Routing4DB 使用文档

一一 谷宝剑

目录

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
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
负载均衡示例	15
自定义数据源路由策略	
如何指定数据源	16
事务的处理	16
如何设置某些方法不执行路由	16
针对 Mybatis 的增强功能	17
问题汇总	
9.1 oracle 下 master-standby 模式未生效	17
	1.1 写 Master,读多个 Slaves,示意图如下: 1.2 写 Master,读 Master 和多个 Slave,示意图如下: 1.3 Master-Standby-Slaves 实现,此方式示意图如下: 分库功能 2.1 单机分库功能,示意图如下: 2.2 多机集群分库,构建分布式数据库,示意图如下: 2.3 高可用多级分布式集群,示意图如下: 负载均衡示例 自定义数据源路由策略 如何指定数据源 事务的处理 如何设置某些方法不执行路由 针对 Mybatis 的增强功能

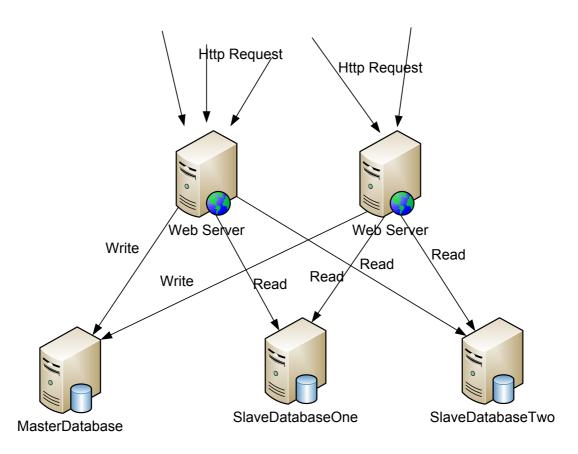
版本 1.1.0

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	1、增加获取数据源的路由 log 提示
		无花	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:



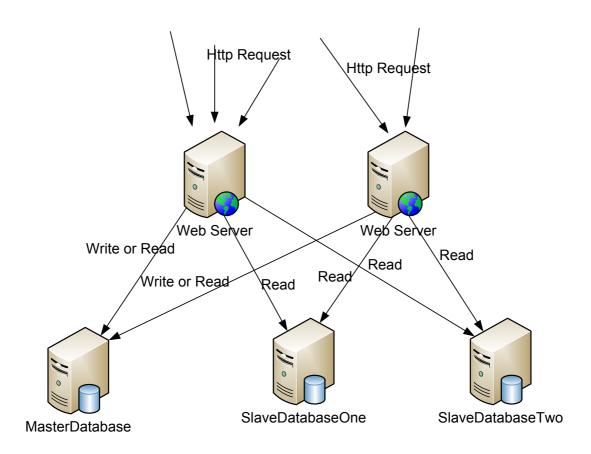
1.1 读 Master 写 Slave 示意图

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

```
cproperty name="maxStatements" value="0" />
</bean>
<!-- Slave数据源 -->
<bean id="slaveDataSourceOne"</pre>
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;characterEncodin
g=UTF-8&autoReconnect=true&failOverReadOnly=false</value>
      </property>
      property name="user" value="root" />
      property name="password" value="lovejava" />
      property name="initialPoolSize" value="2" />
      property name="minPoolSize" value="2" />
      cproperty name="maxPoolSize" value="10" />
      cproperty name="acquireIncrement" value="5" />
      cproperty name="maxIdleTime" value="30" />
      cproperty name="maxStatements" value="0" />
</bean>
<bean id="slaveDataSourceTwo"</pre>
class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">
      driverClass" value="com.mysql.jdbc.Driver"/>
      property name="jdbcUrl">
<value>jdbc:mysql://localhost:3306/test?useUnicode=true&amp;characterEncodin
g=UTF-8&autoReconnect=true&failOverReadOnly=false</value>
      </property>
      property name="user" value="root" />
      property name="password" value="lovejava" />
      property name="initialPoolSize" value="2" />
      cproperty name="minPoolSize" value="2" />
      cproperty name="maxPoolSize" value="10" />
      cproperty name="acquireIncrement" value="5" />
      property name="maxIdleTime" value="30" />
      property name="maxStatements" value="0" />
</bean>
<!-- 配置Routing4DB 数据源 -->
<bean id="routing4DBDataSource"</pre>
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>
  </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"</pre>
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
 cproperty name="dataSource" ref="routing4DBDataSource" />
</bean>
<tx:annotation-driven transaction-manager="transactionManager"/>
<!-- 配置路由策略 -->
<bean id="masterSlaveStrategy"</pre>
class="com.google.code.routing4db.strategy.impl.MasterSlaveStrategy">
   property name="readMethodPatterns">
      st>
         <value>*get*</value>
         <value>*find*</value>
      </list>
   </property>
   <!-- Master数据源标识符 -->
   <!-- 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"</pre>
```

