Spring 声明式事务管理

声明式事务管理方法允许你在配置的帮助下而不是源代码硬编程来管理事务。这意味着你可以将事务管理从事务代码中隔离出来。你可以只使用注释或基于配置的 XML 来管理事务。

bean 配置会指定事务型方法。下面是与声明式事务相关的步骤：

- 我们使用标签，它创建一个事务处理的建议，同时，我们定义一个匹配所有方法的切入点，我们希望这些方法是事务型的并且会引用事务型的建议。
- 如果在事务型配置中包含了一个方法的名称，那么创建的建议在调用方法之前就会在事务中开始进行。
- 目标方法会在 try / catch 块中执行。
- 如果方法正常结束，AOP 建议会成功的提交事务，否则它执行回滚操作。

让我们看看上述步骤是如何实现的。在我们开始之前，至少有两个数据库表是至关重要的，在事务的帮助下，我们可以实现各种 CRUD 操作。以 **Student** 表为例，该表是使用下述 DDL 在 MySQL TEST 数据库中创建的。

```
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**，我们用来存储基于年份的学生标记。在这里，**SID** 是 Student 表的外键。

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

```
);
```

现在让我们编写 Spring JDBC 应用程序来在 Student 和 Marks 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	在项目中添加其它必需的库 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();
}
```

以下是 **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;
   }
}
```

下面是 **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;
public class StudentJDBCTemplate implements StudentDAO{
   private JdbcTemplate jdbcTemplateObject;
   public void setDataSource(DataSource dataSource) {
      this.jdbcTemplateObject = new JdbcTemplate(dataSource);
   }
   public void create(String name, Integer age, Integer marks, Integer year){
      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);
         // to simulate the exception.
         throw new RuntimeException("simulate Error condition") ;
      } catch (DataAccessException e) {
         System.out.println("Error in creating record, rolling back");
         throw e;
      }
   }
   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;
public class MainApp {
   public static void main(String[] args) {
      ApplicationContext context =
            new ClassPathXmlApplicationContext("Beans.xml");
      StudentDAO studentJDBCTemplate =
      (StudentDAO)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());
      }
   }
```

```
}
```

以下是配置文件 **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"
   xmlns:tx="http://www.springframework.org/schema/tx"
   xmlns:aop="http://www.springframework.org/schema/aop"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
   http://www.springframework.org/schema/tx
   http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
   http://www.springframework.org/schema/aop
   http://www.springframework.org/schema/aop/spring-aop-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="cohondob"/>
   </bean>

   <tx:advice id="txAdvice"  transaction-manager="transactionManager">
      <tx:attributes>
      <tx:method name="create"/>
      </tx:attributes>
   </tx:advice>

   <aop:config>
      <aop:pointcut id="createOperation"
      expression="execution(* com.tutorialspoint.StudentJDBCTemplate.create(..))"/>
      <aop:advisor advice-ref="txAdvice" pointcut-ref="createOperation"/>
   </aop:config>

   <!-- 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" />
   </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，那么会输出如下所示的异常。在这种情况下，事务会回滚并且在数据库表中不会创建任何记录。

```
------Records creation--------
Created Name = Zara, Age = 11
Exception in thread "main" java.lang.RuntimeException: simulate Error condition
```

在删除异常后，你可以尝试上述示例，在这种情况下，会提交事务并且你可以在数据库中看见一条记录。