**Routing4DB使用文档**

**—— 谷宝剑**

目录

一、Master-Slave读写分离实现 2

1.1写Master，读多个Slaves，示意图如下： 2

1.2写Master，读Master和多个Slave，示意图如下 5

1.3 Master-Standby-Slaves实现，此方式示意图如下： 6

二、分库功能 8

2.1单机分库功能，示意图如下： 8

2.2 多机集群分库，构建分布式数据库，示意图如下： 12

2.3 高可用多级分布式集群，示意图如下： 14

三、负载均衡示例 15

四、自定义数据源路由策略 15

五、如何指定数据源 16

六、事务的处理 16

七、如何设置某些方法不执行路由 16

八、针对Mybatis的增强功能 17

九、问题汇总 17

9.1 oracle下master-standby模式未生效 17

十、BUG反馈及交流 18

版本1.1.0

**修订历史记录表：**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **参与者** | **说明** |
| 1.1.0 | 2013-8-10 | 谷宝剑  无花 | 1. 增加获取数据源的路由log提示 2. checkAvailableSql数据源有效性检查sql可配置化，支持oracle、postgres等数据库 3. 默认lenientFallback属性为false，防止配置出错时选用默认数据源作为路由数据源，而导致的错误。 4. 增加路由方法过滤功能，可指定接口中某些方法不执行路由 |
| 1.0.0 | 2013-5-8 | 谷宝剑 | 基于接口的代理策略的数据源路由框架发布 |

# 一、Master-Slave读写分离实现

## 1.1写Master，读多个Slaves，示意图如下：



**1.1 读Master写 Slave示意图**

Spring配置如下，详情见test/resource目录下write-master-read-slaves.xml及WriteMasterReadSlavesTest.java：

<!-- Master数据源 -->

<bean id=*"masterDataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*> <value>jdbc:mysql://localhost:3306/test?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

<!-- Slave数据源 -->

<bean id=*"slaveDataSourceOne"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://localhost:3306/test?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

<bean id=*"slaveDataSourceTwo"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*> <value>jdbc:mysql://localhost:3306/test?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

<!-- 配置Routing4DB 数据源 -->

<bean id=*"routing4DBDataSource"* class=*"com.google.code.routing4db.datasource.Routing4DBDataSource"*>

<property name=*"targetDataSources"*>

<!-- 配置数据源标示符号 -->

<map>

<!-- master datasource -->

<entry key="masterDataSource" value-ref="masterDataSource"></entry>

<!-- slave数据源编号 -->

<entry key="slaveDataSourceOne" value-ref="slaveDataSourceOne"/>

<entry key="slaveDataSourceTwo" value-ref="slaveDataSourceTwo"/>

</map>

</property>

<property name=*"defaultTargetDataSource"* ref=*"masterDataSource"*/>

</bean>

<!-- JdbcTemplate -->

<bean id=*"jdbcTemplate"* class=*"org.springframework.jdbc.core.JdbcTemplate"*>

<property name=*"dataSource"* ref=*"routing4DBDataSource"*/>

</bean>

<!-- 事务配置, 事务注解@Transactional要放到实现类上，不支持放到接口上 -->

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

<property name=*"dataSource"* ref=*"routing4DBDataSource"* />

</bean>

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

<!-- 配置路由策略 -->

<bean id=*"masterSlaveStrategy"* class=*"com.google.code.routing4db.strategy.impl.MasterSlaveStrategy"*>

<property name=*"readMethodPatterns"*>

<list>

<value>\*get\*</value>

<value>\*find\*</value>

</list>

</property>

<!-- Master数据源标识符 -->

<property name=*"masterDataSourceKey"* value=*"masterDataSource"*></property>

<!-- Slaves数据源 从0开始，以此编号 -->

<property name=*"dataSourceKeyMap"*>

<map>

<entry key=*"0"* value=*"slaveDataSourceOne"*></entry>

<entry key=*"1"* value=*"slaveDataSourceTwo"*></entry>

</map>

</property>

</bean>

<!-- Dao实现 -->

<bean id=*"userDaoTarget"* class=*"com.google.code.routing4db.dao.UserDaoJdbcTemplateImpl"*></bean>

