

Département Mathématique informatique

Rapport

filière :

“ Ingénierie Informatique - Big Data & Cloud Computing ”

II-BDCC

Mapping objet relationnel: JPA , Hibernate, Spring Data

Réalisation :

Zakaria Mansouri

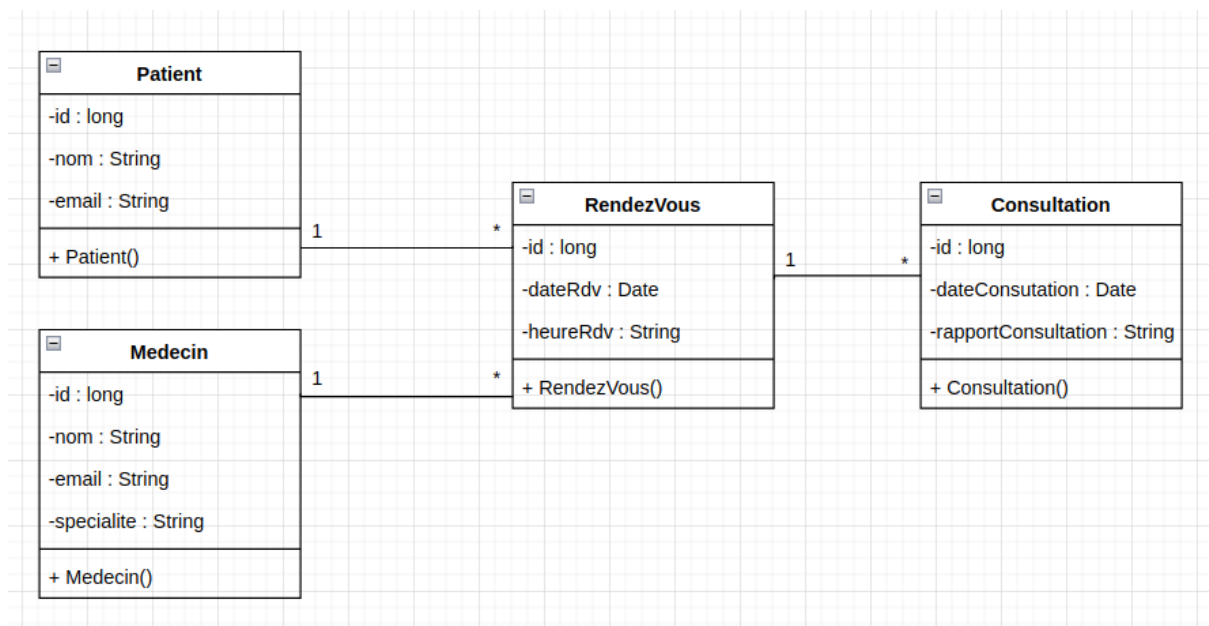
Année Universitaire : 2022 - 2023

Introduction:

Le mapping objet relationnel est le processus qui permet de faire la correspondance entre les données d'une base de données et les données de l'application.

→ Cas 1 : Gestion des Patients

nous allons traiter l'exemple suivant :



Nous allons faire le mapping objet relationnel de ces classes en utilisant JPA, Hibernate , Spring Data.

→ Notre projet sera un projet spring data.

Création des entités :

→ Patient

```
@Entity
@Data
@NoArgsConstructor @AllArgsConstructor
public class Patient {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    @Temporal(TemporalType.DATE)
    private Date dateNaissance;
    private boolean malade;

    @OneToMany(mappedBy = "patient", fetch = FetchType.LAZY)
    private Collection<RendezVous> rendezVous;
}
```

→ Medecin

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Medecin {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private String email;
    private StatusRDV status;
    private String specialite;

    @OneToMany(mappedBy = "medecin", fetch = FetchType.LAZY)
    private Collection<RendezVous> rendezVous;
}
```

→ Consultation

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Consultation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date consultationDate;
    private String rapport;

    @OneToOne
    private RendezVous rendezvous;
}
```

→ RendezVous

```
@Entity
@Data
@NoArgsConstructor
public class RendezVous {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date date;
    private StatusRDV status;

    @ManyToOne
    private Patient patient;

    @ManyToOne
    private Medecin medecin;

    @OneToOne(mappedBy = "rendezvous")
    private Consultation consultation;
}
```

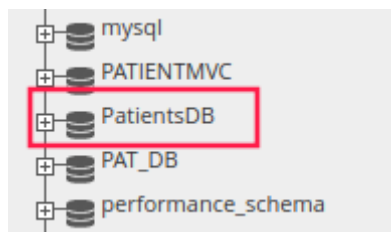
→ application.properties :

