Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Université Hassan II de Casablanca

ENSET

ا لمدرسة العليا لأساتذة التعليم التقني المحمدية

جامعة الحسن الثاني بالدار البيضاء

DÉPARTEMENT MATHÉMATIQUES ET INFORMATIQUE

Filière :

« Ingénierie Informatique : Big Data et Cloud Computing »

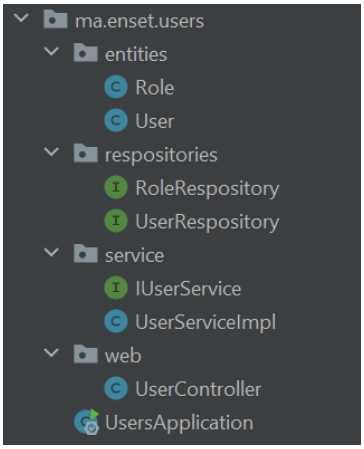
II-BDCC

TP 2: Jpa, Hibernate et Spring Data : Part 2

Réalisé Par: Laila Sad Elbouyoud

Classe: ii-BDCC 2

1. Hierarchie du projet :



2. Les entités :

a. classe User :

```java
@Entity
@Data @AllArgsConstructor @NoArgsConstructor
public class User {
    @Id
    private String UserId;
    @Column(unique = true, length = 25)
    private String username;
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private String password;
    @ManyToMany(mappedBy = "users", fetch = FetchType.EAGER)
    private List<Role> Roles = new ArrayList<>();
}
```

b. classe Rôle :

```java
@Entity
@Data @AllArgsConstructor @NoArgsConstructor
public class Role {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "DESCRIPTION")
    private String desc;
    @Column(unique = true, length = 25)
    private String roleName;
    @ManyToMany(fetch = FetchType.EAGER)
    @ToString.Exclude
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<User> users = new ArrayList<>();
}
```

3. Respositories :

a. UserRespository:

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

b. RoleRespository:

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

4. Couche Service:

a. Interface Service:

```
public interface IUserService {
    User addNewUser(User user);
    Role addNewRole(Role role);
    User findUserByUserName(String username);
    Role findRoleByRoleName(String roleName);
    Role addRoleToUser(String username, String rolename);
    User authenticate(String username, String password);
}
```

b. Implementation de l'interface Service :

```java
@Service
@Transactional
@AllArgsConstructor
public class UserServiceImpl implements IUserService {
    private RoleRespository roleRespository;
    private UserRespository userRespository;

    @Override
    public User addNewUser(User user) {
        user.setUserId(UUID.randomUUID().toString());
        return userRespository.save(user);
    }

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

    @Override
    public User findUserByUserName(String username) {
        return userRespository.findByUsername(username);
    }
```

```java
@Override
public Role findRoleByRoleName(String roleName) {
    return roleRespository.findByRoleName(roleName);
}

@Override
public Role 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);
    }
    return null;
}

@Override
public User authenticate(String username, String password) {
    User user = userRespository.findByUsername(username);
    if(user==null) throw new RuntimeException("Bad Credentials!");
    if(user.getPassword().equals(password)){
        return user;
    }
    throw new RuntimeException("Bad Credentials!");
}
```

5. Application Properties:

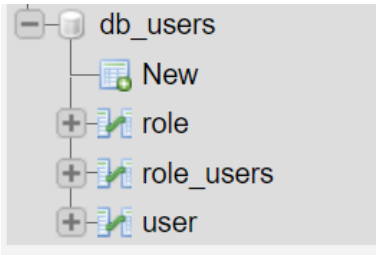
```
spring.datasource.url=jdbc:mysql://localhost:3306/db_Users?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
server.port=8082
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto= create
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect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

6. Application Main:

```
@SpringBootApplication
public class UsersApplication {

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

    @Bean
    CommandLineRunner start(IUserService service) {
        return args -> {
            User u = new User();
            u.setUsername("user1");
            u.setPassword("12345");
            service.addNewUser(u);

            User u2 = new User();
            u2.setUsername("admin");
            u2.setPassword("12345");
            service.addNewUser(u2);

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

```
    service.addRoleToUser( username: "user1",  rolename: "USER");
    service.addRoleToUser( username: "user1",  rolename: "STUDENT");
    service.addRoleToUser( username: "admin",  rolename: "USER");
    service.addRoleToUser( username: "admin",  rolename: "ADMIN");

    try{
        User user = service.authenticate( username: "user1",  password: "12345");
        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 e){
        e.printStackTrace();
    }
```

7. Base de données:



8. Couche Web:

a. User Controller:

```java
@RestController
public class UserController {
    @Autowired
    private IUserService userService;

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

b. Resultat:

{"username":"admin","userId":"ccc69bd9-1255-4527-82ec-de2fa49fb2ee","roles":[{"id":2,"desc":null,"roleName":"USER"},{"id":3,"desc":null,"roleName":"ADMIN"}]}