Java 9

Hibernate 5

Mysql connector 5

**Database : MySQL**

CREATE DATABASE 'bookstore';

USE 'bookstore';

CREATE TABLE `book` (

`book\_id` int(11) NOT NULL AUTO\_INCREMENT,

`title` varchar(128) NOT NULL,

`author` varchar(45) NOT NULL,

`price` float NOT NULL,

PRIMARY KEY (`book\_id`),

UNIQUE KEY `book\_id\_UNIQUE` (`book\_id`),

UNIQUE KEY `title\_UNIQUE` (`title`)

) ENGINE=InnoDB;

**Creating project**

Download maven plugin

Create java project

Configure to maven

**Maven dependencies in POM file**

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.2.6.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.40</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-impl -->

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-impl</artifactId>

<version>2.2.11</version>

</dependency>

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-core</artifactId>

<version>2.2.11</version>

</dependency>

<dependency>

<groupId>javax.transaction</groupId>

<artifactId>javax.transaction-api</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>org.hibernate.javax.persistence</groupId>

<artifactId>hibernate-jpa-2.0-api</artifactId>

<version>1.0.1.Final</version>

</dependency>

</dependencies>

**Hibernate.cfg.file : location source folder**

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

<!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>

<!-- Database connection settings -->

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

<property name="connection.url">jdbc:mysql://localhost:3306/bookstore</property>

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

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

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

<mapping class="com.example.hibernate.Book" />

</session-factory>

</hibernate-configuration>

**Book.java Entity class**

package com.example.hibernate;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "book")

public class Book {

private long id;

private String title;

private String author;

private float price;

public Book() {

}

@Id

@Column(name = "book\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

}

**BookManager.java**

package com.example.hibernate;

//--add-modules java.xml.bind : add this in run configurations -> Arguements ->- vm arguments

import java.util.List;

import javax.persistence.TypedQuery;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class BookManager {

protected SessionFactory sessionFactory;

protected void setup() {

final StandardServiceRegistry registry = new StandardServiceRegistryBuilder()

.configure() // configures settings from hibernate.cfg.xml

.build();

try {

sessionFactory = new MetadataSources(registry).buildMetadata().buildSessionFactory();

System.out.println("connection = "+sessionFactory.isOpen());

} catch (Exception ex) {

StandardServiceRegistryBuilder.destroy(registry);

}

}

protected void exit() {

sessionFactory.close();

}

protected void create() {

Book book = new Book();

book.setTitle("CompleteReference Java");

book.setAuthor("Herbert sheldt");

book.setPrice(50.59f);

Session session = sessionFactory.openSession();

session.beginTransaction();

session.save(book);

session.getTransaction().commit();

session.close();

/\*Book book = new Book();

book.setTitle("Effective Java");

book.setAuthor("Joshua Bloch");

book.setPrice(32.59f);

Session session = sessionFactory.openSession();

session.beginTransaction();

session.save(book);

session.getTransaction().commit();

session.close();\*/

}

protected void search(long bookID) {

System.out.println("first session");

Session session = sessionFactory.openSession();

Book book = session.get(Book.class, bookID);

System.out.println("Title: " + book.getTitle());

System.out.println("Author: " + book.getAuthor());

System.out.println("Price: " + book.getPrice());

session.close();

System.out.println("second session");

Session session2 = sessionFactory.openSession();

Book book2 = session2.get(Book.class, bookID);

System.out.println("Title: " + book2.getTitle());

System.out.println("Author: " + book2.getAuthor());

System.out.println("Price: " + book2.getPrice());

session.close();

}

protected void read() {

Session session = sessionFactory.openSession();

TypedQuery<Book> query = session.createQuery("FROM Book");

List<Book> result = query.getResultList();

/\*List<Book> bookList = new ArrayList();

List<Book> list = session.createQuery("FROM Book").getResultList();\*/

result.forEach(book -> System.out.println(book.getTitle()));

session.close();

}

protected void update() {

Book book = new Book();

book.setId(1);

book.setTitle("Ultimate Java Programming");

book.setAuthor("Nam Ha Minh");

book.setPrice(19.99f);

Session session = sessionFactory.openSession();

session.beginTransaction();

session.update(book);

session.getTransaction().commit();

session.close();

}

protected void delete() {

Book book = new Book();

book.setId(2);

Session session = sessionFactory.openSession();

session.beginTransaction();

session.delete(book);

session.getTransaction().commit();

session.close();

}

public static void main(String[] args) {

BookManager manager = new BookManager();

manager.setup();

//test one by one by uncommenting

//manager.create();

//manager.update();

//manager.read();

//manager.delete();

//without second cache

manager.search(1);

manager.exit();

}

}

**Note: --add-modules java.xml.bind : add this in varibale argument - vm arguments**

**For java9**