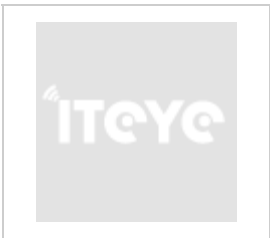


lujia35

永久域名 <http://lujia35.iteye.com>



lujia35

浏览: 1912 次

性别:

来自: 北京

我现在离线

[详细资料](#) [留言簿](#)

搜索本博客

最近访客 [>>更多访客](#)



[zxwu](#)



[swen00](#)



[yasaso](#)



[lzq3267373](#)

博客分类

- 全部博客 (4)

我的留言簿 [>>更多留言](#)

其他分类

- 我的收藏 (7)
- 我的代码 (0)
- 我的论坛主题帖 (0)
- 我的所有论坛帖 (0)
- 我的精华良好帖 (0)

最近加入群组

- [系统架构与架构应用](#)
- [Google App Engine](#)
- [网络爬虫](#)

存档

2011-03-21

[深入了解MySQL 5.5分区功能增强](#) ▶

[Spring+Hibernate框架下Mysql读写分离、主从数据库配置](#)

MySQL 框架 Spring Hibernate 配置管理

介绍下mysql数据库读写分离在spring,hibernate框架下的配置。

1.mysql连接配置文件jdbc.properties

master.*表示主数据库连接参数，负责增，删，改；

slave.*表示从数据库连接参数，只负责读取；

jdbc.properties

Java代码

```
1. master.jdbc.driverClassName=com.mysql.jdbc.Driver
2. master.jdbc.url=*****
3. master.jdbc.username=*****
4. master.jdbc.password=*****
5.
6. slave.jdbc.driverClassName=com.mysql.jdbc.Driver
7. slave.jdbc.url=*****
8. slave.jdbc.username=*****
9. slave.jdbc.password=*****
```

把**改成你所需的连接参数；

2.配置AOP切面类 DataSourceAdvice.java

Java代码

```
1. import java.lang.reflect.Method;
2. import org.springframework.aop.AfterReturningAdvice;
3. import org.springframework.aop.MethodBeforeAdvice;
4. import org.springframework.aop.ThrowsAdvice;
5.
6. import com.company.datasource.DataSourceSwitcher;
7.
8. public class DataSourceAdvice implements MethodBeforeAdvice, AfterReturningAdvice, ThrowsAdvice
9. {
10.     // service方法执行之前被调用
11.     public void before(Method method, Object[] args, Object target) throws Throwable {
12.         System.out.println("切入点: " + target.getClass().getName() + "类中" + method.getName() + "方法");
13.         if(method.getName().startsWith("add")
14.             || method.getName().startsWith("create")
15.             || method.getName().startsWith("save")
16.             || method.getName().startsWith("edit")
17.             || method.getName().startsWith("update")
18.             || method.getName().startsWith("delete")
19.             || method.getName().startsWith("remove")){
20.             System.out.println("切换到: master");
21.             DataSourceSwitcher.setMaster();
22.         }
23.         else {
24.             System.out.println("切换到: slave");
25.             DataSourceSwitcher.setSlave();
26.         }
27.     }
28. }
```

- [2011-03](#) (1)
- [2010-07](#) (2)
- [2010-04](#) (1)
- [更多存档...](#)

评论排行榜

- [Spring+Hibernate框架下Mysql读写分离、主...](#)



```
25.         }
26.     }
27.
28.     // service方法执行完之后被调用
29.     public void afterReturning(Object arg0, Method method, Object[] args, Object target) throws
30.     Throwable {
31.
32.     // 抛出Exception之后被调用
33.     public void afterThrowing(Method method, Object[] args, Object target, Exception ex) throws
34.     Throwable {
35.         DataSourceSwitcher.setSlave();
36.         System.out.println("出现异常,切换到: slave");
37.     }
38. }
```

数据源选择类 DataSourceSwitcher.java

Java代码 ☆

```
1. package com.company.datasources;
2. import org.springframework.util.Assert;
3.
4. public class DataSourceSwitcher {
5.     @SuppressWarnings("rawtypes")
6.     private static final ThreadLocal contextHolder = new ThreadLocal();
7.
8.     @SuppressWarnings("unchecked")
9.     public static void setDataSource(String dataSource) {
10.         Assert.notNull(dataSource, "dataSource cannot be null");
11.         contextHolder.set(dataSource);
12.     }
13.
14.     public static void setMaster(){
15.         clearDataSource();
16.     }
17.
18.     public static void setSlave() {
19.         setDataSource("slave");
20.     }
21.
22.     public static String getDataSource() {
23.         return (String) contextHolder.get();
24.     }
25.
26.     public static void clearDataSource() {
27.         contextHolder.remove();
28.     }
29. }
```

DynamicDataSource.java数据源动态切换类

Java代码 ☆

```
1. package com.company.datasources;
2.
3. import org.springframework.jdbc.datasource.lookup.AbstractRoutingDataSource;
4.
5. public class DynamicDataSource extends AbstractRoutingDataSource {
6.
7.     @Override
8.     protected Object determineCurrentLookupKey() {
```

```
9.         return DataSourceSwitcher.getDataSource();
10.     }
11.
12. }
```

