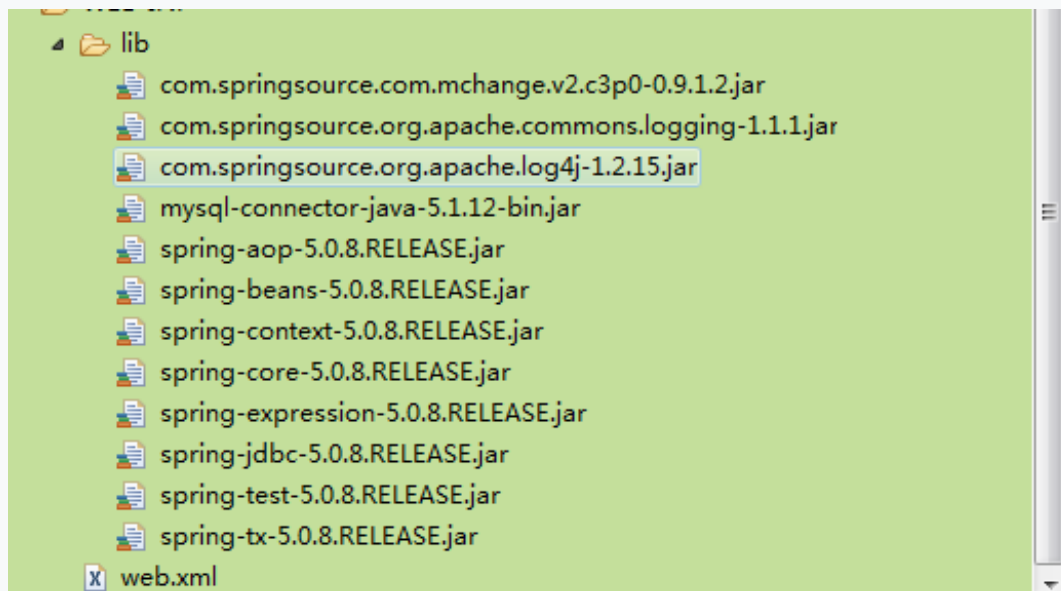


18 Spring_事务管理_HelloWord

Spring

一:准备工作

1、 : 导入jar包



2、 实体类

```
1. public class Users {  
2.     private Integer id;  
3.     private String name;  
4.     private float money;  
5.     //对应的get set 方法  
6. }
```

3、 dao实现类

1、接口就为两个抽象方法

```
1. import org.springframework.jdbc.core.support.JdbcDaoSupport;
2. import com.spring.Dao.IUsersDao;
3. public class UsersDaoImp extends JdbcDaoSupport implements IUsersDao {
4.     @Override
5.     public void addMoney(Integer id, Float money) {
6.
7.         String sql="update users set money=money+? where id=?";
8.         super.getJdbcTemplate().update(sql,money,id);
9.     }
10.    @Override
11.    public void updateMoney(Integer id, Float money) {
12.        // TODO Auto-generated method stub
13.        String sql="update users set money=money-? where id=?";
14.        super.getJdbcTemplate().update(sql,money,id);
15.    }
16. }
```

4、Serivce实现类

1、接口就为一个抽象方法

```
1. import com.spring.Dao.IUsersDao;
2. import com.spring.serivce.IUsersSerivce;
3. public class UsersSerivceImp implements IUsersSerivce {
4.
5.     // 必须有get set 方法
6.     private IUsersDao usersDao;
7.
8.     @Override
9.     public void getTran(Integer in, Integer in2, Float money) {
10.         usersDao.updateMoney(in, money);
11.         usersDao.addMoney(in2, money);
12.
13.     }
14.
15.     public IUsersDao getUsersDao() {
16.         return usersDao;
17.     }
```

```

18.
19.     public void setUsersDao(IUsersDao usersDao) {
20.         this.usersDao = usersDao;
21.     }
22. }

