**Hibernate 6 Example Step-By-Step**

**1. Create a Simple Maven Project**

**2. Add jar Dependencies to pom.xml**

Next, let's add the required dependencies to the pom.xml file:

<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.32</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>6.1.7.Final</version>

</dependency>

Here is the complete pom.xml file for your reference:

<dependencies>

<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.32</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>6.1.7.Final</version>

</dependency>

</dependencies>

<build>

<sourceDirectory>src/main/java</sourceDirectory>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.5.1</version>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**3. Creating the JPA Entity Class(Persistent class)**

Let's create a *Book* persistent class that is mapped to a database "books" table:

package net.pack.hibernate.entity;

import jakarta.persistence.\*;

@Entity

@Table(name = "books")

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private String description;

public Book(){

}

public Book(String name, String description) {

this.name = name;

this.description = description;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

}

**4. Create a Hibernate Configuration file - hibernate.cfg.xml**

Before creating Hibernate configuration file, make sure that you create a *demo* database in the MySQL server.

create database demo

Let's create an XML file named *hibernate.cfg.xml* under the */resources* folder and write the following code in it.

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- JDBC Database connection settings -->

<property name="connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="connection.url">jdbc:mysql://localhost:3306/demo?useSSL=false</property>

<property name="connection.username">root</property>

<property name="connection.password">root</property>

<!-- JDBC connection pool settings ... using built-in test pool -->

<property name="connection.pool\_size">1</property>

<!-- Echo the SQL to stdout -->

<property name="show\_sql">true</property>

<!-- Set the current session context -->

<property name="current\_session\_context\_class">thread</property>

<!-- Drop and re-create the database schema on startup -->

<property name="hbm2ddl.auto">create-drop</property>

<!-- dbcp connection pool configuration -->

<property name="hibernate.dbcp.initialSize">5</property>

<property name="hibernate.dbcp.maxTotal">20</property>

<property name="hibernate.dbcp.maxIdle">10</property>

<property name="hibernate.dbcp.minIdle">5</property>

<property name="hibernate.dbcp.maxWaitMillis">-1</property>

<mapping class="net.pack.hibernate.entity.Book" />

</session-factory>

</hibernate-configuration>

The above configuration file contains information about the database and mapping file. Conventionally, its name should be *hibernate.cfg.xml*.

In Hibernate 6, we don't have to specify the dialect because Hibernate 6 will automatically configure it based on the database JDBC driver that we add to the classpath.

**5. Create a Hibernate Utility Class**

Next, let's create a helper class to bootstrap hibernate *SessionFactory*. In most Hibernate applications, the *SessionFactory* should be instantiated once during application initialization. The single instance should then be used by all code in a particular process, and any *Session* should be created using this single SessionFactory.  
  
The *SessionFactory* is thread-safe and can be shared; a *Session* is a single-threaded object.

Let's create *HibernateUtil* class to configure singleton *SessionFactory* and use it throughout the application:

package net.pack.hibernate.util;

import org.hibernate.SessionFactory;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class HibernateUtil {

private static StandardServiceRegistry registry;

private static SessionFactory sessionFactory;

public static SessionFactory getSessionFactory() {

if (sessionFactory == null) {

try {

// Create registry

registry = new StandardServiceRegistryBuilder().configure().build();

// Create MetadataSources

MetadataSources sources = new MetadataSources(registry);

// Create Metadata

Metadata metadata = sources.getMetadataBuilder().build();

// Create SessionFactory

sessionFactory = metadata.getSessionFactoryBuilder().build();

} catch (Exception e) {

e.printStackTrace();

if (registry != null) {

StandardServiceRegistryBuilder.destroy(registry);

}

}

}

return sessionFactory;

}

public static void shutdown() {

if (registry != null) {

StandardServiceRegistryBuilder.destroy(registry);

}

}

}

**6 Create the main App class and Run an Application**

Here is the main *App* class which is used to connect the MySQL database and persist the *Student* object in a database table.

package net.pack.hibernate;

import java.util.List;

import net.pack.hibernate.entity.Book;

import org.hibernate.Session;

import org.hibernate.Transaction;

import net.pack.hibernate.util.HibernateUtil;

public class App {

public static void main(String[] args) {

Book book = new Book("Core Java", "Learn Core Java with Coding Examples");

Book book1 = new Book("Learn Hibernate", "Learn Hibernate with building projects");

Transaction transaction = null;

try (Session session = HibernateUtil.getSessionFactory().openSession()) {

// start a transaction

transaction = session.beginTransaction();

// save the book objects

session.persist(book);

session.persist(book1);

// commit transaction

transaction.commit();

} catch (Exception e) {

if (transaction != null) {

transaction.rollback();

}

e.printStackTrace();

}

try (Session session = HibernateUtil.getSessionFactory().openSession()) {

List<Book> books = session.createQuery("from Book", Book.class).list();

books.forEach(b -> {

System.out.println("Print book name : " + b.getName());

});

} catch (Exception e) {

if (transaction != null) {

transaction.rollback();

}

e.printStackTrace();

}

}

}

Note that we are using *Session.persist()* method to save the Student entity object into the database.