<!-- 配置DAO接口代理 ）-->

<bean id=*"userDao"* class=*"com.google.code.routing4db.spring.RoutingSpringFactoryBean"*>

<!-- 代理接口 -->

<property name=*"targetInterface"* value=*"com.google.code.routing4db.dao.UserDao"*></property>

<!-- 代理的DAO实际对象 -->

<property name=*"targetObject"* ref=*"userDaoTarget"*></property>

<!-- 路由策略 -->

<property name=*"routingStrategy"* ref=*"masterSlaveStrategy"*></property>

</bean>

## 1.2写Master，读Master和多个Slave，示意图如下



**1.2写Master读Master和Slaves示意图**

此方式配置和1.1方式相同，仅仅在路由策略Slaves节点中加入Master即可。不同的配置如下:

<!-- 配置路由策略 -->

<bean id=*"masterSlaveStrategy"* class=*"com.google.code.routing4db.strategy.impl.MasterSlaveStrategy"*>

<!-- 读接口的方法通配符 -->

<property name=*"readMethodPatterns"*>

<list>

<value>\*get\*</value>

<value>\*find\*</value>

</list>

</property>

<!-- Master数据源 -->

<property name=*"masterDataSourceKey"* value=*"masterDataSource"*></property>

<!-- Slaves数据源 从0开始，以此编号 -->

<property name=*"dataSourceKeyMap"*>

<map>

<entry key=*"0"* value=*"slaveDataSourceOne"*></entry>

<entry key=*"1"* value=*"slaveDataSourceTwo"*></entry>

<!-- 读master -->

<entry key=*"2"* value=*"masterDataSource"*></entry>

</map>

</property>

</bean>

## 1.3 Master-Standby-Slaves实现，此方式示意图如下：



该方式在Master节点出故障时，会自动切换到Standby节点。将写入操作转移到Standby节点，在Master节点故障恢复正常后，自动把写操作迁移到Master节点。此方式配置如下：

<!-- Master数据源 -->

<bean id=*"masterDataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://192.168.56.102:3306/test?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

<property name=*"checkoutTimeout"* value=*"2000"*/>

<property name=*"acquireRetryAttempts"* value=*"3"*></property>

</bean>

<!-- Standby数据源 -->

<bean id=*"standbyDataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://localhost:3306/test?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

<property name=*"checkoutTimeout"* value=*"2000"*/>

<property name=*"acquireRetryAttempts"* value=*"3"*></property>

</bean>

<!-- 配置Master Standby 的数据源 -->

<bean id=*"masterStandbyDataSource"* class=*"com.google.code.routing4db.datasource.MasterStrandbyDataSource"*>

<property name=*"standbyDataSource"* ref=*"standbyDataSource"* />

<property name=*"masterDataSource"* ref=*"masterDataSource"*/>

</bean>

<!-- 配置Routing4DB 数据源 -->

<bean id=*"routing4DBDataSource"* class=*"com.google.code.routing4db.datasource.Routing4DBDataSource"*>

<property name=*"targetDataSources"*>

<map>

<entry key=*"slaveDataSourceOne"* value-ref=*"slaveDataSourceOne"*/>

<entry key=*"slaveDataSourceTwo"* value-ref=*"slaveDataSourceTwo"*/>

</map>

</property>

<property name=*"defaultTargetDataSource"* ref=*"masterStandbyDataSource"*/>

</bean>

其它配置与1.1及1.2相同，可参考1.1及1.2。

# 二、分库功能

## 2.1单机分库功能，示意图如下：



**2.1 单机分库示意图**

此方式配置如下，具体可参考测试工程中的：ShardDatabaseByModTest.java 及shard-database-by-mod.xml

<!-- mode one 数据源 -->

<bean id=*"dataSourceOne"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://localhost:3306/test0?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

<!-- mode two 数据源 -->

<bean id=*"dataSourceTwo"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://localhost:3306/test1?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

<!-- mode three 数据源 -->

<bean id=*"dataSourceThree"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://localhost:3306/test2?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

<!-- mode four 数据源 -->

<bean id=*"dataSourceFour"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://localhost:3306/test3?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

