JPA AVEC HIBERNATE

BERNIER Allan 2023

QU'EST CE QUE C'EST ?

UTILISER LA BASE DE DONNÉES SANS JPA HIBERNATE

CREATE

sauvegarder notre entité

READ

Récupérer une ou toutes nos entités

UPDATE

mettre à jour une entité

DELETE

supprimer une entité

m2i Formation

CONNECTION AVEC LA BASE DE DONNÉES

```java
public class UtilConnexion {

    public static Connection seConnecter() throws Exception {

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection("jdbc:mysql://localhost:3306/tuto","root","root");

        } catch (ClassNotFoundException e) {
            throw new Exception("ClassNotFoundException :" + e.getMessage());
        } catch (SQLException e) {
            throw new Exception("SQLException :" + e.getMessage());
        }
    }
}
```

CREATE

```java
try {
    Connection con = UtilConnexion.seConnecter();

    PreparedStatement ps = con.prepareStatement("INSERT INTO user(login, email, password) VALUE( ?, ?, ?);");

    ps.setString(1, u.getLogin());
    ps.setString(2, u.getEmail());
    ps.setString(3, u.getPassword());

    ps.executeUpdate();

    con.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

READ

```java
List<User> res = new ArrayList<>();

try {
    Connection con = UtilConnexion.seConnecter();
    ResultSet rs = con.createStatement().executeQuery("SELECT * FROM user;");
    while ( rs.next() ) {
        res.add(new User(
            rs.getInt("id"),
            rs.getString("email"),
            rs.getString("login"),
            rs.getString("password")
        ));

    }

    rs.close();
    con.close();
} catch (Exception e) {
    e.printStackTrace();
}
return res;
```

```java
User u = null;
try {
    Connection con = UtilConnexion.seConnecter();

    PreparedStatement ps = con.prepareStatement("SELECT * FROM user WHERE id=?;");
    ps.setInt(1, id);
    ResultSet rs = ps.executeQuery();

    if ( rs.next()) {
        u = new User(
            rs.getInt("id"),
            rs.getString("email"),
            rs.getString("login"),
            rs.getString("password")
        );
    }
    rs.close();
    con.close();
} catch (Exception e) {
    e.printStackTrace();
}
return u;
```

UPDATE

```java
try {
    Connection con = UtilConnexion.seConnecter();

    PreparedStatement ps = con.prepareStatement("UPDATE user SET login=?, email=?, password=? WHERE id=?;");
    ps.setString(1, u.getLogin());
    ps.setString(2, u.getEmail());
    ps.setString(3, u.getPassword());
    ps.setInt(4, u.getId());

    ps.executeUpdate();

    con.close();
    return true;

} catch (Exception e) {
    e.printStackTrace();
    return false;
}
```

DELETE

```
try {
    Connection con = UtilConnexion.seConnecter();

    PreparedStatement ps = con.prepareStatement("DELETE FROM user WHERE id=?;");
    ps.setInt(1, id);
    ps.executeUpdate();

    con.close();


} catch (Exception e) {
    e.printStackTrace();
}
```

UTILISATION DE JPA / HIBERNATE

Retour à la page du programme

1. CONFIGURATION

```
<!-- /src/main/ressources/META-INF/persistence.xml  -->

<?xml version="1.0" encoding="UTF-8"?>

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
    version="2.0">

    <persistence-unit name="my-jpa-conf">
        <properties>
            <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver" />
            <property name="javax.persistence.jdbc.url"
                      value="jdbc:mysql://localhost:3306/CoursJpa?useSSL=false&amp;serverTimezone=UTC"/>
            <property name="javax.persistence.jdbc.user" value="root" />
            <property name="javax.persistence.jdbc.password" value="root" />
        </properties>
    </persistence-unit>

</persistence>
```

2. CONNEXION

```
EntityManagerFactory emf = null;
EntityManager em = null;
try {
    emf = Persistence.createEntityManagerFactory("my-jpa-conf");
    em = emf.createEntityManager();

    /**
        QUERIES
    */
} finally {
    if ( em != null ) em.close();
    if ( emf != null ) emf.close();
}
```

3. DECLARATION DE L'ENTITÉE

```java
@Entity( name="entity_name") @Table( name="users")
public class User {

    @Id @GeneratedValue( strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name="first_name")
    private String firstName;

    @Column(name="last_name")
    private String lastName;

    private String email;

    private int age;


    /**
    CONSTRUCTORS
    GETTERS
    SETTERS
    */
}
```

@Entity

Définis le nom de l'entité utilisée par JPA Hibernate

@Table

Le nom de la table dans notre base de données

@Id @GeneratedValue

Id Précise que ce champ est notre clé primaire et GeneratedValue précise la manière dont il s'incrémente

@Column

Sert à choisir le nom du champ dans la base de données

4. REQUETER LA BASE DE DONNÉES

```
EntityManagerFactory emf = null;
EntityManager em = null;
try {
    emf = Persistence.createEntityManagerFactory("my-jpa-cor
    em = emf.createEntityManager();

    User user = em.find(User.class, 1);
    System.out.println(user);
} finally {
    if ( em != null ) em.close();
    if ( emf != null ) emf.close();
}
```

```
EntityManagerFactory emf = null;
EntityManager em = null;
try {
    emf = Persistence.createEntityManagerFactory("my-jpa-conf");
    em = emf.createEntityManager();

    User user = em.find(User.class, 4);
    EntityTransaction trans = em.getTransaction();
    trans.begin();

    user.setFirstName("Toto");
    user.setAge(2);
    em.persist(user);

    trans.commit();

} finally {
    if ( em != null ) em.close();
    if ( emf != null ) emf.close();
}
```

LES DIFFÉRENTES RÉLATIONS

OneToOne

ManyToMany

ManyToOne

OneToMany

ONE TO ONE

```java
@Entity @Table(name = "T_Users")
public class User {

    @Id  @GeneratedValue( strategy=GenerationType.IDENTITY )
    private int idUser;

    @OneToOne( mappedBy = "user" )
    private UserInformations userInformations;

}
```

```java
@Entity @Table(name = "T_UserInformations")
public class UserInformations {

    @OneToOne
    @JoinColumn( name="idUser", nullable=false )
    private User user;

}
```

MANY TO MANY

```java
@Entity @Table(name = "T_Users")
public class User {

    @Id  @GeneratedValue( strategy=GenerationType.IDENTITY )
    private int idUser;

    @ManyToMany
    @JoinTable( name = "T_Users_Roles_Associations",
                joinColumns = @JoinColumn( name = "idUser" ),
                inverseJoinColumns = @JoinColumn( name = "idRole" ) )
    private List<Role> roles = new ArrayList<>();

}
```

```java
@Entity @Table(name="T_Roles")
public class Role {

    @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
    private int idRole;

    @ManyToMany
    @JoinTable( name = "T_Users_Roles_Associations",
                joinColumns = @JoinColumn( name = "idRole" ),
                inverseJoinColumns = @JoinColumn( name = "idUser" ) )
    private List<User> users = new ArrayList<>();

}
```

MANY TO ONE ET ONE TO MANY

```java
@Entity  @Table(name="T_Commands")
public class Command {

    @ManyToOne @JoinColumn(name="idUser", nullable=false)
    private User user;
```

```java
@Entity @Table(name = "T_Users")
public class User {

    @Id  @GeneratedValue( strategy=GenerationType.IDENTITY )
    private int idUser;

    @OneToMany( targetEntity=Command.class, mappedBy="user" )
    private List<Command> commands = new ArrayList<>();
}
```