虫二的专栏~~在路上~~~ 河流之所以能够到达目的地,是因为它懂得怎样避开障碍。

■ 目录视图

₩ 摘要视图



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

文章搜索

Q

文章分类

FLEX(3)

Google Android(4)

Delphi(33)

Java(103)

JavaScript(26)

Liferay Portal(5)

Linux(22)

MySQL(5)

Oracle(12)

Web/JS(7)

Windows(5)

Windows技术(1)

ARM专题(5)

乱七八糟(14)

服务器搭建(26)

未分类(1)

系统分析师(3)

管理(2)

版本控制器(0)

SOLR(1)

♀ CSDN&ITeye 招贤榜

新版下载频道用户权限及积分规则详解

bShare分享,迅速提升10倍流量

♥ Spring 多数据源事务配置问题

分类: Java

2011-07-30 15:39 9 102人阅读 平论(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:

01. create database puretext_index;
02.
03. use puretext_index;
04.
05. create table articles(
06. id int primary key auto_increment,
07. subject varchar(256),
08. 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...

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

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?

```
09. );
```

第一步测试,配置多个DataSource,配置文件如下:

application.properties:

02.

01. 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:

```
01.
             < xml version="1.0" encoding="UTF-8" >
             <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi=</pre>
02.
             "http://www.w3.org/2001/XMLSchema-
             instance" xmlns:jee="http://www.springframework.org/schema/jee" xmlns:tx="http://www.springfram
             -lazv-
            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/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.org/schema/tx/springframework.o
            tx-
            2.5.xsd http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spr
            2.5.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context
            context-2.5.xsd">
03.
94.
                     <description>Spring公共配置文件 </description>
05.
                     <!-- 定义受环境影响易变的变量 -->
06.
07.
                     <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
                             cproperty name="systemPropertiesModeName" value="SYSTEM_PROPERTIES_MODE_OVERRIDE">
98.
                             cproperty name="ignoreResourceNotFound" value="true">
99
10.
                             cproperty name="locations">
                                      t>
11.
                                              <!-- 标准配置 -->
12.
13.
                                              <value>classpath*:/application.properties</value>
                                              <!-- 本地开发环境配置 -->
14.
                                              <value>classpath*:/application.local.properties</value>
15
                                              <!-- 服务器生产环境配置 -->
16.
17.
                                              <!-->file:/var/myapp/application.server.properties -->
                                     </!--></list>
18.
19.
                             </property>
20.
                     </property></property></bean>
21.
                     <!-- 使用annotation 自动注册bean,并保证@Required,@Autowired的属性被注入 -->
22.
23.
                     <context:component-scan base-package="cn.puretext">
24.
                     <!-- 数据源配置,使用应用内的DBCP数据库连接池 -->
25.
                     <bean id="dataSourceContent" class="org.apache.commons.dbcp.BasicDataSource" destroy-</pre>
26.
            method="close">
27.
                             <!-- Connection Info -->
                             cproperty name="driverClassName" value="com.mysql.jdbc.Driver">
28.
29.
                             cproperty name="url" value="${jdbc.urlContent}">
30.
                             cproperty name="username" value="${jdbc.username}">
31.
                             cproperty name="password" value="${jdbc.password}">
32.
33.
                             <!-- Connection Pooling Info -->
34.
                             cproperty name="initialSize" value="5">
35.
                             cproperty name="maxActive" value="100">
```

我...
WTK把所有工程都放在WTK2...
??哪里哪里?发上来看看;)
厉害,这都被你发现了。-_# ...

ARM学习资源

21IC电子工程师社区-ARM

电子产品世界-论坛首页

小白杨的Blog (RSS)

H-JTAG(电子产品世界)-ARM开 发总论坛斑竹的

robertchai的笔记

J2ME

搭建OTA下载服务器

我的好友

土豆的 (RSS)

花开的地方

文凭's BLog(准备考系分)

学习资源

使用 Eclipse 平台共享代码

Ehlib 的使用

eXtremComponents (RSS)

掌控上传进度的AJAX Upload

一个搞java的,技术文章也不错 (RSS)