<!-- 配置Routing4DB 数据源 -->

<bean id=*"routing4DBDataSource"* class=*"com.google.code.routing4db.datasource.Routing4DBDataSource"*>

<property name=*"targetDataSources"*>

<map>

<entry key=*"dataSourceOne"* value-ref=*"dataSourceOne"*/>

<entry key=*"dataSourceTwo"* value-ref=*"dataSourceTwo"*/>

<entry key=*"dataSourceThree"* value-ref=*"dataSourceThree"*/>

<entry key=*"dataSourceFour"* value-ref=*"dataSourceFour"*/>

</map>

</property>

<!-- 无路由配置时默认的datasource -->

<property name=*"defaultTargetDataSource"* ref=*"dataSourceOne"*/>

</bean>

<!-- JdbcTemplate -->

<bean id=*"jdbcTemplate"* class=*"org.springframework.jdbc.core.JdbcTemplate"*>

<property name=*"dataSource"* ref=*"routing4DBDataSource"*/>

</bean>

<!-- 事务配置, 事务注解@Transactional要放到实现类上，不支持放到接口上 -->

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

<property name=*"dataSource"* ref=*"routing4DBDataSource"* />

</bean>

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

<!-- 配置路由策略 -->

<bean id=*"modRoutingStrategy"* class=*"com.google.code.routing4db.strategy.impl.ModRoutingStrategy"*>

<!-- 分4个数据库, 采用Id属性进行分库路由 -->

<property name=*"dataSourceNum"* value=*"4"*></property>

<property name=*"propertyName"* value=*"id"*></property>

<!--取模的数据源 从0开始，以此编号 -->

<property name=*"dataSourceKeyMap"*>

<map>

<entry key=*"0"* value=*"dataSourceOne"*></entry>

<entry key=*"1"* value=*"dataSourceTwo"*></entry>

<entry key=*"2"* value=*"dataSourceThree"*></entry>

<entry key=*"3"* value=*"dataSourceFour"*></entry>

</map>

</property>

</bean>

<!-- Dao实现 -->

<bean id=*"userDaoTarget"* class=*"com.google.code.routing4db.dao.UserDaoJdbcTemplateImpl"*></bean>

<!-- 配置DAO接口代理 ）-->

<bean id=*"userDao"* class=*"com.google.code.routing4db.spring.RoutingSpringFactoryBean"*>

<!-- 代理接口 -->

<property name=*"targetInterface"* value=*"com.google.code.routing4db.dao.UserDao"*></property>

<!-- 代理的DAO实际对象 -->

<property name=*"targetObject"* ref=*"userDaoTarget"*></property>

<!-- 路由策略 -->

<property name=*"routingStrategy"* ref=*"modRoutingStrategy"*></property>

</bean>

## 2.2 多机集群分库，构建分布式数据库，示意图如下：



**2.2 多机分布式机器分库示意图**

此方式与单机分库没什么区别，原理及配置方式相同，唯一区别就是把数据源的IP修改一下，其配置可参考2.1单机版分库，参考测试工程中的：ShardDatabaseByModTest.java及shard-database-by-mod.xml实例。配置如下，：

<!-- mode one 数据源 -->

<bean id=*"dataSourceOne"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://192.168.3.25:3306/test?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

<!-- mode two 数据源 -->

<bean id=*"dataSourceTwo"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://192.168.3.26:3306/test1?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

<!-- mode three 数据源 -->

<bean id=*"dataSourceThree"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://192.168.3.27:3306/test2?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

<!-- mode four 数据源 -->

<bean id=*"dataSourceFour"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"jdbcUrl"*>

<value>jdbc:mysql://192.168.3.28:3306/test3?useUnicode=true&amp;characterEncoding=UTF-8&amp;autoReconnect=true&amp;failOverReadOnly=false</value>

</property>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"lovejava"* />

<property name=*"initialPoolSize"* value=*"2"* />

<property name=*"minPoolSize"* value=*"2"* />

<property name=*"maxPoolSize"* value=*"10"* />

<property name=*"acquireIncrement"* value=*"5"* />

<property name=*"maxIdleTime"* value=*"30"* />

<property name=*"maxStatements"* value=*"0"* />

</bean>

## 2.3 高可用多级分布式集群，示意图如下：



**2.3 高可用分布式集群**

此方式结合master-slaves及mod模式，具体可参考mod-master-slaves-example.xml及ModMasterSlavesTest.java 部分配置如下：

