个人资料

xzknet

访问：271762次
积分：4961分
排名：第526名

原创：161篇 转载：136篇
译文：2篇 评论：259条

转 Spring 多数据源事务配置问题

分类：Java 2011-07-30 15:39 102人阅读 评论(0) 收藏 举报

在SpringSide 3 中，白衣提供的预先配置好的环境非常有利于用户进行快速开发，但是同时也会为扩展带来一些困难。最直接的例子就是关于在项目中使用多个数据源的问题，似乎 很难搞。在上一篇中，我探讨了SpringSide 3 中的数据访问层，在这一篇中，我立志要解决多数据源配置的难题，我的思路是这样的：

第一步、测试能否配置多个DataSource
第二步、测试能否配置多个SessionFactory
第三步、测试能否配置多个TransactionManager
第四步、测试能否使用多个TransactionManager，也就是看能否配置多个

基本上到第四步就应该走不通了，因为Spring中似乎不能配置多个，而且@transactional注解也无法让用户选择具体使用哪个TransactionManager。也就是说，在SpringSide的应用中，不能让不同的数据源分别属于不同的事务管理器，多数据源只能使用分布式事务管理器，那么测试思路继续如下进行：

第五步、测试能否配置JTATransactionManager

如果到这一步，项目还能顺利在Tomcat中运行的话，我们就算大功告成了。但我总认为事情不会那么顺利，我总觉得JTATransactionManager需要应用服务器的支持，而且需要和JNDI配合使用，具体是不是这样，那只有等测试后才知道。如果被我不幸言中，那么进行下一步：

第六步、更换Tomcat为GlassFish，更换JDBC的DataSource为JNDI查找的DataSource，然后配置JTATransactionManager

下面测试开始，先假设场景，还是继续用上一篇中提到的简单的文章发布系统，假设该系统运行一段时间后非常火爆，单靠一台服务器已经无法支持巨大的用户数， 这时候，站长想到了把数据进行水平划分，于是，需要建立一个索引数据库，该索引数据库需保存每一篇文章的Subject及其内容所在的Web服务器，而每 一个Web服务器上运行的项目，需要同时访问索引数据库和内容数据库。所以，需要创建索引数据库，如下：

```
create database puretext_index;

use puretext_index;

create table articles(
id int primary key auto_increment,
subject varchar(256),
webserver varchar(30)
```

文章存档

2011年08月(5)
2011年07月(7)
2011年06月(3)
2011年05月(5)
2011年04月(2)
展开

阅读排行

联想一键恢复自己装(4.6) (6864)
Google Android 初体验（吼... (5778)
用 OpenSessionInViewI... (5342)
Ubuntu安装GCC编译器 (5216)
使用FLEX3开发大型多人在线游戏 (4945)
Eclipse和MyEclipse的里程... (4457)
Eclpise3.3中使用JadClip... (3987)
可选择也可以输入的下拉列表框 (3846)
可免费分发的Oracle小巧客户端：Or... (3591)
在Eclipse中将Java项目打包为j... (3481)

评论排行

Google Android 初体验（吼... (21)
Oracle 批处理启动关闭服务 (18)
解析eclipse下生成Hibernat... (15)
联想一键恢复自己装(4.6) (13)
Eclpise3.3中使用JadClip... (9)
Eclipse和MyEclipse的里程... (9)
exe4j视频教程 (8)
J2ME里面的一些未实现的三角函数 (8)
使用FLEX3开发大型多人在线游戏 (8)
让IE和Firefox都支持innerT... (7)

最新评论

挺好的 学习了
谢了，我下了openoffic...
@taoqiadai:顺便说下...
坑爹那，我看到有个举办按钮做的...
这个漂亮～
真服了你了 文章中那么多错的也...
总结也要说转载么？没必要吧？

```
);
```

第一步测试，配置多个DataSource，配置文件如下：

application.properties：

```
jdbc.urlContent=jdbc:mysql://localhost:3306/PureText useUnicode=true&characterEncoding=utf8
jdbc.urlIndex=jdbc:mysql://localhost:3306/PureText_Index useUnicode=true&characterEncoding=utf8
```

applicationContext.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:jee="http://www.springframework.org/schema/jee" xmlns:tx="http://www.springfram
-lazy-
init="true" xsi:schemalocation=
"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring
beans-
2.5.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/sprin
tx-
2.5.xsd http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spr
jee-
2.5.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/con
context-2.5.xsd">

    <description>Spring公共配置文件 </description>

    <!-- 定义受环境影响易变的变量 -->
    <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
        <property name="systemPropertiesModeName" value="SYSTEM_PROPERTIES_MODE_OVERRIDE">
        <property name="ignoreResourceNotFound" value="true">
        <property name="locations">
            <list>
                <!-- 标准配置 -->
                <value>classpath*:/application.properties</value>
                <!-- 本地开发环境配置 -->
                <value>classpath*:/application.local.properties</value>
                <!-- 服务器生产环境配置 -->
                <!-->file:/var/myapp/application.server.properties -->
            </!--></list>
        </property>
    </property></property></bean>