```

5、配置文件

```

1.     <!--指定读取properties文件 -->
2.     <context:property-placeholder location="classpath:db.properties"
3.     />
4.     <!--1.将连接池放入Spring容器中 -->
5.     <bean name="dataSource"
6.     class="com.mchange.v2.c3p0.ComboPooledDataSource">
7.         <property name="driverClass" value="${jdbc.driverClass}"></prop
8.     erty>
9.         <property name="jdbcUrl" value="${jdbc.jdbcUrl}"></property>
10.        <property name="user" value="${jdbc.user}"></property>
11.        <property name="password" value="${jdbc.password}"></property>
12.    </bean>
13.    <!--2、将IUsesDao放入Spring容器中 -->
14.    <bean name="usersDao" class="com.spring.Dao.imp.UsersDaoImp">
15.        <!-- <property name="jt" ref="jdbcTemplate"></property> -->
16.        <property name="dataSource" ref="dataSource"></property>
17.    </bean>
18.    <!--3、将usersSerivce放入Spring容器中 -->
19.    <bean name="usersSerivce"
20.    class="com.spring.serivce.impl.UsersSerivceImp">
21.        <property name="usersDao" ref="usersDao"></property>
22.    </bean>

```

6、测试类

```

1.     @RunWith(SpringJUnit4ClassRunner.class)
2.     @ContextConfiguration("classpath:applicationContext.xml")
3.     public class UserDemo {
4.
5.         @Resource(name="usersSerivce")
6.         private IUsersSerivce usersSerivce;
7.         @Test
8.         public void getVoid01() throws Exception{

```

```
9.         usersSerivce.getTran(15,16,1000F);
10.     }
11. }
```

二:事务的管理方式

1、测试同上

1、方式一_编码 Transaction模板(了解)

1-1、核心的事务管理器

1、将核心的事务管理器配置到容器中

```
1.     <!--将核心的事务管理器配置到容器中 -->
2.         <bean name="transactionManager"
3.
4.             class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
5.             >
6.                 <property name="dataSource" ref="dataSource"></property>
7.             </bean>
8.         <!-- 事务模版对象 -->
9.         <bean name="transactionTemplate"
10.
11.             class="org.springframework.transaction.support.TransactionTemplate">
12.                 <property name="transactionManager" ref="transactionManager"></
13.                 property>
14.             </bean>
```

1-2、模版对像

1、将事务模版对象 组装到Serivce层

```
1.     import org.springframework.transaction.TransactionStatus;
2.     import
3.         org.springframework.transaction.support.TransactionCallbackWithoutResult
4.         ;
5.     import org.springframework.transaction.support.TransactionTemplate;
```

```

4.    import com.spring.Dao.IUsersDao;
5.    import com.spring.servive.IUsersServive;
6.    public class UsersServiveImp implements IUsersServive {
7.        // 必须有get set 方法
8.        private IUsersDao usersDao;
9.        private TransactionTemplate tt;
10.
11.        @Override
12.        public void getTran(final Integer in, final Integer in2, final Float money) throws Exception {
13.
14.            tt.execute(new TransactionCallbackWithoutResult() {
15.                @Override
16.                protected void doInTransactionWithoutResult(TransactionStatus arg0) {
17.
18.                    usersDao.updateMoney(in, money);
19.                    int i=1/0;
20.                    usersDao.addMoney(in2, money);
21.                }
22.            });
23.        }

