Service dans le module Data

- @Service Profession
  - Service

```
public interface ProfessionService {  
    Profession save(ProfessionDTO dto, BindingResult result, Locale locale);  
    List<Profession> delete(int id, Locale locale);  
    Profession getRecordById(int id);  
    List<Profession> getRecords(boolean actif, String keyword);  
}
```

- o ServiceImpl
 

```

@Service
public class ProfessionServiceImpl implements ProfessionService {

    @Autowired
    private ProfessionRepo repo;

    @Autowired
    private ProfessionRepoCustom repoC;

    @Autowired
    private ProfessionMapper mapper;

    @Autowired
    private AppUtils utils;

    @Autowired
    private MessageSource msgSrc;

    @Transactional(rollbackFor = Exception.class)
    @Override
    public Profession save(ProfessionDTO dto, BindingResult result, Locale locale) {
        if (result.hasErrors()) {
            throw new MyInvalidEntryException(utils.getErrors(result));
        } else if (!repoC.isUnique(dto.getId(), dto.getDescription())) {
            throw new MyDuplicatedEntryException(
                msgSrc.getMessage("error.duplicatedEntry", new String[] { dto.getDescription() }, locale));
        }
        return repo.saveAndFlush(mapper.mapToEntity(dto));
    }

    @Transactional(rollbackFor = Exception.class)
    @Override
    public List<Profession> delete(int id, Locale locale) {
        repo.deleteById(id);
        repo.flush();
        return this.getRecords(true, null);
    }

    @Transactional(readOnly = true)
    @Override
    public Profession getRecordById(int id) {
        return repoC.getRecordById(id);
    }

    @Transactional(readOnly = true)
    @Override
    public List<Profession> getRecords(boolean actif, String keyword) {
        return repoC.getRecords(actif, keyword);
    }
}

```
- @Service Formation
  - o Service
 

```

public interface FormationService {

    Formation save(FormationDTO dto, BindingResult result, Locale locale);

    List<Formation> delete(int id, Locale locale);

    Formation getRecordById(int id);

    List<Formation> getRecords(boolean actif, String keyword);
}

```

- o *ServiceImpl*

```

@Service
public class FormationServiceImpl implements FormationService {

    @Autowired
    private FormationRepo repo;

    @Autowired
    private FormationRepoCustom repoC;

    @Autowired
    private FormationMapper mapper;

    @Autowired
    private AppUtils utils;

    @Autowired
    private MessageSource msgSrc;

    @Transactional(rollbackFor = Exception.class)
    @Override
    public Formation save(FormationDTO dto, BindingResult result, Locale locale) {
        if (result.hasErrors()) {
            throw new MyInvalidEntryException(utils.getErrors(result));
        } else if (!repoC.isUnique(dto.getId(), dto.getDescription())) {
            throw new MyDuplicatedEntryException(
                msgSrc.getMessage("error.duplicatedEntry", new String[] { dto.getDescription() }, locale));
        }
        return repo.saveAndFlush(mapper.mapToEntity(dto));
    }

    @Transactional(rollbackFor = Exception.class)
    @Override
    public List<Formation> delete(int id, Locale locale) {
        repo.deleteById(id);
        repo.flush();
        return this.getRecords(true, null);
    }

    @Transactional(readOnly = true)
    @Override
    public Formation getRecordById(int id) {
        return repoC.getRecordById(id);
    }

    @Transactional(readOnly = true)
    @Override
    public List<Formation> getRecords(boolean actif, String keyword) {
        return repoC.getRecords(actif, keyword);
    }
}

```

- *@Service Etudiant*

- o *Service*

```

public interface EtudiantService {

    Etudiant save(EtudiantDTO dto, BindingResult result, Locale locale);

    List<Etudiant> delete(int id, Locale locale);

    Etudiant getRecordById(int id);

    List<Etudiant> getRecords(boolean actif, String keyword);
}

```

```

o ServiceImpl
@Service
public class EtudiantServiceImpl implements EtudiantService {

    @Autowired
    private EtudiantRepo repo;

    @Autowired
    private EtudiantRepoCustom repoC;

    @Autowired
    private EtudiantMapper mapper;

    @Autowired
    private AppUtils utils;

    @Autowired
    private MessageSource msgSrc;

    @Transactional(rollbackFor = Exception.class)
    @Override
    public Etudiant save(EtudiantDTO dto, BindingResult result, Locale locale) {
        if (result.hasErrors()) {
            throw new MyInvalidEntryException(utils.getErrors(result));
        }
        else if (!repoC.isUnique(dto.getId(), dto.getPrenom(), dto.getNom(), dto.getPostnom())) {
            throw new MyDuplicatedEntryException(msgSrc.getMessage("error.duplicatedEntry",
                new String[] { dto.getPrenom().concat(" " + dto.getNom()).concat(" " + dto.getPostnom() }, locale));
        }
        return repo.saveAndFlush(mapper.mapToEntity(dto));
    }

    @Transactional(rollbackFor = Exception.class)
    @Override
    public List<Etudiant> delete(int id, Locale locale) {
        repo.deleteById(id);
        repo.flush();
        return this.getRecords(true, null);
    }

    @Transactional(readOnly = true)
    @Override
    public Etudiant getRecordById(int id) {
        return repoC.getRecordById(id);
    }

    @Transactional(readOnly = true)
    @Override
    public List<Etudiant> getRecords(boolean actif, String keyword) {
        return repoC.getRecords(actif, keyword);
    }
}

```

- @Service EtudiantFormation
  - o Service

```

public interface EtudiantFormationService {

    Etudiant save(EtudiantFormationDTO dto, BindingResult result, Locale locale);

    Etudiant delete(int id, Locale locale);
}

```

```

o ServiceImpl
@Service
public class EtudiantFormationServiceImpl implements EtudiantFormationService {

    @Autowired
    private EtudiantFormationRepo repo;

    @Autowired
    private EtudiantRepoCustom repoE;

    @Autowired
    private EtudiantFormationMapper mapper;

    @Autowired
    private AppUtils utils;

    @Autowired
    private MessageSource msgSrc;

    @Transactional(rollbackFor = Exception.class)
    @Override
    public Etudiant save(EtudiantFormationDTO dto, BindingResult result, Locale locale) {
        if (result.hasErrors()) {
            throw new MyInvalidEntryException(utils.getErrors(result));
        }

        final EtudiantFormation row = repo.saveAndFlush(mapper.mapToEntity(dto));

        return row != null ? repoE.getRecordById(row.getEtudiantPk()) : null;
    }

    @Transactional(rollbackFor = Exception.class)
    @Override
    public Etudiant delete(int id, Locale locale) {
        final Optional<EtudiantFormation> _row = repo.findById(id);
        if (!_row.isPresent() || _row == null) {
            throw new MyNotFoundException(msgSrc.getMessage("error.notFoundEntry", null, locale));
        }

        final EtudiantFormation row = _row.get();
        repo.deleteById(id);
        repo.flush();

        return repoE.getRecordById(row.getEtudiantPk());
    }
}

```

#### 4.7.17. Implémenter la couche @Controller dans le module Web

- Créer le @Component AppViewUtils pour gérer les noms de fichiers .html

```

@Component
public class AppViewUtils {

    public final String professions_view = "professions-view";
    public final String professions_form = "professions-form";

    public final String formations_view = "formations-view";
    public final String formations_form = "formations-form";

    public final String etudiants_view = "etudiants-view";
    public final String etudiants_form = "etudiants-form";

    public final String etudiants_formations_view = "etudiants-formations-view";
    public final String etudiants_formations_form = "etudiants-formations-form";
}

```

- @Controller Profession

```
@Controller
@RequestMapping("/professions")
public class ProfessionController {

    @Autowired
    private ProfessionService service;

    @Autowired
    private ProfessionMapper mapper;

    @Autowired
    private LocaleResolver resolver;

    @Autowired
    private MessageSource msgSrc;

    @Autowired
    private AppViewUtils viewUtils;

    @Autowired
    private AppUtils appUtils;

    /**
     * @GetMapping("/get-form/{id}")
     */
    public String getForm(@PathVariable("id") int id, Model model, HttpServletRequest request) {
        try {
            final ProfessionDTO row = (id != 0) ? mapper.mapToDTO(service.getRecordById(id)) : new ProfessionDTO();
            model.addAttribute("title", this.getFormTitle(row, request));
            model.addAttribute("entryBean", row);
        } catch (Exception ex) {
            model.addAttribute("error", ex.getLocalizedMessage());
        }
        return viewUtils.professions_form;
    }

    /**
     * @GetMapping
     */
    public String getRecords(@RequestParam(value = "keyword", required = false, defaultValue = "") String keyword,
                           @RequestParam(value = "active", required = false, defaultValue = "true") boolean active,
                           HttpServletRequest request, Model model) {
        try {
            if (request.getParameter("active") == null) { // Handle Unchecked checkbox not submitted
                active = false;
            }
            final List<Profession> data = service.getRecords(active, keyword);
            model.addAttribute("data", data);
            model.addAttribute("active", active);
            model.addAttribute("keyword", keyword);
        } catch (Exception ex) {
            model.addAttribute("error", ex.getLocalizedMessage());
        }
        return viewUtils.professions_view;
    }
}
```

```

@PostMapping
public String doSave(@Valid @ModelAttribute("entryBean") ProfessionDTO dto, BindingResult result, Model model,
    RedirectAttributes redAttr, HttpServletRequest request) {
    final Locale locale = resolver.resolveLocale(request);

    // Process
    try {
        long id = dto != null ? dto.getId() : 0;
        service.save(dto, result, locale);
        if (id == 0) { // New
            model.addAttribute("title", this.getFormTitle(new ProfessionDTO(), request));
            model.addAttribute("entryBean", new ProfessionDTO());
            model.addAttribute("message", msgSrc.getMessage("message.taskSuccessfullyCompleted", null, locale));

            return viewUtils.professions_form;
        } else {
            redAttr.addFlashAttribute("message",
                msgSrc.getMessage("message.taskSuccessfullyCompleted", null, locale));
        }
        return "redirect:/professions/?active=true";
    }
    } catch (Exception ex) {
        model.addAttribute("title", this.getFormTitle(dto, request));
        result.reject("error", appUtils.getErrorMsg(ex, locale));
        return viewUtils.professions_form;
    }
}

@RequestMapping("/delete/{id}")
public String doDelete(@PathVariable("id") int id, Model model, HttpServletRequest request) {
    final Locale locale = resolver.resolveLocale(request);
    try {
        service.delete(id, locale);

    } catch (Exception ex) {
        model.addAttribute("error", ex.getLocalizedMessage());
    }

    return "redirect:/professions";
}

// Helper
private String getFormTitle(ProfessionDTO entity, HttpServletRequest request) {
    final Locale locale = resolver.resolveLocale(request);
    final int id = entity != null ? entity.getId() : 0;

    return msgSrc.getMessage("form.occupations", null, locale)
        .concat(" / " + (id != 0 ? msgSrc.getMessage("form.edit", null, locale).concat("# " + id)
            : msgSrc.getMessage("form.new", null, locale)));
}
}