    <!-- 使用annotation 自动注册bean,并保证@Required,@Autowired的属性被注入 -->
    <context:component-scan base-package="cn.puretext">

    <!-- 数据源配置,使用应用内的DBCP数据库连接池 -->
    <bean id="dataSourceContent" class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
        <!-- Connection Info -->
        <property name="driverClassName" value="com.mysql.jdbc.Driver">
        <property name="url" value="${jdbc.urlContent}">
        <property name="username" value="${jdbc.username}">
        <property name="password" value="${jdbc.password}">

        <!-- Connection Pooling Info -->
        <property name="initialSize" value="5">
        <property name="maxActive" value="100">
```

```
        <property name="maxIdle" value="30">
        <property name="maxWait" value="1000">
        <property name="poolPreparedStatements" value="true">
        <property name="defaultAutoCommit" value="false">
    </property></property></property></property></property></property></property></property></p
    <bean id="dataSourceIndex" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <!-- Connection Info -->
        <property name="driverClassName" value="com.mysql.jdbc.Driver">
        <property name="url" value="${jdbc.urlIndex}">
        <property name="username" value="${jdbc.username}">
        <property name="password" value="${jdbc.password}">

        <!-- Connection Pooling Info -->
        <property name="initialSize" value="5">
        <property name="maxActive" value="100">
        <property name="maxIdle" value="30">
        <property name="maxWait" value="1000">
        <property name="poolPreparedStatements" value="true">
        <property name="defaultAutoCommit" value="false">
    </property></property></property></property></property></property></property></property></p

    <!-- 数据源配置,使用应用服务器的数据库连接池 -->
    <!--<jee:jndi-lookup id="dataSource" jndi-name="java:comp/env/jdbc/ExampleDB">-->

    <!-- Hibernate配置 -->
    <bean id="sessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSe>
        <property name="dataSource" ref="dataSourceContent">
        <property name="namingStrategy">
            <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
        </bean></property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQL5InnoDBDialect</prop>
                <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
                <prop key="hibernate.format_sql">${hibernate.format_sql}</prop>
                <prop key="hibernate.cache.provider_class">org.hibernate.cache.EhCacheProvider
                </prop>
                <prop key="hibernate.cache.provider_configuration_file_resource_path">${hibernate.ehcache_config_file}</prop>
            </props>
        </property>
        <property name="packagesToScan" value="cn.puretext.entity.*">
    </property></property></bean>

    <!-- 事务管理器配置,单数据源事务 -->
    <bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactio>
        <property name="sessionFactory" ref="sessionFactory">
    </property></bean>

    <!-- 事务管理器配置,多数据源JTA事务-->
    <!--
 id="transactionManager" class="org.springframework.transaction.jta.JtaTransactionManager or
        WebLogicJtaTransactionManager">
    -->

    <!-- 使用annotation定义事务 -->
    <tx:annotation-driven transaction-manager="transactionManager">
</tx:annotation-driven></!--></!--<jee:jndi-lookup></context:component-scan></beans>
```

这个时候运行上一篇文章中写好的单元测试DaoTest.java，结果发现还是会出错，错误原因如下：

org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'cn.puretext.unit.service.DaoTest': Autowiring of methods failed; nested exception is org.springframework.beans.factory.BeanCreationException: Could not autowire method: public void org.springframework.test.context.junit4.AbstractTransactionalJUnit4SpringContextTests.setDataSource(javax.sql.Dat nested exception is org.springframework.beans.factory.NoSuchBeanDefinitionException: No unique bean of type [javax.sql.DataSource] is defined: expected single matching bean but found 2: [dataSourceContent, dataSourceIndex]

经过分析，发现是测试类的基类需要注入DataSource，而现在配置了多个DataSource，所以Spring不知道哪个DataSource匹配了，所以需要改写DaoTest.java，如下：

```
01. package cn.puretext.unit.service;
02.
03. import java.util.List;
04.
05. import javax.annotation.Resource;
06. import javax.sql.DataSource;
07.
08. import org.junit.Test;
09. import org.springframework.beans.factory.annotation.Autowired;
10. import org.springside.modules.orm.Page;
11. import org.springside.modules.test.junit4.SpringTxTestCase;
12.
13. import cn.puretext.dao.ArticleDao;
14. import cn.puretext.entity.web.Article;
15.
16. public class DaoTest extends SpringTxTestCase {
17.     @Autowired
18.     private ArticleDao articleDao;
19.
20.     public ArticleDao getArticleDao() {
21.         return articleDao;
22.     }
23.
24.     public void setArticleDao(ArticleDao articleDao) {
25.         this.articleDao = articleDao;
26.     }
27.
28.     @Override
29.     @Resource(name = "dataSourceContent")
30.     public void setDataSource(DataSource dataSource) {
31.         // TODO Auto-generated method stub
32.         super.setDataSource(dataSource);
33.     }
34.
35.     @Test
36.     public void addArticle() {
37.         Article article = new Article();
38.         article.setSubject("article test");
39.         article.setContent("article test");
40.         articleDao.save(article);
41.     }
42.
43.     @Test
44.     public void pageQuery() {
45.         Page<article> page = new Page<article>();
46.         page.setPageSize(10);
47.         page.setPageNo(2);
48.         page = articleDao.getAll(page);
```