Google Maps API 中文使用说明 文档

友情链节

老顽童-程序员考试 河南雅思科技有限公司 联想一键恢复高手

```
cproperty name="maxIdle" value="30">
37.
                    cproperty name="maxWait" value="1000">
                    cproperty name="poolPreparedStatements" value="true">
38.
                    cproperty name="defaultAutoCommit" value="false">
39.
40.
               </property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></pro>
41.
               <bean id="dataSourceIndex" class="org.apache.commons.dbcp.BasicDataSource" destroy-</pre>
         method="close">
42.
                    <!-- Connection Info -->
43.
                    cproperty name="driverClassName" value="com.mysql.jdbc.Driver">
44.
                    cproperty name="url" value="${jdbc.urlIndex}">
                    cproperty name="username" value="${jdbc.username}">
45.
                    cproperty name="password" value="${jdbc.password}">
46.
47.
48.
                    <!-- Connection Pooling Info -->
49.
                    cproperty name="initialSize" value="5">
50.
                    cproperty name="maxActive" value="100">
51.
                    cproperty name="maxIdle" value="30">
                    cproperty name="maxWait" value="1000">
52.
53.
                    cproperty name="poolPreparedStatements" value="true">
                    cproperty name="defaultAutoCommit" value="false">
54.
55.
               </property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></pro>
56.
               <!-- 数据源配置,使用应用服务器的数据库连接池 -->
57.
               <!--<jee:jndi-lookup id="dataSource" jndi-name="java:comp/env/jdbc/ExampleDB">-->
58.
59.
60.
              <!-- Hibernate配置 -->
61.
               <bean id="sessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSe</pre>
62.
                    cproperty name="dataSource" ref="dataSourceContent">
63.
                    cproperty name="namingStrategy">
64.
                          <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
65.
                    </bean></property>
66.
                    cproperty name="hibernateProperties">
67.
                          cprops>
                                 68.
         >org.hibernate.dialect.MySQL5InnoDBDialect</prop>
69.
                                <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
70.
                                 key="hibernate.format_sql">${hibernate.format_sql}
71.
                                cprop key="hibernate.cache.provider_class"
         >org.hibernate.cache.EhCacheProvider
72.
                                 key="hibernate.cache.provider configuration file resource path"
73.
         >${hibernate.ehcache_config_file}</prop>
74.
                          </props>
75.
                    cproperty name="packagesToScan" value="cn.puretext.entity.*">
76.
77.
               </property></property></bean>
78.
              <!-- 事务管理器配置,单数据源事务 -->
79.
               \verb|\class="org.springframework.orm.hibernate3.HibernateTransactions| \\
80.
         >
81.
                     cproperty name="sessionFactory" ref="sessionFactory">
82.
              </property></bean>
83.
              <!-- 事务管理器配置,多数据源JTA事务-->
84.
85.
              <!--
          86.
                    WebLogicJtaTransactionManager">
87.
              -->
88.
               <!-- 使用annotation定义事务 -->
89.
              <tx:annotation-driven transaction-manager="transactionManager">
90.
         </tx:annotation-driven></!--></!--<jee:jndi-lookup></context:component-scan></beans>
91.
```

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.Dataested 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.
      public class DaoTest extends SpringTxTestCase {
16.
17.
          @Autowired
18.
          private ArticleDao articleDao;
19.
20.
           public ArticleDao getArticleDao() {
               return articleDao;
21.
22.
23.
24.
           public void setArticleDao(ArticleDao articleDao) {
               this.articleDao = articleDao;
25.
26.
27.
28.
          @Override
           @Resource(name = "dataSourceContent")
29.
30.
          public void setDataSource(DataSource dataSource) {
31.
               // TODO Auto-generated method stub
32.
               super.setDataSource(dataSource);
33.
          }
34.
35.
           @Test
          public void addArticle() {
36.
               Article article = new Article();
37.
              article.setSubject("article test");
38.
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);
```

```
49. List<article> articles = page.getResult();
50. }
51. }
52. 
53. </article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></article></a
```

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:

```
01.
      < xml version="1.0" encoding="UTF-8" >
      <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi=</pre>
      "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
      2.5.xsd http://www.springframework.org/schema/tx/http://www.springframework.org/schema/tx/spring
      2.5.xsd http://www.springframework.org/schema/jee/spr
      2.5.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context
      context-2.5.xsd">
03.
          <description>Spring公共配置文件 </description>
04.
05.
          <!-- 定义受环境影响易变的变量 -->
06.
          <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
97.
              cproperty name="systemPropertiesModeName" value="SYSTEM_PROPERTIES_MODE_OVERRIDE">
08.
09.
              cproperty name="ignoreResourceNotFound" value="true">
              cproperty name="locations">
10.
11.
                  t>
12.
                      <!-- 标准配置 -->
                       <value>classpath*:/application.properties</value>
13.
                      <!-- 本地开发环境配置 -->
14.
15.
                       <value>classpath*:/application.local.properties</value>
                      <!-- 服务器生产环境配置 -->
16.
                       <!-->file:/var/myapp/application.server.properties -->
17.
18.
                  </!--></list>
19.
              20.
          </property></property></bean>
21.
          <!-- 使用annotation 自动注册bean,并保证@Required,@Autowired的属性被注入 -->
22.
          <context:component-scan base-package="cn.puretext">
23.
24.
          <!-- 数据源配置,使用应用内的DBCP数据库连接池 -->
25.
          <bean id="dataSourceContent" class="org.apache.commons.dbcp.BasicDataSource" destroy-</pre>
26.
      method="close">
27.
              <!-- Connection Info -->
              cproperty name="driverClassName" value="com.mysql.jdbc.Driver">
28.
29.
              cproperty name="url" value="${jdbc.urlContent}">
30.
              cproperty name="username" value="${jdbc.username}">
31.
              cproperty name="password" value="${jdbc.password}">
32.
              <!-- Connection Pooling Info -->
33.
34.
              cproperty name="initialSize" value="5">
35.
              cproperty name="maxActive" value="100">
              cproperty name="maxIdle" value="30">
36.
37.
              cproperty name="maxWait" value="1000">
```

```
38.
                    cproperty name="poolPreparedStatements" value="true">
                     cproperty name="defaultAutoCommit" value="false">
               </property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></pro>
40.
41.
               <bean id="dataSourceIndex" class="org.apache.commons.dbcp.BasicDataSource" destroy-</pre>
         method="close">
42.
                    <!-- Connection Info -->
43.
                     cproperty name="driverClassName" value="com.mysql.jdbc.Driver">
44.
                    cproperty name="url" value="${jdbc.urlIndex}">
45.
                     cproperty name="username" value="${jdbc.username}">
                    cproperty name="password" value="${jdbc.password}">
46.
47.
48.
                    <!-- Connection Pooling Info -->
                     cproperty name="initialSize" value="5">
49.
                     cproperty name="maxActive" value="100">
50.
                     cproperty name="maxIdle" value="30">
51.
                     cproperty name="maxWait" value="1000">
52.
53.
                     cproperty name="poolPreparedStatements" value="true">
54.
                     cproperty name="defaultAutoCommit" value="false">
55.
               </property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></pro>
56.
               <!-- 数据源配置,使用应用服务器的数据库连接池 -->
57.
               <!--<jee:jndi-lookup id="dataSource" jndi-name="java:comp/env/jdbc/ExampleDB">-->
58.
59.
60.
               <!-- Hibernate配置 -->
               <bean id="sessionFactoryContent" class="org.springframework.orm.hibernate3.annotation.Annot</pre>
61.
                     cproperty name="dataSource" ref="dataSourceContent">
62.
63.
                     cproperty name="namingStrategy">
64.
                          <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
65.
66.
                     roperty name="hibernateProperties">
67.
                          ops>
                                  key="hibernate.dialect"
68.
         >org.hibernate.dialect.MySQL5InnoDBDialect</prop>
69.
                                 key="hibernate.show_sql">${hibernate.show_sql}
                                <prop key="hibernate.format_sql">${hibernate.format_sql}</prop>
70.
71.
                                    key="hibernate.cache.provider_class"
         >org.hibernate.cache.EhCacheProvider
72.
                                </prop>
73.
                                key="hibernate.cache.provider_configuration_file_resource_path"
         >${hibernate.ehcache_config_file}</prop>
74.
                          </props>
75.
                     </property>
                     cproperty name="packagesToScan" value="cn.puretext.entity.*">
76.
77.
               </property></property></bean>
78.
               <bean id="sessionFactoryIndex" class="org.springframework.orm.hibernate3.annotation.Annotat</pre>
79.
                     cproperty name="dataSource" ref="dataSourceIndex">
80.
                     cproperty name="namingStrategy">
81.
                          <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
82.
                     </bean></property>
                     cproperty name="hibernateProperties">
83.
84
                          <props>
85.
                                cprop key="hibernate.dialect"
         >org.hibernate.dialect.MySQL5InnoDBDialect</prop>
86.
                                 key="hibernate.show_sql">${hibernate.show_sql}
87.
                                   key="hibernate.format_sql">${hibernate.format_sql}
88.
                                cprop key="hibernate.cache.provider_class"
         >org.hibernate.cache.EhCacheProvider
89.
90.
                                key="hibernate.cache.provider_configuration_file_resource_path"
         >${hibernate.ehcache_config_file}</prop>
91.
                          </props>
92.
93.
                     cproperty name="packagesToScan" value="cn.puretext.entity.*">
94.
               </property></property></bean>
95.
```

