关于hibernate的一些事 （此博文为[http://my.oschina.net/u/555061/blog/506216](http://www.okbase.net/file/link?u=http://my.oschina.net/u/555061/blog/506216)后续）

简单说一下SessionFactory、Session 源自： [http://blog.csdn.net/javaloveiphone/article/details/8155340](http://www.okbase.net/file/link?u=http://blog.csdn.net/javaloveiphone/article/details/8155340)

1、Hibernate中SessionFactory对象的创建代价很高，它是线程安全的对象，被设计成可以为所有的应用程序线程所共享。通常，SessionFactory会在应用程序启动时创建，一旦创建了SessionFactory将不会轻易关闭，只有当应用关闭时，SessionFactory才会关闭。  
2、而Session的对象是轻量级的，它是线程不安全的。对于单个业务进程单个工作单元而言，Session只被使用一次。创建Session时，并不会立即打开与数据库之间的连接，Session只在需要进行数据库操作时，才会获取JDBC连接。因此，打开和关闭Session，并不会对性能造成很大的影响。甚至即使无法确定一个请求是否需要数据访问，也可以打开Session对象，因为如果不进行数据库访问，Session不会获取JDBC连接。  
使用Spring管理hibernate的事务，在每个dao操作中使用SessionFactory.getCurrentSession()方法，该方法可以得到当前事务绑定的session。同时当前的Session和关联的Hibernate事务被绑定到当前线程上，虽然session不是线程安全的，但是通过这样的方式，每一个session都处于单线程中，避免session线程安全问题

3、不通过Spring管理事务，开启事务的主动性：

在sessionFactory.openSession()中，Hibernate会初始化数据库连接，与此同时，将其 AutoCommit设为关闭状态，这就是说，从SessionFactory获得session，其自动提交属性就已经被关闭了，事务需要主动、显示的调用才能生效，下面的代码不会对事务性数据库产生任何效果。  
session=sessionFactory.openSession();  
   session.save(user);  
   session.close();  
如果要使得代码真正作用到数据库，必须显示的调用Transaction指令  
   session=sessionFactory.openSession();  
   Transaction tx = session.beginTransaction();  
   session.save(user);  
   tx.commit();  
   session.close();

前面1、2点都是没什么问题，但是第3点事务的问题，我跟该博主意见不一致，首先Spring可以管理由Hibernate的SessionFactory产生的事务，所以通过Spring管理Hibernate的事务是可行的。其次，在开启事务配置的同时，框架已经在自动帮你管理事务，也就是刚刚连续两个操作，一个save一个sqlquery，由于sqlquery失败导致save同时失败回滚（或者说就没有commit的后续操作），而如果再使用session.beginTransaction()则会导致org.hibernate.TransactionException: nested transactions not supported错误，分析字面上的意思和根据本人查询结果综合来看，就是因为多次开启事务导致错误，反过来说事务不需要在session.beginTransaction()的情况下才开启，框架自动已经就在配置好以后就开启了。

注意，在以下情况中spring的事务管理会失效：

private 方法无法添加事务管理.

final 方法无法添加事务管理.

static 方法无法添加事务管理.

当绕过代理对象, 直接调用添加事务管理的方法时, 事务管理将无法生效.

**重点**：txManager 事物拦截器中可以分别对jdbctemplate进行事物，也可以对mySessionFactory进行事物管理。不同的实现方式（jdbctemplate/mySessionFactory）对应的事物管理器的实现类不同

在Spring托管中，使用getCurrentSession()获得的session并不是程序员自己控制的，session的生命周期交由Spring管理。而且要配置Spring事务管理器，并且使用getCurrentSession()的方法要加入事务管理器中。

当然也可以使用openSession(),自己来管理事务好session的生命周期，这样会有很多冗余代码，也没有应用Spring的特性

<!-- 事务管理器 -->  
<!--<bean id="txManager"-->  
 <!--class="org.springframework.jdbc.datasource.DataSourceTransactionManager">-->  
 <!--<property name="dataSource" ref="dataSource" />-->  
<!--</bean>-->  
  
<bean id="txManager"  
 class="org.springframework.orm.hibernate3.HibernateTransactionManager">  
 <!--class="org.springframework.jdbc.datasource.DataSourceTransactionManager">-->  
 <property name="sessionFactory" ref="mySessionFactory" />  
</bean>