```
        List<article> articles = page.getResult();
    }
}

</article></article></article>
```

改变的内容主要为重写了基类中的setDataSource方法，并使用@Resource注解指定使用的DataSource为dataSourceContent。经过修改后，单元测试成功运行。

第二步，配置多个SessionFactory，配置文件如下：

```
< 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:jee="http://www.springframework.org/schema/jee" xmlns:tx="http://www.springfram
-lazy-
init="true" xsi:schemalocation=
"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring
beans-
2.5.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/sprin
tx-
2.5.xsd http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spr
jee-
2.5.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/con
context-2.5.xsd">

    <description>Spring公共配置文件 </description>

    <!-- 定义受环境影响易变的变量 -->
    <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
        <property name="systemPropertiesModeName" value="SYSTEM_PROPERTIES_MODE_OVERRIDE">
        <property name="ignoreResourceNotFound" value="true">
        <property name="locations">
            <list>
                <!-- 标准配置 -->
                <value>classpath*:/application.properties</value>
                <!-- 本地开发环境配置 -->
                <value>classpath*:/application.local.properties</value>
                <!-- 服务器生产环境配置 -->
                <!-->file:/var/myapp/application.server.properties -->
            </!--></list>
        </property>
    </property></property></bean>

    <!-- 使用annotation 自动注册bean,并保证@Required,@Autowired的属性被注入 -->
    <context:component-scan base-package="cn.puretext">

    <!-- 数据源配置,使用应用内的DBCP数据库连接池 -->
    <bean id="dataSourceContent" class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
        <!-- Connection Info -->
        <property name="driverClassName" value="com.mysql.jdbc.Driver">
        <property name="url" value="${jdbc.urlContent}">
        <property name="username" value="${jdbc.username}">
        <property name="password" value="${jdbc.password}">

        <!-- Connection Pooling Info -->
        <property name="initialSize" value="5">
        <property name="maxActive" value="100">
        <property name="maxIdle" value="30">
        <property name="maxWait" value="1000">
```

```
        <property name="poolPreparedStatements" value="true">
        <property name="defaultAutoCommit" value="false">
    </property></property></property></property></property></property></property></property></p
    <bean id="dataSourceIndex" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <!-- Connection Info -->
        <property name="driverClassName" value="com.mysql.jdbc.Driver">
        <property name="url" value="${jdbc.urlIndex}">
        <property name="username" value="${jdbc.username}">
        <property name="password" value="${jdbc.password}">

        <!-- Connection Pooling Info -->
        <property name="initialSize" value="5">
        <property name="maxActive" value="100">
        <property name="maxIdle" value="30">
        <property name="maxWait" value="1000">
        <property name="poolPreparedStatements" value="true">
        <property name="defaultAutoCommit" value="false">
    </property></property></property></property></property></property></property></property></p

    <!-- 数据源配置,使用应用服务器的数据库连接池 -->
    <!--<jee:jndi-lookup id="dataSource" jndi-name="java:comp/env/jdbc/ExampleDB">-->

    <!-- Hibernate配置 -->
    <bean id="sessionFactoryContent" class="org.springframework.orm.hibernate3.annotation.Annot>
        <property name="dataSource" ref="dataSourceContent">
        <property name="namingStrategy">
            <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
        </bean></property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQL5InnoDBDialect</prop>
                <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
                <prop key="hibernate.format_sql">${hibernate.format_sql}</prop>
                <prop key="hibernate.cache.provider_class">org.hibernate.cache.EhCacheProvider
                </prop>
                <prop key="hibernate.cache.provider_configuration_file_resource_path">${hibernate.ehcache_config_file}</prop>
            </props>
        </property>
        <property name="packagesToScan" value="cn.puretext.entity.*">
    </property></property></bean>
    <bean id="sessionFactoryIndex" class="org.springframework.orm.hibernate3.annotation.Annotat>
        <property name="dataSource" ref="dataSourceIndex">
        <property name="namingStrategy">
            <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
        </bean></property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQL5InnoDBDialect</prop>
                <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
                <prop key="hibernate.format_sql">${hibernate.format_sql}</prop>
                <prop key="hibernate.cache.provider_class">org.hibernate.cache.EhCacheProvider
                </prop>
                <prop key="hibernate.cache.provider_configuration_file_resource_path">${hibernate.ehcache_config_file}</prop>
            </props>
        </property>
        <property name="packagesToScan" value="cn.puretext.entity.*">
    </property></property></bean>

```

```
    <!-- 事务管理器配置,单数据源事务 -->
    <bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactio
>
        <property name="sessionFactory" ref="sessionFactoryContent">
    </property></bean>

    <!-- 事务管理器配置,多数据源JTA事务-->
    <!--
 id="transactionManager" class="org.springframework.transaction.jta.JtaTransactionManager or
        WebLogicJtaTransactionManager">
    -->