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

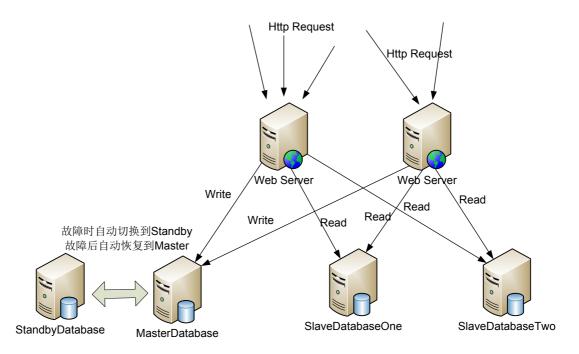
此方式配置和 1.1 方式相同,仅仅在路由策略 Slaves 节点中加入 Master 即可。不同的配置如下:

<!-- 配置路由策略 -->

<bean id="masterSlaveStrategy"</pre>

```
class="com.google.code.routing4db.strategy.impl.MasterSlaveStrategy">
  <!-- 读接口的方法通配符 -->
   property name="readMethodPatterns">
         <value>*get*</value>
         <value>*find*</value>
      </list>
   </property>
   <!-- Master数据源 -->
   <!-- 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 -->

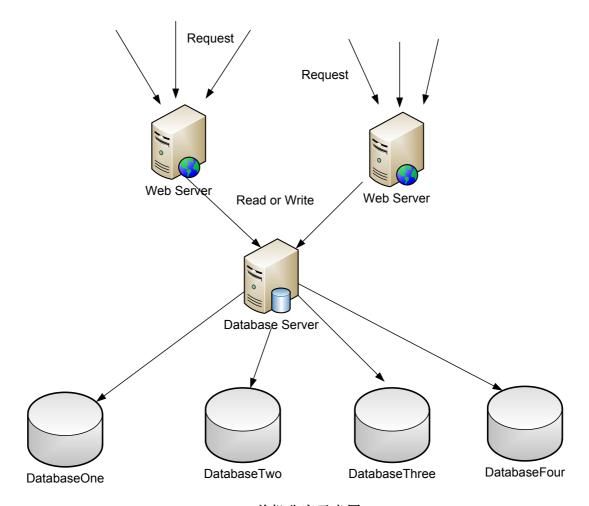
<bean id="masterDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"</pre>

