

DÉPARTEMENT MATHÉMATIQUES ET INFORMATIQUE

Compte rendu de TP

Filière :

« Génie du Logiciel et des Systèmes Informatiques Distribués »

GLSID

TP6 : MAPPING ORM Héritage

Réalisé par :

- Hajar ZARGUAN

Encadré par :

Pr. Azeddine KHIAT

Année Universitaire : 2021-2022

Package metier:

```
9 import java.util.ArrayList;
10 import java.util.*;
11
12
13 @Entity
14 @Data @AllArgsConstructor @NoArgsConstructor
15 public class Role {
16
17     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
18     private Long Id ;
19     private String RoleName ;
20
21     @Column(name = "Description")
22     private String Desc ;
23
24     @ToString.Exclude
25     @ManyToMany(fetch = FetchType.EAGER)
26     private List<User> users = new ArrayList<User>();
27
28
29
30 }
31
```

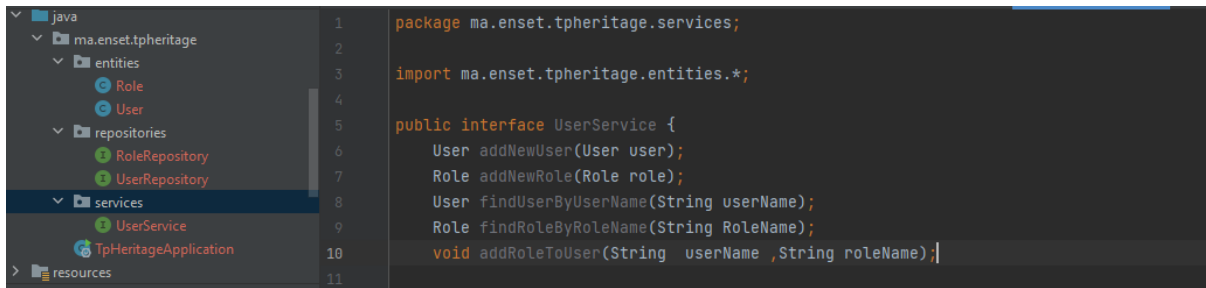
```
11 @Entity
12 @Data @AllArgsConstructor @NoArgsConstructor
13 public class User {
14
15     @Id
16     @GeneratedValue(strategy = GenerationType.IDENTITY)
17     private String userId ;
18
19     private String userName;
20     private String password ;
21
22
23     @ManyToMany( mappedBy = "users" ,fetch = FetchType.EAGER)
24     public List<Role> roles = new ArrayList<Role>();
25
26
27 }
```

Package Repositories

```
1 package ma.enset.tpheritage.repositories;
2
3 import ma.enset.tpheritage.entities.Role;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface RoleRepository extends JpaRepository<Role, Long> {
7 }
8
```

```
1 package ma.enset.tpheritage.repositories;
2
3 import ma.enset.tpheritage.entities.User;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface UserRepository extends JpaRepository<User, String> {
7 }
8
9
```

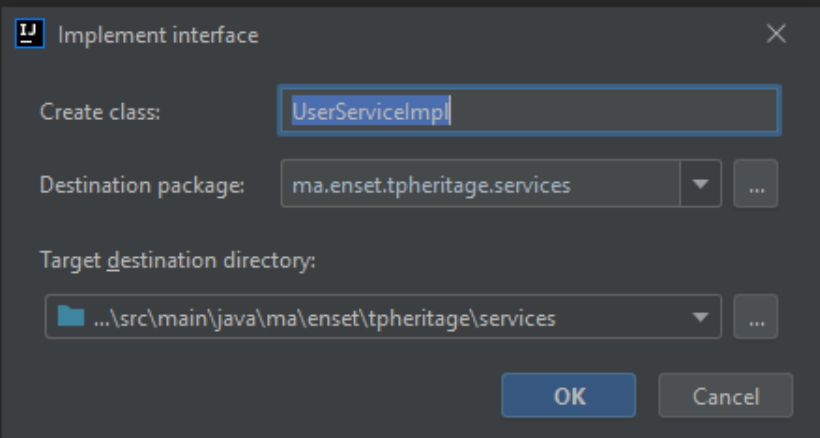
Package Services



```
1 package ma.enset.tpheritage.services;
2
3 import ma.enset.tpheritage.entities.*;
4
5 public interface UserService {
6     User addNewUser(User user);
7     Role addNewRole(Role role);
8     User findUserByUserName(String userName);
9     Role findRoleByRoleName(String roleName);
10    void addRoleToUser(String userName, String roleName);
11}
```