```

- @Controller Formation
- @Controller Etudiant
- @Controller EtudiantFormation

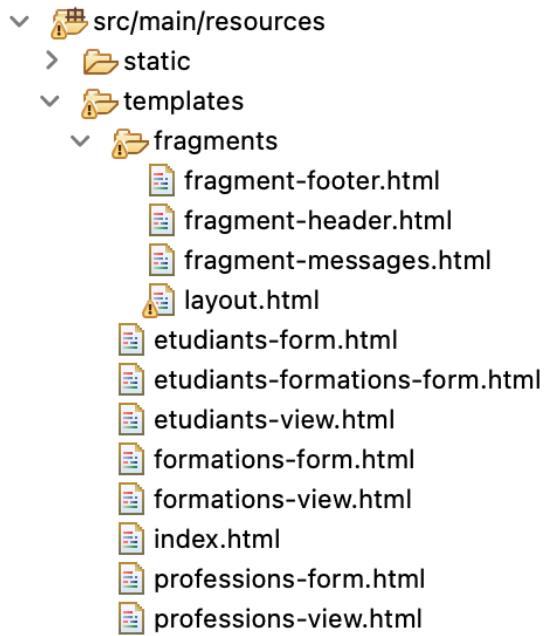
#### 4.7.18. Implémenter le Thymeleaf Layout

- Inclure la dépendance Layout Dialect/Thymeleaf dans le POM du module Data
 

```
<dependency>
        <groupId>nz.net.ultraq.thymeleaf</groupId>
        <artifactId>thymeleaf-layout-dialect</artifactId>
      </dependency>
```
- Inclure cette @Bean dans la @Configuration AppConfig

```
@Bean
public LayoutDialect layoutDialect() {
    return new LayoutDialect();
}
```

- Créer les fichiers ci-après dans le dossier /templates/fragments du module Web



#### 4.7.19. Mettre à jour les fichiers .html

- Définir le fragment fragment-header.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<div th:fragment="header">

    <div>
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
            <div class="container-fluid">
                <a class="navbar-brand" th:href="@{@{webUrls.home}}>App02</a>
                <button class="navbar-toggler" type="button"
                    data-bs-toggle="collapse" data-bs-target="#navbarScroll"
                    aria-controls="navbarScroll" aria-expanded="false"
                    aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarScroll">
                    <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll"
                        style="--bs-scroll-height: 100px;">
                        <li class="nav-item"><a class="nav-link active"
                            aria-current="page" th:href="@{@{webUrls.home}}"
                            th:text="#{form.home}">Home</a></li>
                        <li class="nav-item dropdown"><a
                            class="nav-link dropdown-toggle" href="#"
                            id="navbarDropdownScrollingDropdown" role="button"
                            data-bs-toggle="dropdown" aria-expanded="false"
                            th:text="#{form.file}"></a>
                            <ul class="dropdown-menu"
                                aria-labelledby="navbarDropdownScrollingDropdown">
                                <li><a class="dropdown-item"
                                    th:href="@{@{webUrls.professions_view}}"
                                    th:text="#{form.professions}"></a></li>
                                <li><a class="dropdown-item"
                                    th:href="@{@{webUrls.formations_view}}"
                                    th:text="#{form.trainings}"></a></li>
                                <li><hr class="dropdown-divider"></li>
                                <li><a class="dropdown-item"
                                    th:href="@{@{webUrls.etudiants_view}}"
                                    th:text="#{form.students}"></a></li>
                            </ul></li>
                        </ul>
                        <div class="text-light">
                            <a class="link-light" style="text-decoration: none;">
                                th:href="@{?locale=en}" th:text="#{form.english}"</a> | <a
                                class="link-light" style="text-decoration: none;">
                                th:href="@{?locale=fr}" th:text="#{form.french}"</a>
                            </div>
                    </div>
                </nav>
            </div>
        </div>
    </div>
</html>

```

- Définir le fragment fragment-footer.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<div th:fragment="footer">

    <footer class="bg-dark text-center text-white">
        <div class="align-middle p-2">
            <p>
                © 2023 <span>Omer Pitou Ntumba Katuala</span>
            </p>
        </div>
    </footer>

</div>

</html>
```

- Définir le fragment fragment-messages.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:fragment="msg-validation">

    <div th:if="${#fields.hasErrors('*')}" th:remove="tag">
        <br/>
        <div class="alert alert-danger alert-dismissible fade show"
            role="alert">
            <strong th:text="#{form.error}">Error</strong>
            <button type="button" class="close" data-dismiss="alert"
                aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
            <ul th:if="${#fields.hasErrors('*')}">
                <li th:each="err : ${#fields.errors('*')}" th:if="${err} != ''"
                    th:text="${err}"></li>
            </ul>
        </div>
    </div>

    <div th:if="${message != null}">
        <div class="alert alert-success alert-dismissible fade show"
            role="alert">
            [[${message}]]
            <button type="button" class="close" data-dismiss="alert"
                aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
        </div>
    </div>

    <div th:if="${error != null}">
        <div class="alert alert-danger alert-dismissible fade show"
            role="alert">
            [[${error}]]
            <button type="button" class="close" data-dismiss="alert"
                aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
        </div>
    </div>
</th:block>
</html>
```

- Définir le layout.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
<meta charset="utf-8" />
<meta
      content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no"
      name="viewport" />
<meta content="" name="description" />
<meta content="" name="author" />

<title layout:title-pattern="$LAYOUT_TITLE | $CONTENT_TITLE"></title>

<link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-GhltTQ8iRABdZLl603oVMWSktQ00p6b7In1zI3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
    crossorigin="anonymous" />

<link rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

</head>
<body>

    <div class="container">

        <div id="header">
            <div th:replace="fragments/fragment-header :: header"></div>
        </div>

        <div id="content">
            <th:block layout:fragment="page-content"></th:block>
        </div>

        <div id="footer">
            <div th:replace="fragments/fragment-footer :: footer"></div>
        </div>

    </div>

    <script
        src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN"
        crossorigin="anonymous"></script>

    <th:block layout:fragment="app-js"></th:block>

</body>
```

- Appliquer le Layout dans index.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~/fragments/layout">

<head>
</head>
<body>

    <th:block layout:fragment="page-content">

        <div
            class="cover-container d-flex w-100 h-100 p-5 mx-auto flex-column">
            <main role="main" class="inner cover">
                <h1 class="cover-heading">G3 FASI / UPC</h1>
                <h1 class="cover-heading">Spring Framework</h1>
                <h1 class="cover-heading">Module 01</h1>
                <h1 class="cover-heading">App02</h1>
                <ul class="lead">
                    <li th:text="#{title.multiModulesProject}">Multi-Module
                        Project</li>
                    <li th:text="#{title.tymeleafLayout}">Tymeleaf Layout</li>
                    <li th:text="#{title.advancedJdbcTemplate}">Advanced
                        JdbcTemplate</li>
                    <li th:text="#{title.pageNavigation}">Page Navigation</li>
                    <li th:text="#{title.security}">Security</li>
                    <li th:text="#{title.others} + '...'">Others...</li>
                </ul>
            </main>
        </div>
    </th:block>

