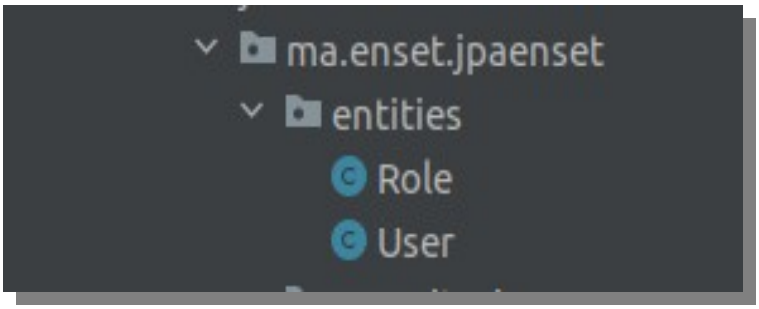
Activité pratique : La gestion des rôles de utilisateurs

Créé par : Mohamed Ait Lahcen

1. Les entités de l’application :

Cette application contient deux entités :



- **Role :**

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

- **User :**

```
22 usages
@Entity
    @Table(name="USERS")
@Data @AllArgsConstructor @NoArgsConstructor
public class User {
    @Id
    private String userId;
    @Column(name="USER_NAME",unique = true, length =20 )
    private String userName;
    private String password;
    @ManyToMany( fetch = FetchType.EAGER)
    private List<Role> roles=new ArrayList<>();
}
```

2. **Les interfaces repository :**

   Cette application contient une interface repository pour chaque entité :

   - **RoleRepository:**

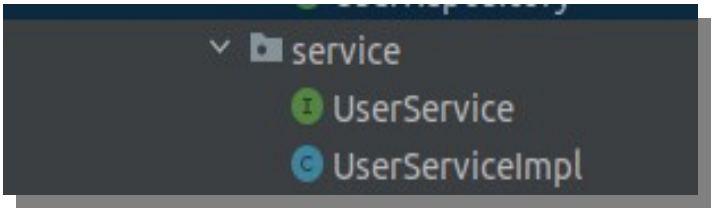
```
4 usages
public interface RoleRepository extends JpaRepository<Role,Long> {
    1 usage
    Role findByRoleName(String roleName);
}
```

- **UserRepository :**

```
2 usages
public interface UserRepository extends JpaRepository<User,String> {
    3 usages
    User findByUserName(String userName);
}
```

3. La couche service (Metier) :

Ce package contient une interface et son implémentation en fait il représente la couche metier de l'application :



- **UserService :**

cette interface qui contient les fonctions qui permettent d'ajouter les instances des entités ou bien les enregistrements des entités dans la base de données :

```java
3 usages  1 implementation
public interface UserService {
    2 usages  1 implementation
    User addNewUser(User user);
    1 usage  1 implementation
    Role addNewRole(Role role);
    1 implementation
    User findUserByUserName(String userName);
    //Role findByRoleName(String roleName);
    4 usages  1 implementation
    void addRoleToUser(String userName,String roleName);
    1 usage  1 implementation
    User authenticate(String username,String password);

}
```

- **UserServiceImpl:**

  cette classe contient la déclaration des fonctions de l’interface IhospitalService :

```java
@Service
@Transactional
@AllArgsConstructor
public class UserServiceImpl implements UserService {
    5 usages
    UserRepository userRepository;
    2 usages
    RoleRepository roleRepository;
    2 usages
```

```java
    2 usages
    @Override
    public User addNewUser(User user) {
        user.setUserId(UUID.randomUUID().toString());
        return userRepository.save(user);
    }
    1 usage
    @Override
    public Role addNewRole(Role role) {
        return roleRepository.save(role);
    }
    @Override
    public User findUserByUserName(String userName) {
        return userRepository.findByUserName(userName);
    }
```

4. **La connexion avec la base de données :**

Dans cette application j'ai utilisé la base de donnée mySql, voici le fichier configuration.properties qui contient le type de la base de données utilisé :

```
spring.datasource.url=jdbc:mysql://localhost:3306/users_db?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MariaDBDialect
spring.h2.console.enabled=true
server.port=8082
spring.jpa.show-sql=true
```

- **jpaEnsetApplication :**

  Cette classe permet d'interagir avec les table de la base de données voici une partie de code de cette classe dans laquelle j'ai ajouté des enregistrements dans la base de données :

La création des utilisateurs :

```
@Bean
CommandLineRunner start(UserService userService, RoleRepository roleRepository){
    return args->{
        //create the first user
     User user1= new User();
     user1.setUserName("user");
     user1.setPassword("12345");
     userService.addNewUser(user1);
     //create the second user :
        User user2= new User();
        user2.setUserName("admin");
        user2.setPassword("12345");
        userService.addNewUser(user2);
```

La création des rôles :

```
        //create roles :
        Stream.of( ...ts: "USER","STUDENT","ADMIN").forEach(name->{
            Role role1=new Role();
            role1.setRoleName(name);
            role1.setDesc(name+"_desc");
            userService.addNewRole(role1);
        });
```

■ L’ajout des rôles aux utilisateurs :

```
//adding roles to users :
try{
    userService.addRoleToUser( userName: "admin", roleName: "ADMIN");
}catch(Exception e){
    e.printStackTrace();
}
userService.addRoleToUser( userName: "admin", roleName: "USER");
userService.addRoleToUser( userName: "user", roleName: "USER");
userService.addRoleToUser( userName: "user", roleName: "STUDENT");
try{
   User user= userService.authenticate( username: "admin", password: "1234");
    System.out.println("userId:"+user.getUserId());
    System.out.println("username:"+user.getUserName());
    System.out.println("roles=>");
    user.getRoles().forEach(r->{
        System.out.println("role=>"+r.getRoleName());
    });
```

◆ Les tables de la bases de données :

1. La table Role :

	id	description	role_name
Éditer Copier Supprimer	1	USER_desc	USER
Éditer Copier Supprimer	2	STUDENT_desc	STUDENT
Éditer Copier Supprimer	3	ADMIN_desc	ADMIN

2. **La table User :**

Options

←T→	user_id	password	user_name
Éditer Copier Supprimer	cc3baf92-117c-4c88-b1ad-ab3b038f56fb	12345	user
Éditer Copier Supprimer	e40699a2-2179-4460-97f3-715423aab783	12345	admin