Implémentation de l'interface

```
5 public interface UserService {
6     User addNewUser(User user);
7     Role addNewRole(Role role);
8     User findUserByUserName(String userName);
9
10 }
11
12
13
```



```
3 import ...
4
5
6 public class UserServiceImpl implements UserService {
7     @Override
8     public User addNewUser(User user) {
9         return null;
10    }
11
12    @Override
13    public Role addNewRole(Role role) { return null; }
14
15
16    @Override
17    public User findUserByUserName(String userName) { return null; }
18
19
20
21    @Override
22    public Role findRoleByRoleName(String roleName) { return null; }
23
24
25
26    @Override
27    public void addRoleToUser(String userName, String roleName) {
28
29    }
30 }
31
```

Les annotations ajoutées: Contrainte d'intégrité

```
13 @Entity
14 @Data @AllArgsConstructor @NoArgsConstructor
15 public class Role {
16
17     @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
18     private Long Id ;
19
20     @Column(name = "USER_NAME", unique = true, length = 20)
21     private String RoleName ;
22
23     @Column(name = "Description")
24     private String Desc ;
25
26     @ToString.Exclude
27     @ManyToMany(fetch = FetchType.EAGER)
28     private List<User> users = new ArrayList<User>();
29 }
```

```
1 @Entity
2 @Data @AllArgsConstructor @NoArgsConstructor
3 public class User {
4
5     @Id
6     @GeneratedValue(strategy = GenerationType.IDENTITY)
7     private String userId ;
8
9     @Column(name = "USER_NAME", unique = true, length = 20)
10     private String userName;
11     private String password ;
12
13
14     @ManyToMany( mappedBy = "users" ,fetch = FetchType.EAGER)
15     public List<Role> roles = new ArrayList<Role>();
16 }
```

Package service : l'implémentation

```
10
11 import java.util.UUID;
12
13 @Service @Transactional @AllArgsConstructor
14 public class UserServiceImpl implements UserService {
15
16     UserRepository userRepository ;
17     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);
    }
}
```

