

# Tp JEE Hibernate : Mapping des associations

**Objectif du TP :** Créer une application qui gère des produits classés par catégories, et pouvant être commandés. L'objectif est de mettre en œuvre les relations **@ManyToOne**, **@OneToOne**, **@ManyToMany**, avec persistance en base de données.

**Entités à modéliser :**

➤ **Catégorie**

- Attributs : id (Long) et nom (String)
- Relation : OneToMany avec Produit

➤ **Produit**

- Attributs : id (Long), nom (String) et prix (double)
- Relations : ManyToOne avec Catégorie et ManyToMany avec Commande

➤ **Commande**

- Attributs : id (Long) et dateCommande (LocalDate)
- Relation : ManyToMany avec Produit

## Étape 1 : Configuration de la base de données

1. Crée une base de données nommée gestion\_produits.
2. Aucun schéma nécessaire : Hibernate créera les tables automatiquement.

## Étape 2 : Création du projet Maven

1. Crée un projet Web Maven.
2. Configure le fichier persistence.xml :

```

<?xml version="1.0" encoding="UTF-8"?>
<persistence version="3.0" xmlns="https://jakarta.ee/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="https://jakarta.ee/xml/ns/persistence https://jakarta.ee/xml/ns/persistence_3_0.xsd">
    <!-- Define Persistence Unit -->
    <persistence-unit name="my_persistence_unit">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <properties>
            <property name="hibernate.connection.url" value="jdbc:mysql://localhost:3306/dbHibernate2?useSSL=false"/>
            <property name="hibernate.connection.username" value="root"/>
            <property name="hibernate.connection.password" value="RootPass"/>
            <property name="hibernate.connection.driver_class" value="com.mysql.cj.jdbc.Driver"/>
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>
            <property name="hibernate.hbm2ddl.auto" value="update"/>
            <property name="hibernate.show_sql" value="true"/>
        </properties>
    </persistence-unit>
</persistence>

```

```

public class JPAUtil {
    private static final EntityManagerFactory emf = Persistence.createEntityManagerFactory("my_persistence_unit");

    public static EntityManager getEntityManager() {
        return emf.createEntityManager();
    }
}

```

### Étape 3 : Créer les entités

```

@Entity
public class Categorie {

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

    private String nom;

    @OneToMany(mappedBy = "categorie", cascade = CascadeType.ALL)
    private Set<Produit> produits = new HashSet<>();
}

```

```

@Entity
public class Produit {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nom;
    private double prix;

    @ManyToOne
    private Categorie categorie;

    @ManyToMany(mappedBy = "produits")
    private Set<Commande> commandes = new HashSet<>();
}

@Entity
public class Commande {
    @Id @GeneratedValue
    private Long id;

    private LocalDate dateCommande;

    @ManyToMany
    @JoinTable(
        name = "commande_produit",
        joinColumns = @JoinColumn(name = "commande_id"),
        inverseJoinColumns = @JoinColumn(name = "produit_id")
    )
    private Set<Produit> produits = new HashSet<>();
}

```

#### Étape 4 : Créer les DAO

Créer un DAO par entité avec :

- ajouter(objet)
- lister()
- trouver(Long id)

**Exemple :**

```

public class CategorieDAO {
    private EntityManager em = JPAUtil.getEntityManager();

    public void ajouter(Categorie c) {
        em.getTransaction().begin();
        em.persist(c);
        em.getTransaction().commit();
    }

    public List<Categorie> lister() {
        return em.createQuery("FROM Categorie", Categorie.class).getResultList();
    }

    public Categorie trouver(Long id) {
        return em.find(Categorie.class, id);
    }
}

```

## Étape 5 : Créer les Servlets

### CategorieServlet :

- GET : affiche formulaire pour ajouter une catégorie et liste les catégories existantes
- POST : enregistre une nouvelle catégorie

### ProduitServlet :

- GET : liste des produits avec leur catégorie
- POST : ajouter un produit en sélectionnant la catégorie

### CommandeServlet :

- GET : affiche formulaire avec liste de produits
- POST : ajoute une commande avec les produits sélectionnés

## Étape 6 : Créer les pages JSP

### categories.jsp

- Formulaire d'ajout d'une nouvelle catégorie
- Liste des catégories existantes

### produits.jsp

- Liste des produits avec leur catégorie

- Formulaire avec champ nom, prix, et select pour choisir la catégorie, ce qui permet d'ajouter un nouveau produit

### **commandes.jsp**

- Formulaire avec checkbox produits
- Liste des commandes avec leurs produits