关于事务管理：

1、自己实现一个TransactionManager事务管理工具（工厂）类，通过Spring的AOP来管理事务

1.1、自定义事务类里面有事务的开启，提交，回滚，资源的释放以及获取连接和绑定Connection到线程上（见事务工厂类附件）

1.2、配置事务工厂类的IOC以及数据源的注入

```
<!-- 通知也是一个bean:TransactionManager  主要作用是用于事务管理，自己写的一个事务管事类 -->
<bean id="transactionManager" class="com.itheima.utils.TransactionManager">
    <!-- 注入数据源 -->
    <property name="dataSource" ref="driverManagerDataSource"></property>
</bean>

<!-- 配置一个数据源 -->
<bean id="driverManagerDataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <!-- 注入数据 -->
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/day36_szee03_spring"></property>
    <property name="username" value="root"></property>
    <property name="password" value="1234"></property>
</bean>
```

1.3、配置从当前线程获取Connection的IOC

```
<!-- 把资源都交给spring来管理 -->
<bean id="accountDao" class="com.itheima.dao.impl.AccountDaoImpl">
    <property name="conn" ref="conn"></property>
</bean>

<!-- 此处要的conn比较特殊：它必须得是从当前线程上得到那个 -->
<bean id="conn" factory-bean="transactionManager" factory-method="getConnection"></bean>
```

从事务工厂的方法里获取

1.4、配置事务对业务类的AOP管理

```
<aop:config>
    <!-- 全局切入点表达式 -->
    <aop:pointcut expression="execution(* com.itheima.service.impl.*.*(..))" id="pt1"/>
    <!-- 配置切面 -->
    <aop:aspect id="txAdvice" ref="transactionManager">
        <!-- 配置通知的类型 -->
        <!-- 开启事务是前置通知 -->
        <aop:before method="startTransaction" pointcut-ref="pt1"/>
        <!-- 提交事务是后置通知 -->
        <aop:after-returning method="commit" pointcut-ref="pt1"/>
        <!-- 回滚事务是异常通知 -->
        <aop:after-throwing method="rollback" pointcut-ref="pt1"/>
        <!-- 释放资源是最终通知 -->
        <aop:after method="release" pointcut-ref="pt1"/>
    </aop:aspect>
</aop:config>
```

2、用Spring内置的事务管理TransactionManager

通过XML声明式配置使用

Spring事务管理器提供的三个主要接口

PlatformTransactionManager 对事务的操作，只有三步操作：获取事务，提交事务，回滚事务

常见的事管理器模板（实现类）有：

DataSourceTransactionManager（jdbc开发时事务管理器）

HibernateTransactionManager（hibernate开发时事务管理器）

TransactionStatus Spring用于记录当前事务运行状态

TransactionDefinition Spring用于确定事务的具体详情，如：隔离级别、是否只读、超时时间等

开发步骤：

1、导入**IOC,AOP**和**TX**的**jar**包

2、导入**ioc,aop**和**tx**的约束

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd">
```

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/tx
                           http://www.springframework.org/schema/tx/spring-tx.xsd
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop.xsd">
```

3、配置内置的事务管理器

```
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
 <!-- 注入数据源 -->
 <property name="dataSource" ref="driverManagerDataSource"></property>
</bean>
```

```
<!-- 配置一个数据源 -->
<bean id="driverManagerDataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <!-- 注入数据 -->
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/day36_szee03_spring"></property>
    <property name="username" value="root"></property>
    <property name="password" value="1234"></property>
</bean>
```

使用内置事务管理器 - 有数据源的

```
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <!-- 注入数据源 -->
    <property name="dataSource" ref="driverManagerDataSource"></property>
</bean>
```