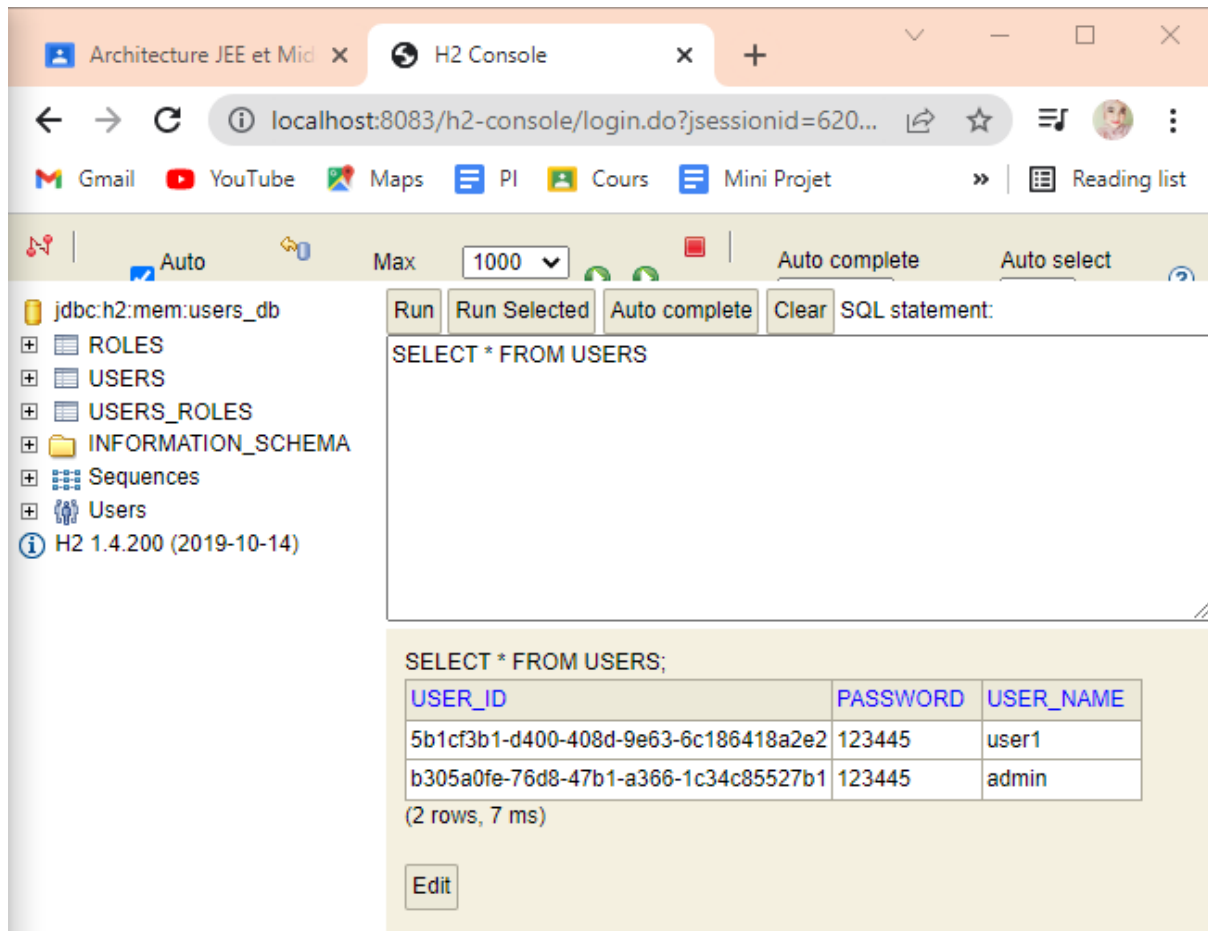
```
1 package ma.enset.tpheritage.repositories;
2
3 import ma.enset.tpheritage.entities.Role;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface RoleRepository extends JpaRepository<Role, Long> {
7     Role findRoleByRoleName(String roleName);
8 }
9
```

```
3 import ma.enset.tpheritage.entities.User;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface UserRepository extends JpaRepository<User, String> {
7     User findUserByUsername(String username);
8 }
9
```

```
31      @Override
32      public User findUserByUserName(String userName) {
33          return userRepository.findUserByUserName(userName);
34      }
35
36      @Override
37      public Role findRoleByRoleName(String roleName) {
38          return roleRepository.findRoleByRoleName(roleName);
39      }
```

```
31      @Override
32      public User findUserByUserName(String userName) {
33          return userRepository.findUserByUserName(userName);
34      }
35
36      @Override
37      public Role findRoleByRoleName(String roleName) {
38          return roleRepository.findRoleByRoleName(roleName);
39      }
40
41      @Override
42      public void addRoleToUser(String userName, String roleName) {
43          User user = findUserByUserName(userName);
44          Role role = findRoleByRoleName(roleName);
45          user.getRoles().add(role);
46          role.getUsers().add(user);
47      }
48  }
```