```
<!-- 事务管理器配置,单数据源事务 -->
 97.
           <bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactio</pre>
               cproperty name="sessionFactory" ref="sessionFactoryContent">
 98.
 99.
           </property></bean>
100.
           <!-- 事务管理器配置,多数据源JTA事务-->
101.
102.
           <1--
        id="transactionManager" class="org.springframework.transaction.jta.JtaTransactionManager or
103.
               WebLogicJtaTransactionManager">
104.
105.
106.
           <!-- 使用annotation定义事务 -->
107.
            <tx:annotation-driven transaction-manager="transactionManager">
108.
       </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

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;
01.
02.
03.
04.
      import javax.annotation.Resource;
05.
      import org.hibernate.SessionFactory;
06.
07.
      import org.springframework.stereotype.Repository;
      import org.springside.modules.orm.hibernate.HibernateDao;
08.
09.
10.
      import cn.puretext.entity.web.Article;
11.
12.
      @Repository
13.
      public class ArticleDao extends HibernateDao<article, long=""> {
14.
15.
          @Override
       @Resource(name = "sessionFactoryContent")
16.
17.
          public void setSessionFactory(SessionFactory sessionFactory) {
18.
             // TODO Auto-generated method stub
19.
               super.setSessionFactory(sessionFactory);
20.
21.
22.
      }
      </article,>
23.
```

运行单元测试,成功。

第三步、配置多个TransactionManager, 如下:

```
01.
        < xml version="1.0" encoding="UTF-8" >
        <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi=</pre>
        "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"
        2.5.xsd http://www.springframework.org/schema/tx/http://www.springframework.org/schema/tx/spring
        2.5.xsd http://www.springframework.org/schema/jee/spr
        2.5.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context
        context-2.5.xsd">
03.
             <description>Spring公共配置文件 </description>
94.
05.
             <!-- 定义受环境影响易变的变量 -->
06.
              <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
07.
                  cproperty name="systemPropertiesModeName" value="SYSTEM_PROPERTIES_MODE_OVERRIDE">
08.
                  cproperty name="ignoreResourceNotFound" value="true">
99.
10.
                  cproperty name="locations">
11.
                        t>
                             <!-- 标准配置 -->
12.
                             <value>classpath*:/application.properties</value>
13.
14.
                             <!-- 本地开发环境配置 -->
15.
                             <value>classpath*:/application.local.properties</value>
                             <!-- 服务器生产环境配置 -->
16.
17.
                             <!-->file:/var/myapp/application.server.properties -->
18.
                        </!--></list>
19.
                  </property>
20.
             </property></property></bean>
21.
             <!-- 使用annotation 自动注册bean,并保证@Required,@Autowired的属性被注入 -->
22.
             <context:component-scan base-package="cn.puretext">
23.
24.
             <!-- 数据源配置,使用应用内的DBCP数据库连接池 -->
25.
26.
             <bean id="dataSourceContent" class="org.apache.commons.dbcp.BasicDataSource" destroy-</pre>
        method="close">
                  <!-- Connection Info -->
27.
                  cproperty name="driverClassName" value="com.mysql.jdbc.Driver">
28.
29.
                  cproperty name="url" value="${jdbc.urlContent}">
30.
                  cproperty name="username" value="${jdbc.username}">
31.
                  cproperty name="password" value="${jdbc.password}">
32.
                  <!-- Connection Pooling Info -->
33.
34.
                  cproperty name="initialSize" value="5">
35.
                  cproperty name="maxActive" value="100">
                  cproperty name="maxIdle" value="30">
36.
37.
                  cproperty name="maxWait" value="1000">
38.
                  cproperty name="poolPreparedStatements" value="true">
39.
                  cproperty name="defaultAutoCommit" value="false">
40.
             </property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></property></pro>
41.
             <bean id="dataSourceIndex" class="org.apache.commons.dbcp.BasicDataSource" destroy-</pre>
        method="close">
42.
                 <!-- Connection Info -->
43.
                  cproperty name="driverClassName" value="com.mysql.jdbc.Driver">
44.
                  cproperty name="url" value="${jdbc.urlIndex}">
```

