Spring-ORM and JDBC

1. **What is Spring ORM ?**

   ▼ **Ans**

   - **It is a framework which used for data integration this framework can integrate with varies ORM framework and simplifies object relation mapping.**
   - **Example → Hibernate, JPS etc.**

2. **What is Spring Hibernate Template ?**

   ▼ **Ans**

   - **It is a template provided for ORM by integrating Spring and Hibernate.**
   - **Using Hibernate Template, We can avoid the steps before the saving the data and after saving the data which was used in hibernate.**

3. **Code - 1 Steps to Create a project Using Spring ORM ?**

   ▼ **Ans**

   1. **Create a Simple Maven Project.**
   2. **Add following dependencies in pom.xml** → (We will it in Satish Sir Git-Hub (FileName → "pomforSpringHibernate.txt") or link → "https://github.com/sathishnyadav/supporting-files/blob/master/pomforSpringHibernate.txt")

      a. **Commons-dbcp**

      b. **Spring ORM**

      c. **Spring Context**

      d. **Hibernate core Reloation**

      e. **MySQL-Connector-java**

      ▼ **pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocatio
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.jsp</groupId>
  <artifactId>Spring-Hibernate-Demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>commons-dbcp</groupId>
      <artifactId>commons-dbcp</artifactId>
      <version>1.4</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.18</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>5.6.15.Final</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-orm -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-orm</artifactId>
      <version>5.3.18</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.28</version>
    </dependency>
  </dependencies>
</project>
```

3. **Create a XML configuration file to configure the Hibernate Template.**
4. **Create Entity Class.**

▼ **org.jsp.dto**

▼ **User.java**

```
package org.jsp.dto;

public class User {
  private int id;
  private String name;
  private long phone;
  private String password;

  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 long getPhone() {
    return phone;
  }

  public void setPhone(long phone) {
    this.phone = phone;
  }

  public String getPassword() {
    return password;
  }

  public void setPassword(String password) {
    this.password = password;
  }

}
```

5. **Create an hibernate mapping file to map the Entity class with data table.**
6. **Add the mapping resources in SessionFactory bean.**

▼ **applicationContext.xml** → (We will it in Satish Sir Git-Hub (FileName → “applicationContext.xml”) or link → “https://github.com/sathishnyadav/supporting-files/blob/master/applicationContext.xml”)

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:tx="http://www.springframework.org/schema/tx" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:p="http://www.springframework.org/schema/p"
  xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd
```

```
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-3.0.xsd">


  <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"></property>
    <property name="url"
      value="jdbc:mysql://localhost:3306/springOrm_demo?createDatabaseIfNotExist=true"></property>
    <property name="username" value="root"></property>
    <property name="password" value="admin"></property>
  </bean>
  <bean id="sessionFactory"
    class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource"></property>
    <property name="hibernateProperties">
      <props>
        <prop key="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</prop>
        <prop key="hibernate.hbm2ddl.auto">update</prop>
        <prop key="hibernate.show_sql">true</prop>
      </props>
    </property>
    <property name="mappingResources">
      <list>
        <value>user.hbm.xml</value>
      </list>
    </property>
  </bean>

  <bean id="hibernateTemplate" class="org.springframework.orm.hibernate5.HibernateTemplate">
    <property name="sessionFactory" ref="sessionFactory"></property>
    <property name="checkWriteOperations" value="true"></property>
  </bean>

  <tx:annotation-driven />
  <bean id="transactionManager" class="org.springframework.orm.hibernate5.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
  </bean>
  <bean id="dao" class="org.jsp.dao.UserDao">
    <property name="hibernateTemplate" ref="hibernateTemplate" />
  </bean>
</beans>
```

▼ user.hbm.xml → (We will get it in Satish Sir Git-hub)

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="org.jsp.dto.User" table="user">
    <id name="id" column="id">
      <generator class="identity"></generator>
    </id>
    <property name="name" column="name" />
    <property name="phone" column="phone" />
    <property name="password" column="password" />
  </class>
</hibernate-mapping>
```

▼ org.jsp.dao

▼ UserDao.java

```
package org.jsp.dao;

import java.util.List;

import org.jsp.dto.User;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;
import org.springframework.transaction.annotation.Transactional;

public class UserDao extends HibernateDaoSupport {
  @Transactional
  public User Saveuser(User user) {
    getHibernateTemplate().save(user);
    return user;
  }

  @Transactional
  public User UpdateUser(User user) {
    getHibernateTemplate().update(user);
    return user;
  }

  public User getUserById(int id) {
    return getHibernateTemplate().get(User.class, id);
  }

  @Transactional
  public boolean deleteUser(int id) {
    User u = getUserById(id);
    if (u != null) {
      getHibernateTemplate().delete(u);
      return true;
    }
    return false;
  }

  public List<User> getAllUsers() {
    return getHibernateTemplate().loadAll(User.class);
  }
}
```

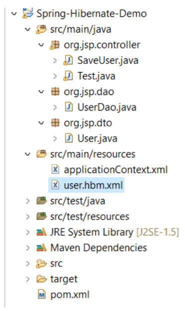
org.jsp.controller

SaveUser.java

