Hands-On Exercise: Implementing a CRUD Application with Spring ORM (Hibernate) and MySQL

Objective:

Create a simple CRUD (Create, Read, Update, Delete) application using **Spring ORM** with **Hibernate** as the persistence provider, and **MySQL** as the database.

Exercise Steps:

Part 1: Setting Up the MySQL Database

1. **Create a new database in MySQL** called spring_orm_crud
2. **Design a table User** with the following fields:
    - id (BIGINT, Primary Key, Auto Increment)
    - username (VARCHAR)
    - email (VARCHAR)

Part 2: Setting Up the Spring Project

3. **Create a new Spring project**. What steps would you follow to create a Spring-based project with the necessary dependencies for Hibernate and MySQL in pom.xml?
4. **Add dependencies** for Spring ORM, Hibernate, MySQL Connector, and Spring Transaction Management in your pom.xml file. **What are the key dependencies needed for this setup?**

Part 3: Configuring Hibernate and DataSource

5. **Configure the DataSource** to connect to the spring_orm_crud database.
6. **Set up the SessionFactory** in Spring to work with Hibernate. What properties do you need to define in the SessionFactory to connect Hibernate with the MySQL database?
7. **Enable transaction management** in Spring. How do you configure Spring to manage transactions for Hibernate?

Part 4: Creating the User Entity

8. **Create a User entity class** annotated with JPA annotations (@Entity, @Id, etc.). What annotations and fields are needed to map the User class to the User table in MySQL?

Part 5: Implementing the DAO Layer

9. **Create a DAO class (UserDao)** with methods for CRUD operations. Define the saveUser, getUser, updateUser, and deleteUser methods using Hibernate?
10. **Implement transaction management** for the CRUD methods. How do you ensure that Hibernate transactions are properly committed and rolled back in case of errors?

Part 6: Writing the Service Layer

11. **Write a service class (UserService)** to expose the CRUD operations. Expose the methods from UserDao in a service class.

Part 7: Writing the Main Application

12. **Write a main class** to test your application. How would you create instances of User and interact with the database using the UserService class?
13. **Test the CRUD operations**:
    - Create a new User in the database.
    - Read the details of a User.
    - Update the User's email address.
    - Delete a User from the database.

Part 8: Verifying the Transactions

14. **Test transaction rollback**. What would happen if an error occurs while saving a user (e.g., trying to save a user with a null email)? How does Spring and Hibernate handle transaction rollback for such errors?

Part 9: Exception Handling

15. **Handle exceptions in the CRUD operations**. How would you handle exceptions (such as **DataIntegrityViolationException**) in the DAO layer and propagate them to the service layer?