<!-- 配置Routing4DB 数据源 -->

<bean id=*"routing4DBDataSource"* class=*"com.google.code.routing4db.datasource.Routing4DBDataSource"*>

<property name=*"targetDataSources"*>

<map>

<entry key=*"dataSourceOneMaster"* value-ref=*"dataSourceOneMaster"*/>

<entry key=*"dataSourceOneSlaveOne"* value-ref=*"dataSourceOneSlaveOne"*/>

<entry key=*"dataSourceOneSlaveTwo"* value-ref=*"dataSourceOneSlaveTwo"*/>

<entry key=*"dataSourceTwoMaster"* value-ref=*"dataSourceTwoMaster"*/>

<entry key=*"dataSourceSlaveOne"* value-ref=*"dataSourceSlaveOne"*/>

<entry key=*"dataSourceSlaveTwo"* value-ref=*"dataSourceSlaveTwo"*/>

</map>

</property>

<!-- 默认的datasource -->

<property name=*"defaultTargetDataSource"* ref=*"dataSourceOneMaster"*/>

</bean>

<!-- 配置路由策略 -->

<bean id=*"modMasterSlaveRoutingStrategy"* class=*"com.google.code.routing4db.strategy.impl.ModMasterSlaveRoutingStrategy"*>

<!-- 分2集群组, 采用Id属性进行分库路由 -->

<property name=*"dataSourceNum"* value=*"2"*></property>

<property name=*"propertyName"* value=*"id"*></property>

<!--取模的数据源 从0开始，以此编号 -->

<property name=*"dataSourceKeyMap"*>

<map>

<entry key=*"0"* value=*"dataSourceOneMaster,dataSourceOneSlaveOne,dataSourceOneSlaveTwo"*></entry>

<entry key=*"1"* value=*"dataSourceTwoMaster,dataSourceTwoSlaveOne,dataSourceTwoSlaveTwo"*></entry>

</map>

</property>

</bean>

# 三、负载均衡示例

负责均衡，可参考Master-Slaves模式的实现。也可利用数据库驱动自带的功能实现，如Mysql的Connector-J，本身驱动就也实现了负载均衡的功能。

# 四、自定义数据源路由策略

通过扩展RoutingStrategy接口，实现route方法，来实现自定义的数据源路由策略。在route方法可根据DAO对象、方法、以及方法参数选择需要的数据源，将选择的数据源的key设置到RoutingHolder中。 在自定义扩展数据源路由策略时可参考：MasterSlaveStrategy，ModRoutingStrategy及ModMasterSlaveRoutingStrategy代码。route方法参数说明如下：

/\*\*

\* 执行此策略，选择对应的数据源，并将其key设置到RoutingHolder中，

\* 如果未设置，则采用默认数据源

\* **@param** target 代理的DAO对象

\* **@param** method DAO对象上执行的方法

\* **@param** args 方法执行所需的参数

\* \*/

**public** **void** route(Object target, Method method, Object[] args);

# 五、如何指定数据源

你可以通过RoutingHolder的setCurrentDataSourceKey(String Key)方法手动指定数据源。其中key为你注册到Routing4DBDataSource中的key，设置key为null，则采用默认的数据源。示例如下：

采用dataSourceOne数据源：

RoutingHolder.setCurrentDataSourceKey(“dataSourceOne”);

采用默认数据源：

RoutingHolder.setCurrentDataSourceKey(null);

# 六、事务的处理

Routing4DB采用接口代理的方式实现数据源的路由，可实现单数据源事务支持，不支持多数据源事务。事务处理方式和普通处理方式没有什么差别；如果需要事务支持，只需在对应接口**实现类**的方法上，只需加入@Transactional或AOP的方式实现。示例如下：

@Transactional()

**public** **void** insertWithTransaction(User user) {

String sql ="insert sql";

jdbcTemplate.execute(sql);