</body>
</html>
```

- Définir le fichier professions-view.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{fragments/layout}">

<head>
</head>
<body>

    <th:block layout:fragment="page-content">

        <div class="container-fluid">

            <h3 class="text-dark my-3" th:text="#{form.occupations}"></h3>

            <div class="row my-3">

                <div class="col-md-6 mt-2">
                    <a id="btnNew" class="btn btn-primary btn-sm"
                       th:href="@{@{webUrls.professions_form}(id=0)}"
                       th:text="#{form.new}"></a>
                </div>
                <div class="col-md-6 mt-2">
                    <form id="myFilterForm" th:action="@{@{webUrls.professions_view}}"
                          enctype="multipart/form-data" method="get">
                        <div class="input-group d-flex align-items-center">
                            <div class="form-check form-check-inline form-switch">
                                <input id="chkActif" class="form-check-input" type="checkbox"
                                       name="active" th:checked="${active}"
                                       onclick="submit()" /><label
                                       class="form-check-label" for="chkActif"
                                       th:text="#{form.active}"></label>
                            </div>
                            <input id="keyword" type="text" class="form-control"
                                   name="keyword" th:value="${keyword}"
                                   th:placeholder="#{form.search}" />
                            <div class="input-group-append">
                                <button class="btn btn-secondary" type="submit">
                                    <i class="fa fa-search"></i>
                                </button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>

    </th:block>
</body>
```

```

<div class="row">
    <table class="table">
        <thead>
            <tr>
                <th scope="col" th:text="#{form.id}"></th>
                <th scope="col" th:text="#{form.description}"></th>
                <th scope="col" th:text="#{form.active}"></th>
                <th scope="col" th:text="#{form.actions}"></th>
            </tr>
        </thead>
        <tbody th:if="${data != null && !data.isEmpty()}">
            <tr th:each="item : ${data}">
                <td scope="row" th:text="${item.id}"></td>
                <td scope="row" th:text="${item.description}"></td>
                <td><div class="form-check form-switch">
                    <input class="form-check-input" type="checkbox"
                           disabled="disabled" th:checked="${item.actif}" />
                </div></td>
                <td><a href="@{@{${webUrls.professions_form} (id=${item.id})}}"
                      class="btn btn-secondary btn-sm"><i
                        class="fa fa-pen-to-square"></i> </a> <a
                      th:href="@{/employees/delete/{id}(id=${item.id})}"
                      th:itemID="${item.id}" class="btn btn-danger btn-sm btn-delete">
                        <i class="fa fa-trash" aria-hidden="true"></i>
                    </a></td>
                </tr>
            </tbody>
            <tbody th:if="${data == null || data.isEmpty()}">
                <tr>
                    <td colspan="4" th:text="#{error.noDataFound}"></td>
                </tr>
            </tbody>
        </table>
    </div>
</div>

<th:block layout-fragment="app-js">
    <script type="text/javascript" th:inline="javascript">
        /*<![CDATA[*]-->
        $(document).ready(function() {
            function submit(e) {
                e.preventDefault();
                $('#myFilterForm').submit();
            };
            });
        /*]]>*/
    </script>
</th:block>

</th:block>

</body>
</html>

```

- Définir le fichier professions-form.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{fragments/layout}">
<head>
</head>
<body>
    <th:block layout:fragment="page-content">
        <div class="container-fluid">

            <div class="my-3 col-md-6">
                <div class="p-3">
                    <h3 class="text-left" th:text="${title}"></h3>

                    <form id="myform" th:action="@{@{webUrls.professions_save}}"
                           method="post" enctype="multipart/form-data"
                           th:object="${entryBean}">

                        <!-- FIELDS -->
                        <div class="form-group">
                            <label for="description" th:text="#{form.description}"></label> <input
                                id="description" type="text"
                                class="form-control form-control-sm" th:field="*{description}"
                                th:errorclass="has-error" />
                        </div>
                        <div class="form-check">
                            <input id="actif" type="checkbox" class="form-check-input"
                                   th:field="*{actif}" /><label class="form-check-label"
                                   for="actif" th:text="#{form.active}"></label>
                        </div>
                        <br/>
                        <div class="form-group">
                            <button id="btnSave" type="submit" class="btn btn-primary btn-sm"
                                   th:text="#{form.save}"></button>
                            <a id="btnCancel" type="button" class="btn btn-danger btn-sm"
                               th:href="@{@{webUrls.professions_view}{(active=true)})"
                               th:text="#{form.cancel}"></a>
                        </div>

                        <!-- MESSAGE -->
                        <div th:replace="fragments/fragment-messages :: msg-validation"></div>

                        <!-- HIDDEN FIELDS -->
                        <input id="id" type="hidden" th:field="*{id}" />
                        <input id="error" type="hidden"/>
                    </form>
                </div>
            </div>
        </th:block>
    </body>

```

- Définir le fichier formations-view.html
- Définir le fichier formations-form.html
- Définir le fichier etudiants-view.html
- Définir le fichier etudiants-form.html

#### 4.7.20. Tester l'application Web

- Créer un @Controller AppController dans le module Web

```
@Controller
public class AppController {

    @RequestMapping("/")
    public String doHome() {
        return "index";

    }

}
```

- Exécuter le module DemoUpcG3FasiApp02StarterWeb

## 5. PRATIQUE: APPLICATION WEB SERVICE (API)

### 5.1. Concepts

#### 5.1.1. Généralités : Http

- [https://www.tutorialspoint.com/http/http\\_overview.htm](https://www.tutorialspoint.com/http/http_overview.htm)

#### 5.1.2. Différence entre Application Web & API

- @RestController pour l'API.
- @Controller pour le Web
- L'API retourne des données structurées, et le format par défaut est JSON (JavaScript Object Notation)
- Le Web retourne un fichier .html
- Configuration Sécurité

#### 5.1.3. Méthodes les plus utilisées (POST, PUT, DELETE, GET...)

- Reference Web : [https://www.tutorialspoint.com/http/http\\_methods.htm](https://www.tutorialspoint.com/http/http_methods.htm)

#### 5.1.4. Status de Réponses http (CREATED, OK, CONFLICT, UNAUTHORIZED, BAD REQUEST...)

- Reference Web : <https://httpstatus.in>

### 5.2. Pratique

#### 5.2.1. Personnaliser la réponse Http

- Créer la classe « MyResponse» pour uniformiser les réponses http dans le module Lib/package edu.upc.dtos

```

@Getter
@Setter
@NoArgsConstructor
public class MyResponse {

    private boolean success = false;
    private int status;
    private String path;
    private String message;

    @JsonInclude(JsonInclude.Include.NON_NULL)
    private List<?> data;

}

```

- Créer un @Component ApiUtils dans le même package

```

@Component
public class ApiUtils {

    @Autowired
    private MessageSource messageSource;

    @SuppressWarnings("unchecked")
    public <T> ResponseEntity<?> getResponse(HttpServletRequest request, HttpStatus status, String message, T data,
                                              Locale locale) {
        List<?> _data = new ArrayList<?>();
        if (data != null) {
            if (data instanceof List<?>) {
                _data = ((List<T>) data).stream().collect(Collectors.toList());
            } else {
                _data.add(data);
            }
        }

        final String _message = status.is2xxSuccessful() && StringUtils.isEmpty(message)
            ? messageSource.getMessage("message.taskSuccessfullyCompleted", null, locale)
            : message;

        MyResponse row = new MyResponse();
        row.setSuccess(status.is2xxSuccessful());
        row.setStatus(status.value());
        row.setPath(request.getServerName().concat(request.getRequestURI()));
        row.setMessage(_message);
        row.setData(_data);

        return new ResponseEntity<?>(row, status);
    }

    public HttpStatus getErrorStatus(Exception ex) {
        final HttpStatus status;

        if (ex instanceof MyNotFoundException) {
            status = HttpStatus.NOT_FOUND;
        } else if (ex instanceof MyInvalidEntryException || ex instanceof MyDuplicatedEntryException) {
            status = HttpStatus.CONFLICT;
        } else if (ex instanceof MyAccessDeniedException || ex instanceof MyInvalidCredsException) {
            status = HttpStatus.UNAUTHORIZED;
        } else {
            status = HttpStatus.INTERNAL_SERVER_ERROR;
        }

        return status;
    }
}

```

### 5.2.2. Gérer les Exceptions

- Créer un @RestControllerAdvice dans le package edu.upc.controllers.utils, pour gérer toutes les exceptions

```
@RestControllerAdvice
public class MyExceptionHandler {

    @Autowired
    private LocaleResolver resolver;

    @Autowired
    private AppUtils appUtils;

    @Autowired
    private ApiUtils apiUtils;

    @ExceptionHandler({ MyNotFoundException.class, MyInvalidEntryException.class,
                        MyDuplicatedEntryException.class, MyAccessDeniedException.class })
    public ResponseEntity<?> handleCustomException(RuntimeException ex, WebRequest request,
                                                HttpServletRequest hrequest) {
        // Locale
        final Locale locale = resolver.resolveLocale(hrequest);

        // Process
        final String error = appUtils.getErrorMsg(ex, locale);

        final HttpStatus status = apiUtils.getErrorStatus(ex);

        return apiUtils.getResponse(hrequest, status, error, null, locale);
    }

    @ExceptionHandler
    protected ResponseEntity<?> handleAnyOtherException(RuntimeException ex, WebRequest request,
                                                HttpServletRequest hrequest) {
        // Locale
        final Locale locale = resolver.resolveLocale(hrequest);

        // Process
        final String error = appUtils.getErrorMsg(ex, locale);

        return apiUtils.getResponse(hrequest, HttpStatus.INTERNAL_SERVER_ERROR, error, null, locale);
    }
}
```

### 5.2.3. Implémenter la couche @RestController

- @RestController Profession

```

@RestController
@RequestMapping("/api/professions")
public class ProfessionRestController {

    @Autowired
    private ProfessionService service;

    @Autowired
    private ApiUtils utils;

    @Autowired
    private LocaleResolver resolver;