```
destroy-method="close">
      driverClass" value="com.mysql.jdbc.Driver"/>
      property name="jdbcUrl">
<value>jdbc:mysql://192.168.56.102:3306/test?useUnicode=true&amp;characterEn
coding=UTF-8&autoReconnect=true&failOverReadOnly=false/value>
      </property>
      property name="user" value="root" />
      property name="password" value="lovejava" />
      property name="initialPoolSize" value="2" />
      property name="minPoolSize" value="2" />
      cproperty name="maxPoolSize" value="10" />
      cproperty name="acquireIncrement" value="5" />
      property name="maxIdleTime" value="30" />
      cproperty name="maxStatements" value="0" />
      property name="checkoutTimeout" value="2000"/>
      cproperty name="acquireRetryAttempts" value="3">
</bean>
<!-- Standby数据源 -->
<bean id="standbyDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"</pre>
destroy-method="close">
      driverClass" value="com.mysql.jdbc.Driver"/>
      property name="jdbcUrl">
<value>jdbc:mysql://localhost:3306/test?useUnicode=true&amp;characterEncodin
g=UTF-8&autoReconnect=true&failOverReadOnly=false</value>
      </property>
      property name="user" value="root" />
      property name="password" value="lovejava" />
      property name="initialPoolSize" value="2" />
      cproperty name="minPoolSize" value="2" />
      cproperty name="maxPoolSize" value="10" />
      cproperty name="acquireIncrement" value="5" />
      cproperty name="maxIdleTime" value="30" />
      property name="maxStatements" value="0" />
      property name="checkoutTimeout" value="2000"/>
      </bean>
<!-- 配置Master Standby 的数据源 -->
<bean id="masterStandbyDataSource"</pre>
class="com.google.code.routing4db.datasource.MasterStrandbyDataSource">
  property name="standbyDataSource" ref="standbyDataSource" />
```

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

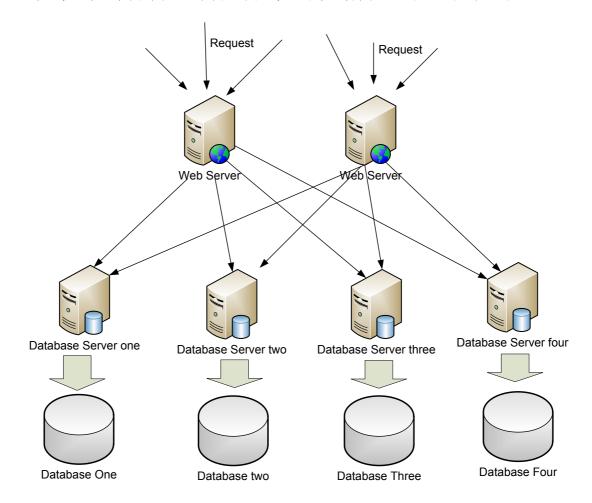
```
<bean id="dataSourceOne" class="com.mchange.v2.c3p0.ComboPooledDataSource"</pre>
destroy-method="close">
      property name="driverClass" value="com.mysql.jdbc.Driver"/>
      property name="jdbcUrl">
<value>jdbc:mysql://localhost:3306/test0?useUnicode=true&amp;characterEncodi
ng=UTF-8&autoReconnect=true&failOverReadOnly=false</value>
      </property>
      property name="user" value="root" />
      property name="password" value="lovejava" />
      property name="initialPoolSize" value="2" />
      cproperty name="minPoolSize" value="2" />
      cproperty name="maxPoolSize" value="10" />
      property name="acquireIncrement" value="5" />
      property name="maxIdleTime" value="30" />
      cproperty name="maxStatements" value="0" />
</bean>
<!-- mode two 数据源 -->
<bean id="dataSourceTwo" class="com.mchange.v2.c3p0.ComboPooledDataSource"</pre>
destroy-method="close">
      cproperty name="driverClass" value="com.mysql.jdbc.Driver"/>
      property name="jdbcUrl">
<value>jdbc:mysql://localhost:3306/test1?useUnicode=true&amp;characterEncodi
ng=UTF-8&autoReconnect=true&failOverReadOnly=false</value>
      </property>
      property name="user" value="root" />
      property name="password" value="lovejava" />
      property name="initialPoolSize" value="2" />
      cproperty name="minPoolSize" value="2" />
      cproperty name="maxPoolSize" value="10" />
      property name="acquireIncrement" value="5" />
      cproperty name="maxIdleTime" value="30" />
      property name="maxStatements" value="0" />
</bean>
<!-- mode three 数据源 -->
<bean id="dataSourceThree" class="com.mchange.v2.c3p0.ComboPooledDataSource"</pre>
destroy-method="close">
      cproperty name="driverClass" value="com.mysql.jdbc.Driver"/>
      property name="jdbcUrl">
<value>jdbc:mysql://localhost:3306/test2?useUnicode=true&amp;characterEncodi
ng=UTF-8&autoReconnect=true&failOverReadOnly=false</value>
      </property>
      property name="user" value="root" />
```

<!-- mode one 数据源 -->