}

**注意：要把@Transactional放到实现类，而非接口上。原因是Routing4DB采用基于接口代理的方式实现数据源路由，若@Transactional放到接口上，Spring FactoryBean会在创建接口代理后，针对该代理创建事务处理拦截，事务处理在代理执行（路由策略执行）之前执行，导致未知异常。**

# 七、如何设置某些方法不执行路由

你可以配置某些方法不执行路由操作，只需配置对应路由策略中的excludeMethodPatterns属性，，设置不需要执行路由的方法规则。 示例配置如下。

<!-- 配置路由策略 -->

<bean id=*"masterSlaveStrategy"* class=*"com.google.code.routing4db.strategy.impl.MasterSlaveStrategy"*>

<!—- 其它配置省略 -->

<!-- 指定某些接口的方法不执行路由，若接口所有方法都执行路由，则无需配置该属性 -->

<property name=*"excludeMethodPatterns"*>

<list>

<value>excludeMethodPatterns\*</value>

<value>\*exclude\*</value>

</list>

</property>

</bean>

# 八、针对Mybatis的增强功能

Mybatis 3.0提供了基于工厂代理创建DAO的方式，您只需书写接口，无需实现类。Routing4DB针对Mybatis这一功能提供了增强; 你只需实现简单修改即可实现数据源的路由。具体参考：write-master-read-slaves-mybatis.xml及MybatisWriteMasterReadSlavesTest.java代码，示例如下：

原Mybatis配置：

<!-- (此配置方式不支持事务，详情见6事务说明) -->

<bean id=*"userMapper"* class=*"org.mybatis.spring.mapper.MapperFactoryBean"*>

<property name=*"mapperInterface"* value=*"com.google.code.routing4db.dao.UserMapper"* />

<property name=*"sqlSessionFactory"* ref=*"sqlSessionFactory"* />

</bean>

采用Routing4DB路由数据源的Mybatis配置：

<!-- (此配置方式不支持事务，详情见6事务说明) -->

<bean id=*"userMapper"* class=*"com.google.code.routing4db.mybatis.RoutingMapperFactoryBean"*>

<property name=*"mapperInterface"* value=*"com.google.code.routing4db.dao.UserMapper"* />

<property name=*"sqlSessionFactory"* ref=*"sqlSessionFactory"* />

<!-- 路由策略 -->

<property name=*"routingStrategy"* ref=*"masterSlaveStrategy"*></property>

</bean>

# 九、问题汇总

## 9.1 oracle下master-standby模式未生效

Routing4DB默认检查数据源是否有效执行sql为：select 1; 该语句在mysql下执行正常，但在oracle无法执行。因此你需要配置一条可以在oracle执行sql，比如 select 1 from dual。如果你采用的是其他数据库，只需配置一条在该数据库下执行的sql即可。示例如下：

<!-- 配置Master Standby 的数据源 -->

<bean id=*"masterStandbyDataSource"* class=*"com.google.code.routing4db.datasource.MasterStrandbyDataSource"*>

<property name=*"standbyDataSource"* ref=*"standbyDataSource"* />

<property name=*"masterDataSource"* ref=*"masterDataSource"*/>

<!-- 其它配置属性（可选） -->

<property name=*"configProperties"*>

<props>

<!-- 检查数据源的时间间隔，单位为毫秒，默认配置为10秒钟检查一次 -->

<prop key=*"checkTimeInterval"*>10000</prop>

<!-- 检查数据源是否有效时执行的sql，配置一条可执行的sql即可；请根据数据库不同进行配置。比如mysql下可配置为 select 1 而select 1 在oracle无法执行，oracle则可配置为 select 1 from dual -->

<prop key=*"checkAvailableSql"*>select 1</prop>

</props>

</property>

</bean>

# 十、BUG反馈及交流

你有BUG反馈，项目建议，或你想参与其中，改进Routing4DB项目。请与我联系，Routing4DB期待您的参与。

项目贡献者：无花

项目作者：谷宝剑

作者联系方式：

**email:** [efurture@gmail.com](mailto:efurture@gmail.com) 或 [gubaojian@163.com](mailto:gubaojian@163.com)

**QQ:**  787277208