    @PostMapping
    public ResponseEntity<?> doCreate(@Valid @RequestBody ProfessionDTO entity, BindingResult result,
                                         HttpServletRequest request) {
        final Profession row = service.save(entity, result, resolver.resolveLocale(request));
        return utils.getResponse(request, HttpStatus.OK, null, row, resolver.resolveLocale(request));
    }

    @PutMapping("/{id}")
    public ResponseEntity<?> doUpdate(@PathVariable("id") int id,
                                         @Valid @RequestBody ProfessionDTO entity, BindingResult result,
                                         HttpServletRequest request) {
        entity.setId(id);
        final Profession row = service.save(entity, result, resolver.resolveLocale(request));
        return utils.getResponse(request, HttpStatus.OK, null, row, resolver.resolveLocale(request));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<?> doDelete(@PathVariable("id") int id, HttpServletRequest request) {
        final List<Profession> data = service.delete(id, resolver.resolveLocale(request));
        return utils.getResponse(request, HttpStatus.OK, null, data, resolver.resolveLocale(request));
    }

    @GetMapping("/{id}")
    public ResponseEntity<?> getRecordById(@PathVariable("id") int id, HttpServletRequest request) {
        return utils.getResponse(request, HttpStatus.OK, null, service.getRecordById(id),
                                 resolver.resolveLocale(request));
    }

    @GetMapping
    public ResponseEntity<?> getRecords(
            @RequestParam(value = "active", required = false, defaultValue = "true") boolean active,
            @RequestParam(value = "keyword", required = false, defaultValue = "") String keyword,
            HttpServletRequest request) {
        return utils.getResponse(request, HttpStatus.OK, null, service.getRecords(active, keyword),
                                 resolver.resolveLocale(request));
    }
}

```

- @RestController Formation
- @RestController Etudiant

#### 5.2.4. Installer Postman

- Reference Web : <https://www.postman.com/downloads/>

#### 5.2.5. Tester l'API

- Exécuter le module DemoUpcG3FasiApp02StarterWeb

- Tester la fonctionnalité Ajouter

The screenshot shows the Postman interface for testing a Spring Framework API. The request is a POST to `http://localhost:8080/api/formations`. The body is set to `JSON` and contains the following JSON payload:

```
1 {  
2     "description": "Spring Framework",  
3     "duree": 30,  
4     "notes": "Aucune"  
5 }
```

The response status is `200 OK` with a `447 ms` duration and `361 B` size. The response body is:

```
1 {  
2     "success": true,  
3     "status": 200,  
4     "path": "localhost/api/formations",  
5     "message": "Tache exécutée avec succès",  
6     "data": [  
7         {  
8             "id": 1,  
9             "description": "Spring Framework",  
10            "duree": 30,  
11            "notes": "Aucune",  
12            "actif": true  
13        }  
14    ]  
15 }
```

- Tester la fonctionnalité Modifier

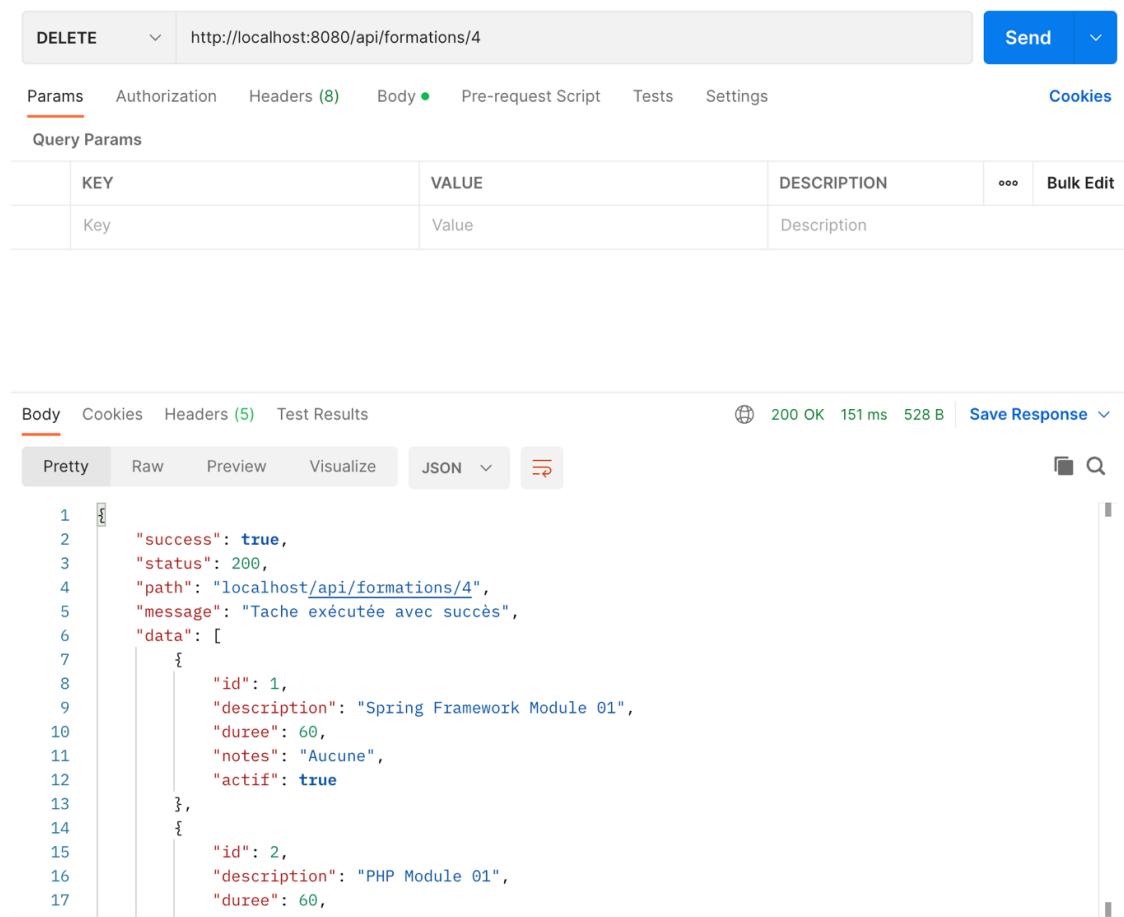
The screenshot shows the Postman interface for a PUT request to `http://localhost:8080/api/formations/1`. The request body is a JSON object:

```
1 {  
2   "description": "Spring Framework Module 01",  
3   "duree": 60,  
4   "notes": "Aucune"  
5 }
```

The response status is 200 OK, with a response time of 116 ms and a response size of 373 B. The response body is:

```
1 {  
2   "success": true,  
3   "status": 200,  
4   "path": "localhost/api/formations/1",  
5   "message": "Tache exécutée avec succès",  
6   "data": [  
7     {  
8       "id": 1,  
9       "description": "Spring Framework Module 01",  
10      "duree": 60,  
11      "notes": "Aucune",  
12      "actif": true  
13    }  
14  ]  
15 }
```

- Tester la fonctionnalité Supprimer



DELETE http://localhost:8080/api/formations/4 Send

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK 151 ms 528 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "success": true,
2 "status": 200,
3 "path": "localhost/api/formations/4",
4 "message": "Tache exécutée avec succès",
5 "data": [
6     {
7         "id": 1,
8         "description": "Spring Framework Module 01",
9         "duree": 60,
10        "notes": "Aucune",
11        "actif": true
12    },
13    {
14        "id": 2,
15        "description": "PHP Module 01",
16        "duree": 60,
```

- Tester la fonctionnalité Lire un enregistrement

GET http://localhost:8080/api/formations/1

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON 🔍

200 OK 30 ms 373 B Save Response

```
1 {
2   "success": true,
3   "status": 200,
4   "path": "localhost/api/formations/1",
5   "message": "Tache exécutée avec succès",
6   "data": [
7     {
8       "id": 1,
9       "description": "Spring Framework Module 01",
10      "duree": 60,
11      "notes": "Aucune",
12      "actif": true
13    }
14  ]
15 }
```

- Tester la fonctionnalité Lire tous les enregistrements

```

1 {
2   "success": true,
3   "status": 200,
4   "path": "localhost/api/formations/",
5   "message": "Tache ex\u00e9cut\u00e9e avec succ\u00e8s",
6   "data": [
7     {
8       "id": 1,
9       "description": "Spring Framework Module 01",
10      "duree": 60,
11      "notes": "Aucune",
12      "actif": true
13    },
14    {
15      "id": 2,
16      "description": "PHP Module 01",
17      "duree": 60,
18      "notes": "Aucune",
19      "actif": true
20    },
21    {
22      "id": 3
23    }
24  ]
25}
  
```

### 5.3. Travail Pratique

- Test les fonctionnalités CRUD à partir de Postman du projet App02.

## 6. SPRING SECURITY

### 6.1. Concepts

#### 6.1.1. Terminologie

- « Authentication » : Processus de vérification de l'identité d'un utilisateur partant du code utilisateur (username) et du password. C'est la réponse à la question « Qui êtes-vous ? ».
- « Autorisation » : Processus par lequel on vérifie si l'utilisateur a les droits nécessaires pour exécuter une tâche donnée. C'est la réponse à la question « Qu'est-ce que vous êtes autorisés à faire ? ».
- « Principal » : C'est l'utilisateur présentement authentifié.

