# Note：编程式事务请参考示例项目spring-tx83(编程式事务)

# 定义Dao层与数据库的交互的代码请参照示例项目

# 使用xml配置声明式事务

## 定义业务接口：

**package** com.yang.springtx.service;

**import** java.sql.SQLException;

**import** java.util.List;

**import** com.yang.springtx.model.Emp;

**public** **interface** EmpService {

**void** addEmp(List<Emp> emps) **throws** SQLException;

}

## 定义业务实现类：

**package** com.yang.springtx.service;

**import** java.sql.SQLException;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.TransactionStatus;

**import** org.springframework.transaction.support.TransactionCallback;

**import** org.springframework.transaction.support.TransactionTemplate;

**import** com.yang.springtx.dao.EmpDao;

**import** com.yang.springtx.model.Emp;

/\*\*

\* 业务实现类

\*

\* 此处作为xml配置声明事务的目标类

\* **@author** Administrator

\*

\*/

@Component("empService")

**public** **class** EmpServiceImpl **implements** EmpService {

@Autowired

EmpDao empDao;

@Autowired

TransactionTemplate transactionTemplate;

@Override

**public** **void** addEmp(List<Emp> emps) **throws** SQLException {

**for** (**int** index=0; index<emps.size(); index++){

Emp emp = emps.get(index);

empDao.insertEmp(emp);

**if**(index>11){

**throw** **new** RuntimeException();

}

}

}

}

## 配置xml：

<?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:context=*"http://www.springframework.org/schema/context"*

xmlns:p=*"http://www.springframework.org/schema/p"* 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-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd"*>

<!-- 用annotation自动注入bean -->

<context:component-scan base-package=*"com"*/>

<!--加载属性文件方式一:使用context命名空间的配置引用属性文件 -->

<context:property-placeholder

location=*"classpath:/jdbc.properties"*

file-encoding=*"utf8"* />

<bean id=*"utf8"* class=*"java.lang.String"*>

<constructor-arg value=*"utf-8"* />

</bean>

<!--加载属性文件方式一:使用传统的“属性文件占位符配置类”PropertyPlaceholderConfigurer引用属性文件 -->

<!-- <bean

class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"

p:fileEncoding="utf-8">

<property name="locations">

<list>

<value>classpath:com/springioc/placeholder/jdbc.properties</value>

</list>

</property>

</bean> -->

<!-- 使用占位符，加载DataSource-->

<bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"*

destroy-method=*"close"*

p:driverClassName=*"${driverClassName}"*

p:url=*"${url}"*

p:username=*"${username}"*

p:password=*"${password}"* />

<!-- 配置生成JdbcTemplete -->

<bean id=*"jdbcTemplate"*

class=*"org.springframework.jdbc.core.JdbcTemplate"*

p:dataSource-ref=*"dataSource"*/>

<bean id=*"txManager"*

class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*

p:dataSource-ref=*"dataSource"*/>

<bean id=*"transactionTemplate"*

class=*"org.springframework.transaction.support.TransactionTemplate"*

p:transactionManager-ref=*"txManager"*/>

<bean id=*"transactonEmpService"*

class=*"org.springframework.transaction.interceptor.TransactionProxyFactoryBean"*

p:transactionManager-ref=*"txManager"*

p:target-ref=*"empService"*>

<property name=*"transactionAttributes"*>

<props>

<prop key=*"addEmp"*>

PROPAGATION\_REQUIRED,-RuntimeException

</prop>

</props>

</property>

</bean>

</beans>

## Test：

**package** com.yang.springtx.service;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.test.context.ContextConfiguration;

**import** org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

**import** com.yang.springtx.model.Emp;

@RunWith(SpringJUnit4ClassRunner.**class**)

@ContextConfiguration(locations= "/beans.xml")

**public** **class** TestEmpService {

@Autowired

**private** EmpService transactonEmpService;

/\*\*

\* 尝试批量插入数据，由于出现异常(参考EmpServiceImpl类)，回滚，

\* 最终没有向数据库插入任何数据

\*/

@Test

**public** **void** testAddEmp(){

List<Emp> emps = **new** ArrayList<Emp>();

**for** (**int** index = 1;index <= 1000;index ++) {

Emp emp = **new** Emp();

emp.setEname("yang"+index);

emps.add(emp);

}

**try** {

transactonEmpService.addEmp(emps);

System.***out***.println(emps.toString());

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

# 使用annotation配置声明式事务（更简便，较常用）

## 定义业务接口：

**package** com.yang.springtx.service;

**import** java.sql.SQLException;

**import** java.util.List;

**import** com.yang.springtx.model.Emp;

**public** **interface** EmpService {

**void** addEmp(List<Emp> emps) **throws** SQLException;

}

## 定义业务实现类：

**package** com.yang.springtx.service;

**import** java.sql.SQLException;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.dao.PessimisticLockingFailureException;

**import** org.springframework.stereotype.Component;

**import** org.springframework.transaction.annotation.Isolation;

**import** org.springframework.transaction.annotation.Propagation;

**import** org.springframework.transaction.annotation.Transactional;

**import** org.springframework.transaction.support.TransactionTemplate;

**import** com.yang.springtx.dao.EmpDao;

**import** com.yang.springtx.model.Emp;

/\*\*

\* 业务实现类

\*

\* 此处作为annotation配置声明事务的目标类

\* **@author** Administrator

\*

\*/

@Component("empService2")

**public** **class** EmpServiceImpl2 **implements** EmpService {

@Autowired

EmpDao empDao;

@Autowired

TransactionTemplate transactionTemplate;