实现的实例，供以后参考

<?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:p="http://www.springframework.org/schema/p"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">  
  
 <bean id="propertyConfigurer"  
 class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  
 <property name="locations">  
 <list>  
 <value>classpath:srv/spring/db.properties</value>  
 </list>  
 </property>  
 </bean>  
 <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">  
 <property name="url" value="${db.jdbcUrl}" />   
 <property name="username" value="${db.user}" />   
 <property name="password" value="${db.password}" />  
 <property name="filters" value="stat" />  
 <property name="maxActive" value="400" />  
 <property name="initialSize" value="20" />  
 <property name="maxWait" value="60000" />  
 <property name="minIdle" value="1" />  
 <property name="timeBetweenEvictionRunsMillis" value="60000" />  
 <property name="minEvictableIdleTimeMillis" value="300000" />  
 <property name="validationQuery" value="SELECT \* from dual" />  
 <property name="testWhileIdle" value="true" />  
 <property name="testOnBorrow" value="false" />  
 <property name="testOnReturn" value="false" />  
 <property name="poolPreparedStatements" value="true" />  
 <property name="maxPoolPreparedStatementPerConnectionSize" value="100" />  
 <property name="removeAbandonedTimeout" value="1800" />  
 <property name="logAbandoned" value="false" />  
 </bean>  
   
 <bean id="mySessionFactory"  
 class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">  
 <property name="dataSource" ref="dataSource" />  
 <!-- <property name="mappingDirectoryLocations"> -->  
 <property name="mappingLocations">  
 <list>  
 <value>classpath:srv/hibernate/mappings/hub/\*.hbm.xml</value>  
 </list>  
 </property>  
  
 <property name="hibernateProperties">  
 <props>  
 <prop key="hibernate.dialect">${db.hibernate.dialect}</prop>  
 <prop key="hibernate.show\_sql">true</prop>  
 <prop key="hibernate.generate\_statistics">true</prop>  
 <prop key="hibernate.connection.release\_mode">auto</prop>  
 <prop key="hibernate.autoReconnect">true</prop>  
 <prop key="hibernate.cache.use\_second\_level\_cache">false</prop>  
 <prop key="hibernate.format\_sql">false</prop>  
 </props>  
 </property>  
 </bean>  
 <!-- 事务管理器 -->  
 <!--<bean id="txManager"-->  
 <!--class="org.springframework.jdbc.datasource.DataSourceTransactionManager">-->  
 <!--<property name="dataSource" ref="dataSource" />-->  
 <!--</bean>-->  
  
 <bean id="txManager"  
 class="org.springframework.orm.hibernate3.HibernateTransactionManager">  
 <!--class="org.springframework.jdbc.datasource.DataSourceTransactionManager">-->  
 <property name="sessionFactory" ref="mySessionFactory" />  
 </bean>  
  
 <bean id="transactionInterceptor"  
 class="org.springframework.transaction.interceptor.TransactionInterceptor">  
 <!-- 事务拦截器bean需要依赖注入一个事务管理器 -->  
 <property name="transactionManager" ref="txManager" />  
 <property name="transactionAttributes">  
 <!-- 下面定义事务传播属性 -->  
 <props>  
 <prop key="find\*">PROPAGATION\_REQUIRED,readOnly</prop>  
 <prop key="list\*">PROPAGATION\_REQUIRED,readOnly</prop>  
 <prop key="get\*">PROPAGATION\_REQUIRED,readOnly</prop>  
 <prop key="del\*">PROPAGATION\_REQUIRED,-Exception</prop>  
 <prop key="add\*">PROPAGATION\_REQUIRED,-Exception</prop>  
 <prop key="save\*">PROPAGATION\_REQUIRED,-Exception</prop>  
 <prop key="update\*">PROPAGATION\_REQUIRED,-Exception</prop>  
 <prop key="call\*">PROPAGATION\_REQUIRED,-Exception</prop>  
 <prop key="\*">PROPAGATION\_REQUIRED,readOnly</prop>  
 </props>  
 </property>  
 </bean>  
 <bean id="transaction"  
 class="org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator">  
 <!-- 定义BeanNameAutoProxyCreator -->  
 <property name="beanNames">  
 <!-- 下面是所有需要自动创建事务代理的bean 所有以ServiceImpl 结尾的都会创建事务代理 -->  
 <value>\*ServiceImpl\*</value>  
 </property>  
 <!-- 下面定义BeanNameAutoProxyCreator所需的事务拦截器 -->  
 <property name="interceptorNames">  
 <list>  
 <!-- 此处可增加其他新的Interceptor -->  
  
