**[Spring, MyBatis 多数据源的配置和管理](https://www.cnblogs.com/digdeep/p/4512368.html)**

同一个项目有时会涉及到多个数据库，也就是多数据源。多数据源又可以分为两种情况：

1）两个或多个数据库没有相关性，各自独立，其实这种可以作为两个项目来开发。比如在游戏开发中一个数据库是平台数据库，其它还有平台下的游戏对应的数据库；

2）两个或多个数据库是master-slave的关系，比如有mysql搭建一个 master-master，其后又带有多个slave；或者采用MHA搭建的master-slave复制；

目前我所知道的 Spring 多数据源的搭建大概有两种方式，可以根据多数据源的情况进行选择。

1. 采用spring配置文件直接配置多个数据源

比如针对两个数据库没有相关性的情况，可以采用直接在spring的配置文件中配置多个数据源，然后分别进行事务的配置，如下所示：

<context:component-scan base-package="net.aazj.service,net.aazj.aop" />

<context:component-scan base-package="net.aazj.aop" />

<!-- 引入属性文件 -->

<context:property-placeholder location="classpath:config/db.properties" />

<!-- 配置数据源 -->

<bean name="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">

<property name="url" value="${jdbc\_url}" />

<property name="username" value="${jdbc\_username}" />

<property name="password" value="${jdbc\_password}" />

<!-- 初始化连接大小 -->

<property name="initialSize" value="0" />

<!-- 连接池最大使用连接数量 -->

<property name="maxActive" value="20" />

<!-- 连接池最大空闲 -->

<property name="maxIdle" value="20" />

<!-- 连接池最小空闲 -->

<property name="minIdle" value="0" />

<!-- 获取连接最大等待时间 -->

<property name="maxWait" value="60000" />

</bean>

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<property name="configLocation" value="classpath:config/mybatis-config.xml" />

<property name="mapperLocations" value="classpath\*:config/mappers/\*\*/\*.xml" />

</bean>

<!-- Transaction manager for a single JDBC DataSource -->

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource" />

</bean>

<!-- 使用annotation定义事务 -->

<tx:annotation-driven transaction-manager="transactionManager" />

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="net.aazj.mapper" />

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>

</bean>

<!-- Enables the use of the @AspectJ style of Spring AOP -->

<aop:aspectj-autoproxy/>

<!-- ===============第二个数据源的配置=============== -->

<bean name="dataSource\_2" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">

<property name="url" value="${jdbc\_url\_2}" />

<property name="username" value="${jdbc\_username\_2}" />

<property name="password" value="${jdbc\_password\_2}" />

<!-- 初始化连接大小 -->

<property name="initialSize" value="0" />

<!-- 连接池最大使用连接数量 -->

<property name="maxActive" value="20" />

<!-- 连接池最大空闲 -->

<property name="maxIdle" value="20" />

<!-- 连接池最小空闲 -->

<property name="minIdle" value="0" />

<!-- 获取连接最大等待时间 -->

<property name="maxWait" value="60000" />

</bean>

<bean id="sqlSessionFactory\_slave" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource\_2" />

<property name="configLocation" value="classpath:config/mybatis-config-2.xml" />

<property name="mapperLocations" value="classpath\*:config/mappers2/\*\*/\*.xml" />

</bean>

<!-- Transaction manager for a single JDBC DataSource -->

<bean id="transactionManager\_2" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource\_2" />

</bean>

<!-- 使用annotation定义事务 -->

<tx:annotation-driven transaction-manager="transactionManager\_2" />

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="net.aazj.mapper2" />

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory\_2"/>

</bean>

[复制代码](javascript:void(0);)

如上所示，我们分别配置了两个 dataSource，两个sqlSessionFactory，两个transactionManager，以及关键的地方在于MapperScannerConfigurer 的配置——使用**sqlSessionFactoryBeanName**属性，注入不同的sqlSessionFactory的名称，这样的话，就为不同的数据库对应的 mapper 接口注入了对应的 sqlSessionFactory。

