TP 2 – Premiers web-services

1 Mon premier web-service

1.1 Création du projet

Sous IntellIJ,

1. **File** → **New Project**
2. Choisissez **Spring Intialzr**
3. Renseignez les champs comme suit :
   (a) Name : *monPremierWebService*
   (b) Type : *Maven*
   (c) Group : *fr.orleans.m1miage.wsi*
   (d) Artifact : *mon-premier-webservice*
   (e) JDK : Choisissez une version supérieure ou égale à 17
   (f) Java : Choisissez une version supérieure ou égale à 17
   (g) Packaging : *jar*
4. dans la section *Web*, cochez uniquement *Spring Web*
5. Validez la création avec Finish

Votre projet est prêt.

1.2 Création du modèle

1. Créez un package modele au même niveau que la classe principale
2. Créez les classes ci-dessous dans ce package :

```
 @Component
public class FacadePromotion {

    Map<String,Etudiant> etudiants;

    public FacadePromotion() {
        this.etudiants = new HashMap<>();
    }

    public String enregistrerEtudiant(String nom, String prenom, String adresse) {
        Etudiant etudiant = new Etudiant(nom,prenom,adresse);
        this.etudiants.put(etudiant.getNumeroEtudiant(),etudiant);
        return etudiant.getNumeroEtudiant();
    }

    public Collection<Etudiant> getEtudiants() {
        return this.etudiants.values();
    }

    public Etudiant getEtudiantById(String numeroEtudiant) throws EtudiantInexistantException {
        if (this.etudiants.containsKey(numeroEtudiant)) {
            return this.etudiants.get(numeroEtudiant);
        }
        throw new EtudiantInexistantException();
    }
}
```

```java
public class Etudiant {

    private String numeroEtudiant;
    private String nomEtudiant;
    private String prenomEtudiant;
    private String adresse;

    public Etudiant(String nomEtudiant, String prenomEtudiant, String adresse) {
        this.numeroEtudiant = UUID.randomUUID().toString();
        this.nomEtudiant = nomEtudiant;
        this.prenomEtudiant = prenomEtudiant;
        this.adresse = adresse;
    }

    public String getNumeroEtudiant() {
        return numeroEtudiant;
    }

    public void setNumeroEtudiant(String numeroEtudiant) {
        this.numeroEtudiant = numeroEtudiant;
    }

    public String getNomEtudiant() {
        return nomEtudiant;
    }

    public void setNomEtudiant(String nomEtudiant) {
        this.nomEtudiant = nomEtudiant;
    }

    public String getPrenomEtudiant() {
        return prenomEtudiant;
    }

    public void setPrenomEtudiant(String prenomEtudiant) {
        this.prenomEtudiant = prenomEtudiant;
    }

    public String getAdresse() {
        return adresse;
    }

    public void setAdresse(String adresse) {
        this.adresse = adresse;
    }
}

public class EtudiantInexistantException extends Exception {
    // NOP
}
```

3. A quoi sert l'annotation **@Component** ?
4. Créez un package controleur au même niveau que la classe principale.
5. Créez une classe Controleur que vous annoterez avec **@RestController**.
6. Déclarez un champ de **FacadePromotion** annoté avec **@Autowired**. A quoi sert cette annotation ?
7. Créez un fichier **exercice1.http** dans le dossier `/src/main/resources`. Développez les URIs nécessaires pour exécuter le script suivant :

```
### Inscription d'un nouvel étudiant
POST http://localhost:8080/mpws/etudiant
Content-Type: application/x-www-form-urlencoded

nom=Boichut&prenom=Yohan&adresse=Somewhere in Orleans
```

```
> {%
client.global.set("locationEtudiant", response.headers.valueOf("Location"));
client.test("Request executed successfully", function() {
  client.assert(response.status === 201, "L'étudiant aurait dû être créé");
});
%}

### Récupération des informations de l'étudiant créé
GET {{locationEtudiant}}

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "L'étudiant aurait dû être retrouvé !");
});
%}

### Récupération d'un étudiant inexistant (404 attendu)
GET http://localhost:8080/mpws/etudiant/aucunechance

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 404, "L'étudiant n'aurait pas dû être retrouvé");
});
%}

### Récupération d'une collection de tous les étudiants
GET http://localhost:8080/mpws/etudiant

> {%
client.test("Request executed successfully", function() {
  client.assert(response.status === 200, "Les étudiants aurait dû être récupérés");
});
%}
```

8. A quoi sert à votre avis `client.global.set(...)` ?

2 Web-service un peu plus élaboré

Nous voulons gérer une plate-forme de vidéos.

2.1 Fonctionnalités attendues

— Un utilisateur doit s'inscrire (nom, mot de passe) pour pouvoir déposer des vidéos, créer des playlists, supprimer des playlists, ajouter une vidéo à une playlist, supprimer une vidéo d'une playlist.
— Chaque utilisateur inscrit peut déposer des vidéos (url, description, titre) sur cette plate-forme à partir du moment où il est authentifié (nom et mot de passe vont être requis pour chaque requête). Ces vidéos deviennent disponibles pour tout le monde.
— Seuls les utilisateurs inscrits peuvent composer et gérer des playlists. Une playlist peut contenir des vidéos de toute origine (pas nécessaire que l'utilisateur en soit le propriétaire).
— Un utilisateur inscrit peut récupérer l'intégralité de son profil, et uniquement de son profil (son nom, ses playlists et toutes les vidéos qu'il a posté).
— Un utilisateur inscrit peut récupérer l'intégralité de ses playlists (uniquement les siennes).
— Un utilisateur inscrit peut récupérer l'intégralité de ses vidéos postées (uniquement les siennes).

2.2 Travail à réaliser

1. Proposez des URIs pour votre Web-service qui permettront de couvrir les différentes fonctionnalités attendues.
2. Après validation de vos URIs, vous devez mettre en place le Web-service :
   (a) Créez votre application Springboot, comme dans l'exercice 2 du TP1.
   (b) Développez à l'intérieur un modèle (sans cryptographie) offrant toutes les intéractions requises : créer un compte, s'authentifier, ajouter une vidéo, ...
   (c) Créez votre contrôleur REST en injectant le modèle (comme vu dans l'exercice 2 du TP1).