Hands-On: ORM Implementation with Hibernate for Java Applications

For this exercise, you will create a simple application that manages the books' information. You will use Hibernate to create and manage the database tables. Follow the steps below to complete the exercise:

Step 1: Create a Java project in your IDE (Eclipse or IntelliJ IDEA) and add the following dependencies to your pom.xml file:

```
<dependencies>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>5.4.32.Final</version>
    </dependency>
    <dependency>
        <groupId>javax.persistence</groupId>
        <artifactId>persistence-api</artifactId>
        <version>2.2</version>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <version>1.4.200</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Step 2: Create a new Java class called "Book" with the following fields:

```
@Entity
@Table(name = "Book")
public class Book {

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

    @Column(name = "Title")
    private String title;

    @Column(name = "Author")
    private String author;

    @Column(name = "Price")
    private double price;

    // Constructors, getters, and setters
}
```

Step 3: Create a Hibernate configuration file called "hibernate.cfg.xml" in the "src/main/resources" folder and add the following configuration:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
   <session-factory>
      <property name="hibernate.connection.driver_class">org.h2.Driver</property>
      <property name="hibernate.connection.url">jdbc:h2:mem:test</property>
      <property name="hibernate.connection.username">sa</property>
      <property name="hibernate.connection.password"></property>
      <property name="hibernate.dialect">org.hibernate.dialect.H2Dialect</property>
      <property name="hibernate.hbm2ddl.auto">create</property>
      <property name="hibernate.show_sql">true</property>
      <property name="hibernate.format_sql">true</property>
      <mapping class="com.example.Book"/>
   </session-factory>
</hibernate-configuration>
```

Step 4: Create a new Java class called "App" and add the following code to create a new book:

```
public class App {

   public static void main(String[] args) {
      SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
      Session session = sessionFactory.openSession();

      Book book = new Book();
      book.setTitle("The Great Gatsby");
      book.setAuthor("F. Scott Fitzgerald");
      book.setPrice(25.0);

      Transaction transaction = session.beginTransaction();
      session.save(book);
      transaction.commit();

      session.close();
      sessionFactory.close();
   }
}
```

Step 5: Run the "App" class to create a new book in the database.

Step 6: Add the following code to the "App" class to retrieve the book from the database:

```
public class App {
```

```
    public static void main(String[] args) {
        SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();

        // Create a new book
        Book book = new Book();
        book.setTitle("The Great Gatsby");
        book.setAuthor("F. Scott Fitzgerald");
        book.setPrice(25.0);

        // Save the book to the database
        Transaction transaction = session.beginTransaction();
        session.save(book);
        transaction.commit();

        // Retrieve the book from the database
        int bookId = book.getId();
        Book retrievedBook = session.get(Book.class, bookId);
        System.out.println("Retrieved Book: " + retrievedBook);

        session.close();
        sessionFactory.close();
    }
}
```

Step 7: Run the "App" class to retrieve the book from the database.

Step 8: Add the following code to the "App" class to update the book's price:

```
public class App {

    public static void main(String[] args) {
        SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();

        // Create a new book
        Book book = new Book();
        book.setTitle("The Great Gatsby");
        book.setAuthor("F. Scott Fitzgerald");
        book.setPrice(25.0);

        // Save the book to the database
        Transaction transaction = session.beginTransaction();
        session.save(book);
        transaction.commit();

        // Retrieve the book from the database
        int bookId = book.getId();
        Book retrievedBook = session.get(Book.class, bookId);
        System.out.println("Retrieved Book: " + retrievedBook);

        // Update the book's price
        transaction = session.beginTransaction();
        retrievedBook.setPrice(30.0);
```

```
        session.update(retrievedBook);
        transaction.commit();

        session.close();
        sessionFactory.close();
    }
}
```

Step 9: Run the "App" class to update the book's price.

Step 10: Add the following code to the "App" class to delete the book:

```
public class App {

    public static void main(String[] args) {
        SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();

        // Create a new book
        Book book = new Book();
        book.setTitle("The Great Gatsby");
        book.setAuthor("F. Scott Fitzgerald");
        book.setPrice(25.0);

        // Save the book to the database
        Transaction transaction = session.beginTransaction();
        session.save(book);
        transaction.commit();

        // Retrieve the book from the database
        int bookId = book.getId();
        Book retrievedBook = session.get(Book.class, bookId);
        System.out.println("Retrieved Book: " + retrievedBook);

        // Update the book's price
        transaction = session.beginTransaction();
        retrievedBook.setPrice(30.0);
        session.update(retrievedBook);
        transaction.commit();

        // Delete the book
        transaction = session.beginTransaction();
        session.delete(retrievedBook);
        transaction.commit();

        session.close();
        sessionFactory.close();
    }
}
```

Step 11: Run the "App" class to delete the book.

Congratulations! You have successfully completed the hands-on exercise based on ORM implementation with Hibernate. In this exercise, you have created a Java project, added Hibernate dependencies, created a database table, created a Java class to represent the table, and used Hibernate to create, retrieve, update, and delete a book from the database. This exercise provides a basic understanding of how Hibernate