Compte rendu du TP5 :

Mon GitHub : https://github.com/salma-didi

1. Les Entités :
   - Classe User :

```java
package com.example.jpa_tp5.entites;

import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
//@Table(schema = "users")
@Table(name = "USERS")
@Data @AllArgsConstructor @NoArgsConstructor
public class User {
    @Id
    private String UserID;
    @Column(name = "USER_NAME", unique = true,length = 20) //
c'est un index unique
    private String Username;
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private String password;
    @ManyToMany(mappedBy = "users", fetch = FetchType.EAGER)
    private List<Role> roles = new ArrayList<>();
}
```

   - Classe Role:

```java
package com.example.jpa_tp5.entites;

import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Role {
    @Id @GeneratedValue(strategy= GenerationType.IDENTITY)
    private Long id;
    @Column(name = "DESCRIPTION")
    private String desc;
    @Column(unique = true,length = 20)
    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<>();
}
```

2. Repositories:
   - UserRepository:

```
package com.example.jpa_tp5.repositories;

import com.example.jpa_tp5.entites.Role;
import com.example.jpa_tp5.entites.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;


public interface UserRepository extends JpaRepository<User,
String> {
    User findByUserName(String username);
}
```

   - RoleRepository:

```
package com.example.jpa_tp5.repositories;

import com.example.jpa_tp5.entites.Role;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;


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

3. Services:
   - Interface UserService:

```
package com.example.jpa_tp5.services;

import com.example.jpa_tp5.entites.Role;
import com.example.jpa_tp5.entites.User;

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);  //
l'importance de l'unicité des arguments
    User authenticate(String userName, String password);
}
```

   - Classe UserServiceImpl:

```java
package com.example.jpa_tp5.services;

import com.example.jpa_tp5.entites.Role;
import com.example.jpa_tp5.entites.User;
import com.example.jpa_tp5.repositories.RoleRepository;
import com.example.jpa_tp5.repositories.UserRepository;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;

import javax.transaction.Transactional;
import java.util.UUID;

@Service
@Transactional @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) {
        //role.setId(UUID.randomUUID().getMostSignificantBits());
        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); // pour être plus correcte
en POO
        }
        //userRepository.save(user); //c'est pas nécessaire, la
transaction fait le job elle-même
    }

    @Override
    public User authenticate(String userName, String password) {
        User user = userRepository.findByUserName(userName);
        if(user == null){
            throw new RuntimeException("Bad Credentials");
        }
```

```
        if(user.getPassword().equals(password)){
            return user;
        }
        throw new RuntimeException("Bad Credentials");
    }
}
```

4. Web:
   - Classe UserController:

```
package com.example.jpa_tp5.web;

import com.example.jpa_tp5.entites.User;
import com.example.jpa_tp5.services.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;

@RestController
public class UserController {
    @Autowired
    private UserService userService;

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

5. Application Properties:
   - H2:

```
spring.datasource.url=jdbc:h2:mem:user_db
spring.h2.console.enabled=true
server.port=8083
```

   - MySQL:

```
server.port=8083
spring.datasource.url =
jdbc:mysql://localhost:3306/USERS_DB?createBatabaseIfNotExist=true
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
```

6. Application d'affichage:

```
package com.example.jpa_tp5;

import com.example.jpa_tp5.entites.Role;
import com.example.jpa_tp5.entites.User;
import com.example.jpa_tp5.services.UserService;
```

```java
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import java.util.stream.Stream;

@SpringBootApplication
public class JpaTp5Application {

   public static void main(String[] args) {
      SpringApplication.run(JpaTp5Application.class, args);
   }
   @Bean
   CommandLineRunner start(UserService userService){
      return args -> {
         User u=new User();
         u.setUsername("user1");
         u.setPassword("123456");
         userService.addNewUser(u);

         User u1=new User();
         u1.setUsername("admin");
         u1.setPassword("123456");
         userService.addNewUser(u1);

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

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