    <!-- 使用annotation定义事务 -->
    <tx:annotation-driven transaction-manager="transactionManager">
</tx:annotation-driven></!--></!--<jee:jndi-lookup></context:component-scan></beans>
```

运行单元测试，报错，错误代码如下：

org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'cn.puretext.unit.service.DaoTest': Autowiring of fields failed; nested exception is org.springframework.beans.factory.BeanCreationException: Could not autowire field: private cn.puretext.dao.ArticleDao cn.puretext.unit.service.DaoTest.articleDao; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'articleDao': Autowiring of methods failed; nested exception is org.springframework.beans.factory.BeanCreationException: Could not autowire method: public void org.springside.modules.orm.hibernate.SimpleHibernateDao.setSessionFactory(org.hibernate.SessionFactory); nested exception is org.springframework.beans.factory.NoSuchBeanDefinitionException: No unique bean of type [org.hibernate.SessionFactory] is defined: expected single matching bean but found 2: [sessionFactoryContent, sessionFactoryIndex]

这和上面出现的错误是异曲同工的，只不过这次是ArticleDao类里面不知道注入哪一个SessionFactory，因此，需要修改ArticleDao类，重写setSessionFactory方法，并用@Resource注解指定，如下：

```
package cn.puretext.dao;


import javax.annotation.Resource;

import org.hibernate.SessionFactory;
import org.springframework.stereotype.Repository;
import org.springside.modules.orm.hibernate.HibernateDao;

import cn.puretext.entity.web.Article;

@Repository
public class ArticleDao extends HibernateDao<article, long=""> {

    @Override
    @Resource(name = "sessionFactoryContent")
    public void setSessionFactory(SessionFactory sessionFactory) {
        // TODO Auto-generated method stub
        super.setSessionFactory(sessionFactory);
    }

}
</article,>
```

运行单元测试，成功。

第三步、配置多个TransactionManager，如下：

```
< 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:jee="http://www.springframework.org/schema/jee" xmlns:tx="http://www.springfram
-lazy-
init="true" xsi:schemalocation=
"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring
beans-
2.5.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/sprin
tx-
2.5.xsd http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spr
jee-
2.5.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/con
context-2.5.xsd">

    <description>Spring公共配置文件 </description>

    <!-- 定义受环境影响易变的变量 -->
    <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
        <property name="systemPropertiesModeName" value="SYSTEM_PROPERTIES_MODE_OVERRIDE">
        <property name="ignoreResourceNotFound" value="true">
        <property name="locations">
            <list>
                <!-- 标准配置 -->
                <value>classpath*:/application.properties</value>
                <!-- 本地开发环境配置 -->
                <value>classpath*:/application.local.properties</value>
                <!-- 服务器生产环境配置 -->
                <!-->file:/var/myapp/application.server.properties -->
            </!--></list>
        </property>
    </property></property></bean>

    <!-- 使用annotation 自动注册bean,并保证@Required,@Autowired的属性被注入 -->
    <context:component-scan base-package="cn.puretext">

    <!-- 数据源配置,使用应用内的DBCP数据库连接池 -->
    <bean id="dataSourceContent" class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
        <!-- Connection Info -->
        <property name="driverClassName" value="com.mysql.jdbc.Driver">
        <property name="url" value="${jdbc.urlContent}">
        <property name="username" value="${jdbc.username}">
        <property name="password" value="${jdbc.password}">

        <!-- Connection Pooling Info -->
        <property name="initialSize" value="5">
        <property name="maxActive" value="100">
        <property name="maxIdle" value="30">
        <property name="maxWait" value="1000">
        <property name="poolPreparedStatements" value="true">
        <property name="defaultAutoCommit" value="false">
    </property></property></property></property></property></property></property></property></p
    <bean id="dataSourceIndex" class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
        <!-- Connection Info -->
        <property name="driverClassName" value="com.mysql.jdbc.Driver">
        <property name="url" value="${jdbc.urlIndex}">
```

```
        <property name="username" value="${jdbc.username}">
        <property name="password" value="${jdbc.password}">

        <!-- Connection Pooling Info -->
        <property name="initialSize" value="5">
        <property name="maxActive" value="100">
        <property name="maxIdle" value="30">
        <property name="maxWait" value="1000">
        <property name="poolPreparedStatements" value="true">
        <property name="defaultAutoCommit" value="false">
    </property></property></property></property></property></property></property></property></p

    <!-- 数据源配置,使用应用服务器的数据库连接池 -->
    <!--<jee:jndi-lookup id="dataSource" jndi-name="java:comp/env/jdbc/ExampleDB">-->

