Spring 编程式事务管理

编程式事务管理方法允许你在对你的源代码编程的帮助下管理事务。这给了你极大地灵活性，但是它很难维护。

在我们开始之前，至少要有两个数据库表，在事务的帮助下我们可以执行多种 CRUD 操作。

以 **Student** 表为例，用下述 DDL 可以在 MySQL TEST 数据库中创建该表：

```
CREATE TABLE Student(
   ID   INT NOT NULL AUTO_INCREMENT,
   NAME VARCHAR(20) NOT NULL,
   AGE  INT NOT NULL,
   PRIMARY KEY (ID)
);
```

第二个表是 **Marks**，用来存储基于年份的学生的标记。这里 **SID** 是 Student 表的外键。

```
CREATE TABLE Marks(
   SID INT NOT NULL,
   MARKS  INT NOT NULL,
   YEAR   INT NOT NULL
);
```

让我们直接使用 *PlatformTransactionManager* 来实现编程式方法从而实现事务。要开始一个新事务，你需要有一个带有适当的 transaction 属性的 *TransactionDefinition* 的实例。这个例子中，我们使用默认的 transaction 属性简单的创建了 *DefaultTransactionDefinition* 的一个实例。

当 TransactionDefinition 创建后，你可以通过调用 *getTransaction()* 方法来开始你的事务，该方法会返回 *TransactionStatus* 的一个实例。 *TransactionStatus* 对象帮助追踪当前的事务状态，并且最终，如果一切运行顺利，你可以使

用 *PlatformTransactionManager* 的 *commit()* 方法来提交这个事务，否则的话，你可以使用 *rollback()* 方法来回滚整个操作。

现在让我们编写我们的 Spring JDBC 应用程序，它能够在 Student 和 Mark 表中实现简单的操作。让我们适当的使用 Eclipse IDE，并按照如下所示的步骤来创建一个 Spring 应用程序：

步骤	描述
1	创建一个名为 *SpringExample* 的项目，并在创建的项目中的 **src** 文件夹下创
2	使用 *Add External JARs* 选项添加必需的 Spring 库，解释见 *Spring Hello*
3	在项目中添加 Spring JDBC 指定的最新的库 **mysql-connector-java.jar**，**org.springframework.jdbc.jar** 和 **org.springframework.transaction.jar**。 们。
4	创建 DAO 接口 *StudentDAO* 并列出所有需要的方法。尽管它不是必需的并且你可 写 *StudentJDBCTemplate* 类，但是作为一个好的实践，我们还是做吧。
5	在 *com.tutorialspoint* 包下创建其他必需的 Java 类 *StudentMarks*，*Studen StudentJDBCTemplate* 和 *MainApp*。如果需要的话，你可以创建其他的 POJO 类
6	确保你已经在 TEST 数据库中创建了 **Student** 和 **Marks** 表。还要确保你的 M 出的用户名和密码可以读/写访问数据库。
7	在 **src** 文件夹下创建 Beans 配置文件 *Beans.xml* 。
8	最后一步是创建所有 Java 文件和 Bean 配置文件的内容并按照如下所示的方法运

下面是数据访问对象接口文件 **StudentDAO.java** 的内容：

```
package com.tutorialspoint;
import java.util.List;
import javax.sql.DataSource;
public interface StudentDAO {
   /**
    * This is the method to be used to initialize
    * database resources ie. connection.
    */
   public void setDataSource(DataSource ds);
   /**
    * This is the method to be used to create
    * a record in the Student and Marks tables.
    */
   public void create(String name, Integer age, Integer marks, Integer year);
   /**
    * This is the method to be used to list down
    * all the records from the Student and Marks tables.
    */
```

```
   public List<StudentMarks> listStudents();
}
```

下面是 **StudentMarks.java** 文件的内容：

```
package com.tutorialspoint;
public class StudentMarks {
   private Integer age;
   private String name;
   private Integer id;
   private Integer marks;
   private Integer year;
   private Integer sid;
   public void setAge(Integer age) {
      this.age = age;
   }
   public Integer getAge() {
      return age;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getName() {
      return name;
   }
   public void setId(Integer id) {
      this.id = id;
   }
   public Integer getId() {
      return id;
   }
   public void setMarks(Integer marks) {
      this.marks = marks;
   }
   public Integer getMarks() {
      return marks;
   }
   public void setYear(Integer year) {
      this.year = year;
   }
   public Integer getYear() {
      return year;
   }
   public void setSid(Integer sid) {
      this.sid = sid;
```

```
   }
   public Integer getSid() {
      return sid;
   }
}
```

以下是 **StudentMarksMapper.java** 文件的内容：

```
package com.tutorialspoint;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;
public class StudentMarksMapper implements RowMapper<StudentMarks> {
   public StudentMarks mapRow(ResultSet rs, int rowNum) throws SQLException {
      StudentMarks studentMarks = new StudentMarks();
      studentMarks.setId(rs.getInt("id"));
      studentMarks.setName(rs.getString("name"));
      studentMarks.setAge(rs.getInt("age"));
      studentMarks.setSid(rs.getInt("sid"));
      studentMarks.setMarks(rs.getInt("marks"));
      studentMarks.setYear(rs.getInt("year"));
      return studentMarks;
   }
}
```