```
package org.jsp.controller;

import org.jsp.dao.UserDao;
import org.jsp.dto.User;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SaveUser {
  public static void main(String[] args) {
    User user = new User();
    user.setName("ABC");
    user.setPhone(99999);
    user.setPassword("A13456");
    ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
    UserDao dao = context.getBean(UserDao.class);
    dao.Saveuser(user);
  }
}
```

Test.java

```
package org.jsp.controller;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.orm.hibernate5.HibernateTemplate;

public class Test
{
  public static void main(String[] args) {
    ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
    HibernateTemplate template = context.getBean(HibernateTemplate.class);
    System.out.println(template);
  }
}
```



```
Spring-Hibernate-Demo
  src/main/java
    org.jsp.controller
      SaveUser.java
      Test.java
    org.jsp.dao
      UserDao.java
    org.jsp.dto
      User.java
  src/main/resources
    applicationContext.xml
    user.hbm.xml
  src/test/java
  src/test/resources
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
  target
  pom.xml
```

1. **What is JdbcTemplate ?**

   ▼ Ans

   - **It is a class which is integrating JDBC API springframework. JdbcTemplate provides methods using which we can directly execute handling, establish the connection, load and register, creating the statement and closing costly recourses.**

2. **What is Anonymous class ?**

   ▼ Ans

   - **It is a class but it does not have any name. if we use nextLine() for string we have to call 2 times when we pass string after int long, float..**

3. **Spring-JDBC Project ?**

   ▼ Ans

   ▼ pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="h
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.jsp</groupId>
  <artifactId>Spring-JDBC</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.18</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-orm -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-orm</artifactId>
      <version>5.3.18</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.28</version>
    </dependency>
  </dependencies>
</project>
```

   ▼ src/main/resource

   ▼ spring-jdbc.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:tx="http://www.springframework.org/schema/tx" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:p="http://www.springframework.org/schema/p"
  xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.x
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-3.0.xsd">
  <bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://localhost:3306/spring_jdbc"></property>
    <property name="username" value="root"></property>
    <property name="password" value="admin"></property>
  </bean>
  <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource"></property>
  </bean>
</beans>
```

src/main/java

org.jsp

CreateTable.java

```
package org.jsp;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.jdbc.core.JdbcTemplate;

public class CreateTable {
  public static void main(String[] args) {
    ApplicationContext context = new ClassPathXmlApplicationContext("spring-jdbc.xml");
    JdbcTemplate template = context.getBean(JdbcTemplate.class);
    template.execute(
        "create table user(id int not null,name varchar(45) " + "null,phone bigint(20),primary key(id))");
  }
}
```

User.java

```
package org.jsp;

public class User {
  private int id;
  private String name;
  private long phone;

  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 long getPhone() {
    return phone;
  }

  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

SaveUser.java

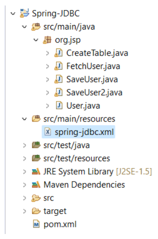
```
package org.jsp;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.jdbc.core.JdbcTemplate;

public class SaveUser {
  public static void main(String[] args) {
    ApplicationContext context = new ClassPathXmlApplicationContext("spring-jdbc.xml");
    JdbcTemplate template = context.getBean(JdbcTemplate.class);
    template.execute("insert into user values(1,'ABC',888)");
  }
}
```

SaveUser2.java

```
package org.jsp;

import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementCallback;

public class SaveUser2 {
  public static void main(String[] args) {
    String qry = "insert into user values(?,?,?)";
    ApplicationContext context = new ClassPathXmlApplicationContext("spring-jdbc.xml");
    JdbcTemplate template = context.getBean(JdbcTemplate.class);
    int r = template.execute(qry, new MyPSCB());
    System.out.println(r + " rows are affected");
  }
}

class MyPSCB implements PreparedStatementCallback<Integer> {

  public Integer doInPreparedStatement(PreparedStatement ps) throws SQLException, DataAccessException {
    ps.setInt(1, 4);
    ps.setString(2, "XYXZ");
    ps.setLong(3, 88888L);
    return ps.executeUpdate();
  }

}
```

FetchUser.java

```
package org.jsp;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.ResultSetExtractor;

public class FetchUser {
  public static void main(String[] args) {
    String qry = "select * from user where id=1";
    ApplicationContext context = new ClassPathXmlApplicationContext("spring-jdbc.xml");
    JdbcTemplate template = context.getBean(JdbcTemplate.class);
    User u = template.query(qry, new MyRSE());
    System.out.println("ID : " + u.getId());
    System.out.println("Name : " + u.getName());
    System.out.println("Phone : " + u.getPhone());
  }
}

class MyRSE implements ResultSetExtractor<User> {
  public User extractData(ResultSet rs) throws SQLException, DataAccessException {
    User u = new User();
    while (rs.next()) {
      u.setId(rs.getInt(1));
      u.setName(rs.getString(2));
      u.setPhone(rs.getLong(3));
    }
    return u;
  }
}
```



```
Spring-JDBC
  src/main/java
    org.jsp
      CreateTable.java
      FetchUser.java
      SaveUser.java
      SaveUser2.java
      User.java
  src/main/resources
    spring-jdbc.xml
  src/test/java
  src/test/resources
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
  target
  pom.xml
```