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  [IMG_256](http://lujia35.iteye.com/blog/javascript:void())**

1. master.jdbc.driverClassName=com.mysql.jdbc.Driver
2. master.jdbc.url=\*\*\*\*\*\*\*\*
3. master.jdbc.username=\*\*\*\*\*\*\*\*
4. master.jdbc.password=\*\*\*\*\*\*\*\*
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  [IMG_257](http://lujia35.iteye.com/blog/javascript:void())**

1. **import** java.lang.reflect.Method;
2. **import** org.springframework.aop.AfterReturningAdvice;
3. **import** org.springframework.aop.MethodBeforeAdvice;
4. **import** org.springframework.aop.ThrowsAdvice;
6. **import** com.company.datasource.DataSourceSwitcher;
8. **public** **class** DataSourceAdvice **implements** MethodBeforeAdvice, AfterReturningAdvice, ThrowsAdvice {
9. // service方法执行之前被调用
10. **public** **void** before(Method method, Object[] args, Object target) **throws** Throwable {
11. System.out.println("切入点: " + target.getClass().getName() + "类中" + method.getName() + "方法");
12. **if**(method.getName().startsWith("add")
13. || method.getName().startsWith("create")
14. || method.getName().startsWith("save")
15. || method.getName().startsWith("edit")
16. || method.getName().startsWith("update")
17. || method.getName().startsWith("delete")
18. || method.getName().startsWith("remove")){
19. System.out.println("切换到: master");
20. DataSourceSwitcher.setMaster();
21. }
22. **else**  {
23. System.out.println("切换到: slave");
24. DataSourceSwitcher.setSlave();
25. }
26. }
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** Throwable {
30. }
32. // 抛出Exception之后被调用
33. **public** **void** afterThrowing(Method method, Object[] args, Object target, Exception ex) **throws** Throwable {
34. DataSourceSwitcher.setSlave();
35. System.out.println("出现异常,切换到: slave");
36. }
38. }

数据源选择类 DataSourceSwitcher.java

**Java代码  [IMG_258](http://lujia35.iteye.com/blog/javascript:void())**

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

DynamicDataSource.java数据源动态切换类

**Java代码  [IMG_259](http://lujia35.iteye.com/blog/javascript:void())**

1. **package** com.company.datasource;
3. **import** org.springframework.jdbc.datasource.lookup.AbstractRoutingDataSource;
5. **public** **class** DynamicDataSource **extends** AbstractRoutingDataSource {
7. @Override
8. **protected** Object determineCurrentLookupKey() {
9. **return** DataSourceSwitcher.getDataSource();
10. }
12. }

下面配置spring applicationContext.xml文件

**Xml代码  [IMG_260](http://lujia35.iteye.com/blog/javascript:void())**

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"**>**
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" **/>**
21. <!-- 加载properties配置文件 -->
22. **<bean** id="propertyConfigurer"
23. class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"**>**
24. **<property** name="locations"**>**
25. **<list>**
26. **<value>**classpath:jdbc.properties**</value>**
27. **</list>**
28. **</property>**
29. **</bean>**
31. **<bean** id="parentDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"**>**
32. //\*\*\*c3p0配置
33. **</bean>**
34. <!-- 主数据源-->
35. **<bean** id="masterDataSource" parent="parentDataSource"**>**
36. **<property** name="driverClass" value="${master.jdbc.driverClassName}" **/>**
37. **<property** name="jdbcUrl" value="${master.jdbc.url}" **/>**
38. **<property** name="user" value="${master.jdbc.username}" **/>**
39. **<property** name="password" value="${master.jdbc.password}" **/>**
40. **</bean>**
41. <!-- 从数据源-->
42. **<bean** id="slaveDataSource" parent="parentDataSource"**>**
43. **<property** name="driverClass" value="${slave.jdbc.driverClassName}" **/>**
44. **<property** name="jdbcUrl" value="${slave.jdbc.url}" **/>**
45. **<property** name="user" value="${slave.jdbc.username}" **/>**
46. **<property** name="password" value="${slave.jdbc.password}" **/>**
47. **</bean>**
49. **<bean** id="dataSource" class="com.company.datasource.DynamicDataSource"**>**
50. **<property** name="targetDataSources"**>**
51. **<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>**
58. <!-- 配置sessionFactory -->
59. **<bean** id="sessionFactory"
60. class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"**>**
61. **<property** name="dataSource" ref="dataSource"**></property>**
62. **<property** name="packagesToScan" value="cn.com.company.entity" **/>**
63. **<property** name="hibernateProperties"**>**
64. **<props>**
66. **<prop>**//\*\*\*hibernate一些参数这里不写了**</prop>**
67. **</props>**
68. **</property>**
69. **</bean>**
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>**
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>**

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