```

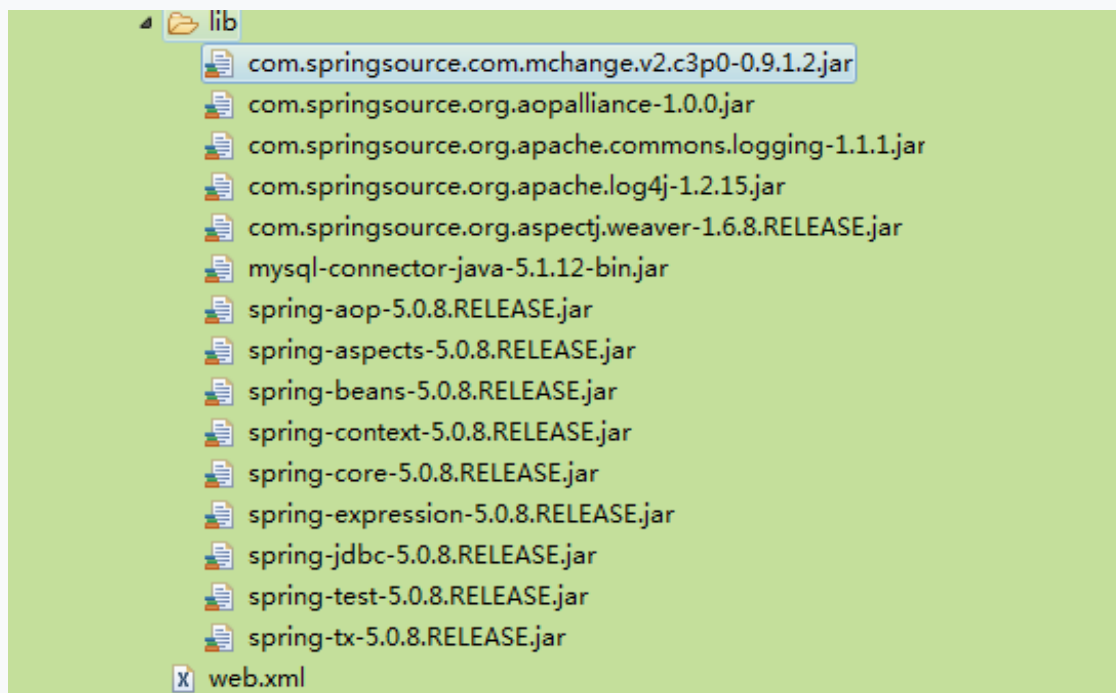
```

1.        <!--3、将usersServive放入Spring容器中 -->
2.        <bean name="usersServive"
3.            class="com.spring.servive.impl.UsersServiveImp">
4.            <property name="usersDao" ref="usersDao"></property>
5.            <!--事务模版对象 组装到Servive层 -->
6.            <property name="tt" ref="transactionTemplate"></property>
7.        </bean>

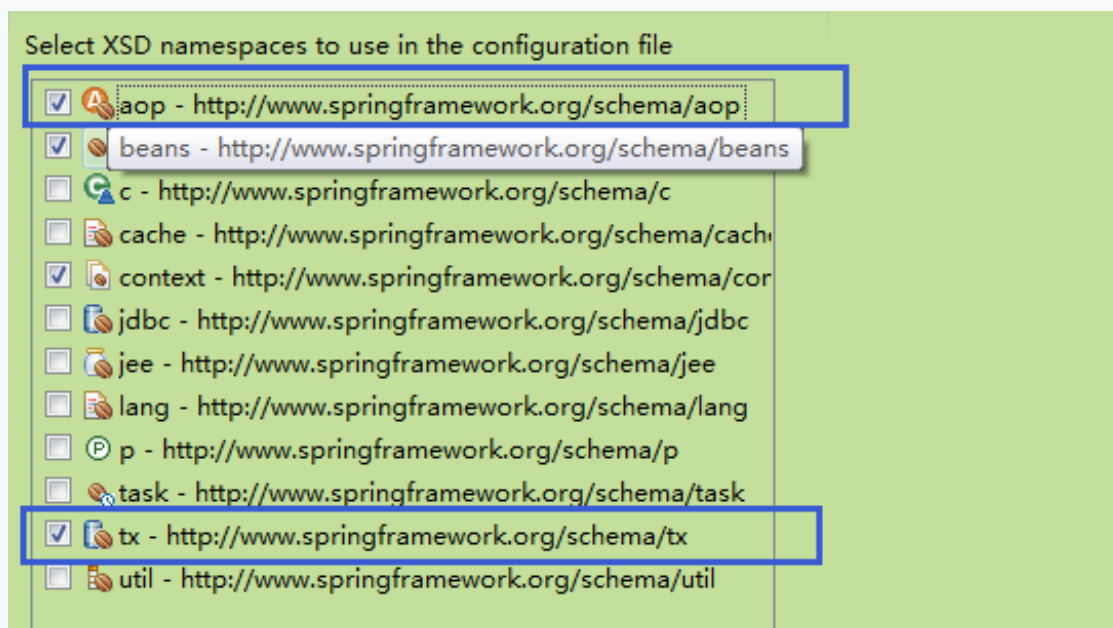
```

2、方式二_xml配置aop事务(重点)