需要注意的是，多个数据库的这种配置是不支持分布式事务的，也就是同一个事务中，不能操作多个数据库。这种配置方式的优点是很简单，但是却不灵活。对于master-slave类型的多数据源配置而言不太适应，master-slave性的多数据源的配置，需要特别灵活，需要根据业务的类型进行细致的配置。比如对于一些耗时特别大的select语句，我们希望放到slave上执行，而对于update,delete等操作肯定是只能在master上执行的，另外对于一些实时性要求很高的select语句，我们也可能需要放到master上执行——比如一个场景是我去商城购买一件兵器，购买操作的很定是master，同时购买完成之后，需要重新查询出我所拥有的兵器和金币，那么这个查询可能也需要防止master上执行，而不能放在slave上去执行，因为slave上可能存在延时，我们可不希望玩家发现购买成功之后，在背包中却找不到兵器的情况出现。

所以对于master-slave类型的多数据源的配置，需要根据业务来进行灵活的配置，哪些select可以放到slave上，哪些select不能放到slave上。所以上面的那种所数据源的配置就不太适应了。

**2. 基于 AbstractRoutingDataSource 和 AOP 的多数据源的配置**

基本原理是，我们自己定义一个DataSource类ThreadLocalRountingDataSource，来继承AbstractRoutingDataSource，然后在配置文件中向ThreadLocalRountingDataSource注入 master 和 slave 的数据源，然后通过 AOP 来灵活配置，在哪些地方选择  master 数据源，在哪些地方需要选择 slave数据源。下面看代码实现：

**1）**先定义一个enum来表示不同的数据源：

[复制代码](javascript:void(0);)

package net.aazj.enums;

/\*\*

\* 数据源的类别：master/slave

\*/

public enum DataSources {

MASTER, SLAVE

}

[复制代码](javascript:void(0);)

**2）**通过 TheadLocal 来保存每个线程选择哪个数据源的标志(key)：

[复制代码](javascript:void(0);)

package net.aazj.util;

import net.aazj.enums.DataSources;

public class DataSourceTypeManager {

private static final ThreadLocal<DataSources> dataSourceTypes = new ThreadLocal<DataSources>(){

@Override

protected DataSources initialValue(){

return DataSources.MASTER;

}

};

public static DataSources get(){

return dataSourceTypes.get();

}

public static void set(DataSources dataSourceType){

dataSourceTypes.set(dataSourceType);

}

public static void reset(){

dataSourceTypes.set(DataSources.MASTER0);

}

}

[复制代码](javascript:void(0);)

**3）**定义 ThreadLocalRountingDataSource，继承AbstractRoutingDataSource：

[复制代码](javascript:void(0);)

package net.aazj.util;

import org.springframework.jdbc.datasource.lookup.AbstractRoutingDataSource;

public class ThreadLocalRountingDataSource extends AbstractRoutingDataSource {

@Override

protected Object determineCurrentLookupKey() {

return DataSourceTypeManager.get();

}

}

[复制代码](javascript:void(0);)

**4）**在配置文件中向 ThreadLocalRountingDataSource 注入 master 和 slave 的数据源：

[复制代码](javascript:void(0);)

<context:component-scan base-package="net.aazj.service,net.aazj.aop" />

<context:component-scan base-package="net.aazj.aop" />

<!-- 引入属性文件 -->

<context:property-placeholder location="classpath:config/db.properties" />

<!-- 配置数据源Master -->

<bean name="dataSourceMaster" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">

<property name="url" value="${jdbc\_url}" />

<property name="username" value="${jdbc\_username}" />

<property name="password" value="${jdbc\_password}" />

<!-- 初始化连接大小 -->

<property name="initialSize" value="0" />

<!-- 连接池最大使用连接数量 -->

<property name="maxActive" value="20" />

<!-- 连接池最大空闲 -->

<property name="maxIdle" value="20" />

<!-- 连接池最小空闲 -->

<property name="minIdle" value="0" />

<!-- 获取连接最大等待时间 -->

<property name="maxWait" value="60000" />

</bean>

<!-- 配置数据源Slave -->

<bean name="dataSourceSlave" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">