 <!-- 事务拦截器 -->  
 <value>transactionInterceptor</value>  
  
 </list>  
 </property>  
 </bean>  
  
 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate" >  
 <property name="dataSource">  
 <ref bean="dataSource" />  
 </property>  
 </bean>  
 <!--  
 <bean id = "commonJdbcDaoImpl4Ysgl" class = "core.source.dao.CommonJdbcDaoImpl" >   
 <property name="jdbcTemplate" ref="jdbcTemplate" />   
 </bean>  
 -->  
   
</beans>

Spring事务配置的五种方式

    前段时间对Spring的事务配置做了比较深入的研究，在此之间对Spring的事务配置虽说也配置过，但是一直没有一个清楚的认识。通过这次的学习发觉Spring的事务配置只要把思路理清，还是比较好掌握的。

    总结如下：

    Spring配置文件中关于事务配置总是由三个组成部分，分别是DataSource、TransactionManager和代理机制这三部分，无论哪种配置方式，一般变化的只是代理机制这部分。

    DataSource、TransactionManager这两部分只是会根据数据访问方式有所变化，比如使用Hibernate进行数据访问时，DataSource实际为SessionFactory，TransactionManager的实现为HibernateTransactionManager。

    具体如下图：

    根据代理机制的不同，总结了五种Spring事务的配置方式，配置文件如下：

**第一种方式：每个Bean都有一个代理**

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

    xsi:schemaLocation="http://www.springframework.org/schema/beans

           http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

           http://www.springframework.org/schema/context

           http://www.springframework.org/schema/context/spring-context-2.5.xsd

           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

    <bean id="sessionFactory"

            class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

        <property name="configLocation" value="classpath:hibernate.cfg.xml" />

        <property name="configurationClass" value="org.hibernate.cfg.AnnotationConfiguration" />

    </bean>

    <!-- 定义事务管理器（声明式的事务） -->

    <bean id="transactionManager"

        class="org.springframework.orm.hibernate3.HibernateTransactionManager">

        <property name="sessionFactory" ref="sessionFactory" />

    </bean>

    <!-- 配置DAO -->

    <bean id="userDaoTarget" class="com.bluesky.spring.dao.UserDaoImpl">

        <property name="sessionFactory" ref="sessionFactory" />

    </bean>

    <bean id="userDao"

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

           <!-- 配置事务管理器 -->

           <property name="transactionManager" ref="transactionManager" />

        <property name="target" ref="userDaoTarget" />

         <property name="proxyInterfaces" value="com.bluesky.spring.dao.GeneratorDao" />

        <!-- 配置事务属性 -->

        <property name="transactionAttributes">

            <props>

                <prop key="\*">PROPAGATION\_REQUIRED</prop>

            </props>

        </property>

    </bean>

</beans>

**第二种方式：所有Bean共享一个代理基类**

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

    xsi:schemaLocation="http://www.springframework.org/schema/beans

           http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

           http://www.springframework.org/schema/context

           http://www.springframework.org/schema/context/spring-context-2.5.xsd

           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

    <bean id="sessionFactory"

            class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

        <property name="configLocation" value="classpath:hibernate.cfg.xml" />

        <property name="configurationClass" value="org.hibernate.cfg.AnnotationConfiguration" />

    </bean>

    <!-- 定义事务管理器（声明式的事务） -->

    <bean id="transactionManager"

        class="org.springframework.orm.hibernate3.HibernateTransactionManager">

        <property name="sessionFactory" ref="sessionFactory" />

    </bean>

    <bean id="transactionBase"

