事务管理

描述

无论是使用JDBC，hibernate还是mybatis，都会用到事务。

什么是事务

事务就是一系列的动作, 它们被当做一个单独的工作单元. 这些动作要么全部完成, 要么全部不起作用

事务的四个关键属性(ACID)

- 原子性(atomicity): 事务是一个原子操作, 由一系列动作组成. 事务的原子性确保动作要么全部完成要么完全不起作用.
- 一致性(consistency): 一旦所有事务动作完成, 事务就被提交. 数据和资源就处于一种满足业务规则的一致性状态中.
- 隔离性(isolation): 可能有许多事务会同时处理相同的数据, 因此每个事物都应该与其他事务隔离开来, 防止数据损坏.
- 持久性(durability): 一旦事务完成, 无论发生什么系统错误, 它的结果都不应该受到影响. 通常情况下, 事务的结果被写到持久化存储器中.

使用spring的事务管理

作为企业级应用程序框架, Spring 在不同的事务管理 API 之上定义了一个抽象层. 而应用程序开发人员不必了解底层的事务管理 API, 就可以使用 Spring 的事务管理机制.

如何在spring中使用事务管理

- Spring 既支持编程式事务管理, 也支持声明式的事务管理. 编程式事务管理: 将事务管理代码嵌入到业务方法中来控制事务的提交和回滚. 在编程式管理事务时, 必须在每个事务操作中包含额外的事务管理代码.
- 声明式事务管理: 大多数情况下比编程式事务管理更好用. 它将事务管理代码从业务方法中分离出来, 以声明的方式来实现事务管理. 事务管理作为一种横切关注点, 可以通过 AOP 方法模块化. Spring 通过 Spring AOP 框架支持声明式事务管理.

spring事务管理的核心类TransformManager,去启动事务。

事务的步骤：

1. 创建事务。我们采用jdbc的事务管理，所以需要spring-jdbc的包。
2. 启动事务。[xml配置方式，注解的方式]依赖spring-tx包。
3. 事务的绑定。

注解方式：

```
<!-- 配置一个 事务管理器 -->
<bean id="transactionManager"
 class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
 <!-- 数据源 dataSource在application-dao.xml中配置 -->
 <property name="dataSource" ref="dataSource" />
</bean>
<!-- 注解的事务 -->
<tx:annotation-driven transaction-manager="transactionManager" />
```

方法体中使用@Transactional[加载使用事务的方法上]。

非注解的配置：

1. 注册一个事务管理的Bean
2. 启动事务，如果不配置tx:method，则采用默认的的配置。

```
<tx:advice id="txAdvice" transaction-manager="transactionManager">
 <!-- 事物的传播行为 -->
 <tx:attributes>
 <!-- 指定事务的传播行为是新开一个事务,还有其他的事务属性设置 -->
  <tx:method name="*" propagation="REQUIRES_NEW"/>
  <tx:method name="get*" read-only="true"/>
```

```
   <tx:method name="updata*" propagation="REQUIRED"/>
  </tx:attributes>
 </tx:advice>
```

3. 事物的绑定。[利用aop将所需要进行事务管理的方法进行事务管理]

```
<!-- 配置事物切入点 -->
 <aop:config>
  <aop:pointcut expression="切点" id="切点的id名字" />
  <!-- 那些方法需要添加事物 -->
  <aop:advisor advice-ref="txAdvice"  pointcut-ref="切点的id名字"/>
 </aop:config>
```

案例

```
<!-- 事务管理器
         对mybatis操作数据库的事务控制，spring使用jdbc的事务控制
         -->
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <!-- 数据源
         dataSource在application-dao.xml中配置
         -->
        <property name="dataSource" ref="dataSource" />
    </bean>

    <!-- 通知 -->
    <tx:advice id="txAdvice" transaction-manager="transactionManager">
        <tx:attributes>
            <!-- 传播行为 -->
            <tx:method name="save*" propagation="REQUIRED" />
            <tx:method name="delete*" propagation="REQUIRED" />
            <tx:method name="insert*" propagation="REQUIRED" />
            <tx:method name="update*" propagation="REQUIRED" />
            <tx:method name="find*" propagation="SUPPORTS" read-only="true" />
            <tx:method name="get*" propagation="SUPPORTS" read-only="true" />
            <tx:method name="select*" propagation="SUPPORTS" read-only="true" />
        </tx:attributes>
    </tx:advice>

    <!-- aop -->
    <aop:config>
        <!-- pointcut：切点，com.swxc.ssm.service.impl下的所有类的下的所有方法，不限制传入参数 -->
        <aop:advisor advice-ref="txAdvice" pointcut="execution(* com.swxc.ssm.service.impl.*.*(..))" />
    </aop:config>
```

补充

为了提高代码的可维护性，项目中尽可能使用xml配置的方式。