<property name="url" value="${jdbc\_url\_slave}" />

<property name="username" value="${jdbc\_username\_slave}" />

<property name="password" value="${jdbc\_password\_slave}" />

<!-- 初始化连接大小 -->

<property name="initialSize" value="0" />

<!-- 连接池最大使用连接数量 -->

<property name="maxActive" value="20" />

<!-- 连接池最大空闲 -->

<property name="maxIdle" value="20" />

<!-- 连接池最小空闲 -->

<property name="minIdle" value="0" />

<!-- 获取连接最大等待时间 -->

<property name="maxWait" value="60000" />

</bean>

<bean id="dataSource" class="net.aazj.util.ThreadLocalRountingDataSource">

<property name="defaultTargetDataSource" ref="dataSourceMaster" />

<property name="targetDataSources">

<map key-type="net.aazj.enums.DataSources">

<entry key="MASTER" value-ref="dataSourceMaster"/>

<entry key="SLAVE" value-ref="dataSourceSlave"/>

<!-- 这里还可以加多个dataSource -->

</map>

</property>

</bean>

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<property name="configLocation" value="classpath:config/mybatis-config.xml" />

<property name="mapperLocations" value="classpath\*:config/mappers/\*\*/\*.xml" />

</bean>

<!-- Transaction manager for a single JDBC DataSource -->

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource" />

</bean>

<!-- 使用annotation定义事务 -->

<tx:annotation-driven transaction-manager="transactionManager" />

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="net.aazj.mapper" />

<!-- <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/> -->

</bean>

[复制代码](javascript:void(0);)

上面spring的配置文件中，我们针对master数据库和slave数据库分别定义了dataSourceMaster和dataSourceSlave两个dataSource，然后注入到<bean id="dataSource" class="net.aazj.util.ThreadLocalRountingDataSource"> 中，这样我们的dataSource就可以来根据 key 的不同来选择dataSourceMaster和 dataSourceSlave了。

**5）**使用Spring AOP 来指定 dataSource 的 key ，从而dataSource会根据key选择 dataSourceMaster 和 dataSourceSlave：

[复制代码](javascript:void(0);)

package net.aazj.aop;

import net.aazj.enums.DataSources;

import net.aazj.util.DataSourceTypeManager;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.Pointcut;

import org.springframework.stereotype.Component;

@Aspect // for aop

@Component // for auto scan  
@Order(0)  // execute before @Transactional

public class DataSourceInterceptor {

@Pointcut("execution(public \* net.aazj.service..\*.getUser(..))")

public void dataSourceSlave(){};

@Before("dataSourceSlave()")

public void before(JoinPoint jp) {

DataSourceTypeManager.set(DataSources.SLAVE);

}  
 // ... ...

}

[复制代码](javascript:void(0);)

这里我们定义了一个 Aspect 类，我们使用 @Before 来在符合 @Pointcut("execution(public \* net.aazj.service..\*.getUser(..))") 中的方法被调用之前，调用 DataSourceTypeManager.set(DataSources.SLAVE) 设置了 key 的类型为 DataSources.SLAVE，所以 dataSource 会根据key=DataSources.SLAVE 选择 dataSourceSlave 这个dataSource。所以该方法对于的sql语句会在slave数据库上执行(经网友[老刘1987](http://www.cnblogs.com/readylau1987/" \t "_blank)提醒，这里存在多个Aspect之间的一个执行顺序的问题，必须保证切换数据源的Aspect必须在@Transactional这个Aspect之前执行，所以这里使用了@Order(0)来保证切换数据源先于@Transactional执行)。

我们可以不断的扩充 DataSourceInterceptor  这个 Aspect，在中进行各种各样的定义，来为某个service的某个方法指定合适的数据源对应的dataSource。