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

            lazy-init="true" abstract="true">

        <!-- 配置事务管理器 -->

        <property name="transactionManager" ref="transactionManager" />

        <!-- 配置事务属性 -->

        <property name="transactionAttributes">

            <props>

                <prop key="\*">PROPAGATION\_REQUIRED</prop>

            </props>

        </property>

    </bean>

    <!-- 配置DAO -->

    <bean id="userDaoTarget" class="com.bluesky.spring.dao.UserDaoImpl">

        <property name="sessionFactory" ref="sessionFactory" />

    </bean>

    <bean id="userDao" parent="transactionBase" >

        <property name="target" ref="userDaoTarget" />

    </bean>

</beans>

**第三种方式：使用拦截器**

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

    xsi:schemaLocation="http://www.springframework.org/schema/beans

           http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

           http://www.springframework.org/schema/context

           http://www.springframework.org/schema/context/spring-context-2.5.xsd

           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

    <bean id="sessionFactory"

            class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

        <property name="configLocation" value="classpath:hibernate.cfg.xml" />

        <property name="configurationClass" value="org.hibernate.cfg.AnnotationConfiguration" />

    </bean>

    <!-- 定义事务管理器（声明式的事务） -->

    <bean id="transactionManager"

        class="org.springframework.orm.hibernate3.HibernateTransactionManager">

        <property name="sessionFactory" ref="sessionFactory" />

    </bean>

    <bean id="transactionInterceptor"

        class="org.springframework.transaction.interceptor.TransactionInterceptor">

        <property name="transactionManager" ref="transactionManager" />

        <!-- 配置事务属性 -->

        <property name="transactionAttributes">

            <props>

                <prop key="\*">PROPAGATION\_REQUIRED</prop>

            </props>

        </property>

    </bean>

    <bean class="org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator">

        <property name="beanNames">

            <list>

                <value>\*Dao</value>

            </list>

        </property>

        <property name="interceptorNames">

            <list>

                <value>transactionInterceptor</value>

            </list>

        </property>

    </bean>

    <!-- 配置DAO -->

    <bean id="userDao" class="com.bluesky.spring.dao.UserDaoImpl">

        <property name="sessionFactory" ref="sessionFactory" />

    </bean>

</beans>

**第四种方式：使用tx标签配置的拦截器**

<?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: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-2.5.xsd

           http://www.springframework.org/schema/context

           http://www.springframework.org/schema/context/spring-context-2.5.xsd

           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.5.xsd

           http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">

    <context:annotation-config />

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

    <bean id="sessionFactory"

            class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

        <property name="configLocation" value="classpath:hibernate.cfg.xml" />

        <property name="configurationClass" value="org.hibernate.cfg.AnnotationConfiguration" />

    </bean>

    <!-- 定义事务管理器（声明式的事务） -->

    <bean id="transactionManager"

        class="org.springframework.orm.hibernate3.HibernateTransactionManager">

        <property name="sessionFactory" ref="sessionFactory" />

    </bean>

    <tx:advice id="txAdvice" transaction-manager="transactionManager">

        <tx:attributes>

            <tx:method name="\*" propagation="REQUIRED" />

        </tx:attributes>

    </tx:advice>

    <aop:config>

        <aop:pointcut id="interceptorPointCuts"

            expression="execution(\* com.bluesky.spring.dao.\*.\*(..))" />

        <aop:advisor advice-ref="txAdvice"

            pointcut-ref="interceptorPointCuts" />

    </aop:config>

</beans>

**第五种方式：全注解**

<?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: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-2.5.xsd

           http://www.springframework.org/schema/context

           http://www.springframework.org/schema/context/spring-context-2.5.xsd

           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.5.xsd

           http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">

    <context:annotation-config />

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

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

    <bean id="sessionFactory"

            class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

        <property name="configLocation" value="classpath:hibernate.cfg.xml" />

        <property name="configurationClass" value="org.hibernate.cfg.AnnotationConfiguration" />

    </bean>

    <!-- 定义事务管理器（声明式的事务） -->

    <bean id="transactionManager"

        class="org.springframework.orm.hibernate3.HibernateTransactionManager">

        <property name="sessionFactory" ref="sessionFactory" />

    </bean>

</beans>

此时在DAO上需加上@Transactional注解，如下：

package com.bluesky.spring.dao;

import java.util.List;

import org.hibernate.SessionFactory;

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

import org.springframework.orm.hibernate3.support.HibernateDaoSupport;

import org.springframework.stereotype.Component;

import com.bluesky.spring.domain.User;

@Transactional

@Component("userDao")

public class UserDaoImpl extends HibernateDaoSupport implements UserDao {

    public List<User> listUsers() {

        return this.getSession().createQuery("from User").list();