ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA

UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Filière :
« Génie du Logiciel et des Systèmes Informatiques Distribués »
GLSID

Compte rendu
Activité 5: Mapping des associations(ManyToMany) en JPA Hibernate Spring Data

Année Universitaire : 2021-2022

Réalisé par :

Amina MAAKOUL

Encadré par :

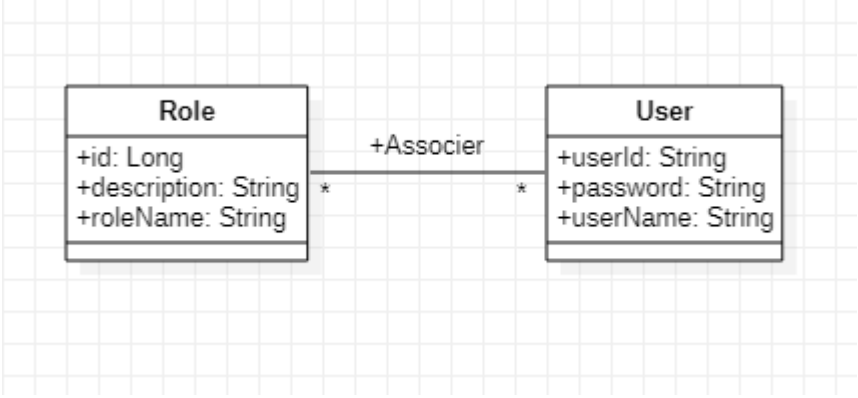
M. Mohamed YOUSSFI

I. Introduction

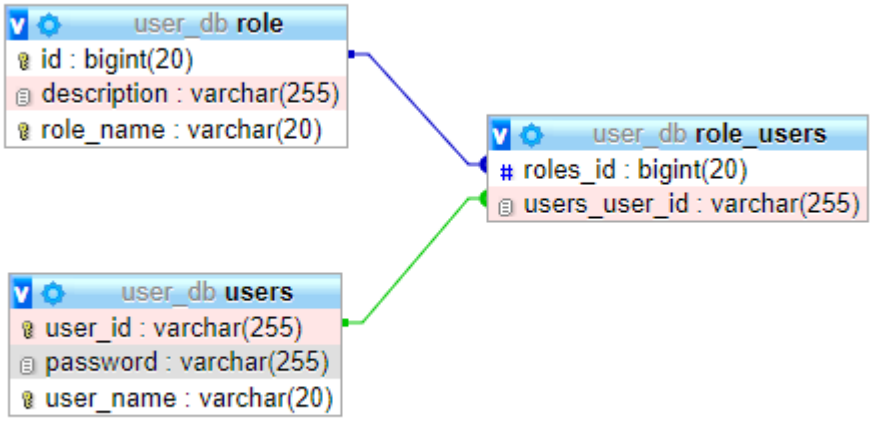
Cette activité permet de démonter la manipulation des différentes fonctionnalités sur des entités qui ont une associations(**ManyToMany**) en **JPA Hibernate Spring Data.** Et l'affichage des tables et leur association dans la base de données h2 et MySQL.

L'architecture d'activité se compose de trois couche : métier, DAO et présentation.

- Diagramme de classe

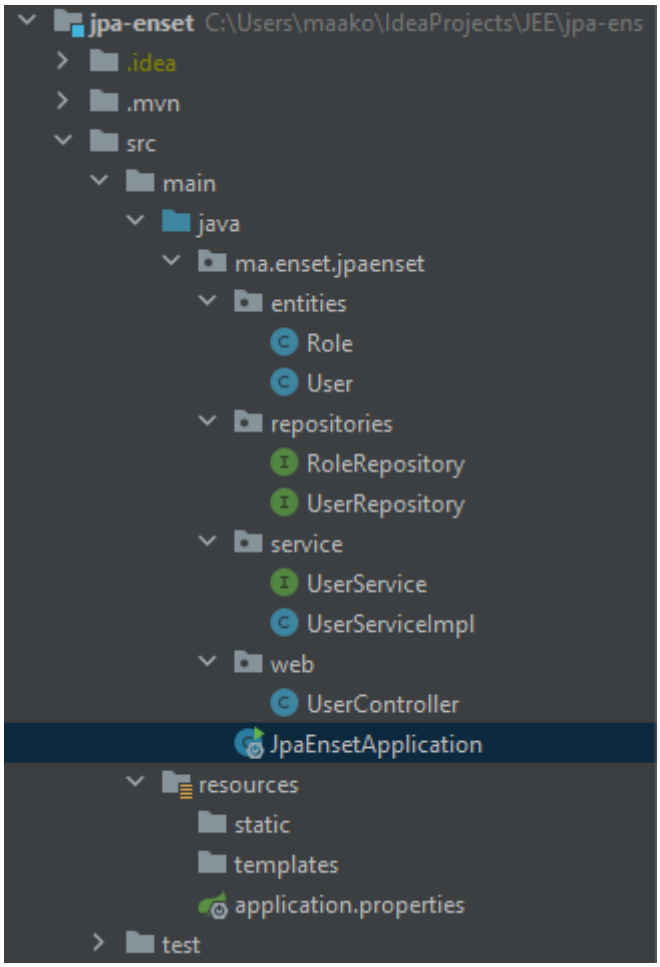


- MLD (PhpMyAdmin)