这样我们就可以使用 Spring AOP 的强大功能来，十分灵活进行配置了。

**6）AbstractRoutingDataSource原理剖析**

ThreadLocalRountingDataSource继承了AbstractRoutingDataSource，实现其抽象方法protected abstract Object determineCurrentLookupKey(); 从而实现对不同数据源的路由功能。我们从源码入手分析下其中原理：

public abstract class AbstractRoutingDataSource extends AbstractDataSource implements InitializingBean

AbstractRoutingDataSource **实现了 InitializingBean 那么spring在初始化该bean时，会调用InitializingBean的接口**  
**void afterPropertiesSet() throws Exception;** 我们看下AbstractRoutingDataSource是如何实现这个接口的：

[复制代码](javascript:void(0);)

@Override

public void afterPropertiesSet() {

if (this.targetDataSources == null) {

throw new IllegalArgumentException("Property 'targetDataSources' is required");

}

this.resolvedDataSources = new HashMap<Object, DataSource>(this.targetDataSources.size());

for (Map.Entry<Object, Object> entry : this.targetDataSources.entrySet()) {

Object lookupKey = resolveSpecifiedLookupKey(entry.getKey());

DataSource dataSource = resolveSpecifiedDataSource(entry.getValue());

this.resolvedDataSources.put(lookupKey, dataSource);

}

if (this.defaultTargetDataSource != null) {

this.resolvedDefaultDataSource = resolveSpecifiedDataSource(this.defaultTargetDataSource);

}

}

[复制代码](javascript:void(0);)

targetDataSources 是我们在xml配置文件中注入的 dataSourceMaster 和 dataSourceSlave. **afterPropertiesSet方法就是使用注入的**  
**dataSourceMaster 和 dataSourceSlave来构造一个HashMap——resolvedDataSources。方便后面根据 key 从该map 中取得对应的dataSource**。  
我们在看下 AbstractDataSource 接口中的 Connection getConnection() throws SQLException; 是如何实现的：

@Override

public Connection getConnection() throws SQLException {

return determineTargetDataSource().getConnection();

}

关键在于 determineTargetDataSource()，根据方法名就可以看出，应该此处就决定了使用哪个 dataSource ：

[复制代码](javascript:void(0);)

protected DataSource determineTargetDataSource() {

Assert.notNull(this.resolvedDataSources, "DataSource router not initialized");

Object lookupKey = determineCurrentLookupKey();

DataSource dataSource = this.resolvedDataSources.get(lookupKey);

if (dataSource == null && (this.lenientFallback || lookupKey == null)) {

dataSource = this.resolvedDefaultDataSource;

}

if (dataSource == null) {

throw new IllegalStateException("Cannot determine target DataSource for lookup key [" + lookupKey + "]");

}

return dataSource;

}

[复制代码](javascript:void(0);)

Object lookupKey = determineCurrentLookupKey(); 该方法是我们实现的，在其中获取ThreadLocal中保存的 key 值。获得了key之后，  
在从afterPropertiesSet()中初始化好了的resolvedDataSources这个map中获得key对应的dataSource。而ThreadLocal中保存的 key 值  
是通过AOP的方式在调用service中相关方法之前设置好的。OK，到此搞定！

**7）扩展 ThreadLocalRountingDataSource**

上面我们只是实现了 master-slave 数据源的选择。如果有多台 master 或者有多台 slave。多台master组成一个HA，要实现当其中一台master挂了是，自动切换到另一台master，这个功能可以使用LVS/Keepalived来实现，也可以通过进一步扩展ThreadLocalRountingDataSource来实现，可以另外加一个线程专门来每个一秒来测试mysql是否正常来实现。同样对于多台slave之间要实现负载均衡，同时当一台slave挂了时，要实现将其从负载均衡中去除掉，这个功能既可以使用LVS/Keepalived来实现，同样也可以通过近一步扩展ThreadLocalRountingDataSource来实现。

**3. 总结**

从本文中我们可以体会到AOP的强大和灵活。

本文使用的是mybatis,其实使用Hibernate也应该是相似的配置。

分类: [Java-ibatis/mybatis](http://www.cnblogs.com/digdeep/category/644918.html),[Java-spring/springmvc](http://www.cnblogs.com/digdeep/category/683618.html)