4、声明事务的通知

```
<!-- 配置事务的通知 -->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
 <tx:attributes>
  <tx:method name="*" propagation="REQUIRED" read-only="false"/>
  <tx:method name="find*" propagation="SUPPORTS" read-only="true"/>
 </tx:attributes>
</tx:advice>
```

```
<!-- 配置事务的通知 -->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
    <tx:attributes>
        <tx:method name="*" propagation="REQUIRED" read-only="false"/>
        <tx:method name="find*" propagation="SUPPORTS" read-only="true"/>
    </tx:attributes>
</tx:advice>
```

tx:method: 指明对哪个方法定义具体的事务详情
propagation: 指定事务的传播行为
read-only: 指定是否是只读事务

5、配置**AOP**，建立通知和切入点表达式的对应关系

```
<!-- 配置AOP -->
 <aop:config>
  <!-- 配置切入点表达式 -->
  <aop:pointcut expression="execution(* com.itheima.service.impl.*.*(..))" id="pt1"/>
  <!-- 建立切入点表达式和通知之间的关系 -->
  <aop:advisor advice-ref="txAdvice" pointcut-ref="pt1"/>
 </aop:config>
```

```
<!-- 配置AOP -->
<aop:config>
    <!-- 配置切入点表达式 -->
    <aop:pointcut expression="execution(* com.itheima.service.impl.*.*(..))" id="pt1"/>
    <!-- 建立切入点表达式和通知之间的关系 -->
    <aop:advisor advice-ref="txAdvice" pointcut-ref="pt1"/>
</aop:config>
```

6、配置事务属性

isolation：指定事务的隔离级别。有默认值：默认值是：**DEFAULT**。含义就是使用数据库的默认隔离级别。

propagation：指定事务的传播行为。有默认值：默认值是**REQUIRED**。

read-only：指定是否是只读事务。有默认值：默认值是**false**。不只读。 只有查询方法才可以设置为只读事务。

timeout：指定事务的超时时间。默认是：**-1**，永不超时。单位是秒

rollback-for：指定的是一个异常。当产生该异常时事务回滚，产生其他异常时，事务不回滚。没有默认值。

no-rollback-for：指定的是一个异常。当产生该异常时，事务不回滚。产生其他异常时，事务回滚。没有默认值。

事务的传播行为：

REQUIRED:(增删改)

如果当前没有事务，就新建一个事务，如果已经存在一个事务中，加入到这个事务中。（默认值）

SUPPORTS:（查）

支持当前事务，如果当前没有事务，就以非事务方式执行（没有事务）

MANDATORY：

使用当前的事务，如果当前没有事务，就抛出异常

REQUERS_NEW:

新建事务，如果当前在事务中，把当前事务挂起。

NOT_SUPPORTED:

以非事务方式执行操作，如果当前存在事务，就把当前事务挂起

NEVER:

以非事务方式运行，如果当前存在事务，抛出异常

NESTED:

如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则执行REQUIRED类似的操作。

基于注解的声明式事务控制

步骤：

1、导入ioc/aop/tx的jar包

2、导入ico/tx/context的名称空间

3、使用spring基于注解的IOC来管理Spring内置资源

```
<!-- 配置jdbcTemplate -->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
 <property name="dataSource" ref="driverManagerDataSource"></property>
</bean>

<!-- 配置一个数据源 -->
<bean id="driverManagerDataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
 <!-- 注入数据 -->
 <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
 <property name="url" value="jdbc:mysql://localhost:3306/day36_szee03_spring"></property>
 <property name="username" value="root"></property>
 <property name="password" value="1234"></property>
</bean>
```

4、配置spring要扫描的包

```
<context:component-scan base-package="com.itheima"></context:component-scan>
```

5、配置事务管理器

```
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
<!-- 注入数据源 -->
<property name="dataSource" ref="driverManagerDataSource"></property>
</bean>
```

6、在需要使用事务的地方使用@Transactional注解

该注解能出现：类上，方法上和接口上。

出现在类上：表明当前类中的所有方法都有事务

出现在方法上：表明当前方法有事务

出现接口上：表明当前接口的所有实现类都有事务

优先级：就近原则

```
@Transactional(propagation=Propagation.REQUIRED,readOnly=false)
public class AccountServiceImpl implements IAccountService {
```

7、开启spring对注解事务的支持

```
<tx:annotation-driven transaction-manager="transactionManager"/>
```