II. *Réalisation*

1. La structure du projet :



2. Le package « entities »

- Classe « Role »

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

- Classe « User »

```java
@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<>();
}
```

3. Le package « repositories »

- L'interface « RoleRepository »

```java
package ma.enset.jpaenset.repositories;

import ma.enset.jpaenset.entities.Role;
import ma.enset.jpaenset.entities.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

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

- L'interface « UserRepository »

```java
package ma.enset.jpaenset.repositories;

import ma.enset.jpaenset.entities.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User,String> {
    User findByUsername(String userName);

}
```

4. Le package « service »

- L'interface « IHospitalService »

```java
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 authentificate(String userName, String password);
}
```

- Classe « HospitalServiceImpl »

```
@Service
@Transactional
@AllArgsConstructor //ne fait le constructeur sans paramètre dans
l'injection
public class UserServiceImpl implements UserService {
    private UserRepository userRepository;
    private RoleRepository roleRepository;

    @Override
    public User addNewUser(User user) {
        user.setUserId(UUID.randomUUID().toString());
        String pw=user.getPassword();
        return userRepository.save(user);
    }
    @Override
    public Role addNewRole(Role role) {
        return roleRepository.save(role);
    }

    @Override
    public User findUserByUserName(String userName) {
        return userRepository.findByUsername(userName);
    }
    @Override
    public Role findRoleByRoleName(String roleName) {

        return roleRepository.findByRoleName(roleName);
    }

    @Override
    public void addRoleToUser(String username, String rolename) {
        User user=findUserByUserName(username);
        Role role=findRoleByRoleName(rolename);
        if(user.getRoles()!=null){
            user.getRoles().add(role);
            role.getUsers().add(user);
        }
    }
    @Override
    public User authentificate(String userName, String password) {
        User user=userRepository.findByUsername(userName);
        if(user==null) throw new RuntimeException("Bad credentilas");
        if(user.getPassword().equals(password)){
            return user;
        }
        throw new RuntimeException("Bad credentilas");
    }
}
```

5. Le package « web »

- Classe « UserRestController »

```
@RestController
public class UserController {
    @Autowired
```

```
    private UserService userService;

    @GetMapping("/users/{username}")
    public User user(@PathVariable String username){
        User user=userService.findUserByUserName(username);
        return user;
    }
}
```

6. Présentation

Dans cette présentation nous avons manipulé les fonctions suivantes :

- La création des utilisateurs et des rôles
- Affectation des rôles aux utilisateurs
- Afficher les informations d'un utilisateur à partir de son userName et le mot de passe

```
@SpringBootApplication
public class JpaEnsetApplication {

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

    @Bean
    CommandLineRunner start(UserService userService){
        return args ->{
            User u  = new User();
            u.setUsername("user1"); u.setPassword("123456");
            userService.addNewUser(u);

            User u2  = new User();
            u2.setUsername("admin"); u2.setPassword("123456");
            userService.addNewUser(u2);

            Stream.of("STUDENT","USER", "ADMIN").forEach(r->{
                Role role1=new Role();
                role1.setRoleName(r);
                userService.addNewRole(role1);
            });

            userService.addRoleToUser("user1","STUDENT");
            userService.addRoleToUser("user1","USER");
            userService.addRoleToUser("admin","USER");
            userService.addRoleToUser("admin","ADMIN");

            try {
                User user=userService.authentificate("user1","123456");
                System.out.println(user.getUserId());
                System.out.println(user.getUsername());
                System.out.println("Roles=>");
                user.getRoles().forEach(r->{
                    System.out.println("Role="+r.toString());
                });
            }catch (Exception exception){

            }
```

```
        };
    }

}
```

7. L'exécution

- L'affichage d'ID du premier utilisateur, et la liste de ses rôles.

```
0b77715c-2e00-4973-b878-4bf54c5a92d1
user1
Roles=>
Role=Role(id=1, desc=null, roleName=STUDENT)
Role=Role(id=2, desc=null, roleName=USER)
```

- Les entités avec « h2 »

SELECT * FROM ROLE_USERS;

ROLES_ID	USERS_USER_ID
1	0b77715c-2e00-4973-b878-4bf54c5a92d1
2	0b77715c-2e00-4973-b878-4bf54c5a92d1
2	2c8be3a5-8809-45da-87af-df667bdfe05f
3	2c8be3a5-8809-45da-87af-df667bdfe05f

(4 rows, 12 ms)

SELECT * FROM ROLE;

ID	DESC	ROLE_NAME
1	*null*	STUDENT
2	*null*	USER
3	*null*	ADMIN

(3 rows, 4 ms)

Edit

SELECT * FROM USERS;

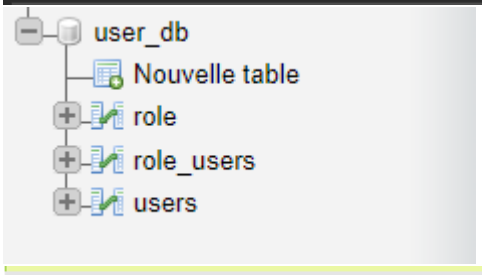
USER_ID	PASSWORD	USER_NAME
0b77715c-2e00-4973-b878-4bf54c5a92d1	123456	user1
2c8be3a5-8809-45da-87af-df667bdfe05f	123456	admin

(2 rows, 4 ms)

Edit

- L'affichage dans MySQL