2-1、到如Jar包



2-2、导入tx & aop匿名空间



2-3、配置通知

```
1.      <!--将核心的事务管理器配置到容器中 -->
2.      <bean name="transactionManager"
3.
4.      class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
      >
      <property name="dataSource" ref="dataSource"></property>
```

```

5.         </bean>
6.         <!--配置事务通知-->
7.         <tx:advice id="txAdvice" transaction-manager="transactionManager">
8.             <tx:attributes>
9.                 <!--
10.                  指定方法以什么样的事务进行执行
11.                  isolation:隔离级别
12.                  propagation :事务的传播行为
13.                  read-only:配置是否只读
14.                  -->
15.                 <!-- 公司的配置方式-->
16.                 <tx:method name="save*" isolation="REPEATABLE_READ" propagation="REQUIRED" read-only="false"/>
17.                 <tx:method name="persist*" isolation="REPEATABLE_READ" propagation="REQUIRED" read-only="false"/>
18.                 <tx:method name="delete*" isolation="REPEATABLE_READ" propagation="REQUIRED" read-only="false"/>
19.                 <tx:method name="remove*" isolation="REPEATABLE_READ" propagation="REQUIRED" read-only="false"/>
20.                 <tx:method name="update*" isolation="REPEATABLE_READ" propagation="REQUIRED" read-only="false"/>
21.                 <tx:method name="modify*" isolation="REPEATABLE_READ" propagation="REQUIRED" read-only="false"/>
22.                 <!--read-only: 只读 true -->
23.                 <tx:method name="get*" isolation="REPEATABLE_READ" propagation="REQUIRED" read-only="true"/>
24.                 <tx:method name="find*" isolation="REPEATABLE_READ" propagation="REQUIRED" read-only="true"/>
25.
26.                 <tx:method name="getTran" isolation="REPEATABLE_READ" propagation="REQUIRED" read-only="false"/>
27.             </tx:attributes>
28.         </tx:advice>

```

2-4、配置将通知织入目标

```

1.         <!--配置织入 -->
2.         <aop:config>
3.             <aop:pointcut expression="execution(* com.spring.serivce.impl.*.*(..))" id="txPointcut"/>
4.             <!-- 配置切面（通知+切点）
5.                  advice-ref:配置通知的名称
6.                  pointcut-ref:配置 切点的名称
7.                  -->
8.             <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointcut" />

```

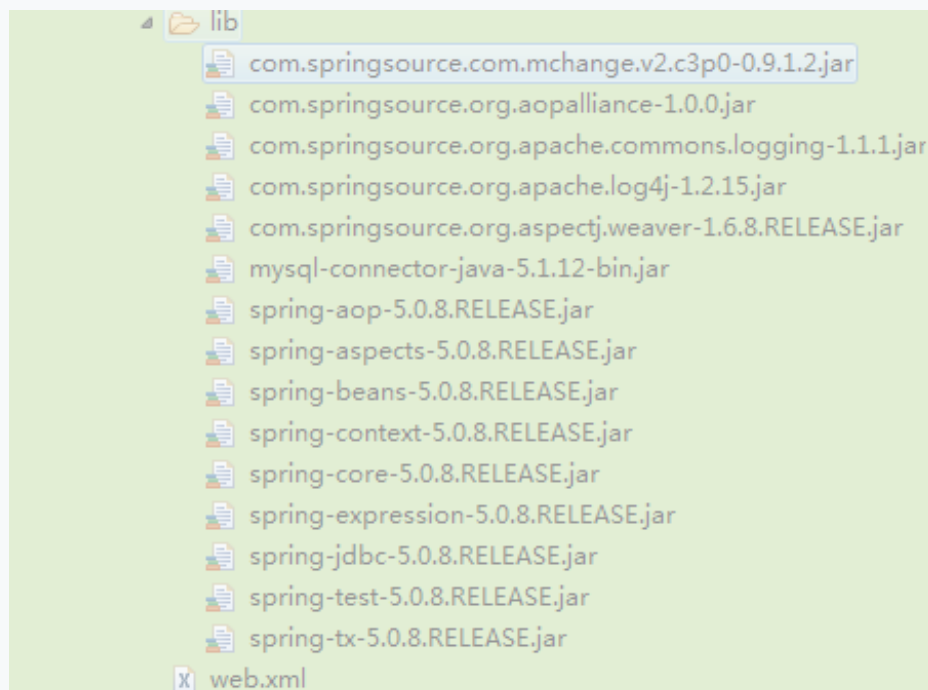
```
9.     </aop:config>
```

```
1.     <!--2、将IUsesDao放入Spring容器中 -->
2.     <bean name="usersDao" class="com.spring.Dao.imp.UsersDaoImp">
3.         <!-- <property name="jt" ref="jdbcTemplate"></property> -->
4.         <property name="dataSource" ref="dataSource"></property>
5.     </bean>
6.     <!--3、将usersSerivce放入Spring容器中 -->
7.     <bean name="usersSerivce"
8.         class="com.spring.serivce.impl.UsersSerivceImp">
9.         <property name="usersDao" ref="usersDao"></property>
10.        <!--事务模版对象 组装到Serivce层 -->
        </bean>
```