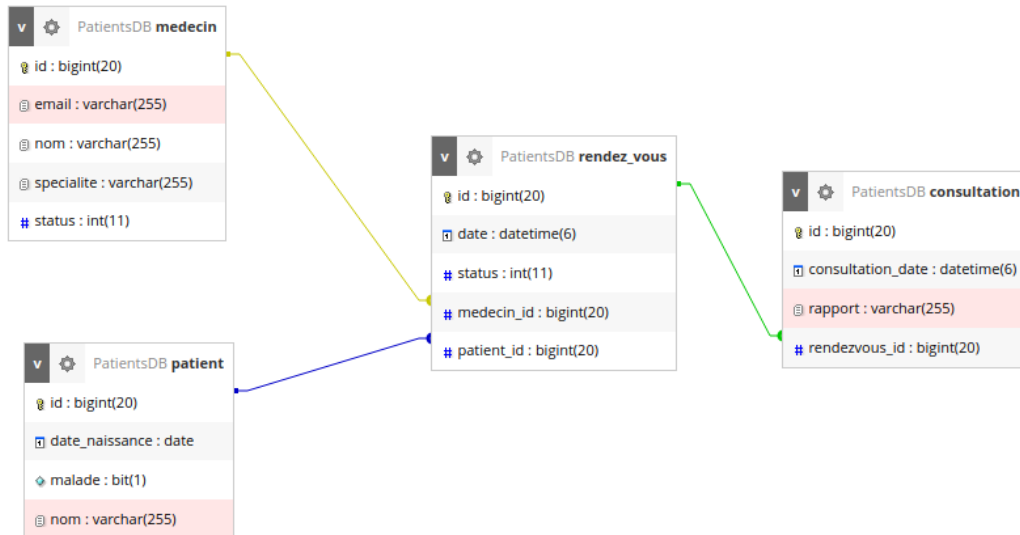
```
spring.datasource.url=jdbc:mysql://localhost:3306/PatientsDB?
createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=create
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MariaDB10Dialect
```

Démarrage de l'application :

[illegible]

→ Vérification de la création de la base de données dans phpmyadmin:





Containing the word:

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> consultation	☆ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
<input type="checkbox"/> medecin	☆ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> patient	☆ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> rendez_vous	☆ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
4 tables	Sum	0	InnoDB	utf8mb4_general_ci	112.0 KiB	0 B

☐ Check all

Création des Repositories qui nous permettra de faire des opérations sur nos entités.

→ PatientRepository :

```

public interface PatientRepository extends JpaRepository<Patient, Long> {

    @Query("select p from Patient p where p.nom like :nom")
    List<Patient> chercherPatients(@Param("nom") String nom);

}
  
```

→ MedecinRepository :

```
public interface MedecinRepository extends JpaRepository<Medecin,Long> {  
  
}
```

→ RendezVousRepository

```
public interface RendezVousRepository extends JpaRepository<RendezVous,Long> {  
    List<RendezVous> findByStatus(StatusRDV staus);  
}
```

→ ConsultationRepository













```
public interface ConsultationRepository extends JpaRepository<Consultation,Long>  
{  
  
}
```

modification de notre application pour l'insertion des données au démarrage.



















```
@SpringBootApplication  
public class OrmJavaApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(OrmJavaApplication.class, args);  
    }  
  
    @Bean  
    CommandLineRunner commandLineRunner(PatientRepository patientRepository, RendezVousRepository rendezVousRepository,  
MedecinRepository medecinRepository, ConsultationRepository consultationRepository) {  
        return args -> {  
  
            //création des patients  
            String patientsnames[] = {"zakaria", "ahmed", "karim", "samia"};  
            for (String patientsname : patientsnames) {  
                Patient patient = new Patient(null, patientsname, new Date(), Math.random() > 0.5, null);  
                patientRepository.save(patient);  
            }  
        }  
    }  
}
```

La base de données après l'Exécution :

→ Table patients:

<div>←T→</div>				id	date_naissance	malade	nom
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	2022-04-01	0	zakaria
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	2022-04-01	0	ahmed
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	2022-04-01	0	karim
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	2022-04-01	1	samia

→ Table rendezvous :

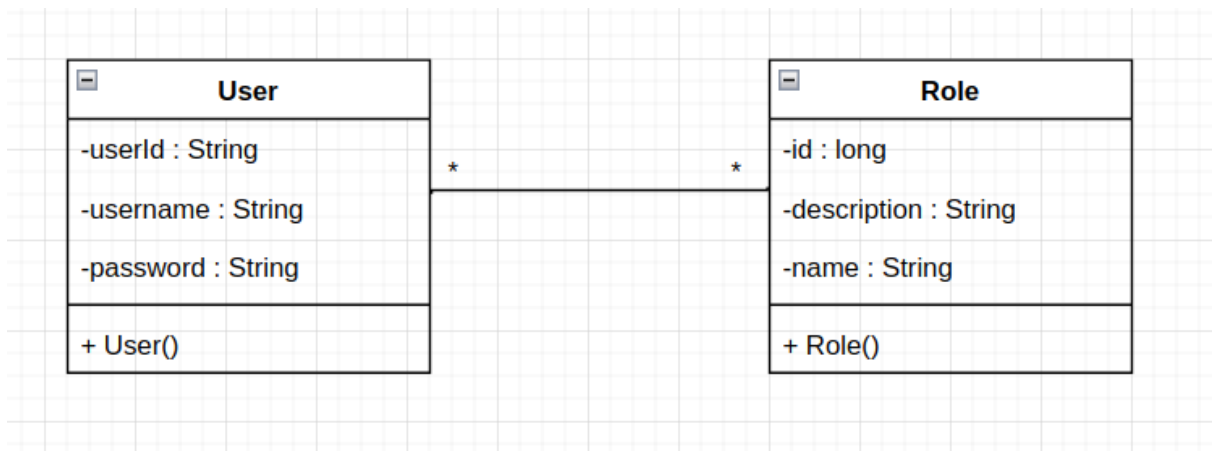
<div><div>←T→</div><div></div></div>				id	date	status	medecin_id	patient_id
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	2022-04-01	0	1	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	2022-04-01	0	2	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	2022-04-01	0	3	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	2022-04-01	0	2	1
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	2022-04-01	0	2	2
<input type="checkbox"/>	 Edit	 Copy	 Delete	6	2022-04-01	0	2	3