```
Hibernate: select roles0_.users_user_id as users_us2_1_0_, roles0_.roles_id as roles_id1_1_0_, role1_.id as id1_0_1_, role1_.description as descript2_0_1_
Hibernate: select users0_.roles_id as roles_id1_1_0_, users0_.users_user_id as users_us2_1_0_, user1_.user_id as user_id1_2_1_, user1_.password as passwor
Hibernate: select roles0_.users_user_id as users_us2_1_0_, roles0_.roles_id as roles_id1_1_0_, role1_.id as id1_0_1_, role1_.description as descript2_0_1_
Hibernate: select users0_.roles_id as roles_id1_1_0_, users0_.users_user_id as users_us2_1_0_, user1_.user_id as user_id1_2_1_, user1_.password as passwor
Hibernate: select role0_.id as id1_0_, role0_.description as descript2_0_, role0_.role_name as role_nam3_0_ from role role0_ where role0_.role_name=?
Hibernate: select users0_.roles_id as roles_id1_1_0_, users0_.users_user_id as users_us2_1_0_, user1_.user_id as user_id1_2_1_, user1_.password as passwor
Hibernate: insert into role_users (roles_id, users_user_id) values (?, ?)
Hibernate: select user0_.user_id as user_id1_2_, user0_.password as password2_2_, user0_.user_name as user_nam3_2_ from users user0_ where user0_.user_nam
Hibernate: select roles0_.users_user_id as users_us2_1_0_, roles0_.roles_id as roles_id1_1_0_, role1_.id as id1_0_1_, role1_.description as descript2_0_1_
Hibernate: select users0_.roles_id as roles_id1_1_0_, users0_.users_user_id as users_us2_1_0_, user1_.user_id as user_id1_2_1_, user1_.password as passwor
Hibernate: select roles0_.users_user_id as users_us2_1_0_, roles0_.roles_id as roles_id1_1_0_, role1_.id as id1_0_1_, role1_.description as descript2_0_1_
Hibernate: select users0_.roles_id as roles_id1_1_0_, users0_.users_user_id as users_us2_1_0_, user1_.user_id as user_id1_2_1_, user1_.password as passwor
Hibernate: select users0_.roles_id as roles_id1_1_0_, users0_.users_user_id as users_us2_1_0_, user1_.user_id as user_id1_2_1_, user1_.password as passwor
7ae21210-4e08-4a93-b970-49e41ca1444b
user1
Roles=>
Role=Role(id=1, desc=null, roleName=STUDENT)
Role=Role(id=2, desc=null, roleName=USER)
```




```
SELECT * FROM `users`
```

Profilage [Éditer en

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier s

+ Options

	user_id	password	user_name
Éditer Copier Supprimer	7ae21210-4e08-4a93-b970-49e41ca1444b	123456	user1
Éditer Copier Supprimer	b20809ac-b208-42ae-b1b9-9569aa01d6c9	123456	admin

```
SELECT * FROM `role`
```

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Cherch

+ Options

	id	description	role_name
Éditer Copier Supprimer	1	*NULL*	STUDENT
Éditer Copier Supprimer	2	*NULL*	USER
Éditer Copier Supprimer	3	*NULL*	ADMIN

```
SELECT * FROM `role_users`
```

Tout afficher | Nombre de lignes : 25 Filtrer les

+ Options

roles_id	users_user_id
1	7ae21210-4e08-4a93-b970-49e41ca1444b
2	7ae21210-4e08-4a93-b970-49e41ca1444b
2	b20809ac-b208-42ae-b1b9-9569aa01d6c9
3	b20809ac-b208-42ae-b1b9-9569aa01d6c9

- L'affichage des informations de l'utilisateur « user1 » sous forme d'un fichier JSON.

- L'affichage des informations de l'utilisateur « admin » sous forme d'un fichier JSON.

III. *Conclusion*

À la fin de cette activité, nous avons pu terminer les tâches concernant la création des entités, la définition des associations, l'injection des dépendances. Ensuite la manipulation d'un ensemble des méthodes sur la gestion des utilisateurs et leurs rôles. En fin la démonstration des données enregistrées dans la base de données h2 et MySQL.