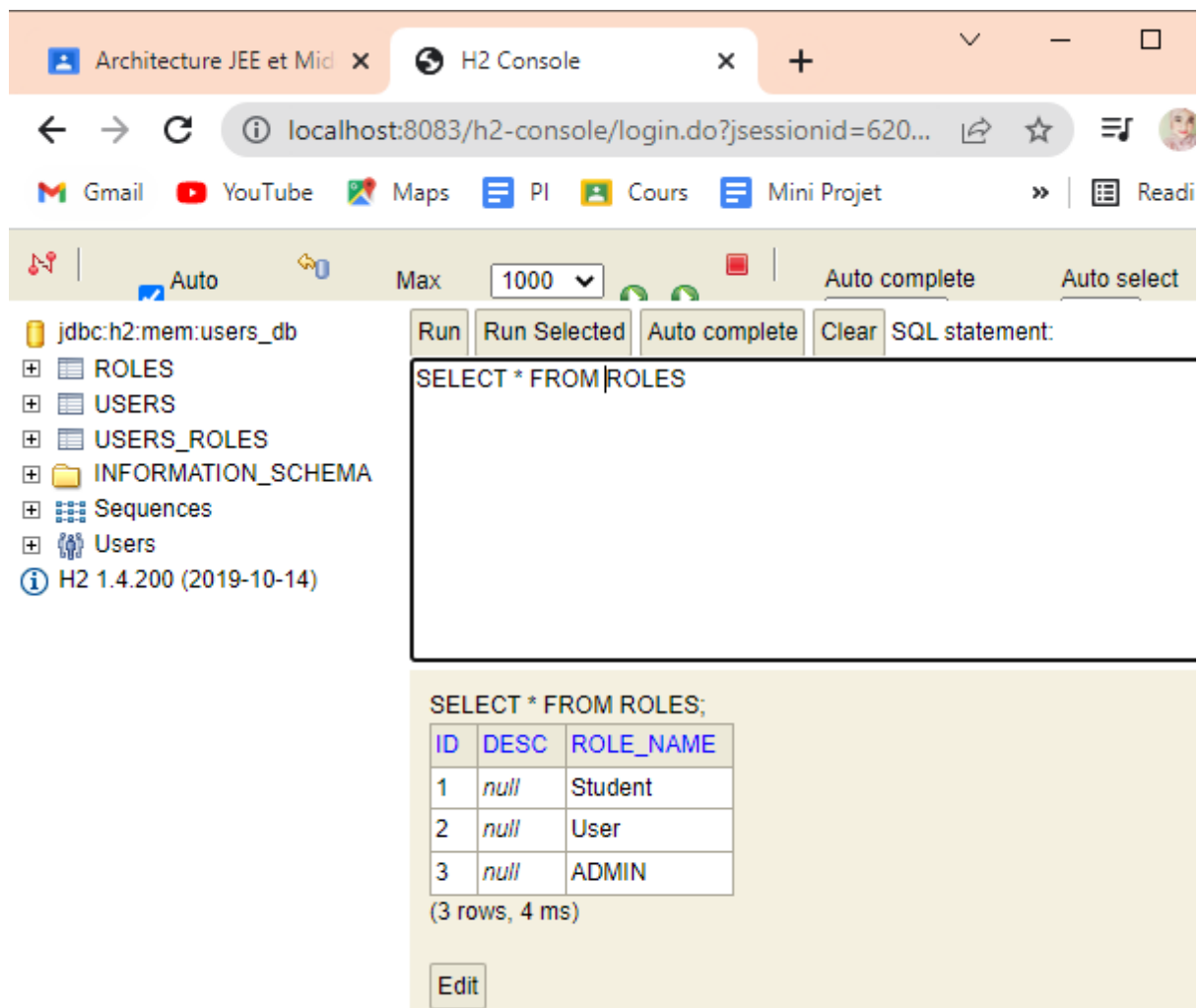
```
31      @Override
32      public User authenticate(String username, String password) throws Exception {
33          User user = userRepository.findUserByUserName(username);
34          if (user.getPassword().equals(password)) {
35              return user;
36          }
37          throw new Exception("Bad Credential");
38      }
```



The screenshot shows the H2 Console web interface in a browser. The URL is `localhost:8083/h2-console/login.do?jsessionid=620...`. The left sidebar shows the database structure for `jdbc:h2:mem:users_db`, including tables like `ROLES`, `USERS`, `USERS_ROLES`, and `INFORMATION_SCHEMA`. The main area displays the SQL statement `SELECT * FROM USERS` and its results in a table format.

USER_ID	PASSWORD	USER_NAME
5b1cf3b1-d400-408d-9e63-6c186418a2e2	123445	user1
b305a0fe-76d8-47b1-a366-1c34c85527b1	123445	admin

(2 rows, 7 ms)



The screenshot shows the H2 Console web interface in a browser. The address bar shows the URL: `localhost:8083/h2-console/login.do?jsessionid=620...`. The interface includes a sidebar with a database tree on the left and a main area for SQL queries on the right. The database tree shows a connection named `jdbc:h2:mem:users_db` with several tables and schemas listed. The main area contains a text input field with the SQL query `SELECT * FROM ROLES`. Below the input field, there are buttons for `Run`, `Run Selected`, `Auto complete`, and `Clear`. The results of the query are displayed in a table below the input field.