[**好文要顶**](javascript:void(0);) [**关注我**](javascript:void(0);) [**收藏该文**](javascript:void(0);) **[https://common.cnblogs.com/images/icon_weibo_24.png](javascript:void(0);)** **[https://common.cnblogs.com/images/wechat.png](javascript:void(0);)**

[https://pic.cnblogs.com/face/sample_face.gif](http://home.cnblogs.com/u/digdeep/)

[digdeep](http://home.cnblogs.com/u/digdeep/)  
[关注 - 46](http://home.cnblogs.com/u/digdeep/followees)  
[粉丝 - 195](http://home.cnblogs.com/u/digdeep/followers)

[+加关注](javascript:void(0);)

9

0

[«](http://www.cnblogs.com/digdeep/p/4510875.html)上一篇：[ThreadLocal 源码剖析](http://www.cnblogs.com/digdeep/p/4510875.html" \o "发布于2015-05-17 23:39)  
[»](http://www.cnblogs.com/digdeep/p/4514156.html)下一篇：[如何挑选适合的前端框架(去哪儿网前端架构师司徒正美)](http://www.cnblogs.com/digdeep/p/4514156.html" \o "发布于2015-05-19 11:44)

posted on 2015-05-18 17:06 [digdeep](http://www.cnblogs.com/digdeep/) 阅读(43025) 评论(25) [编辑](https://i.cnblogs.com/EditPosts.aspx?postid=4512368) [收藏](https://www.cnblogs.com/digdeep/p/4512368.html)

**评论**

[**#1楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3186439)**2015-05-18 17:35**[**yonghu86**](http://www.cnblogs.com/huyong/)

顶了

[支持(0)反对(0)](javascript:void(0);)

[**#2楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3186474)**2015-05-18 18:05**[**老刘2018**](http://www.cnblogs.com/readylau1987/)

对于多数据源aop切面的service方法，如果需要事务控制，不知道lz是怎么处理的。也用的spring事务aop吗？如果是，这两个到一起，会不会有影响？

[支持(0)反对(0)](javascript:void(0);)

[**#3楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3186509)**[楼主] 2015-05-18 19:26 [digdeep](http://www.cnblogs.com/digdeep/" \t "_blank)**

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3186474) 老刘1987  
对，基于注解的 @Transactional  
这个怎么会有影响呢？一个是切换数据源，一个是针对这个数据源的事务控制。不会有影响的。

[支持(0)反对(0)](javascript:void(0);)

[**#4楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3186510)**[楼主] 2015-05-18 19:27 [digdeep](http://www.cnblogs.com/digdeep/" \t "_blank)**

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3186439) yonghu86  
谢谢支持

[支持(0)反对(0)](javascript:void(0);)

[**#5楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3188389)**2015-05-20 17:29**[**老刘2018**](http://www.cnblogs.com/readylau1987/)

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3186509) digdeep  
可能我没描述清楚问题，因为事务是依赖数据源的，你用注解切换数据源的操作 跟 用注解加事务控制的操作应该是注意下先后关系  
  
今天自己写了demo测了下，切换数据源的注解配置是要放在事务注解配置前面的，不然有问题。  
  
ps:也感谢lz的分享，没怎么遇到多数据源，看了博文学到了！！

[支持(1)反对(0)](javascript:void(0);)

[**#6楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3188432)**[楼主] 2015-05-20 17:45 [digdeep](http://www.cnblogs.com/digdeep/" \t "_blank)**

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3188389) 老刘1987  
这里有个先后的逻辑，切换数据源之后，才能从数据源获得 Connection, 有了Connection才能开始事务的。肯定是切换数据源在前面的。  
  
你的意思是说，我上面的贴出的代码是先获得了Connection，再切换数据源的吗？没有吧。  
  
我在DataSourceInterceptor中使用的注解是 @Before 横切在 Service 中的方法上的，它会在这些方法执行之前就先切换数据源，也就是说，切换了数据源之后，再调用Service中的方法，而Service中的方法会获得数据源的Connetion，然后才开启事务。