```
cproperty name="username" value="${jdbc.username}">
46.
                cproperty name="password" value="${jdbc.password}">
47.
48.
               <!-- Connection Pooling Info -->
49.
                cproperty name="initialSize" value="5">
50.
                cproperty name="maxActive" value="100">
51.
                cproperty name="maxIdle" value="30">
52.
                cproperty name="maxWait" value="1000">
53.
                cproperty name="poolPreparedStatements" value="true">
 54.
                cproperty name="defaultAutoCommit" value="false">
 55.
            </property></property></property></property></property></property></property></property></property></property></property></property></property></property></pro>
56.
            <!-- 数据源配置,使用应用服务器的数据库连接池 -->
57.
58.
           <!--<jee:jndi-lookup id="dataSource" jndi-name="java:comp/env/jdbc/ExampleDB">-->
59.
           <!-- Hibernate配置 -->
60.
            <bean id="sessionFactoryContent" class="org.springframework.orm.hibernate3.annotation.Annot</pre>
61.
62.
                cproperty name="dataSource" ref="dataSourceContent">
63.
                cproperty name="namingStrategy">
64.
                    <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
65.
66.
                cproperty name="hibernateProperties">
67.
                    ops>
                         key="hibernate.dialect"
68.
       >org.hibernate.dialect.MySQL5InnoDBDialect</prop>
69.
                         key="hibernate.show_sql">${hibernate.show_sql}
70.
                         key="hibernate.format_sql">${hibernate.format_sql}
71.
                          key="hibernate.cache.provider_class"
       >org.hibernate.cache.EhCacheProvider
72.
                        </prop>
 73.
                        key="hibernate.cache.provider_configuration_file_resource_path"
       >${hibernate.ehcache_config_file}</prop>
74.
                    </props>
75.
                </property>
                cproperty name="packagesToScan" value="cn.puretext.entity.*">
76.
77.
            </property></property></bean>
78.
           <bean id="sessionFactoryIndex" class="org.springframework.orm.hibernate3.annotation.Annotat</pre>
 79.
                cproperty name="dataSource" ref="dataSourceIndex">
80.
                cproperty name="namingStrategy">
                    <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
81.
82.
                cproperty name="hibernateProperties">
83.
                    ops>
84.
85.
                         >org.hibernate.dialect.MySQL5InnoDBDialect</prop>
86.
                         key="hibernate.show_sql">${hibernate.show_sql}
87.
                         key="hibernate.format_sql">${hibernate.format_sql}
88.
                         prop key="hibernate.cache.provider class"
       >org.hibernate.cache.EhCacheProvider
89.
90.
                        key="hibernate.cache.provider_configuration_file_resource_path"
       >${hibernate.ehcache_config_file}</prop>
91.
                    </props>
92.
                </property>
93.
                cproperty name="packagesToScan" value="cn.puretext.entity.*">
94.
           </property></property></bean>
95.
96.
           <!-- 事务管理器配置,单数据源事务 -->
97.
            <bean id="transactionManagerContent" class="org.springframework.orm.hibernate3.HibernateTra</pre>
98.
                cproperty name="sessionFactory" ref="sessionFactoryContent">
99.
            </property></bean>
100.
           <bean id="transactionManagerIndex" class="org.springframework.orm.hibernate3.HibernateTrans</pre>
       >
101.
                cproperty name="sessionFactory" ref="sessionFactoryIndex">
```

```
102.
                        </property></bean>
103.
                        <!-- 事务管理器配置,多数据源JTA事务-->
104.
105.
                              <!--
                      \verb|id="transactionManager"| class="org.springframework.transaction.jta.JtaTransactionManager| org.springframework.transaction.jta.JtaTransactionManager| org.springframework.transactionManager| org.springframework| org.springframework| org.springframework| org.springframework| org.springframework| org.springframework| org.springframework| o
                                        WebLogicJtaTransactionManager">
106.
107.
108.
109.
                              <!-- 使用annotation定义事务 -->
110.
                              <tx:annotation-driven transaction-manager="transactionManagerContent">
111.
                    </tx:annotation-driven></!--></!--<jee:jndi-lookup></context:component-scan></beans>
112.
                    这个时候运行还是会出错, 出错的原因
113.
                    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 -->
114.
                              <bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactio"</pre>
115.
116.
                                        cproperty name="sessionFactory" ref="sessionFactoryContent">
117.
                              </property></bean>
118.
                              <bean id="transactionManagerIndex" class="org.springframework.orm.hibernate3.HibernateTrans</pre>
119.
                                         cproperty name="sessionFactory" ref="sessionFactoryIndex">
120.
                              </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:

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:

```
01.
      package cn.puretext.entity.web;
02.
03.
      import javax.persistence.Column;
04.
      import javax.persistence.Entity;
05.
      import javax.persistence.Table;
06.
      import org.hibernate.annotations.Cache;
07.
08.
      import org.hibernate.annotations.CacheConcurrencyStrategy;
09.
10.
      import cn.puretext.entity.IdEntity;
11.
12.
      @Entity
      // 表名与类名不相同时重新定义表名.
13.
```

```
@Table(name = "articles")
      // 默认的缓存策略.
      @Cache(usage = CacheConcurrencyStrategy.READ_WRITE)
16.
      public class ArticleIndex extends IdEntity {
17.
18.
          private String subject;
19.
          private String webServer;
20.
21.
          public String getSubject() {
22.
               return subject;
23.
24.
25.
           public void setSubject(String subject) {
26.
               this.subject = subject;
27.
28.
          @Column(name = "webserver")
29.
           public String getWebServer() {
30.
               return webServer;
31.
32.
           public void setWebServer(String webServer) {
33.
               this.webServer = webServer;
34.
35.
           }
36.
```

```
01.
      package cn.puretext.dao;
02.
03.
      import javax.annotation.Resource;
04.
05.
      import org.hibernate.SessionFactory;
06.
      import org.springframework.stereotype.Repository;
07.
      import org.springside.modules.orm.hibernate.HibernateDao;
08.
09.
      import cn.puretext.entity.web.ArticleIndex;
10.
11.
      @Repository
12.
      public class ArticleIndexDao extends HibernateDao<articleindex, long=""> {
13.
14.
         @Resource(name = "sessionFactoryIndex")
15.
           public void setSessionFactory(SessionFactory sessionFactory) {
               // TODO Auto-generated method stub
16.
17.
               super.setSessionFactory(sessionFactory);
18.
          }
19.
      }</articleindex,>
```

```
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.springframework.transaction.annotation.Transactional;
11.
       import org.springside.modules.orm.Page;
12.
      import org.springside.modules.test.junit4.SpringTxTestCase;
13.
14.
      import cn.puretext.dao.ArticleDao;
```

```
import cn.puretext.dao.ArticleIndexDao;
16.
      import cn.puretext.entity.web.Article;
      import cn.puretext.entity.web.ArticleIndex;
17.
18.
      import cn.puretext.service.ServiceException;
19.
20.
      public class DaoTest extends SpringTxTestCase {
21.
           @Autowired
22.
           private ArticleDao articleDao;
23.
           @Autowired
24.
           private ArticleIndexDao articleIndexDao;
25.
           public void setArticleIndexDao(ArticleIndexDao articleIndexDao) {
26.
27.
               this.articleIndexDao = articleIndexDao;
28.
29.
30.
           public void setArticleDao(ArticleDao articleDao) {
31.
               this.articleDao = articleDao;
32.
33.
34.
           @Override
           @Resource(name = "dataSourceContent")
35.
           public void setDataSource(DataSource dataSource) {
36.
37.
               // TODO Auto-generated method stub
38.
               super.setDataSource(dataSource);
39.
           }
40.
41.
           @Test
42.
           @Transactional
43.
           public void addArticle() {
               Article article = new Article();
44.
               article.setSubject("article test");
45.
46.
               article.setContent("article test");
47.
               articleDao.save(article);
48.
49.
50.
           @Test
51.
           @Transactional
52.
           public void pageQuery() {
53.
               Page<article> page = new Page<article>();
54.
               page.setPageSize(10);
55.
               page.setPageNo(2);
56.
               page = articleDao.getAll(page);
57.
               List<article> articles = page.getResult();
58.
59.
60.
           @Test
61.
           @Transactional
62.
           public void addIndex() {
               ArticleIndex articleIndex = new ArticleIndex();
63.
64.
               articleIndex.setSubject("test");
65.
               articleIndex.setWebServer("www001");
               articleIndexDao.save(articleIndex);
66.
67.
           }
68.
           @Test
69.
70.
           @Transactional
           public void addArticleAndAddIndex() {
71.
72.
               addArticle();
73.
               addIndex();
74.
               throw new ServiceException("测试事务回滚");
75.
           }
76.
      }
77.
      </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TransactionManager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:

```
< xml version="1.0" encoding="UTF-8" >
01.
       <web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-</pre>
02.
      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.
94.
          <display-name>PureText</display-name>
           <!-- Spring ApplicationContext配置文件的路径,可使用通配符,多个路径用,号分隔
05.
               此参数用于后面的Spring Context Loader -->
06.
07.
           <context-param>
               <param-name>contextConfigLocation</param-name>
08.
               <param-value>classpath*:/applicationContext*.xml</param-value>
99.
10.
           </context-param>
11.
          <!-- Character Encoding filter -->
12.
13.
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>
                   <param-value>UTF-8</param-value>
18.
19.
               </init-param>
20.
               <init-param>
21.
                   <param-name>forceEncoding</param-name>
22.
                   <param-value>true</param-value>
23.
               </init-param>
24.
          </filter>
25.
           <filter>
26.
27.
               <filter-name>hibernateOpenSessionInViewFilterContent</filter-name>
28.
               <filter-class>org.springside.modules.orm.hibernate.OpenSessionInViewFilter</filter-clas</pre>
29.
               <init-param>
30.
                   <param-name>excludeSuffixs</param-name>
31.
                   <param-value>js,css,jpg,gif</param-value>
```

```
</init-param>
33.
               <init-param>
34.
                      <param-name>sessionFactoryBeanName
35.
                   <param-value>sessionFactoryContent
               </init-param>
36.
37.
           </filter>
38.
           <filter>
39.
               <filter-name>hibernateOpenSessionInViewFilterIndex</filter-name>
40.
               <filter-class>org.springside.modules.orm.hibernate.OpenSessionInViewFilter</filter-class</pre>
41.
               <init-param>
                   <param-name>excludeSuffixs</param-name>
42.
43.
                   <param-value>js,css,jpg,gif</param-value>
               </init-param>
44.
45.
               <init-param>
46.
                       <param-name>sessionFactoryBeanName</param-name>
47.
                   <param-value>sessionFactoryIndex</param-value>
48.
               </init-param>
49.
           </filter>
          <!-- SpringSecurity filter-->
50.
           <filter>
51.
               <filter-name>springSecurityFilterChain</filter-name>
52.
53.
               <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
54.
           </filter>
55.
          <!-- Struts2 filter -->
56.
57.
58.
               <filter-name>struts2Filter</filter-name>
               <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</fi>
59.
           </filter>
60.
61.
62.
           <filter-mapping>
               <filter-name>encodingFilter</filter-name>
63.
64.
               <url-pattern>/*</url-pattern>
           </filter-mapping>
65.
66.
67.
68.
           <filter-mapping>
               <filter-name>springSecurityFilterChain</filter-name>
69.
70.
               <url-pattern>/*</url-pattern>
71.
           </filter-mapping>
           <filter-mapping>
72.
73.
               <filter-name>hibernateOpenSessionInViewFilterContent</filter-name>
74.
               <url-pattern>/*</url-pattern>
75.
           </filter-mapping>
76.
           <filter-mapping>
77.
               <filter-name>hibernateOpenSessionInViewFilterIndex</filter-name>
               <url-pattern>/*</url-pattern>
78.
79.
           </filter-mapping>
80.
           <filter-mapping>
81.
               <filter-name>struts2Filter</filter-name>
82.
               <url-pattern>/*</url-pattern>
83.
           </filter-mapping>
84.
           <!--Spring的ApplicationContext 载入 -->
85.
86.
           tener>
87.
               clistener-class>org.springframework.web.context.ContextLoaderListener/listener-class
88.
           </listener>
89.
90.
           <!-- Spring 刷新Introspector防止内存泄露 -->
91.
           tener
92.
               classorg.springframework.web.util.IntrospectorCleanupListener/listener-class
93.
           </listener>
94.
```