@Override

/\*\*

\* **@Transactional** annotation的propagation为必须配置，值一般配为Propagation.REQUIRED

\* 其他的为可选，一般使用过程中不需要配

\* 默认情况下遇到一切异常都回滚

\* 以下配置PessimisticLockingFailureException不回滚

\*/

@Transactional(propagation=Propagation.***REQUIRED***,

isolation=Isolation.***READ\_COMMITTED***,

noRollbackFor = { PessimisticLockingFailureException.**class** },

rollbackFor = { NullPointerException.**class** })

**public** **void** addEmp(List<Emp> emps) **throws** SQLException {

**for** (**int** index=0; index<emps.size(); index++){

Emp emp = emps.get(index);

empDao.insertEmp(emp);

**if**(index>11){

**throw** **new** SQLException();

}

}

}

}

## 配置xml：

<?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:context=*"http://www.springframework.org/schema/context"*

xmlns:p=*"http://www.springframework.org/schema/p"* 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-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd"*>

<!-- 用annotation自动注入bean -->

<context:component-scan base-package=*"com"*/>

<!--对标注@Transactional注解的bean，进行加工处理，比边组织事物管理切面，

只有配置这个才可以使用annotation声明事务 -->

<!-- proxy-target-class=true,说明使用CGLib动态代理 -->

<tx:annotation-driven transaction-manager=*"txManager"*

proxy-target-class=*"true"* />

<!--加载属性文件方式一:使用context命名空间的配置引用属性文件 -->

<context:property-placeholder

location=*"classpath:/jdbc.properties"*

file-encoding=*"utf8"* />

<bean id=*"utf8"* class=*"java.lang.String"*>

<constructor-arg value=*"utf-8"* />

</bean>

<!--加载属性文件方式一:使用传统的“属性文件占位符配置类”PropertyPlaceholderConfigurer引用属性文件 -->

<!-- <bean

class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"

p:fileEncoding="utf-8">

<property name="locations">

<list>

<value>classpath:com/springioc/placeholder/jdbc.properties</value>

</list>

</property>

</bean> -->

<!-- 使用占位符，加载DataSource-->

<bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"*

destroy-method=*"close"*

p:driverClassName=*"${driverClassName}"*

p:url=*"${url}"*

p:username=*"${username}"*

p:password=*"${password}"* />

<!-- 配置生成JdbcTemplete -->

<bean id=*"jdbcTemplate"*

class=*"org.springframework.jdbc.core.JdbcTemplate"*

p:dataSource-ref=*"dataSource"*/>

<!-- 配置生成事务管理器，spring对事务的管理需基于相应的事务管理器实现 -->

<bean id=*"txManager"*

class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*

p:dataSource-ref=*"dataSource"*/>

</beans>

## Test:

**package** com.yang.springtx.service;

**import** java.sql.SQLException;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.test.context.ContextConfiguration;

**import** org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

**import** com.yang.springtx.model.Emp;

@RunWith(SpringJUnit4ClassRunner.**class**)

@ContextConfiguration("/beans-anno.xml")

**public** **class** EmpServiceAnnoTest {

@Autowired

EmpService empService2;

/\*\*

\* 尝试批量插入数据，由于出现异常(参考EmpServiceImpl2类)，回滚，

\* 最终没有向数据库插入任何数据

\*/

@Test

**public** **void** testAddEmp(){

List<Emp> emps = **new** ArrayList<Emp>();

**for** (**int** index = 1;index <= 1000;index ++) {

Emp emp = **new** Emp();

emp.setEname("yang"+index);

emps.add(emp);

}

**try** {

empService2.addEmp(emps);

System.***out***.println(emps.toString());

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

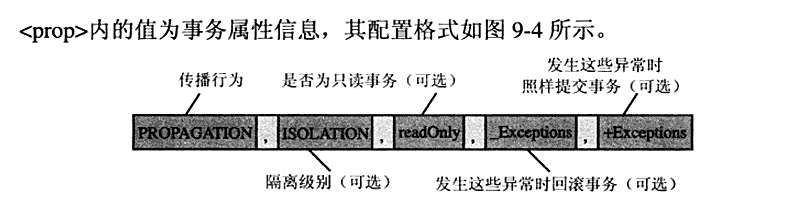
e.printStackTrace();

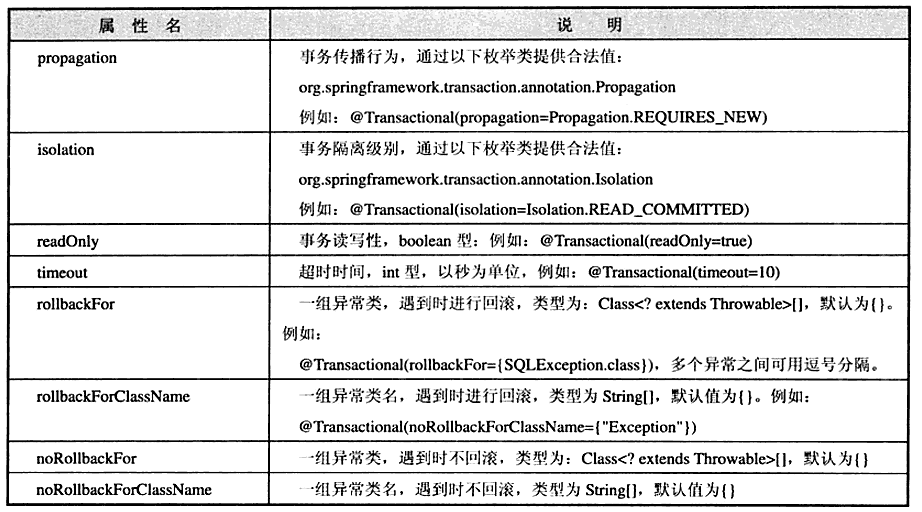
}

}

}

# 事务属性信息：





Isolation（隔离级别）：