[支持(0)反对(0)](javascript:void(0);)

[**#7楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3188486)**2015-05-20 18:18**[**老刘2018**](http://www.cnblogs.com/readylau1987/)

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3188432) digdeep  
跟用@Before没啥关系，还是跟spring初始化的顺序有关，要是你初始化的@Before，再@Transactional那就没问题；但要是顺序相反呢？

[支持(0)反对(0)](javascript:void(0);)

[**#8楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3188623)**[楼主] 2015-05-20 20:40 [digdeep](http://www.cnblogs.com/digdeep/" \t "_blank)**

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3188486) 老刘1987  
原来你是这个意思，哥们，感觉你对spring没有深入的了解吧。Spring的ApplicationContext启动时，会初始化所有的非lazy-init的bean，并进行依赖注入。所以并不存在先初始化有@Transactional注解的service，后初始化有@Before注解的DataSourceInterceptor。在使用这些bean之前，都已经被spring初始化好了。

[支持(0)反对(0)](javascript:void(0);)

[**#9楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3188992)**2015-05-21 09:08**[**老刘2018**](http://www.cnblogs.com/readylau1987/)

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3188623) digdeep  
那奇了怪，为什么我的代码里，两块的配置调个顺序，就会出现事务失效呢？我再研究下吧  
  
另外我对spring的理解，一直是bean的初始化确实是spring完成的，但是初始化也是还有顺序问题

[支持(0)反对(0)](javascript:void(0);)

[**#10楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3189060)**[楼主] 2015-05-21 09:43 [digdeep](http://www.cnblogs.com/digdeep/" \t "_blank)**

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3188992) 老刘1987  
嗯，初始化是有顺序，应该是先初始化被依赖的bean(depends-on)，其它的顺序我们怎么控制呢？getBean之前所有非lazy-init的bean都已经是初始化好了的。  
  
配置调个顺序，怎么调？

[支持(0)反对(0)](javascript:void(0);)

[**#11楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3189176)**2015-05-21 10:57**[**老刘2018**](http://www.cnblogs.com/readylau1987/)

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3189060) digdeep  
application-context.xml:   
<bean id="aspectBean" class="com.suning.demo.DataSourceAspect"/>  
<aop:config>  
<aop:aspect id="dataSourceAspect" ref="aspectBean">  
<aop:pointcut id="dataSourcePoint" expression="execution(\* com.suning.demo.service..\*(..))" />  
<aop:before method="beforeDaoMethod" pointcut-ref="dataSourcePoint"/>  
</aop:aspect>  
</aop:config>  
  
<!-- 配置事务管理器 -->  
<bean id="transactionManager"  
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
<property name="dataSource" ref="txAwareCustomDs" />  
</bean>  
<tx:annotation-driven transaction-manager="transactionManager"/>  
  
如上的配置，我切换数据源时事务是有效的；但如果把<aop:config>部分放到<tx:annotation-driven>配置后面就会出现问题，通过日志会看到先打开事务，后切换数据源  
日志：  
......  
2015-05-21 10:55:49 [日志信息] Returning cached instance of singleton bean 'transactionManager'  
2015-05-21 10:55:49 [日志信息] Creating new transaction with name [com.suning.demo.service.MainServiceImpl.getCountById]: PROPAGATION\_REQUIRED,ISOLATION\_DEFAULT; ''  
2015-05-21 10:55:49 [日志信息] trace com.mchange.v2.resourcepool.BasicResourcePool@45aee222 [managed: 3, unused: 2, excluded: 0] (e.g. com.mchange.v2.c3p0.impl.NewPooledConnection@10359b5d)  
2015-05-21 10:55:49 [日志信息] Acquired Connection [com.mchange.v2.c3p0.impl.NewProxyConnection@702656a] for JDBC transaction  
2015-05-21 10:55:49 [日志信息] Switching JDBC Connection [com.mchange.v2.c3p0.impl.NewProxyConnection@702656a] to manual commit  
2015-05-21 10:55:49 [日志信息] Returning cached instance of singleton bean 'aspectBean'  
[DataSourceAspect]修改数据源，{dataSource = act}  
2015-05-21 10:55:49 [日志信息] Executing prepared SQL query  
2015-05-21 10:55:49 [日志信息] Executing prepared SQL statement [select count(\*) from tf\_b\_paylog where charge\_id=?]  
2015-05-21 10:55:49 [日志信息] com.mchange.v2.c3p0.impl.NewPooledConnection@10359b5d handling a throwable.  
java.sql.SQLException: ORA-00942: 表或视图不存在  
...  
  