- « Granted authority » : C'est la permission de l'utilisateur authentifié.
- « Role » : C'est l'ensemble de permissions de l'utilisateur authentifié.
- « SecurityFilterChain» : C'est la chaîne de filtres de sécurité mise en place aussitôt que la dépendance Spring Security est incluse dans le POM. Cette chaîne de filtres intercepte toutes les requêtes http, et chaque filtre joue un rôle bien déterminé.
- L'ordre d'exécution de filtres de sécurité est croissant partant de la valeur inférieure définie dans l'annotation @Order.
- Si besoin il y a d'implémenter de filtres personnalisés qui doivent être exécutés avant ceux imposés par Spring Security, il faut penser à définir la valeur @Order minimale dans le fichier application.properties : spring.security.filter.order=10, et ceci donne une plage libre de 10 valeurs (0-9).
- « AuthenticationManager» : C'est le coordonnateur qui gère les différentes interfaces d'authentification, dépendant du type de requête http.
- « AuthenticationProvider» : C'est l'interface qui authentifie les requêtes.
- « UserDetailsService » : C'est l'interface qui récupère les informations de l'utilisateur de la base de données.
- « AuthenticationEntryPoint » : C'est l'interface qui intercepte toutes les exceptions d'authentification.
- Spring Security offre deux façons de configurer l' « Authorization » : « URL-Based » , « Annotation-Based » et « Entity-Based ».
- Configuration URL-Based (URL-Level-Security)  

```
.antMatchers("/api/author/**").hasRole(Role.AUTHOR_ADMIN)
.antMatchers("/api/book/**").hasRole(Role.BOOK_ADMIN)
```
- Configuration Annotation-Based (Method-Level-Security)
  - @PreAuthorize : Avant exécution d'une méthode (l'annotation la plus utilisée)  

```
@PreAuthorize("hasRole('ADMIN')")
```

```
@PreAuthorize("hasAnyRole('ADMIN', 'DB-ADMIN')")
```

```
@PreAuthorize("hasAuthority('ROLE_ADMIN')")
```

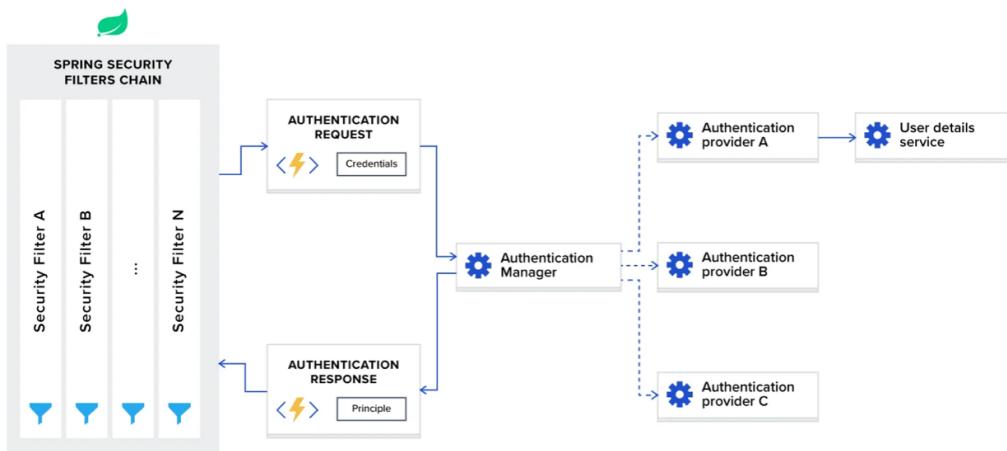
```
@PreAuthorize(GdocsRoleUtil.TACHES_COURRIERS_CREER)
Courrier addRecord(Courrier entity, Locale locale);
```

Note : En cas d'utilisation de hasAuthority, le rôle doit être précédé du préfix ROLE\_. Ce comportement peut être désactivé en ajouter cette @Bean dans le fichier @Configuration

```
@Bean
GrantedAuthorityDefaults grantedAuthorityDefaults() {
    return new GrantedAuthorityDefaults(""); // Remove the ROLE_ prefix
}
```

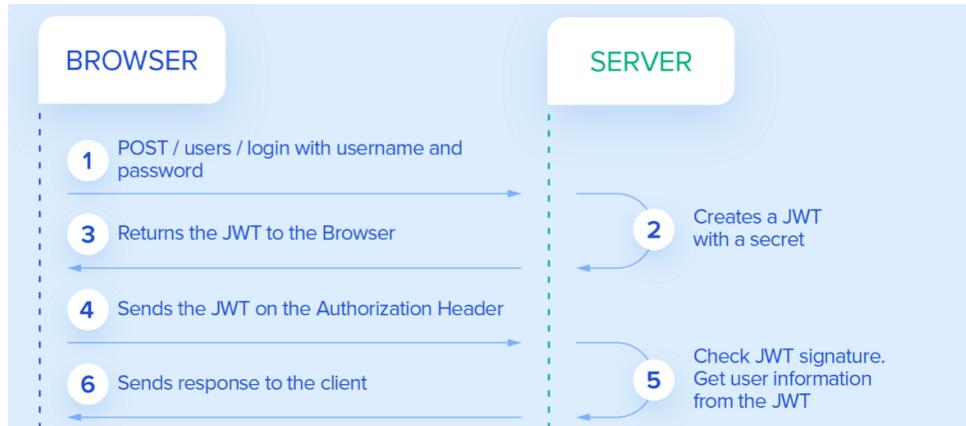
- o @PostAuthorize : Après exécution d'une méthode
  - o @PreFilter
  - o @PostFilter
  - o @Secured
  - o @RolesAllowed
- L'annotation @EnableGlobalMethodSecurity(prePostEnabled = true) sur une classe @Configuration permet de rendre effective la Method-Level-Security au travers des annotations @PreAuthorize et @PostAuthorize.

#### 6.1.2. Architecture



#### 6.1.3. Spring Security avec JWT ( JSON Web Token)

- Architecture



- Le JWT obtenu est envoyé dans le « Authorization Header » de chaque requête  
**Authorization: Bearer <token>**

## 6.2. Préparer l'environnement

### 6.2.1. Dépendances

- Insérer les dépendances Spring Security et JWT dans le POM du module Data

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
```

### 6.2.2. Classe User : @Entity, @Repository, @Service

- Créer une classe SecurityRoleUtils dans le module Data/package  
edu.upc.config.security

```
public class SecurityRoleUtils {

    public static final String ROLE_PREFIX = "ROLE_";
    public static final String ROLE_SEPARATOR = ":";

    public static final String ADMIN = "ADMIN";
    public static final String USER = "USER";

    public static final String TASK_ROLE_ADMIN = "hasRole('ADMIN')";
    public static final String TASK_ROLE_USER = "hasRole('USER')";
    public static final String TASK_ROLE_ANY = "hasAnyRole('ADMIN', 'USER')";

}
```

- Créer une classe @Entity User dans le module Lib

```
@Entity
@Table(name = "admin_users")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class User implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @NotEmpty
    @Size(min = 1, max = 15)
    private String code;

    @NotEmpty
    @Size(min = 1, max = 45)
    private String description;

    @NotEmpty
    @Email
    @Size(min = 1, max = 45)
    private String email;

    @NotEmpty
    @Size(min = 1, max = 45)
    private String telephone;

    @NotNull
    @Builder.Default
    private boolean actif = true;

    @Column(updatable = false)
    @NotNull
    @Builder.Default
    private boolean system=false;

    @JsonProperty(access = Access.WRITE_ONLY)
    private String pwd;

    @NotEmpty
    private String roles;

}
```

- Créer une classe UserDTO dans le module Lib

```
@NoArgsConstructor
@NoArgsConstructor
@ToString
@EqualsAndHashCode
@Builder
public class UserDTO implements Serializable {

    private static final long serialVersionUID = 1L;

    @NotNull(message = "{form.id}")
    @Min(value = 0, message = "{form.id}")
    private int id;

    @NotEmpty(message = "{form.code}")
    @Size(min = 1, max = 15, message = "{form.code}")
    private String code;

    @NotEmpty(message = "{form.description}")
    @Size(min = 1, max = 45, message = "{form.description}")
    private String description;

    @NotEmpty(message = "{form.email}")
    @Size(min = 1, max = 45, message = "{form.email}")
    private String email;

    @NotEmpty(message = "{form.phone}")
    @Size(min = 1, max = 45, message = "{form.phone}")
    private String telephone;

    @NotNull
    @Builder.Default
    private boolean actif = true;

}
```

- Créer un @Component UserMapper dans le module Lib

```
@Component
public class UserMapper {

    public UserDTO mapToDTO(User entity) {
        return UserDTO.builder()
            .id(entity.getId())
            .code(entity.getCode())
            .description(entity.getDescription())
            .email(entity.getEmail())
            .telephone(entity.getTelephone())
            .actif(entity.isActif())
            .build();
    }

    public User mapToEntity(UserDTO dto) {
        return User.builder()
            .id(dto.getId())
            .code(dto.getCode())
            .description(dto.getDescription())
            .email(dto.getEmail())
            .telephone(dto.getTelephone())
            .actif(dto.isActif())
            .build();
    }
}
```

- Implémenter la couche @Repository User

```
@Repository
public interface UserRepo extends JpaRepository<User, Integer> { }
```

```
public interface UserRepoCustom {

    boolean isCodeUnique(int id, String code);

    boolean isDescrUnique(int id, String description);

    boolean isSystemUser(int id);

    User getRecordById(int id);

    List<User> getRecords(boolean actif, String keyword);

