TP3 : SpringBoot et SpringMVC

Cet TP permet de voir certains aspects du starter-web :

- Environnement Spring MVC
- Organisation
  - des ressources statiques
  - des gabarits HTML (Thymeleaf) et la stratégie de résolution de vues

3.1 Application Web et Spring MVC

Ajout des dépendances

Sur le projet précédent, ajouter les starters suivants :

- web
- thymeleaf
- dev-tools

Récupérer les sources fournis (répertoire *static* et *templates* à placer dans /*src/main/resources*)

Configuration

Ajouter une configuration MVC déclarant les viewControleurs, permettant d'accéder aux pages présents dans le répertoire templates : *home.html* et *documents.html*

Contrôleur

Implémenter *MemberController* qui aura pour caractéristique :

- De répondre à une URL GET qui routera vers la page de ***login*** présent dans templates
- De répondre à l'URL POST de la page de login, et vérifier les données encapsulées dans la classe DTO User.
  - Si les données sont correctes, positionner un objet *Member* en session et rediriger vers la page ***documents***
  - Sinon, positionner une erreur

Webjar

Ajouter dans pom.xml la dépendance suivante :

```
<dependency>
     <groupId>org.webjars</groupId>
     <artifactId>bootstrap</artifactId>
     <version>3.3.7-1</version>
```

```
</dependency>
```

Effectuer un ***mvn install*** et recharger la page de login

3.2 API REST

3.2.1 Contrôleurs

Créer un contrôleur REST *MemberRestController* permettant de :

- D'effectuer toutes les méthodes CRUD sur un membre
- Afficher les membres contenant un chaîne particulière

Créer un contrôleur REST *DocumentRestController* permettant de :

- Récupérer tous les documents d'un membre donné
- D'ajouter un document à un membre

Certaines méthodes pourront envoyer des exceptions métier « *MemberNotFoundException* », « *DocumentNotFoundException* »

Ne pas sérialiser la liste des documents d'un membre

3.2.2 Configuration

Configurer le cors

Ajouter un *ControllerAdvice* permettant de centraliser la gestion des exceptions *MemberNotFoundException*

3.2.3 OpenAPI et Swagger

Ajouter les dépendances Swagger dans *pom.xml*

Accéder à la description de notre api REST (*http://server:port/swagger-ui.html*)

Ajouter des annotations OpenAPI pour parfaire la documentation

Faire en sorte que la description OpenAPI ne soit pas disponible dans un profil ***prod***.

3.2 Appels REST

Créer un autre projet qui offre une ressource REST permettant de renvoyer une réponse JSON contenant un Member et tous ces documents associés.
Pour cela, le nouveau projet effectuera des appels REST vers l'application précédente