我还是等晚上再抽时间研究下吧

[支持(0)反对(0)](javascript:void(0);)

[**#12楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3189495)**[楼主] 2015-05-21 15:17 [digdeep](http://www.cnblogs.com/digdeep/" \t "_blank)**

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3189176) 老刘1987  
我测试了下，你说的是对的。确实需要将<aop:config>放到<tx:annotation-driven>的前面。我那个基于注解的DataSourceInterceptor 是有问题的，当时没有仔细测试。看来基于注解@Aspect的AOP配置和基于<aop:config>的AOP配置并不是等价的。谢谢你指出这个问题。

[支持(1)反对(0)](javascript:void(0);)

[**#13楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3189607)**2015-05-21 17:03**[**老刘2018**](http://www.cnblogs.com/readylau1987/)

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3189495) digdeep  
我觉得两种配置还是等价的  
  
你用的是注解配置，我用的是<aop:config>基于xml配置，我的xml里面也是指定初始化顺序对了，才能事务生效。  
而你的注解配置的DataSourceInterceptor是加的@Component，它的初始化是交给spring自动扫描的（这里spring可能按某种路径或规则扫描的），你的事务没问题，可能刚好就是先扫到DataSourceInterceptor并初始化  
  
不过也只是我的猜测，等后面有时间我验证下

[支持(0)反对(0)](javascript:void(0);)

[**#14楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3189718)**[楼主] 2015-05-21 18:24 [digdeep](http://www.cnblogs.com/digdeep/" \t "_blank)**

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3189607) 老刘1987  
我又测试了下，感觉不是 bean 初始化的顺序的问题，而是不同的AOP注解的执行顺序的问题。也就是自定义的 DataSourceInterceptor 和 <tx:annotation-driven> 开启的 @Transactional 的处理器 TransactionInterceptor 哪一个先执行的问题。也就是两个横切面，哪个先执行的问题。原来基于注解@Aspect配置的DataSourceInterceptor是有问题的，但是在其上加上：@Order(0) 之后又成功了：  
@Aspect // for aop  
@Component // for auto scan  
@Order(0) // execute before @Transactional  
public class DataSourceInterceptor {  
@Pointcut("execution(public \* net.aazj.service..\*.getUser(..))")  
public void dataSourceSlave(){};  
  
@Before("dataSourceSlave()")  
public void before(JoinPoint jp) {  
System.out.println("before--------");  
DataSourceTypeManager.set(DataSources.SLAVE);  
System.out.println(DataSourceTypeManager.get());  
}  
}  
所以是哪个AOP先执行的问题。  
<aop:aspectj-autoproxy/> 和 <tx:annotation-driven />在配置文件中的先后顺序没有关系。

[支持(1)反对(0)](javascript:void(0);)

[**#15楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3570367)**2016-12-01 21:53**[**无敌小瘪四**](http://www.cnblogs.com/zxbsping/)

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3189718) digdeep  
这里还有一个问题，<tx:annotation-driven /> 和 <aop:aspectj-autoproxy />开启了proxy-target-class="true"后，order不起作用了，不知您是否遇到过此种情况。

[支持(0)反对(0)](javascript:void(0);)

[**#16楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3615871)**2017-02-10 11:15**[**小小学渣**](http://www.cnblogs.com/zhoumingming/)

可以给我发个demo吗？？？谢谢 156873783@qq.com

[支持(0)反对(0)](javascript:void(0);)