3、方式三_解配置aop事务(重点)

1、导包同上

3-1、到如Jar包



A screenshot of a file explorer window showing a directory named 'lib'. The directory contains the following files:

- com.springsource.com.mchange.v2.c3p0-0.9.1.2.jar
- com.springsource.org.aopalliance-1.0.0.jar
- com.springsource.org.apache.commons.logging-1.1.1.jar
- com.springsource.org.apache.log4j-1.2.15.jar
- com.springsource.org.aspectj.weaver-1.6.8.RELEASE.jar
- mysql-connector-java-5.1.12-bin.jar
- spring-aop-5.0.8.RELEASE.jar
- spring-aspects-5.0.8.RELEASE.jar
- spring-beans-5.0.8.RELEASE.jar
- spring-context-5.0.8.RELEASE.jar
- spring-core-5.0.8.RELEASE.jar
- spring-expression-5.0.8.RELEASE.jar
- spring-jdbc-5.0.8.RELEASE.jar
- spring-test-5.0.8.RELEASE.jar
- spring-tx-5.0.8.RELEASE.jar
- web.xml

3-2、导入tx && aop匿名空间

Select XSD namespaces to use in the configuration file

- ☒ aop - http://www.springframework.org/schema/aop
- ☒ beans - http://www.springframework.org/schema/beans
- ☐ c - http://www.springframework.org/schema/c
- ☐ cache - http://www.springframework.org/schema/cache
- ☒ context - http://www.springframework.org/schema/context
- ☐ jdbc - http://www.springframework.org/schema/jdbc
- ☐ jee - http://www.springframework.org/schema/jee
- ☐ lang - http://www.springframework.org/schema/lang
- ☐ p - http://www.springframework.org/schema/p
- ☐ task - http://www.springframework.org/schema/task
- ☒ tx - http://www.springframework.org/schema/tx
- ☐ util - http://www.springframework.org/schema/util

3-3、配置开启注解

1. `<!--开启使用注解事务 -->`
2. `<tx:annotation-driven/>`

3-4、使用注解

1、方式一：整个类上实现该事务

```
1. @Transactional(isolation=Isolation.REPEATABLE_READ,propagation=Propagation.REQUIRED,readOnly=true)
2. public class UsersServiceImp implements IUsersService {
3.     // 必须有get set 方法
4.     private IUsersDao usersDao;
5.     public void getTran(final Integer in, final Integer in2, final Float money) throws Exception {
6.         usersDao.updateMoney(in, money);
7.         /*int i=1/0; */
8.         usersDao.addMoney(in2, money);
9.     }
10. }
```

2、方式二：指定方法使用该事务注解

```
1. public class UsersSerivceImp implements IUsersSerivce {
2.     // 必须有get set 方法
3.     private IUsersDao usersDao;
4.     @Override
5.
6.     @Transactional(isolation=Isolation.REPEATABLE_READ,propagation=Propagation.REQUIRED,readonly=false)
7.     public void getTran(final Integer in, final Integer in2, final Float money) throws Exception {
8.         usersDao.updateMoney(in, money);
9.         /*int i=1/0; */
10.        usersDao.addMoney(in2, money);
11.    }
12. }
```

3、如在类上使用事务注解,中有方法需要的事务属性不一致,可以方法上在设置事务属性