    <!-- Hibernate配置 -->
    <bean id="sessionFactoryContent" class="org.springframework.orm.hibernate3.annotation.Annot
>
        <property name="dataSource" ref="dataSourceContent">
        <property name="namingStrategy">
            <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
        </bean></property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect"
>org.hibernate.dialect.MySQL5InnoDBDialect</prop>
                <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
                <prop key="hibernate.format_sql">${hibernate.format_sql}</prop>
                <prop key="hibernate.cache.provider_class"
>org.hibernate.cache.EhCacheProvider
                </prop>
                <prop key="hibernate.cache.provider_configuration_file_resource_path"
>${hibernate.ehcache_config_file}</prop>
            </props>
        </property>
        <property name="packagesToScan" value="cn.puretext.entity.*">
    </property></property></bean>
    <bean id="sessionFactoryIndex" class="org.springframework.orm.hibernate3.annotation.Annotat
>
        <property name="dataSource" ref="dataSourceIndex">
        <property name="namingStrategy">
            <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
        </bean></property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect"
>org.hibernate.dialect.MySQL5InnoDBDialect</prop>
                <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
                <prop key="hibernate.format_sql">${hibernate.format_sql}</prop>
                <prop key="hibernate.cache.provider_class"
>org.hibernate.cache.EhCacheProvider
                </prop>
                <prop key="hibernate.cache.provider_configuration_file_resource_path"
>${hibernate.ehcache_config_file}</prop>
            </props>
        </property>
        <property name="packagesToScan" value="cn.puretext.entity.*">
    </property></property></bean>

    <!-- 事务管理器配置,单数据源事务 -->
    <bean id="transactionManagerContent" class="org.springframework.orm.hibernate3.HibernateTra
>
        <property name="sessionFactory" ref="sessionFactoryContent">
    </property></bean>
    <bean id="transactionManagerIndex" class="org.springframework.orm.hibernate3.HibernateTrans
>
        <property name="sessionFactory" ref="sessionFactoryIndex">
```

```
    </property></bean>

    <!-- 事务管理器配置,多数据源JTA事务-->
    <!--
 id="transactionManager" class="org.springframework.transaction.jta.JtaTransactionManager or
        WebLogicJtaTransactionManager">
    -->

    <!-- 使用annotation定义事务 -->
    <tx:annotation-driven transaction-manager="transactionManagerContent">
</tx:annotation-driven></!--></!--<jee:jndi-lookup></context:component-scan></beans>

这个时候运行还是会出错，出错的原因
为 org.springframework.beans.factory.NoSuchBeanDefinitionException: No bean named 'transactionM
，因为该出错信息很短，我也难以找出究竟是哪个地方需要名为“transactionManager”的事务管理器 ，改个名字都
不行，看来Spring的自动注入有时候也错综复杂害人不浅。不过，如果把上面的其中一个名字改
成“transactionManger”， 另外一个名字不改，运行是成功的，如下:
<!-- 事务管理器配置,单数据源事务 -->
    <bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactio
>
        <property name="sessionFactory" ref="sessionFactoryContent">
    </property></bean>
    <bean id="transactionManagerIndex" class="org.springframework.orm.hibernate3.HibernateTrans
>
        <property name="sessionFactory" ref="sessionFactoryIndex">
    </property></bean>
```

这个时候得出结论是，可以配置多个TransactionManager，但是必须有一个的名字是transactionManager。

第四步、配置多个，如下:

```
<!-- 使用annotation定义事务 -->
    <tx:annotation-driven transaction-manager="transactionManager">
    <tx:annotation-driven transaction-manager="transactionManagerIndex">
</tx:annotation-driven></tx:annotation-driven>
```

运行测试，天啦，竟然成功了。和我之前预料的完全不一样，居然在一个配置文件中配置多个一点 问题都没有。那么在使用@Transactional的地方，它真的能够选择正确的事务管理器吗？我不得不写更多的代码来进行测试。那就针对索引数据库中 的表写一个Entity，写一个Dao测试一下吧。

代码如下:

```
package cn.puretext.entity.web;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Table;

import org.hibernate.annotations.Cache;
import org.hibernate.annotations.CacheConcurrencyStrategy;

import cn.puretext.entity.IdEntity;

@Entity
// 表名与类名不相同时重新定义表名.
```

```
@Table(name = "articles")
// 默认的缓存策略.
@Cache(usage = CacheConcurrencyStrategy.READ_WRITE)
public class ArticleIndex extends IdEntity {
    private String subject;
    private String webServer;

    public String getSubject() {
        return subject;
    }

    public void setSubject(String subject) {
        this.subject = subject;
    }
    @Column(name = "webserver")
    public String getWebServer() {
        return webServer;
    }

    public void setWebServer(String webServer) {
        this.webServer = webServer;
    }
}
```

```
package cn.puretext.dao;

import javax.annotation.Resource;

import org.hibernate.SessionFactory;
import org.springframework.stereotype.Repository;
import org.springside.modules.orm.hibernate.HibernateDao;

import cn.puretext.entity.web.ArticleIndex;

@Repository
public class ArticleIndexDao extends HibernateDao<articleindex, long=""> {
    @Override
    @Resource(name = "sessionFactoryIndex")
    public void setSessionFactory(SessionFactory sessionFactory) {
        // TODO Auto-generated method stub
        super.setSessionFactory(sessionFactory);
    }
}</articleindex,>
```