    User getRecord4Auth(String username);

}

@Repository
public class UserRepoImpl implements UserRepoCustom {

    @Autowired
    private JdbcTemplate jdbcTempl;

    @Autowired
    private AppUtils utils;

    private static final ResultSetExtractorImpl<User> rsExtr = JdbcTemplateMapperFactory.newInstance()
        .addKeys("id")
        .ignorePropertyNotFound()
        .newResultSetExtractor(User.class);

    /**
     * @Override
     */
    public boolean isCodeUnique(int id, String description) {
        final String sql = "SELECT NOT EXISTS("
            + "SELECT id FROM admin_users WHERE id=>? AND code=? LIMIT 1"
            + ")";

        final Object[] params = new Object[] { id, description };

        return jdbcTempl.queryForObject(sql, Boolean.class, params);
    }

    /**
     * @Override
     */
    public boolean isDescrUnique(int id, String description) {
        final String sql = "SELECT NOT EXISTS("
            + "SELECT id FROM admin_users WHERE id=>? AND description=? LIMIT 1"
            + ")";

        final Object[] params = new Object[] { id, description };

        return jdbcTempl.queryForObject(sql, Boolean.class, params);
    }

    /**
     * @Override
     */
    public boolean isSystemUser(int id) {
        final String sql = "SELECT EXISTS("
            + "SELECT id FROM admin_users WHERE id=? AND system=1"
            + ")";

        final Object[] params = new Object[] { id };

        return jdbcTempl.queryForObject(sql, Boolean.class, params);
    }
}
```

```

@Override
public User getRecordById(int id) {
    final String sql = "SELECT * FROM admin_users WHERE id=?";
    final Object[] params = new Object[] { id };
    return utils.getObjectFromList(jdbcTempl.query(sql, rsExtr, params));
}

@Override
public List<User> getRecords(boolean actif, String keyword) {
    final String _stmt = "SELECT * FROM admin_users ";
    final String _order = " ORDER BY id";
    String sql = _stmt.concat("WHERE actif=? AND (code LIKE ? OR description LIKE ?)").concat(_order);
    final String _keyword = utils.getFilterKeyword(keyword);
    Object[] params = new Object[] { actif, _keyword, _keyword };
    if (StringUtils.isEmpty(keyword)) {
        sql = _stmt.concat("WHERE actif=?").concat(_order);
        params = new Object[] { actif };
    }
    return jdbcTempl.query(sql, rsExtr, params);
}

@Override
public User getRecord4Auth(String username) {
    final String sql = "SELECT * FROM admin_users WHERE code=?";
    final Object[] params = new Object[] { username };
    return utils.getObjectFromList(jdbcTempl.query(sql, rsExtr, params));
}

```

- Implémenter la couche @Service User

```

public interface UserService {

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_ADMIN)
    User save(UserDTO dto, BindingResult result, Locale locale);

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_ADMIN)
    List<User> delete(int id, Locale locale);

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_ANY)
    User getRecordById(int id);

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_ADMIN)
    List<User> getRecords(boolean actif, String keyword);

}

```

```
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private PasswordEncoder encoder;

    @Autowired
    private UserRepo repo;

    @Autowired
    private UserRepoCustom repoC;

    @Autowired
    private UserMapper mapper;

    @Autowired
    private AppUtils utils;

    @Autowired
    private MessageSource msgSrc;

    @Transactional(rollbackFor = Exception.class)
    @Override
    public User save(UserDTO dto, BindingResult result, Locale locale) {
        //
        if (result.hasErrors()) {
            throw new MyInvalidEntryException(utils.getErrors(result));
        } else if (!repoC.isCodeUnique(dto.getId(), dto.getCode())) {
            throw new MyDuplicatedEntryException(dto.getCode());
        } else if (!repoC.isDescrUnique(dto.getId(), dto.getDescription())) {
            throw new MyDuplicatedEntryException(dto.getDescription());
        }

        final User user = mapper.mapToEntity(dto);
        user.setPwd(encoder.encode(user.getCode())); //Encode
        user.setRoles(RoleUtils.USER);

        return repo.saveAndFlush(user);
    }
}
```

```

@Transactional(rollbackFor = Exception.class)
@Override
public List<User> delete(int id, Locale locale) {
    if (repoC.isSystemUser(id)) {
        throw new MyAccessDeniedException(msgSrc.getMessage("form.systemUser", null, locale));
    }
    repo.deleteById(id);
    repo.flush();
    return this.getRecords(true, null);
}

@Transactional(readOnly = true)
@Override
public User getRecordById(int id) {
    return repoC.getRecordById(id);
}

@Transactional(readOnly = true)
@Override
public List<User> getRecords(boolean actif, String keyword) {
    return repoC.getRecords(actif, keyword);
}
}

```

#### 6.2.3.Sécuriser les @Services : Formation, Profession, Etudiant, EtudiantFormation

- Formation

```

public interface FormationService {

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_USER)
    Formation save(FormationDTO dto, BindingResult result, Locale locale);

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_USER)
    List<Formation> delete(int id, Locale locale);

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_ANY)
    Formation getRecordById(int id);

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_ANY)
    List<Formation> getRecords(boolean actif, String keyword);
}

```

- Profession

```

public interface ProfessionService {

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_USER)
    Profession save(ProfessionDTO dto, BindingResult result, Locale locale);

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_USER)
    List<Profession> delete(int id, Locale locale);

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_ANY)
    Profession getRecordById(int id);

    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_ANY)
    List<Profession> getRecords(boolean actif, String keyword);
}

```

- Etudiant

```
public interface EtudiantService {  
  
    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_USER)  
    Etudiant save(EtudiantDTO dto, BindingResult result, Locale locale);  
  
    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_USER)  
    List<Etudiant> delete(int id, Locale locale);  
  
    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_ANY)  
    Etudiant getRecordById(int id);  
  
    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_ANY)  
    List<Etudiant> getRecords(boolean actif, String keyword);  
}
```

- EtudiantFormation

```
public interface EtudiantFormationService {  
  
    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_USER)  
    Etudiant save(EtudiantFormationDTO dto, BindingResult result, Locale locale);  
  
    @PreAuthorize(SecurityRoleUtils.TASK_ROLE_USER)  
    Etudiant delete(int id, Locale locale);  
}
```

## 6.3. Pratique : Préalables

### 6.3.1. Configurer la sécurité dans le module Data

- Créer la @Configuration SecurityConfigCore

```
@Configuration  
public class SecurityConfigCore {  
  
    @Bean  
    public PasswordEncoder encoder() {  
        return new BCryptPasswordEncoder();  
    }  
  
    @Bean(name = BeanIds.AUTHENTICATION_MANAGER)  
    public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {  
        return config.getAuthenticationManager();  
    }  
}
```

- Créer le @Service SecurityUserDetailsService

```
@Service
@Slf4j
public class SecurityUserDetailsService implements UserDetailsService {

    @Autowired
    private UserRepoCustom userRepoC;

    @Autowired
    private MessageSource messageSource;

    @Autowired
    private HttpServletRequest request;

    @Autowired
    private LocaleResolver localeResolver;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        final Locale locale = localeResolver.resolveLocale(request);
        log.info("Authentication: ".concat("username=" + username + "...Pulling AuthUserDetails"));
        final edu.upc.models.User user = userRepoC.getRecord4Auth(username);
        if (user == null || StringUtils.isEmpty(username)) {
            throw new UsernameNotFoundException(messageSource.getMessage("error.invalidUser", null, locale));
        }
        final List<GrantedAuthority> authorities = new ArrayList<>();
        if (StringUtils.isNotEmpty(user.getRoles())) {
            List<String> _roles = Arrays.asList(user.getRoles().split(SecurityRoleUtils.ROLE_SEPARATOR));
            _roles.forEach((role) -> {
                authorities.add(new SimpleGrantedAuthority(SecurityRoleUtils.ROLE_PREFIX + StringUtils.trim(role)));
            });
        }
        return new org.springframework.security.core.userdetails.User(user.getCode(), user.getPwd(), authorities);
    }
}
```

- Créer le @Component SecurityAuthEntryPoint

```
@Component
@Slf4j
public class SecurityAuthEntryPoint implements AuthenticationEntryPoint { // Called When authentication fails

    @Autowired
    private LocaleResolver localeResolver;

    @Autowired
    private MessageSource messageSource;

    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response,
        AuthenticationException authException) throws IOException, ServletException {
        response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        response.setContentType(MediaType.APPLICATION_JSON_VALUE);

        final String url = request.getRequestURI();
        final Locale locale = localeResolver.resolveLocale(request);
        final String error = messageSource.getMessage("error.authenticationFailed", null, locale);

        log.error("App Unauthorized error: " + authException.getMessage());

        if (StringUtils.isNotEmpty(url) && url.contains("/api")) {
            final MyResponse entity = new MyResponse();
            entity.setSuccess(false);
            entity.setStatus(HttpStatus.UNAUTHORIZED.value());
            entity.setPath(request.getRequestURI());
            entity.setMessage(error);
            entity.setData(null);

            final OutputStream out = response.getOutputStream();

            final ObjectMapper mapper = new ObjectMapper();
            mapper.writeValue(out, entity);
            out.flush();
        } else {
            response.sendError(HttpServletResponse.SC_UNAUTHORIZED, error);
        }
    }
}
```

- Créer le @Component SecurityJwtUtils

