Ecole Normale Supérieure de
l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Compte rendu TP

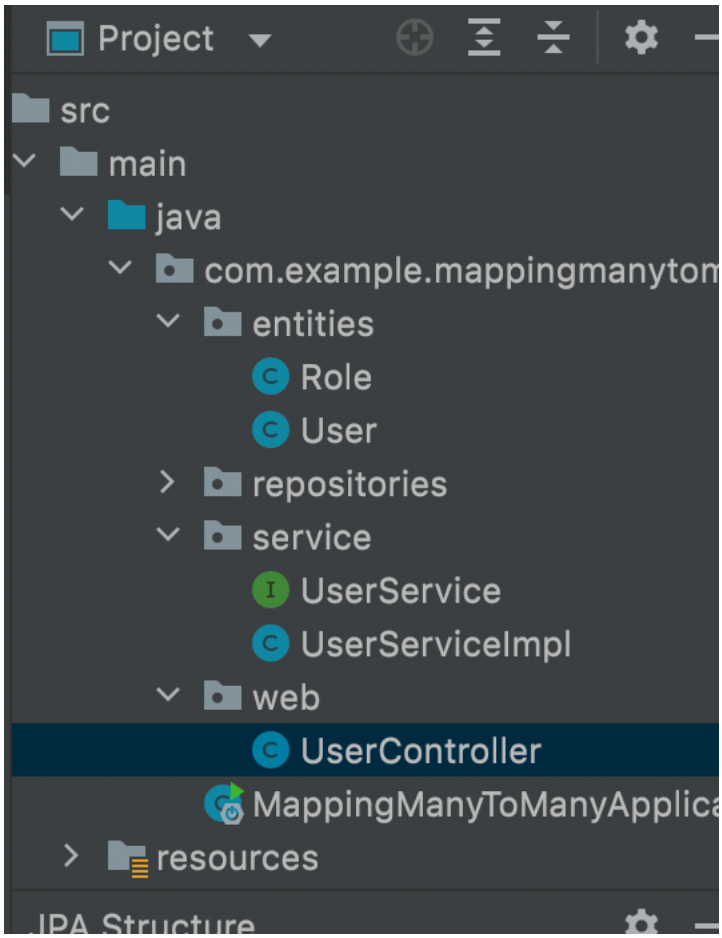
Gérer les associations entre les entités

Réalisé par :
SABRI Wissale

Introduction

Ce TP sert à appliquer nos connaissance en mapping object relationnel.

Structure d'application



1- Les dépendances dans pom

On utilisera :

- Lombok
- Spring data JPA
- mysql driver
- Spring web

2- Les entités JPA

User.java Role.java

```java
package com.example.mappingmanytomany.entities;

import ...

@Entity
@Table(name = "USERS")
@Data @NoArgsConstructor @AllArgsConstructor
public class User {
    @Id
    private String userId;
    @Column(name = "USER_NAME",unique = true, length = 20)
    private String username;
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private String password;
    @ManyToMany (mappedBy = "users", fetch = FetchType.EAGER)
    private List<Role> roles = new ArrayList<>();

}
```

Role.java

```java
package com.example.mappingmanytomany.entities;

import ...

@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Role {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(unique = true,length = 20)
    private String roleName;
    @ManyToMany(fetch = FetchType.EAGER)
    @ToString.Exclude
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<User> users = new ArrayList<>();
    @Column(name = "DESCRIPTION")
    private String desc;
}
```

3- Les repositories

Pour chacune des entités, on crée un repo comme ci-dessous qui hérite de la classe générique JpaRepository en spécifiant la classe et le type de l'id

RoleRepository.java | UserRepository.java

```java
package com.example.mappingmanytomany.repositories;

import com.example.mappingmanytomany.entities.Role;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface RoleRepository extends JpaRepository<Role,Long> {
    Role findByRoleName(String roleName);
}
```

UserRepository.java

```java
package com.example.mappingmanytomany.repositories;

import com.example.mappingmanytomany.entities.User;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface UserRepository extends JpaRepository<User, String> {
    User findByUsername(String userName);
    List<User> findAll();
}
```

4- La couche metier

L'interface service sert à declarer les méthodes de manipulation des objets

UserService.java

```java
package com.example.mappingmanytomany.service;

import com.example.mappingmanytomany.entities.Role;
import com.example.mappingmanytomany.entities.User;


import java.util.List;

public interface UserService {
    User addNewUser(User user);
    Role addNewRole(Role role);
    User findUserByUserName(String userName);
    Role findRoleByRoleNAme(String roleNAme);
    void addRoleToUser(String username, String rolename);
    User autheticate(String username,String password);
    List<User> getAllUsers();
```

On implémente ensuite cette interface, et on se base sur des instances des repositories pour executer les différentes requetes (ajout, recherche, modification, ..) Cette implementation va séparer l'aspect métier de la gestion des objets de la base de données .

UserServiceImpl.java

```java
@AllArgsConstructor
public class UserServiceImpl implements UserService {

    private UserRepository userRepository;
    private RoleRepository roleRepository;

    @Override
    public User addNewUser(User user) {

        user.setUserId(UUID.randomUUID().toString());
        return userRepository.save(user);
    }

    @Override
    public Role addNewRole(Role role) { return roleRepository.save(role); }

    @Override
    public User findUserByUserName(String userName) { return userRepository.

    @Override
    public Role findRoleByRoleNAme(String roleNAme) { return roleRepository.

    @Override
    public void addRoleToUser(String username, String rolename) {
        User user = findUserByUserName(username);
```

5- Controlleur :

UserController.java

```java
import com.example.mappingmanytomany.service.UserService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
public class UserController {
    @Autowired
    private UserService userService;
    @GetMapping("/users/{username}")
    public User user(@PathVariable String username){
        User user = userService.findUserByUserName(username);
        return user;
    }
    @GetMapping("/users")
    public List<User> users(){
        List<User> users = userService.getAllUsers();
        return users;
    }
}
```

6- propriétes de l'application

application.properties

```
server.port=8081
spring.datasource.url=jdbc:mysql://localhost:3306/USERS?createDatabaseIfNotExist=
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=create
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
spring.jpa.show-sql=true
```