

TP JPA 01 : Introduction à Jakarta Persistence

Premiers pas avec JPA : Création et gestion d'entités

Emmanuel BRUNO

Université de Toulon, LIS UMR CNRS 7020

emmanuel.bruno@univ-tln.fr

2026-10-14

Table des matières

1 Objectifs	1
2 Étape 1 : Préparer l'environnement ...	1
3 Étape 2 : Créer une première entité ...	2
4 Étape 3 : Configurer l'EntityManager .	3
5 Étape 5 : Activité finale	4

1 Objectifs

- Comprendre les concepts fondamentaux de JPA
- Savoir configurer un projet avec JPA
- Maîtriser les opérations CRUD de base
- Apprendre à modéliser des entités JPA

2 Étape 1 : Préparer l'environnement

Le TP utilisera une base de données H2 en mémoire pour simplifier la configuration. Vous basculerez vers une base de données réelle dans la suite.

<https://repo1.maven.org/maven2/com/h2/database/h2/2.3.232/h2-2.3.232.jar>

```
java -cp h2*.jar \  
    org.h2.tools.Server \  
        -webAllowOthers \  
        -tcpAllowOthers \  
        -pgAllowOthers \  
        -ifNotExists
```

```
-pgAllowOthers \  
-ifNotExists
```

lancer H2 pour accéder à la console web :
<http://localhost:8082> avec une base de données testdb et le user sa sans mot de passe.

1. Créer un projet Java Maven avec IntelliJ ou VSCode :

- Définissez bien les informations suivantes en les adaptant à votre projet :
 - **Group ID** : `com.example.jpa.tps`
 - **Artifact ID** : `tp01`
 - **Version** : `0.1.0-SNAPSHOT`.

2. Ajouter les dépendances Jakarta Persistence et H2 dans le `pom.xml` :

```
<dependencies>  
  <dependency>  
  
    <groupId>jakarta.persistence</groupId>  
  
    <artifactId>jakarta.persistence-api</artifactId>  
  
    <version>3.2.0</version>  
  </dependency>  
  <dependency>  
  
    <groupId>org.hibernate.orm</groupId>
```

```

groupId>
    <artifactId>hibernate-
core</artifactId>
    <version>6.6.4.Final</
version>
</dependency>
<dependency>
    <groupId>com.h2database</
groupId>
    <artifactId>h2</artifactId>
    <version>2.3.232</version>
    <scope>runtime</scope>
</dependency>
</dependencies>

```

3. Configurer le fichier persistence.xml :

- Créez le fichier dans src/main/resources/META-INF/persistence.xml et ajoutez :

```

<persistence xmlns="https://
jakarta.ee/xml/ns/persistence"
    xmlns:xsi="http://www.w
3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://
jakarta.ee/xml/ns/persistence
https://jakarta.ee/xml/ns/
persistence/persistence_3_0.
xsd"
    version="3.0">

    <persistence-unit
name="tpJakartaUnit">
        <class>fr.univtln.bruno.jpa.Person</
class>

        <properties>
            <property name="jakarta.persistence.jdbc.driver"
value="org.h2.Driver" />
            <property name="jakarta.persistence.jdbc.url"
value="jdbc:h2:tcp://localhost/
~/testdb" />
            <property name="jakarta.persistence.jdbc.user"

```

```

value="sa" />
            <property name="jakarta.persistence.jdbc.
value="" />

            <!--
Database creation: none, create,
drop-and-create, drop -->
            <property name="jakarta.persistence.schem
generation.database.action"
                value="drop-
and-create" />

            <!-- Script
generation: none, create, drop-
and-create, drop -->
            <property name="jakarta.persistence.schem
generation.scripts.action"
                value="drop-
and-create"/>

            <!-- The paths of
scripts generated by JPA -->
            <property name="jakarta.persistence.schem
generation.scripts.create-
target"
                value="generated-create.ddl" />
            <property name="jakarta.persistence.schem
generation.scripts.drop-target"
                value="generated-drop.ddl" />
        </properties>
    </persistence-unit>

</persistence>

```

3 Étape 2 : Créer une première entité

1. Créer une classe Person dans le package :

- Cette classe représente une personne avec un identifiant, un nom et un âge.
- Ajoutez les annotations de base Jakarta Persistence.

Exemple de code :

```
package
com.example.tp_jakarta.entities;

import jakarta.persistence.*;

@Entity
@Table(name = "persons")
public class Person {
    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String name;

    @Column
    private int age;

    public Person() {}

    public Person(String name, int
age) {
        this.name = name;
        this.age = age;
    }

    public Long getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public void setName(String
name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }
}
```

```
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

2. **Ajouter la classe au persistence.xml :**
Vérifiez que la classe Person est bien déclarée dans le fichier persistence.xml.

4 Étape 3 : Configurer l'EntityManager

1. **Créer une classe JpaUtil dans le package com.example.tp_jakarta.utils :**
 - Elle fournira un EntityManagerFactory pour simplifier l'accès au gestionnaire d'entités.

Exemple :

```
import jakarta.persistence.EntityManagerFactory;
import
jakarta.persistence.Persistence;

public class JpaUtil {
    private static final
EntityManagerFactory emf =
Persistence.createEntityManagerFactory("tpJakarta")

    public
static EntityManagerFactory
getEntityManagerFactory() {
        return emf;
    }

    public static void close() {
        emf.close();
    }
}
```

2. **Créer une classe Main pour tester l'enregistrement d'une entité :**

- Insérez une personne dans la base de données en utilisant l'EntityManager. annotations Jakarta Persistence pour la modélisation d'une entité sans association.

Exemple :

```
import jakarta.persistence.EntityManager;

public class Main {
    public static void
    main(String[] args) {
        EntityManager em = JpaUtil.getEntityManagerFactory().createEntityManager();

        // Persistence d'une entité
        em.getTransaction().begin();
        Person person = new
        Person("Alice", 30);
        em.persist(person);

        em.getTransaction().commit();

        // Lecture de l'entité
        persistée
        Person foundPerson =
        em.find(Person.class,
        person.getId());
        System.out.println("Person
        found: " + foundPerson.getName() +
        ", Age: " + foundPerson.getAge());

        em.close();
        JpaUtil.close();
    }
}
```

Ajoutez des personnes supplémentaires, des modifications et des suppressions pour tester les opérations CRUD.

5 Étape 5 : Activité finale

Créez l'entité suivante pour enrichir votre modèle :

Book : - Représente un livre avec un titre, un auteur, une année de publication, un ISBN et une description. - Expérimentez avec les