```
@Component
@Slf4j
public class SecurityJwtUtils {

    String JWT_SECRET = "20G3FasiUpc23";
    final long JWT_TOKEN_VALIDITY = (60 * 60 * 12) * 1000; // Milliseconds 12H

    final Clock clock = DefaultClock.INSTANCE;

    @PostConstruct
    protected void init() {
        JWT_SECRET = Base64.getEncoder().encodeToString(JWT_SECRET.getBytes());
    }

    // HELPER
    private Date getExpirationDate(Date date) {
        return new Date(date.getTime() + JWT_TOKEN_VALIDITY);
    }

    private Claims getAllClaims(String token) {
        return Jwts.parser().setSigningKey(JWT_SECRET).parseClaimsJws(token).getBody();
    }

    // PROCESS
    public String createToken(AuthenticationManager authMgr, String username, String password,
        Locale locale) throws Exception {
        // Auth
        try {
            log.info("AuthCreateToken " + username);

            Authentication auth = authMgr.authenticate(new UsernamePasswordAuthenticationToken(username, password));
            SecurityContextHolder.getContext().setAuthentication(auth); //

            // Token
            final Date date = clock.now();

            Map<String, Object> claims = new HashMap<>();

            return Jwts.builder().setClaims(claims).setSubject(auth.getName()).setIssuedAt(date)
                .setExpiration(this.getExpirationDate(date)).signWith(SignatureAlgorithm.HS512, JWT_SECRET)
                .compact();
        } catch (Exception ex) { //org.springframework.security.authentication.BadCredentialsException:
            throw new MyInvalidCredsException(ex.getMessage());
        }
    }
}
```

```
public String refreshToken(String token) throws Exception {
    final Date date = clock.now();

    final Claims claims = this.getAllClaims(token);
    claims.setIssuedAt(date);
    claims.setExpiration(this.getExpirationDate(date));

    return Jwts.builder().setClaims(claims).signWith(SignatureAlgorithm.HS512, JWT_SECRET).compact();
}

public String resolveToken(HttpServletRequest request) throws Exception {
    String bearerToken = request.getHeader("Authorization");
    if (StringUtils.isNotEmpty(bearerToken) && bearerToken.startsWith("Bearer ")) {
        return bearerToken.substring(7);
    }
    return null;
}

public boolean validateToken(String token) throws Exception {
    Jwts.parser().setSigningKey(JWT_SECRET).parseClaimsJws(token);
    return true;
}

public <T> T getClaim(String token, Function<Claims, T> claimsResolver) throws Exception {
    final Claims claims = getAllClaims(token);
    return claimsResolver.apply(claims);
}

public String getUsername(String token) throws Exception {
    return this.getClaim(token, Claims::getSubject);
}

public Date getIssuedAtDate(String token) throws Exception {
    return this.getClaim(token, Claims::getIssuedAt);
}

public Date getExpirationDate(String token) throws Exception {
    return this.getClaim(token, Claims::getExpiration);
}

public Boolean isTokenExpired(String token) throws Exception {
    final Date expiration = getExpirationDate(token);
    return expiration.before(clock.now());
}
}
```

## 6.4. Pratique : Application Web

### 6.4.1. Configurer la sécurité dans le module Web

- Créer la @Configuration SecurityConfigWeb

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfigWeb {

    @Autowired
    private SecurityUserDetailsService userDetailsService;

    @Autowired
    private SecurityAuthEntryPoint entryPoint;

    @Autowired
    private PasswordEncoder encoder;

    @Bean
    public WebSecurityCustomizer webSecurityCustomizer() {
        return (web) -> web.ignoring().antMatchers("/static/**", "/templates/**");
    }

    @Bean
    @Order(2)
    public SecurityFilterChain filterChainWeb(HttpSecurity http) throws Exception {
        http.csrf().and().authorizeRequests(auth -> {
            auth.antMatchers(
                "/login", "/login/**", "/logout", "/forgot-password",
                "/static/**", "/templates/**"
            ).permitAll();
            auth.anyRequest().authenticated();
        }).exceptionHandling(ex -> {
            ex.authenticationEntryPoint(entryPoint);
        }).authenticationProvider(authProvider());

        http.formLogin().LoginPage("/login").permitAll()
            .successForwardUrl("/")
            .failureUrl("/login?error")
            .usernameParameter("username").passwordParameter("password");

        http.logout().logoutSuccessUrl("/login?logout").deleteCookies("remove")
            .invalidateHttpSession(true)
            .clearAuthentication(true).permitAll();

        return http.build();
    }

    private DaoAuthenticationProvider authProvider() {
        DaoAuthenticationProvider auth = new DaoAuthenticationProvider();
        auth.setUserDetailsService(userDetailsService);
        auth.setPasswordEncoder(encoder);
        return auth;
    }
}
```

- Mettre à jour le @Controller AppController

```
@Controller
public class AppController {

    @RequestMapping("/")
    public String doHome() {
        return "index";
    }

    @RequestMapping("/login")
    public String doLogin(HttpServletRequest request) {
        return "login";
    }

    @RequestMapping("/logout")
    public String doLogout(@RequestParam(value="success", required=false, defaultValue="false") boolean success,
        HttpServletRequest request, HttpServletResponse response) {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if (auth != null) {
            new SecurityContextLogoutHandler().logout(request, response, auth);
        }
        return success ? "redirect:/login?success" : "redirect:/login?logout";
    }
}
```

#### 6.4.2. Mettre à jour les fichiers .html

- Mettre un lien de déconnexion dans le fragment /templates/fragments/fragment-header

```
<div th:fragment="header">

    <div>
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
            <div class="container-fluid">
                <a class="navbar-brand" th:href="@{@{${@webUrls.home}}}">App02</a>
                <button class="navbar-toggler" type="button"
                    data-bs-toggle="collapse" data-bs-target="#navbarScroll"
                    aria-controls="navbarScroll" aria-expanded="false"
                    aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarScroll">
                    <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll"
                        style="-bs-scroll-height: 100px;">
                        <li class="nav-item"><a class="nav-link active"
                            aria-current="page" th:href="@{@{${@webUrls.home}}}"
                            th:text="#{form.home}">Home</a></li>

                        <li class="nav-item dropdown"><a
                            class="nav-link dropdown-toggle" href="#"
                            id="navbarDropdown" role="button"
                            data-bs-toggle="dropdown" aria-expanded="false"
                            th:text="#{form.file}"></a>
                            <ul class="dropdown-menu"
                                aria-labelledby="navbarDropdown">
                                <li><a class="dropdown-item"
                                    th:href="@{@{${@webUrls.professions_view}}}"
                                    th:text="#{form.occupations}"></a></li>
                                <li><a class="dropdown-item"
                                    th:href="@{@{${@webUrls.formations_view}}}"
                                    th:text="#{form.trainings}"></a></li>
                                <li><hr class="dropdown-divider"></li>
                                <li><a class="dropdown-item"
                                    th:href="@{@{${@webUrls.etudiants_view}}}"
                                    th:text="#{form.students}"></a></li>
                            </ul></li>
                        </ul>
                    <div class="text-light">
                        <a class="link-light" style="text-decoration: none;">
                            th:href="@{?locale=en}" th:text="#{form.english}"</a> | <a
                            class="link-light" style="text-decoration: none;">
                            th:href="@{?locale=fr}" th:text="#{form.french}"</a>
                        <br/>
                        <a class="link-light" style="text-decoration: none;">
                            th:href="@{@{${@webUrls.logout}}}" th:text="#{form.logout}"</a>
                    </div>
                </div>
            </div>
        </nav>
    </div>
</div>
```

- Créer un fichier login.html dans le dossier « templates »

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
<meta charset="utf-8" />
<meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no"
      name="viewport" />
<meta content="" name="description" />
<meta content="" name="author" />

<title layout:title-pattern="$LAYOUT_TITLE | $CONTENT_TITLE"></title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJ0Z"
      crossorigin="anonymous">

<link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.min.css"
      integrity="sha512-xH60CQfDfPvWxH6dHkq0oXtqfjzHdIuXnqB8lWVgkqfPvWxH6dHkq0oXtqfjzHdIuXnqB8lWVgkqf"
      crossorigin="anonymous" referrerPolicy="no-referrer" />

<link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/jquery-confirm/3.3.2/jquery-confirm.min.css" />

<link rel="stylesheet" type="text/css" th:href="@{/css/app.css}" />

</head>
<body>

    <div class="container">

        <div class="row align-items-center" style="height: 100vh;">
            <div class="mx-auto col-3" style="background-color: #d9d9d9; padding: 40px;">
                <h2 class="text-center">MyApp02</h2>
                <h3 class="text-center">Login</h3>
                <div>
                    <form id="myform" th:action="@{@{webUrls.login}}" method="post"
                          enctype="multipart/form-data">
                        <!-- FIELDS -->
                        <div class="row mb-3">
                            <label for="username" th:text="#{form.username}"></label> <input
                                id="username" name="username" type="text" class="form-control" />
                        </div>
                        <div class="row mb-3">
                            <label for="password" th:text="#{form.password}"></label> <input
                                id="password" name="password" type="password"
                                class="form-control" />
                        </div>
                        <br />
                        <div class="row mb-3">
                            <button id="btnLogin" type="submit" class="btn btn-primary"
                                   th:text="#{form.login}"></button>
                        </div>
                        <!-- HIDDEN -->
                        <input type="hidden" th:name="${_csrf.parameterName}"
                               th:value="${_csrf.token}" />
                    </form>
                <!-- MESSAGE -->
                <div th:if="${param.logout != null}" class="row mb-3">
                    <div class="alert alert-success">
                        <span th:text="#{message.successfulLogout}"></span>
                    </div>
                </div>
                <div th:if="${param.error != null}" class="row mb-3">
                    <div class="alert alert-danger">
                        <span th:text="#{error.invalidCredentials}"></span>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
```