test du méthode chercherPatients() :

```
2022-04-01 13:01:14.288 INFO 197873 --- [main] o.s.b.w.embedded.tomcat.TomcatWebSe
2022-04-01 13:01:14.306 INFO 197873 --- [main] ma.enset.ormjava.OrmJavaApplication
=====
affichage des patients qui ont le nom : zakaria
zakaria 1 2022-04-01
=====
```


→ Cas 2 : Gestion des Utilisateurs et rôles

◆ diagramme :



Création des entités :

→ User

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class User {
    @Id
    private String userId;
    @Column(unique = true,length = 20)
    private String username;
    private String password;

    @ManyToMany(mappedBy = "users",fetch = FetchType.EAGER)
    List<Role> roles= new ArrayList<>();
}
```

→ Rôle

```
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Role {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String description;
    @Column(unique = true, length = 20)
    private String name;
    @ManyToMany(fetch = FetchType.EAGER)
    List<User> users = new ArrayList<>();
}
```

Création des Repositories qui nous permettra de faire des opérations sur nos entités.

→ UserRepository :

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

→ RoleRepository :

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

→ Interface UserService :

```

public interface UserService {
    User addNewUser(User user);

    Role addNewRole(Role role);

    User findUserByUserName(String username);

    Role findRolByName(String rolename);

    void addRoleToUser(String username, String rolename);
}

```

→ Implémentation de l'interface UserService :

```

@Service
@Transactional
@AllArgsConstructor
public class UserServiceImpl implements UserService {
    private final UserRepository userRepository;
    private final 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);
    }
    @Override
    public User findUserByUserName(String username) {
        return userRepository.findUserByUsername(username);
    }
    @Override
    public Role findRolByName(String rolename) {
        return roleRepository.findRoleByName(rolename);
    }
    @Override
    public void addRoleToUser(String username, String rolename) {
        User user = findUserByUserName(username);
        Role role = findRolByName(rolename);
        user.getRoles().add(role);
        role.getUsers().add(user);
        userRepository.save(user);
    }
}

```

Ajout du code pour ajouter des utilisateurs et des rôles puis
associer les rôles aux utilisateurs
dans la classe principale :

```

User user = new User();
user.setUsername("zakaria");
user.setPassword("123456");
userService.addNewUser(user);

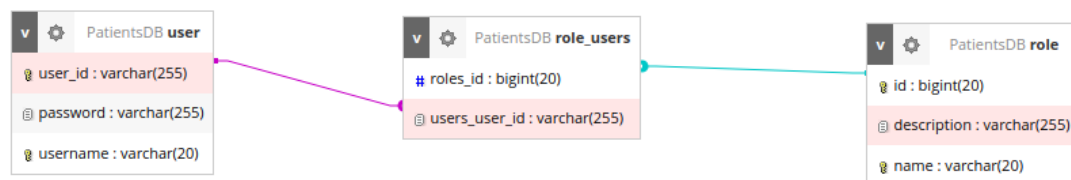
Stream.of("STUDENT", "ADMIN", "USER").forEach(rolename->{
    Role role = new Role();
    role.setName(rolename);
    userService.addNewRole(role);
});

userService.addRoleToUser("zakaria", "ADMIN");

```

Résultat de l'Exécution de l'application :

→ base de données :



→ les rôles créés :

					id	description	name
<input type="checkbox"/>				Delete	1	NULL	STUDENT
<input type="checkbox"/>				Delete	2	NULL	ADMIN
<input type="checkbox"/>				Delete	3	NULL	USER

→ les utilisateurs :

					user_id	password	username
<input type="checkbox"/>				Delete	abdb7e06-e975-4b3f-8f74-24bb04587af7	123456	zakaria
<input type="checkbox"/>				Delete	c86f8aef-7681-4b49-a9a1-d8d274f0c15d	123456	youssef
<input type="checkbox"/>				Delete	d50d99a4-cc92-4e02-9cb9-ca2b169c8d9d	123456	hicham

→ la table généré :

,

roles_id	users_user_id
1	abdb7e06-e975-4b3f-8f74-24bb04587af7