```
package cn.puretext.unit.service;

import java.util.List;

import javax.annotation.Resource;
import javax.sql.DataSource;

import org.junit.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.transaction.annotation.Transactional;
import org.springside.modules.orm.Page;
import org.springside.modules.test.junit4.SpringTxTestCase;

import cn.puretext.dao.ArticleDao;
```

```
import cn.puretext.dao.ArticleIndexDao;
import cn.puretext.entity.web.Article;
import cn.puretext.entity.web.ArticleIndex;
import cn.puretext.service.ServiceException;

public class DaoTest extends SpringTxTestCase {
    @Autowired
    private ArticleDao articleDao;
    @Autowired
    private ArticleIndexDao articleIndexDao;

    public void setArticleIndexDao(ArticleIndexDao articleIndexDao) {
        this.articleIndexDao = articleIndexDao;
    }

    public void setArticleDao(ArticleDao articleDao) {
        this.articleDao = articleDao;
    }

    @Override
    @Resource(name = "dataSourceContent")
    public void setDataSource(DataSource dataSource) {
        // TODO Auto-generated method stub
        super.setDataSource(dataSource);
    }

    @Test
    @Transactional
    public void addArticle() {
        Article article = new Article();
        article.setSubject("article test");
        article.setContent("article test");
        articleDao.save(article);
    }

    @Test
    @Transactional
    public void pageQuery() {
        Page<article> page = new Page<article>();
        page.setPageSize(10);
        page.setPageNo(2);
        page = articleDao.getAll(page);
        List<article> articles = page.getResult();
    }

    @Test
    @Transactional
    public void addIndex() {
        ArticleIndex articleIndex = new ArticleIndex();
        articleIndex.setSubject("test");
        articleIndex.setWebServer("www001");
        articleIndexDao.save(articleIndex);
    }

    @Test
    @Transactional
    public void addArticleAndAddIndex() {
        addArticle();
        addIndex();
        throw new ServiceException("测试事务回滚");
    }
}
</article></article></article>
```

运行测试，结果还是成功的。到目前，发现在一个项目中使用多个TransactionManager可以正常运行，但是有两

个问题需要考虑：

1、为什么必须得有一个TransactionManager名字为transactionManager？

2、这两个TransactionManager真的能正常工作吗？

3、OpenSessionInView的问题怎么解决？

以上的三个问题在单元测试中是不能找出答案的，我只好再去写Action层的代码，期望能够从中得到线索。经过一天艰苦的努力，终于真相大白：

1、并不是必须有一个TransactionManager的名字为transactionMananger，这只是单元测试在搞鬼，在真实的Web环境 中，无论两个TransactionManager取什么名字都可以，运行不会报错。所以这个答案很明确，是因为单元测试的基类需要一个名为 transactionMananger的事务管理器。

2、在单元测试中，只能测试Dao类和Entity类能否正常工作，但是由于单元测试结束后事务会自动回滚，不会把数据写入到数据库中，所以没有办法确定 两个TransactionManager能否正常工作。在真实的Web环境中，问题很快就浮出水面，只有一个数据库中有数据，另外一个数据库中没有，经 过调整的位置并对比分析，发现只有放在前面的TransactionMananger的事务 能够正常提交，放在后面的TransactionManager的事务不能提交，所以永远只有一个数据库里面有数据。

3、如果早一点脱离单元测试而进入真实的Web环境，就会早一点发现OpenSessionInViewFilter的问题，因为只要配置多个 SessionFactory，运行的时候OpenSessionInViewFilter就会报错。为了解决这个问题，我只能去阅读OpenSessionInViewFilter的源代码，发现它在将Session绑定到线程的时候用的是Map，而且使用SessionFactory作为Map的key，这就说明在线程中绑定多个Session不会冲突，也进一步说明可以在web.xml中配置多个 OpenSessionInViewFilter。而我也正是通过配置多个OpenSessionInViewFilter来解决问题的。我的web.xml文件如下：

```
01. < xml version="1.0" encoding="UTF-8" >
02. <web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.4" xsi:schemalocation= "http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
03.
04.     <display-name>PureText</display-name>
05.     <!-- Spring ApplicationContext配置文件的路径,可使用通配符,多个路径用,号分隔
06.         此参数用于后面的Spring Context Loader -->
07.     <context-param>
08.         <param-name>contextConfigLocation</param-name>
09.         <param-value>classpath*:/applicationContext*.xml</param-value>
10.     </context-param>
11.
12.     <!-- Character Encoding filter -->
13.     <filter>
14.         <filter-name>encodingFilter</filter-name>
15.         <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
16.         <init-param>
17.             <param-name>encoding</param-name>
18.             <param-value>UTF-8</param-value>
19.         </init-param>
20.         <init-param>
21.             <param-name>forceEncoding</param-name>
22.             <param-value>true</param-value>
23.         </init-param>
24.     </filter>
25.
26.     <filter>
27.         <filter-name>hibernateOpenSessionInViewFilterContent</filter-name>
28.         <filter-class>org.springside.modules.orm.hibernate.OpenSessionInViewFilter</filter-clas>
29.         <init-param>
30.             <param-name>excludeSuffixs</param-name>
31.             <param-value>js,css,jpg,gif</param-value>
```

