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
搜索

最近访客 >>更多访客

zxwu swen00

yasaso lzg3267373

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

深入了解MySQL 5.5分区功能增强

2011-03-21

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

```
master.jdbc.driverClassName=com.mysql.jdbc.Driver
master.jdbc.url=********
master.jdbc.username=********
master.jdbc.password=********

slave.jdbc.driverClassName=com.mysql.jdbc.Driver
slave.jdbc.url=********
slave.jdbc.username=********
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

```
import java.lang.reflect.Method;
import org.springframework.aop.AfterReturningAdvice;
import org.springframework.aop.MethodBeforeAdvice;
import org.springframework.aop.ThrowsAdvice;

import com.company.datasource.DataSourceSwitcher;

public class DataSourceAdvice implements MethodBeforeAdvice, AfterReturningAdvice, ThrowsAdvice
{
        // service方法执行之前被调用
        public void before(Method method, Object[] args, Object target) throws Throwable {
                System.out.println("切入点: " + target.getClass().getName() + "类中" + method.ge
tName() + "方法");
                if(method.getName().startsWith("add")
                        || method.getName().startsWith("create")
                        || method.getName().startsWith("save")
                        || method.getName().startsWith("edit")
                        || method.getName().startsWith("update")
                        || method.getName().startsWith("delete")
                        || method.getName().startsWith("remove")){
                        System.out.println("切换到: master");
                        DataSourceSwitcher.setMaster();
                }
                else  {
                        System.out.println("切换到: slave");
                        DataSourceSwitcher.setSlave();
```

- 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) th
    rows Throwable {
30.         }
31.
32.         // 抛出Exception之后被调用
33.         public void afterThrowing(Method method, Object[] args, Object target, Exception ex) th
    rows Throwable {
34.                 DataSourceSwitcher.setSlave();
35.                 System.out.println("出现异常,切换到: slave");
36.         }
37.
38. }
```

数据源选择类 DataSourceSwitcher.java

Java代码

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

```
                return DataSourceSwitcher.getDataSource();
        }

}
```

下面配置spring applicationContext.xml文件

Xml代码

```
<?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:aop="http://www.springframework.org/schema/aop"
        xmlns:tx="http://www.springframework.org/schema/tx" xmlns:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="
                http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                http://www.springframework.org/schema/tx
                http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
                http://www.springframework.org/schema/aop
                http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
                http://www.springframework.org/schema/context
                http://www.springframework.org/schema/context/spring-context-3.0.xsd">

        <context:annotation-config />
        <!-- 自动加载SERVICE DAO ACTION -->
        <context:component-scan base-package="cn.com.company.dao.*" />
        <context:component-scan base-package="cn.com.company.service.*" />
        <context:component-scan base-package="cn.com.company.action" />

        <!-- 加载properties配置文件 -->
        <bean id="propertyConfigurer"
                class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

                <property name="locations">
                        <list>
                                <value>classpath:jdbc.properties</value>
                        </list>
                </property>
        </bean>

        <bean id="parentDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
                //***c3p0配置
        </bean>
     <!-- 主数据源-->
        <bean id="masterDataSource" parent="parentDataSource">
                <property name="driverClass" value="${master.jdbc.driverClassName}" />
                <property name="jdbcUrl" value="${master.jdbc.url}" />
                <property name="user" value="${master.jdbc.username}" />
                <property name="password" value="${master.jdbc.password}" />
        </bean>
    <!-- 从数据源-->
        <bean id="slaveDataSource" parent="parentDataSource">
                <property name="driverClass" value="${slave.jdbc.driverClassName}" />
                <property name="jdbcUrl" value="${slave.jdbc.url}" />
                <property name="user" value="${slave.jdbc.username}" />
                <property name="password" value="${slave.jdbc.password}" />
        </bean>

        <bean id="dataSource" class="com.company.datasource.DynamicDataSource">
                <property name="targetDataSources">
                        <map key-type="java.lang.String">
```

```
                                <entry key="slave" value-ref="slaveDataSource" />
                        </map>
                </property>
                <property name="defaultTargetDataSource" ref="masterDataSource" />
        </bean>

        <!-- 配置sessionFactory -->
        <bean id="sessionFactory"
                class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
                <property name="dataSource" ref="dataSource"></property>
                <property name="packagesToScan" value="cn.com.company.entity" />
                <property name="hibernateProperties">
                        <props>

                                <prop>//***hibernate一些参数这里不写了</prop>
                        </props>
                </property>
        </bean>

        <!-- 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>

        <!-- 配置事务管理器 -->
        <bean id="transactionManager"
                class="org.springframework.orm.hibernate3.HibernateTransactionManager">
                <property name="sessionFactory">
                        <ref bean="sessionFactory" />
                </property>
        </bean>
        <!--配置事务的传播特性 -->
        <tx:advice id="txAdvice" transaction-manager="transactionManager">
                <tx:attributes>
                        <!-- 对增、删、改方法进行事务支持 -->
                        <tx:method name="add*" propagation="REQUIRED" />
                        <tx:method name="create*" propagation="REQUIRED" />
                        <tx:method name="save*" propagation="REQUIRED" />
                        <tx:method name="edit*" propagation="REQUIRED" />
                        <tx:method name="update*" propagation="REQUIRED" />
                        <tx:method name="delete*" propagation="REQUIRED" />
                        <tx:method name="remove*" propagation="REQUIRED" />
                        <!-- 对查找方法进行只读事务 -->
                        <tx:method name="loadByUsername*" propagation="SUPPORTS" read-only="true" />
                        <!-- 对其它方法进行只读事务 -->
                        <tx:method name="*" propagation="SUPPORTS" read-only="true" />
                </tx:attributes>
        </tx:advice>
        <!--那些类的哪些方法参与事务 -->
        <aop:config>
                <aop:advisor
                        pointcut="execution(* cn.com.company.service..*Service.*(..))"
                        advice-ref="txAdvice" />
                <aop:advisor
                        pointcut="execution(* cn.com.company.service..*ServiceImpl.*(..))"
                        advice-ref="txAdvice" />
        </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

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