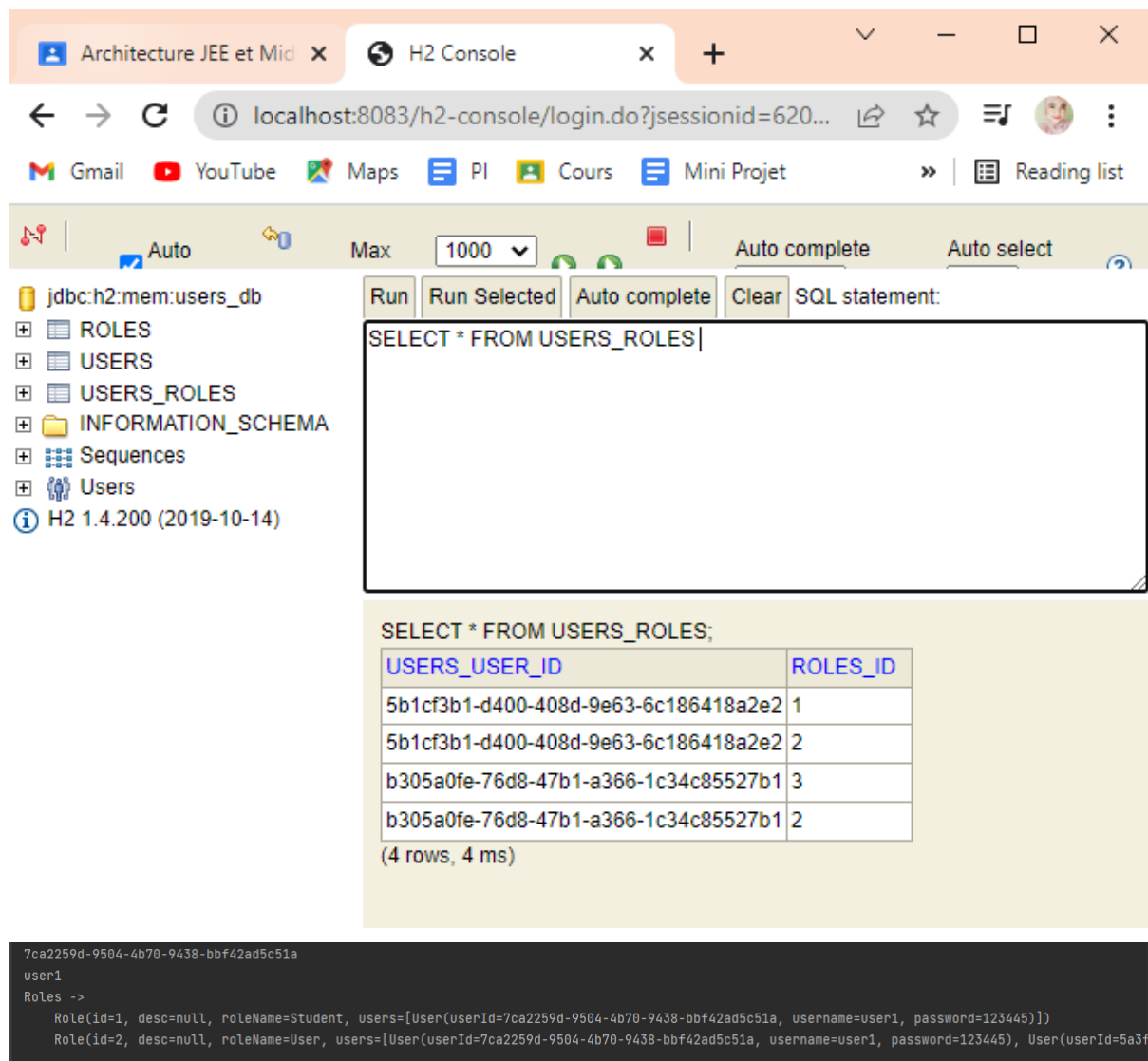
SQL statement:

```
SELECT * FROM ROLES
```

ID	DESC	ROLE_NAME
1	null	Student
2	null	User
3	null	ADMIN

(3 rows, 4 ms)

Edit



The screenshot shows the H2 Console interface in a web browser. The address bar indicates the URL is `localhost:8083/h2-console/login.do?jsessionid=620...`. The left sidebar shows the database structure for `jdbc:h2:mem:users_db`, including tables `ROLES`, `USERS`, and `USERS_ROLES`, as well as `INFORMATION_SCHEMA`, `Sequences`, and `Users`. The main area displays the SQL statement `SELECT * FROM USERS_ROLES` and its results in a table format.

USERS_USER_ID	ROLES_ID
5b1cf3b1-d400-408d-9e63-6c186418a2e2	1
5b1cf3b1-d400-408d-9e63-6c186418a2e2	2
b305a0fe-76d8-47b1-a366-1c34c85527b1	3
b305a0fe-76d8-47b1-a366-1c34c85527b1	2

(4 rows, 4 ms)

Below the table, the console output shows the database schema and user information:

```

7ca2259d-9504-4b70-9438-bbf42ad5c51a
user1
Roles ->
  Role(id=1, desc=null, roleName=Student, users=[User(userId=7ca2259d-9504-4b70-9438-bbf42ad5c51a, username=user1, password=123445)])
  Role(id=2, desc=null, roleName=User, users=[User(userId=7ca2259d-9504-4b70-9438-bbf42ad5c51a, username=user1, password=123445), User(userId=5a3f...
  
```