```
        </init-param>
        <init-param>
                <param-name>sessionFactoryBeanName</param-name>
            <param-value>sessionFactoryContent</param-value>
        </init-param>
    </filter>
    <filter>
        <filter-name>hibernateOpenSessionInViewFilterIndex</filter-name>
        <filter-class>org.springside.modules.orm.hibernate.OpenSessionInViewFilter</filter-clas
>
        <init-param>
            <param-name>excludeSuffixs</param-name>
            <param-value>js,css,jpg,gif</param-value>
        </init-param>
        <init-param>
                <param-name>sessionFactoryBeanName</param-name>
            <param-value>sessionFactoryIndex</param-value>
        </init-param>
    </filter>
    <!-- SpringSecurity filter-->
    <filter>
        <filter-name>springSecurityFilterChain</filter-name>
        <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
    </filter>

    <!-- Struts2 filter -->
    <filter>
        <filter-name>struts2Filter</filter-name>
        <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</fi
>
    </filter>

    <filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>


    <filter-mapping>
        <filter-name>springSecurityFilterChain</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <filter-mapping>
        <filter-name>hibernateOpenSessionInViewFilterContent</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <filter-mapping>
        <filter-name>hibernateOpenSessionInViewFilterIndex</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <filter-mapping>
        <filter-name>struts2Filter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <!--Spring的ApplicationContext 载入 -->
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class
>
    </listener>

    <!-- Spring 刷新Introspector防止内存泄露 -->
    <listener>
        <listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-clas
>
    </listener>

```

```
    <!-- session超时定义,单位为分钟 -->
    <session-config>
        <session-timeout>20</session-timeout>
    </session-config>

    <!-- 出错页面定义 -->
    <error-page>
        <exception-type>java.lang.Throwable</exception-type>
        <location>/common/500.jsp</location>
    </error-page>
    <error-page>
        <error-code>500</error-code>
        <location>/common/500.jsp</location>
    </error-page>
    <error-page>
        <error-code>404</error-code>
        <location>/common/404.jsp</location>
    </error-page>
    <error-page>
        <error-code>403</error-code>
        <location>/common/403.jsp</location>
    </error-page>
</web-app>
```

经过上面的分析，发现使用多个TransactionManager是不可行的（这个时候我在想，也许不使用Annotation就可以使用多个 TransactionMananger吧，毕竟Spring的AOP应该是可以把不同的TransactionManager插入到不同的类和方法中， 但是谁愿意走回头路呢？毕竟都已经是@Transactional的年代了），虽然运行不会报错，但是只有一个TransactionManager的事 务能够正常提交。所以测试进入下一步：

第五步、使用JTATransactionManager

简单地修改配置文件，使用JTATransactionManager做为事务管理器，配置文件我就不列出来了，运行，结果抱错，错误信息如下：
org.springframework.beans.factory.BeanCreationException: Error creating bean with name '_filterChainProxy': Initialization of bean failed; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name '_filterChainList': Cannot create inner bean '(inner bean)' of type [org.springframework.security.config.OrderedFilterBeanDefinitionDecorator$OrderedFilterDecorator] while setting bean property 'filters' with key [10]; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name '(inner bean)': Cannot resolve reference to bean 'filterSecurityInterceptor' while setting constructor argument; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'filterSecurityInterceptor' defined in file [D:Temp1-PureTextWEB-INFclassesapplicationContext-security.xml]: Cannot resolve reference to bean 'databaseDefinitionSource' while setting bean property 'objectDefinitionSource'; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'databaseDefinitionSource': FactoryBean threw exception on object creation; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'org.springframework.transaction.interceptor.TransactionInterceptor#0': Cannot resolve reference to bean 'transactionManager' while setting bean property 'transactionManager'; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'transactionManager' defined in file [D:Temp1-PureTextWEB-INFclassesapplicationContext.xml]: Invocation of init method failed; nested exception is java.lang.IllegalStateException: No JTA UserTransaction available - specify either 'userTransaction' or 'userTransactionName' or 'transactionManager' or

'transactionManagerName'

通过分析，发现其中最关键的一句是No JTA UserTransaction available，看来，我们只能进入到第六步，使用GlassFish了。

第六步、将项目部署到GlassFish中

将项目简单地部署到GlassFish中之后，项目可以成功运行，没有报错，说明JTA UserTransaction问题解决了，但是检查数据库却发现依然没有数据，看来JTATransactionManager不仅要和应用服务器配合 使用，还要和JNDI数据源一起使用。将数据源的配置修改为JNDI后，问题解决。下面是我的配置文件：