下面是定义的 DAO 接口 StudentDAO 实现类文件 **StudentJDBCTemplate.java**：

```
package com.tutorialspoint;
import java.util.List;
import javax.sql.DataSource;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.transaction.PlatformTransactionManager;
import org.springframework.transaction.TransactionDefinition;
import org.springframework.transaction.TransactionStatus;
import org.springframework.transaction.support.DefaultTransactionDefinition;
public class StudentJDBCTemplate implements StudentDAO {
   private DataSource dataSource;
   private JdbcTemplate jdbcTemplateObject;
   private PlatformTransactionManager transactionManager;
   public void setDataSource(DataSource dataSource) {
      this.dataSource = dataSource;
      this.jdbcTemplateObject = new JdbcTemplate(dataSource);
   }
   public void setTransactionManager(
```

```
      PlatformTransactionManager transactionManager) {
      this.transactionManager = transactionManager;
   }
   public void create(String name, Integer age, Integer marks, Integer year){
      TransactionDefinition def = new DefaultTransactionDefinition();
      TransactionStatus status = transactionManager.getTransaction(def);
      try {
         String SQL1 = "insert into Student (name, age) values (?, ?)";
         jdbcTemplateObject.update( SQL1, name, age);
         // Get the latest student id to be used in Marks table
         String SQL2 = "select max(id) from Student";
         int sid = jdbcTemplateObject.queryForInt( SQL2 );
         String SQL3 = "insert into Marks(sid, marks, year) " +
                       "values (?, ?, ?)";
         jdbcTemplateObject.update( SQL3, sid, marks, year);
         System.out.println("Created Name = " + name + ", Age = " + age);
         transactionManager.commit(status);
      } catch (DataAccessException e) {
         System.out.println("Error in creating record, rolling back");
         transactionManager.rollback(status);
         throw e;
      }
      return;
   }
   public List<StudentMarks> listStudents() {
      String SQL = "select * from Student, Marks where Student.id=Marks.sid";
      List <StudentMarks> studentMarks = jdbcTemplateObject.query(SQL,
                                        new StudentMarksMapper());
      return studentMarks;
   }
}
```

现在让我们改变主应用程序文件 **MainApp.java**，如下所示：

```
package com.tutorialspoint;
import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.tutorialspoint.StudentJDBCTemplate;
public class MainApp {
   public static void main(String[] args) {
      ApplicationContext context =
            new ClassPathXmlApplicationContext("Beans.xml");
      StudentJDBCTemplate studentJDBCTemplate =
      (StudentJDBCTemplate)context.getBean("studentJDBCTemplate");
```

```
      System.out.println("------Records creation--------" );
      studentJDBCTemplate.create("Zara", 11, 99, 2010);
      studentJDBCTemplate.create("Nuha", 20, 97, 2010);
      studentJDBCTemplate.create("Ayan", 25, 100, 2011);
      System.out.println("------Listing all the records--------" );
      List<StudentMarks> studentMarks = studentJDBCTemplate.listStudents();
      for (StudentMarks record : studentMarks) {
         System.out.print("ID : " + record.getId() );
         System.out.print(", Name : " + record.getName() );
         System.out.print(", Marks : " + record.getMarks());
         System.out.print(", Year : " + record.getYear());
         System.out.println(", Age : " + record.getAge());
      }
   }
}
```

下面是配置文件 **Beans.xml** 的内容：

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

   <!-- Initialization for data source -->
   <bean id="dataSource"
      class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
      <property name="url" value="jdbc:mysql://localhost:3306/TEST"/>
      <property name="username" value="root"/>
      <property name="password" value="password"/>
   </bean>

   <!-- Initialization for TransactionManager -->
   <bean id="transactionManager"
      class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
      <property name="dataSource"  ref="dataSource" />
   </bean>

   <!-- Definition for studentJDBCTemplate bean -->
   <bean id="studentJDBCTemplate"
      class="com.tutorialspoint.StudentJDBCTemplate">
      <property name="dataSource"  ref="dataSource" />
      <property name="transactionManager"  ref="transactionManager" />
   </bean>
```

```
</beans>
```

当你完成了创建源和 bean 配置文件后，让我们运行应用程序。如果你的应用程序运行顺利的话，那么将会输出如下所示的消息：

```
------Records creation--------
Created Name = Zara, Age = 11
Created Name = Nuha, Age = 20
Created Name = Ayan, Age = 25
------Listing all the records--------
ID : 1, Name : Zara, Marks : 99, Year : 2010, Age : 11
ID : 2, Name : Nuha, Marks : 97, Year : 2010, Age : 20
ID : 3, Name : Ayan, Marks : 100, Year : 2011, Age : 25
```