下面配置spring applicationContext.xml文件

Xml代码



```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3.       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"
4.       xmlns:tx="http://www.springframework.org/schema/tx" xmlns:context="http://www.springframework.org/schema/context"
5.       xsi:schemaLocation="
6.           http://www.springframework.org/schema/beans
7.           http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
8.           http://www.springframework.org/schema/tx
9.           http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
10.          http://www.springframework.org/schema/aop
11.          http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
12.          http://www.springframework.org/schema/context
13.          http://www.springframework.org/schema/context/spring-context-3.0.xsd">
14.
15.     <context:annotation-config />
16.     <!-- 自动加载SERVICE DAO ACTION -->
17.     <context:component-scan base-package="cn.com.company.dao.*" />
18.     <context:component-scan base-package="cn.com.company.service.*" />
19.     <context:component-scan base-package="cn.com.company.action" />
20.
21.     <!-- 加载properties配置文件 -->
22.     <bean id="propertyConfigurer"
23.           class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
24.
25.         <property name="locations">
26.             <list>
27.                 <value>classpath:jdbc.properties</value>
28.             </list>
29.         </property>
30.     </bean>
31.
32.     <bean id="parentDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
33.         <!-- ***c3p0配置 -->
34.     </bean>
35.
36.     <!-- 主数据源-->
37.     <bean id="masterDataSource" parent="parentDataSource">
38.         <property name="driverClass" value="${master.jdbc.driverClassName}" />
39.         <property name="jdbcUrl" value="${master.jdbc.url}" />
40.         <property name="user" value="${master.jdbc.username}" />
41.         <property name="password" value="${master.jdbc.password}" />
42.     </bean>
43.
44.     <!-- 从数据源-->
45.     <bean id="slaveDataSource" parent="parentDataSource">
46.         <property name="driverClass" value="${slave.jdbc.driverClassName}" />
47.         <property name="jdbcUrl" value="${slave.jdbc.url}" />
48.         <property name="user" value="${slave.jdbc.username}" />
49.         <property name="password" value="${slave.jdbc.password}" />
50.     </bean>
51.
52.     <bean id="dataSource" class="com.company.datasources.DynamicDataSource">
53.         <property name="targetDataSources">
54.             <map key-type="java.lang.String">
```

```
52.         <entry key="slave" value-ref="slaveDataSource" />
53.     </map>
54. </property>
55. <property name="defaultTargetDataSource" ref="masterDataSource" />
56. </bean>
57.
58. <!-- 配置sessionFactory -->
59. <bean id="sessionFactory"
60.     class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"
61.     n">
62.     <property name="dataSource" ref="dataSource"></property>
63.     <property name="packagesToScan" value="cn.com.company.entity" />
64.     <property name="hibernateProperties">
65.         <props>
66.             <prop>/**hibernate一些参数这里不写了</prop>
67.         </props>
68.     </property>
69. </bean>
70.
71. <!-- 切换数据源 -->
72. <bean id="dataSourceAdvice" class="com.company.aop.DataSourceAdvice" />
73. <aop:config>
74.     <aop:advisor
75.         pointcut="execution(* cn.com.company.service.*Service.*(..))"
76.         advice-ref="dataSourceAdvice" />
77. </aop:config>
78.
79. <!-- 配置事务管理器 -->
80. <bean id="transactionManager"
81.     class="org.springframework.orm.hibernate3.HibernateTransactionManager">
82.     <property name="sessionFactory">
83.         <ref bean="sessionFactory" />
84.     </property>
85. </bean>
86. <!--配置事务的传播特性 -->
87. <tx:advice id="txAdvice" transaction-manager="transactionManager">
88.     <tx:attributes>
89.         <!-- 对增、删、改方法进行事务支持 -->
90.         <tx:method name="add*" propagation="REQUIRED" />
91.         <tx:method name="create*" propagation="REQUIRED" />
92.         <tx:method name="save*" propagation="REQUIRED" />
93.         <tx:method name="edit*" propagation="REQUIRED" />
94.         <tx:method name="update*" propagation="REQUIRED" />
95.         <tx:method name="delete*" propagation="REQUIRED" />
96.         <tx:method name="remove*" propagation="REQUIRED" />
97.         <!-- 对查找方法进行只读事务 -->
98.         <tx:method name="loadByUsername*" propagation="SUPPORTS" read-only="true" />
99.         <!-- 对其它方法进行只读事务 -->
100.        <tx:method name="*" propagation="SUPPORTS" read-only="true" />
101.    </tx:attributes>
102. </tx:advice>
103. <!--那些类的哪些方法参与事务 -->
104. <aop:config>
105.     <aop:advisor
106.         pointcut="execution(* cn.com.company.service.*Service.*(..))"
107.         advice-ref="txAdvice" />
108.     <aop:advisor
109.         pointcut="execution(* cn.com.company.service.*ServiceImpl.*(..))"
110.         advice-ref="txAdvice" />
111. </aop:config>
112.
113.
```

114. </beans>

```
! 注 applicationContenxt.xml中
<!-- 切换数据源 -->
<bean id="dataSourceAdvice" class="com.company.aop.DataSourceAdvice" />
<aop:config>
<aop:advisor
pointcut="execution(* cn.com.company.service..*Service.*(..))"
advice-ref="dataSourceAdvice" />
</aop:config>
```

一定要配置在事务AOP之上

[Learn Oracle Java at home](#)
The smart choice for your IT Dept. Train more staff, more frequently!
[education.oracle.com/Self-Study](#)

Google border=0 height=16 src=



分享到: [深入了解MySQL 5.5分区功能增强](#) ▶

15:55 | [评论 / 浏览 \(2 / 516\)](#) | 分类: [企业架构](#) | [相关推荐](#) [▶ MORE](#)

评论

2 楼 [rockyxiaogang](#) 2011-03-21
嗯。找了好久了，不错的东东，我收藏一下。

1 楼 [pxchen](#) 2011-03-21
sfsfsf

发表评论



[您还没有登录,请您登录后再发表评论](#)