Configuration DB et Entité

Démonstration 2 du module 5

Les objectifs de cette démonstration sont

- Mise en place des starters de Spring Boot pour JPA
- Intégration du driver de la base de données
- Configuration de la connexion à la base
- Création d'une entité

Contexte

- Compléter le projet de démonstration précédent

Déroulement

Configuration de Spring Boot dans build.gradle

- Ajout du starter de Spring Data JPA
- Ajout du driver de la base de données MySQL

```
dependencies {
	//Starter Spring Data JPA
	implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
	//Driver de MySQL
	runtimeOnly 'mysql:mysql-connector-java'
```

- Faire « Refresh Gradle Project »

Configuration de la connexion à la base de données

- Intégrer dans application.properties :
  - L'URL de la base
    - useSSL pour désactiver le certificat du Serveur en mode développement
    - serverTimezone=UTC pour permettre de préciser la timezone de la machine (*Universal Time Coordinated*)
  - Son user/password
- Et des paramètres pour JPA et Hibernate :
  - La permission de créer les tables à partir des entités (`spring.jpa.hibernate.ddl-auto=create`)
  - L'affichage des requêtes et le format d'affichage

```
#Configuration DB
spring.datasource.url=jdbc:mysql://localhost:3306/db_spring_demo?useSSL=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=Pa$$w0rd

spring.jpa.hibernate.ddl-auto=create
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Première entité : Course

Pour le moment, nous allons transformer le BO : Course en une entité JPA. Nous compléterons au fur et à mesure des démonstrations les entités.

- Sur la classe Course, placer :
  - @Entity → obligatoire pour devenir une entité JPA
  - @Table(nam= «COURSES») → pour customizer le nom de la table créé. Par défaut se serait COURSE ; le nom de la classe

```
package fr.eni.demo.bo;

import javax.persistence.*;

@Entity
@Table(name = "COURSES")
public class Course {
```

- Sur l'attribut id, placer :
  - @Id → pour déclarer cet attribut comme la clef primaire de la table
  - @GeneratedValue(strategy=GenerationType.IDENTITY) → pour déclarer comment créer cette clef. IDENTITY = auto incrémentation de la clef à chaque insertion.

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

- Sur l'attribut title, placer :
  - @Column(length = 250, nullable = false) → pour préciser la taille max de la colonne et le fait qu'il ne peut pas être nul.

```
@Column(length = 250, nullable = false)
private String title;
```

Exécution de l'application :

- Constater dans les traces de démarrage de l'application :

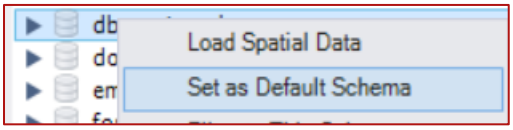
```
Hibernate:

    drop table if exists courses
Hibernate:

    create table courses (
       id bigint not null auto_increment,
        duration integer not null,
        title varchar(250) not null,
        primary key (id)
    ) engine=InnoDB
```

- Spring a exécuté les commandes de suppression et de création de la table courses sans que

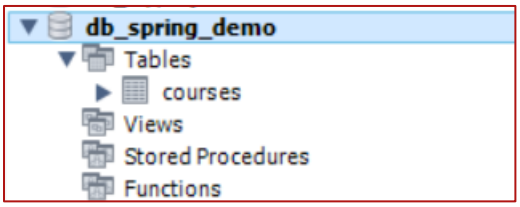
nous ayons rien à faire.

- Par défaut ; dès qu'il y a les annotations @Entity, Spring va essayer de mapper ces classes avec la base de données.

- Vérifier dans MySQL ; la création de la table.
  - Faire un clic droit sur la DB, et cliquer sur « Set as Default Schema »



  - Pour nous permettre de faire du requêtage sur les tables de cette base par défaut
  - Constater que Spring a bien créé la table :



Pour le moment, il n'y a pas de valeurs en base.