```
01. < xml version="1.0" encoding="UTF-8" >
02. <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-
instance" xmlns:jee="http://www.springframework.org/schema/jee" xmlns:tx="http://www.springfram
-lazy-
init="true" xsi:schemalocation=
"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring
beans-
2.5.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/sprin
tx-
2.5.xsd http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spr
jee-
2.5.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/con
context-2.5.xsd">
03.
04.     <description>Spring公共配置文件 </description>
05.
06.     <!-- 定义受环境影响易变的变量 -->
07.     <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
08.         <property name="systemPropertiesModeName" value="SYSTEM_PROPERTIES_MODE_OVERRIDE">
09.         <property name="ignoreResourceNotFound" value="true">
10.         <property name="locations">
11.             <list>
12.                 <!-- 标准配置 -->
13.                 <value>classpath*:/application.properties</value>
14.                 <!-- 本地开发环境配置 -->
15.                 <value>classpath*:/application.local.properties</value>
16.                 <!-- 服务器生产环境配置 -->
17.                 <!-->file:/var/myapp/application.server.properties -->
18.             </!--></list>
19.         </property>
20.     </property></property></bean>
21.
22.     <!-- 使用annotation 自动注册bean,并保证@Required,@Autowired的属性被注入 -->
23.     <context:component-scan base-package="cn.puretext">
24.
25.     <!-- 数据源配置,使用应用服务器的数据库连接池 -->
26.     <jee:jndi-lookup id="dataSourceContent" jndi-name="jdbc/dataSourceContent">
27.     <jee:jndi-lookup id="dataSourceIndex" jndi-name="jdbc/dataSourceIndex">
28.
29.     <!-- Hibernate配置 -->
30.     <bean id="sessionFactoryContent" class="org.springframework.orm.hibernate3.annotation.Annot
>
31.         <property name="dataSource" ref="dataSourceContent">
32.         <property name="namingStrategy">
33.             <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
34.         </bean></property>
35.         <property name="hibernateProperties">
36.             <props>
```

```
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQL5InnoDBDialect</prop>
                <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
                <prop key="hibernate.format_sql">${hibernate.format_sql}</prop>
                <prop key="hibernate.cache.provider_class">org.hibernate.cache.EhCacheProvider
                </prop>
                <prop key="hibernate.cache.provider_configuration_file_resource_path">${hibernate.ehcache_config_file}</prop>
            </props>
        </property>
        <property name="packagesToScan" value="cn.puretext.entity.*">
    </property></property></bean>
    <bean id="sessionFactoryIndex" class="org.springframework.orm.hibernate3.annotation.Annotat>
        <property name="dataSource" ref="dataSourceIndex">
        <property name="namingStrategy">
            <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
        </bean></property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQL5InnoDBDialect</prop>
                <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
                <prop key="hibernate.format_sql">${hibernate.format_sql}</prop>
                <prop key="hibernate.cache.provider_class">org.hibernate.cache.EhCacheProvider
                </prop>
                <prop key="hibernate.cache.provider_configuration_file_resource_path">${hibernate.ehcache_config_file}</prop>
            </props>
        </property>
        <property name="packagesToScan" value="cn.puretext.entity.*">
    </property></property></bean>

    <!-- 事务管理器配置,单数据源事务 -->
    <!--
 id="transactionManagerContent" class="org.springframework.orm.hibernate3.HibernateTransactionM>
        <property name="sessionFactory" ref="sessionFactoryContent">

    <bean id="transactionManagerIndex" class="org.springframework.orm.hibernate3.HibernateTrans>
        <property name="sessionFactory" ref="sessionFactoryIndex">
    </property></bean>
    -->

    <!-- 事务管理器配置,多数据源JTA事务-->

    <bean id="transactionManager" class="org.springframework.transaction.jta.JtaTransactionMana>


    <!-- 使用annotation定义事务 -->
    <tx:annotation-driven transaction-manager="transactionManager">

</tx:annotation-driven></bean></property></!--></jee:jndi-lookup></jee:jndi-lookup></context:component-scan></beans>
```

最后，我得出的结论是：要想使用多个数据库，就必须使用JTATransactionMananger，必须使用GlassFish等应用服务器而不是Tomcat，必须使用JNDI来管理dataSource。

如果一定要使用Tomcat呢？

这确实是一个难题，但是并不代表着没有解决办法。经过广泛的Google一番之后，终于发现了一个好东东，那就是JOTM，它的全称就是Java Open Transaction Mananger，它的作用就是可以单独提供JTA事务管理的功能，不需要应用服务器。JOTM的使用方法有两种，一种就是把它配置到项目中，和 Spring结合起来使用，另外一种就是把它配置到Tomcat中，这时，Tomcat摇身一变就成了和GlassFish一样的能够提供JTA功能的服 务器了。

JOTM的官方网站为http://jotm.ow2.org，这是它的新网站，旧网站为http://jotm.objectweb.org。

我选择了把JOTM 2.0.11整合到Tomcat中的方法进行了测试，结果发现还是不能够正常运行，我使用的是JOTM2.0.11，Tomcat 6.0.20，JKD 6 Update10。看来还得继续折腾下去了。

另外一个开源的JTA事务管理器是Atomikos，它供了事务管理和连接池，不需要应用服务器支持，其官方网站为http://www.atomikos.com/。有兴趣的朋友可以试试。

上一篇：Weblogic 10完美破解（Linux和Windows都适用）
下一篇：eclipse下svn的分支与合并操作
分享到：

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场