[**#17楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3723880)**2017-06-27 17:58**[**4518**](http://home.cnblogs.com/u/1189974/)

老铁，有完整的demo么，新手求罩，万分感谢，1821669628@qq.com

[支持(0)反对(1)](javascript:void(0);)

[**#18楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3760626)**2017-08-17 19:59**[**杜小°**](http://www.cnblogs.com/dumanyi/)

你好，刚好在看这个，但是我本地调试不起来，有完整的demo吗？782226501@qq.com

[支持(0)反对(0)](javascript:void(0);)

[**#19楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3763048)**2017-08-22 08:51**[**加勒比海蓝蓝**](http://home.cnblogs.com/u/1223872/)

lz你这方法经验证不行，数据源切换不了。是我哪点写的不对？

[支持(0)反对(0)](javascript:void(0);)

[**#20楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3817151)**2017-10-20 23:14**[**final变量**](http://home.cnblogs.com/u/1242291/)

Spring, MyBatis 多数据源的配置和管理- digdeep - 博客园，写的不错不错，收藏了。  
  
推荐下，分库分表中间件 Sharding-JDBC 源码解析 17 篇：[http://www.yunai.me/categories/Sharding-JDBC/?cnblog](http://www.yunai.me/categories/Sharding-JDBC/?cnblog" \t "_blank)&602  
  
  
伧

[支持(0)反对(0)](javascript:void(0);)

[**#21楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3843299)**2017-11-17 16:05**[**s灰灰**](http://www.cnblogs.com/youth6966/)

今天也遇到要配置两个数据源的项目。亲测，当配置两个数据源，分别为不同DAO注入不同的sessionFactory是可行的。但是对于事务管理，为两个数据源配置两个事务以后，总是第一个有效，第二个无效，反复试了很多次。

[支持(0)反对(0)](javascript:void(0);)

[**#22楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3855039)**2017-11-30 21:29**[**疯狂的小新FCB**](http://www.cnblogs.com/beyondmessi/)

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3843299) s灰灰  
请问你解决这个问题了吗？我也碰到了这个问题 我的事务都不起作用

[支持(0)反对(0)](javascript:void(0);)

[**#23楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3855464)**2017-12-01 12:17**[**s灰灰**](http://www.cnblogs.com/youth6966/)

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3855039) 疯狂的小新FCB  
已经解决了：假设你的多个数据源已经配置成功了。  
多事务管理配置方法：第二个事务配制的时候

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <!-- 注解驱动的声明式事务管理 -->      <tx:annotation-driven transaction-manager="transactionManager.sys" />        <!-- Tel JDBC事务管理器 -->      <bean id="transactionManager.tel" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">          <property name="dataSource" ref="dataSource.tel" />          <qualifier value="tel"/>      </bean> |

添加：<qualifier value="tel"/>  
在添加事物的时候就可以使用@Transactional("tel")添加对应数据源的事务。  
  
  
  
  
  
个人测试仅仅支持分别控制单个数据源事务分别生效，如果要在一个事务中涉及多个数据源。就要了解一下分布事务控制，又是一个课题。因本人项目中不涉及，就没有再研究。

[支持(0)反对(0)](javascript:void(0);)

[**#24楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3866347)**2017-12-15 10:16**[**疯狂的小新FCB**](http://www.cnblogs.com/beyondmessi/)

[@](https://www.cnblogs.com/digdeep/p/4512368.html#3855464) s灰灰  
你可以试试 你这种方式，事务好像不起作用吧？不回滚

[支持(0)反对(0)](javascript:void(0);)

[**#25楼**](https://www.cnblogs.com/digdeep/p/4512368.html#3869287)**2017-12-19 15:14**[**临渊慕鱼退而结网**](http://www.cnblogs.com/punixi/)

多源数据库，难道大家2套系统不要同时做事务吗？分布式事务基于 AbstractRoutingDataSource 和 AOP 的多数据源的配置这个方法已经没法用了，除非单事务数据源