```
<script
  src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.7/dist/umd/popper.min.js"
  integrity="sha384-zYPOMqeu1DAVkHiLqWBUTcbYfZ8osu1Nd6Z89ify25QV9guujx43ITvfi12/QExE"
  crossorigin="anonymous"></script>

<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-ENjd04Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNs7NTKfAdVQSZe"
  crossorigin="anonymous"></script>

</body>
```

- Mettre à jour les fichiers Bundle si nécessaire

#### 6.4.3. Pratique

- Lancer le module Starter Web
- Taper dans le navigateur : <http://localhost:8081> (dépendant votre port de serveur web).

### 6.5. Pratique : Application API

#### 6.5.1. Configurer la sécurité dans le module API

- Créer le @Component SecurityJwtAuthFilter

```
@Component
@Slf4j
public class SecurityJwtAuthFilter extends OncePerRequestFilter {

    @Autowired
    private SecurityUserDetailsService userService;

    @Autowired
    private SecurityJwtUtils jwtUtils;

    @Autowired
    private MessageSource messageSource;

    @Autowired
    private LocaleResolver localeResolver;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
        throws ServletException, IOException {
        final Locale locale = localeResolver.resolveLocale(request);

        try {
            final String token = jwtUtils.resolveToken(request);

            if (StringUtils.isNotEmpty(token) && jwtUtils.validateToken(token)) {
                final String username = jwtUtils.getUsername(token);
                final String url = request.getRequestURI();

                log.info("Auth Filter JWT User: " + username + " - " + url);

                final UserDetails userDetails = userService.loadUserByUsername(username);
                final UsernamePasswordAuthenticationToken auth = new UsernamePasswordAuthenticationToken(
                    userDetails, null, userDetails.getAuthorities());
                auth.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

                SecurityContextHolder.getContext().setAuthentication(auth);
            }
        } catch (ExpiredJwtException ex) {
            throw new MyAccessDeniedException(messageSource.getMessage("error.tokenExpired", null, locale));
        } catch (Exception ex) { // IMPORTANT To guarantee User Not Authenticated
            SecurityContextHolder.clearContext();
            response.sendError(HttpStatus.FORBIDDEN.value(), ex.getMessage());
            return;
        }
    }

    filterChain.doFilter(request, response);
}
}
```

- Créer la @Configuration SecurityConfigApi

```

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfigApi {

    @Autowired
    private PasswordEncoder encoder;

    @Autowired
    private SecurityJwtAuthFilter jwtFilter;

    @Autowired
    private SecurityAuthEntryPoint entryPoint;

    @Autowired
    private SecurityUserDetailsService userDetailsService;

    @Bean
    @Order(1)
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.antMatcher("/api/**").sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
            .csrf().disable().authorizeRequests(auth -> {
                auth.antMatchers("/api/unsecured/**").permitAll();
                auth.antMatchers(HttpMethod.POST, "/api/auth/**").permitAll();
                auth.anyRequest().authenticated();
            }).exceptionHandling(ex -> {
                ex.authenticationEntryPoint(entryPoint);
            })
            .authenticationProvider(authProvider())
            .addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }

    private DaoAuthenticationProvider authProvider() {
        DaoAuthenticationProvider auth = new DaoAuthenticationProvider();
        auth.setUserDetailsService(userDetailsService);
        auth.setPasswordEncoder(encoder);
        return auth;
    }
}

```

#### 6.5.2. Implémenter @RestController et @Service : Authentification

- Créer AuthDTO dans le module Lib

```

@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
public class AuthDTO implements Serializable {

    private static final long serialVersionUID = 1L;

    @JsonInclude(JsonInclude.Include.NON_NULL)
    @NotEmpty(message = "{form.username}")
    private String username;

    @JsonInclude(JsonInclude.Include.NON_NULL)
    @NotEmpty(message = "{form.password}")
    private String password;

    @JsonInclude(JsonInclude.Include.NON_NULL)
    private String token;

    public AuthDTO(String token) {
        super();
        this.token = token;
    }
}

```

- Créer le @Service AppRestService dans le module Data

```

public interface AppRestService {

    String doAuth(AuthDTO dto, BindingResult result, Locale locale) throws Exception;
}

@Service
public class AppRestServiceImpl implements AppRestService {

    @Autowired
    @Qualifier(BeanIds.AUTHENTICATION_MANAGER)
    private AuthenticationManager authMgr;

    @Autowired
    private AppUtils appUtils;

    @Autowired
    private SecurityJwtUtils jwtUtils;

    @Override
    public String doAuth(AuthDTO dto, BindingResult result, Locale locale) throws Exception {
        if (result.hasErrors()) {
            throw new MyInvalidEntryException(appUtils.getErrors(result));
        }
        return jwtUtils.createToken(authMgr, dto.getUsername(), dto.getPassword(), locale);
    }
}

```

- Créer le @RestController AppRestController dans le module API

```

@RestController
@RequestMapping("/api")
public class AppRestController {

    @Autowired
    private AppRestService service;

    @Autowired
    private ApiUtils appUtils;

    @Autowired
    private LocaleResolver localeResolver;

    @Autowired
    private MessageSource msgSrc;

    @PostMapping("/auth")
    public ResponseEntity<?> doAuth( // Used4Jwt
        @Valid @RequestBody AuthDTO entity,
        BindingResult result, HttpServletRequest request) throws Exception {
        // Locale
        final Locale locale = localeResolver.resolveLocale(request);

        //
        final String token = service.doAuth(entity, result, locale);
        if (StringUtils.isEmpty(token)) {
            throw new MyInvalidEntryException(msgSrc.getMessage("form.token", null, locale));
        }
        return appUtils.getResponse(request, HttpStatus.OK, null, new AuthDTO(token), locale);
    }
}

```

- Mettre à jour les fichiers Bundle si nécessaire

### 6.5.3. Pratique

- Lancer le module Starter Api
- Obtenir un token

POST http://localhost:8080/api/auth

Body (JSON)

```

1 ...
2   ...
3     "username": "admin",
4     "password": "admin"
5 ...

```

Response:

```

1 ...
2   "success": true,
3   "status": 200,
4   "path": "localhost/api/auth",
5   "message": "Tache ex\u00e9cut\u00e9e avec succ\u00e8s",
6   "data": [
7     {
8       "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWlIiJhZG1pbisImV4cCI6MTY4MjA4NDAxMSwiawF0IjoxNjgyMDQwODExfQ.1IImMsM9dT0AGCHyRN9giHMUczypri-P71g0uZ2Z_rtTbafLhRxCqD54PMYLNC9s7aqCkK8fOD0AuZzmzAG6M2A"
9     }
10   ]
11

```

- R\u00e9cup\u00e9rer une ressource s\u00e9curis\u00e9e

GET http://localhost:8080/api/formations

Authorization (Bearer Token)

Response:

```

1 ...
2   "success": true,
3   "status": 200,
4   "path": "localhost/api/formations",
5   "message": "Tache ex\u00e9cut\u00e9e avec succ\u00e8s",
6   "data": [
7     {
8       "id": 1,
9       "description": "Spring Framework Module 01",
10      "duree": 60,
11      "notes": "Aucune",
12      "actif": true
13    },
14    {
15      "id": 2,
16      "description": "PHP Module 01",
17      "duree": 60,
18      "notes": "Aucune",
19    }
20  ]
21

```

- Cr\u00e9er une Profession alors que l'acc\u00e8s est interdit

```

1
2   ...
3     "description": "Operateur economique"
4
5
6
7
  
```

Status: 500 Internal Server Error Time: 55 ms Size: 455 B Save Response

- Créer un Utilisateur (Mot de passe par défaut est le code utilisateur), tout en se rassurant que le token est inséré dans le champ Onglet Autorisation/Type Bearer Token/Token

```

1
2   ...
3     "code": "omer",
4     "description": "Omer Pitou Ntumba",
5     "email": "ntumbakatuala@gmail.com",
6     "telephone": "243814443043",
7     "actif": true
  
```

Status: 200 OK Time: 333 ms Size: 616 B Save Response

- Obtenir un token avec les informations de l'utilisateur créé

The screenshot shows a Postman interface with a POST request to `http://localhost:8080/api/auth`. The request body is set to `JSON` and contains the following JSON:

```
1 {
2   "username": "omer",
3   "password": "omer"
4 }
```

The response status is `200 OK` with a response time of `247 ms` and a size of `639 B`. The response body is:1 {
2 "success": true,
3 "status": 200,
4 "path": "localhost/api/auth",
5 "message": "Tache exécutée avec succès",
6 "data": [
7 {
8 "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWJiOiJvbWVyIiwidXhwIjoxNjgyMDg3NDIzMjIyYXQiojE2ODIwNDQyMjN9.DXIuhJjlAdU\_ZL4x1mmN2w4EPnORWX9htPOX89F55Fga0t3bgjeOA0LGn0gcH\_68E82rgtXFPeWQ4RmeAIVmww"
9 }
10 ]
11 }

## 7. DEPLOYER UNE APPLICATION

- 7.1. Concepts
- 7.2. Pratique

## 8. EXTRA