```
<!-- session超时定义,单位为分钟 -->
     <session-config>
 96.
              <session-timeout>20</session-timeout>
 97.
     </session-config>
 98.
99.
100.
     <!-- 出错页面定义 -->
101.
          <error-page>
102.
              <exception-type>java.lang.Throwable</exception-type>
103.
              <location>/common/500.jsp</location>
104.
       </error-page>
105.
          <error-page>
              <error-code>500</error-code>
106.
107.
              <location>/common/500.jsp</location>
108.
        </error-page>
109.
          <error-page>
110.
              <error-code>404
111.
              <location>/common/404.jsp</location>
112.
        </error-page>
113.
          <error-page>
114.
          <error-code>403</error-code>
115.
              <location>/common/403.jsp</location>
116.
          </error-page>
117. </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 TransactionManager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.lllegalStateException: 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:

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

```
cprop key="hibernate.dialect"
      >org.hibernate.dialect.MySQL5InnoDBDialect</prop>
38.
                        key="hibernate.show_sql">${hibernate.show_sql}
39.
                         key="hibernate.format_sql">${hibernate.format_sql}
40.
                       cprop key="hibernate.cache.provider_class"
      >org.hibernate.cache.EhCacheProvider
41.
                       </prop>
42.
                       key="hibernate.cache.provider_configuration_file_resource_path"
      >${hibernate.ehcache_config_file}</prop>
43.
                   </props>
44.
              </property>
45.
               cproperty name="packagesToScan" value="cn.puretext.entity.*">
46.
          </property></property></bean>
           <bean id="sessionFactoryIndex" class="org.springframework.orm.hibernate3.annotation.Annotat</pre>
47.
48.
               cproperty name="dataSource" ref="dataSourceIndex">
49.
               cproperty name="namingStrategy">
50.
                   <bean class="org.hibernate.cfg.ImprovedNamingStrategy">
51.
               </bean></property>
              cproperty name="hibernateProperties">
52.
53.
                   cprops>
                        key="hibernate.dialect"
54.
      >org.hibernate.dialect.MySQL5InnoDBDialect</prop>
55.
                        key="hibernate.show_sql">${hibernate.show_sql}
56.
                        key="hibernate.format_sql">${hibernate.format_sql}
                       cprop key="hibernate.cache.provider_class"
57.
      >org.hibernate.cache.EhCacheProvider
58.
                       </prop>
                       key="hibernate.cache.provider_configuration_file_resource_path"
59.
      >${hibernate.ehcache_config_file}</prop>
60.
                   </props>
61.
               </property>
               cproperty name="packagesToScan" value="cn.puretext.entity.*">
62.
63.
          </property></property></bean>
64.
65.
          <!-- 事务管理器配置,单数据源事务 -->
          <!--
66.
       id="transactionManagerContent" class="org.springframework.orm.hibernate3.HibernateTransactionM
67.
               cproperty name="sessionFactory" ref="sessionFactoryContent">
68.
69.
           <bean id="transactionManagerIndex" class="org.springframework.orm.hibernate3.HibernateTrans</pre>
70.
               cproperty name="sessionFactory" ref="sessionFactoryIndex">
71.
           </property></bean>
72.
73.
74.
          <!-- 事务管理器配置,多数据源JTA事务-->
75.
76.
          <bean id="transactionManager" class="org.springframework.transaction.jta.JtaTransactionMana</pre>
77.
78.
           <!-- 使用annotation定义事务 -->
79.
           <tx:annotation-driven transaction-manager="transactionManager">
80.
81.
82.
      </tx:annotation-driven></bean></property></!--></jee:jndi-lookup></jee:jndi-</pre>
      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

♦刻♦♦ 捆

华硕推出MeeGo超薄本,价格不高于200美元

推游戏开发者必看,你不看就out啦

推 轻轻松松打包MeeGo的QT程序

推 同属开源Linux 移动市场MeeGo独到之秘

推 Location&Map例子详解

Qt for Android 配置详细

♦剝 ♦ ♦♦ ♦

■ 【网易有道 北京】诚聘研发/前端/web/测试开发工程师!

間【爱立信上海】急聘C++/C软件技术支持工程师!

閱【上海科旭网络】诚招JAVA工程师/软件架构师/项目经

38 【安捷伦科技软件】诚聘软件开发工程师

3 【上海合驿网络】急聘软件开发工程师

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 北京创新乐知信息技术有限公司 版权所有, 京 ICP 证 070598 号 世纪乐知(北京)网络技术有限公司 提供技术支持 江苏乐知网络技术有限公司 提供商务支持

Email:webmaster@csdn.net

Copyright © 1999-2011, CSDN. NET, All Rights Reserved