```
property name="password" value="lovejava" />
      property name="initialPoolSize" value="2" />
      property name="minPoolSize" value="2" />
      cproperty name="maxPoolSize" value="10" />
      property name="acquireIncrement" value="5" />
      cproperty name="maxIdleTime" value="30" />
      cproperty name="maxStatements" value="0" />
</bean>
<!-- mode four 数据源 -->
<bean id="dataSourceFour" class="com.mchange.v2.c3p0.ComboPooledDataSource"</pre>
destroy-method="close">
      cproperty name="driverClass" value="com.mysql.jdbc.Driver"/>
      property name="jdbcUrl">
<value>jdbc:mysql://localhost:3306/test3?useUnicode=true&amp;characterEncodi
ng=UTF-8&autoReconnect=true&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" />
      cproperty name="maxStatements" value="0" />
</bean>
<!-- 配置Routing4DB 数据源 -->
<bean id="routing4DBDataSource"</pre>
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"</pre>
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
 cproperty name="dataSource" ref="routing4DBDataSource" />
</bean>
<tx:annotation-driven transaction-manager="transactionManager"/>
<!-- 配置路由策略 -->
<bean id="modRoutingStrategy"</pre>
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 -->
   cproperty name="dataSourceNum" value="4"></property>
    cproperty 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"</pre>
class="com.google.code.routing4db.dao.UserDaoJdbcTemplateImpl"></bean>
<!-- 配置DAO接口代理 ) -->
<bean id="userDao"</pre>
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 -->
   cproperty name="targetObject" ref="userDaoTarget">
   <!-- 路由策略 -->
```

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

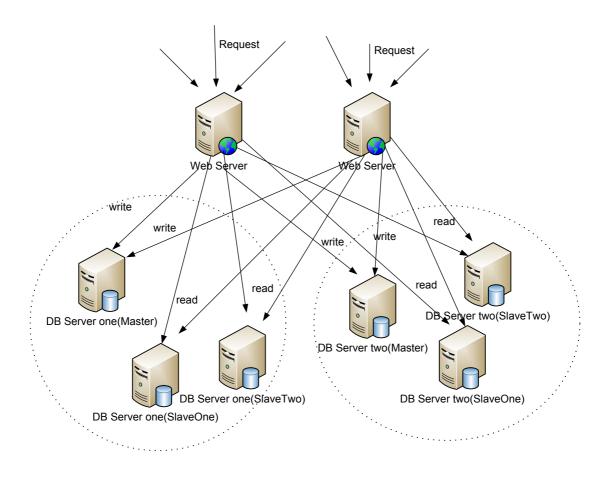
此方式与单机分库没什么区别,原理及配置方式相同,唯一区别就是把数据源的IP修改一下, 其配置可参考2.1单机版分库,参考测试工程中的: ShardDatabaseByModTest.java及 shard-database-by-mod.xml实例。配置如下,:

```
property name="acquireIncrement" value="5" />
      property name="maxIdleTime" value="30" />
      cproperty name="maxStatements" value="0" />
</bean>
<!-- mode two 数据源 -->
<bean id="dataSourceTwo" class="com.mchange.v2.c3p0.ComboPooledDataSource"</pre>
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;characterEnc
oding=UTF-8&autoReconnect=true&failOverReadOnly=false</value>
      </property>
      property name="user" value="root" />
      property name="password" value="lovejava" />
      property name="initialPoolSize" value="2" />
      cproperty name="minPoolSize" value="2" />
      cproperty name="maxPoolSize" value="10" />
      property name="acquireIncrement" value="5" />
      cproperty name="maxIdleTime" value="30" />
      cproperty name="maxStatements" value="0" />
</bean>
<!-- mode three 数据源 -->
<bean id="dataSourceThree" class="com.mchange.v2.c3p0.ComboPooledDataSource"</pre>
destroy-method="close">
      cproperty name="driverClass" value="com.mysql.jdbc.Driver"/>
      property name="jdbcUrl">
<value>jdbc:mysql://192.168.3.27:3306/test2?useUnicode=true&amp;characterEnc
oding=UTF-8&autoReconnect=true&failOverReadOnly=false</value>
      </property>
      property name="user" value="root" />
      property name="password" value="lovejava" />
      property name="initialPoolSize" value="2" />
      cproperty name="minPoolSize" value="2" />
      property name="maxPoolSize" value="10" />
      cproperty 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"</pre>
destroy-method="close">
      cproperty name="driverClass" value="com.mysql.jdbc.Driver"/>
```

```
property name="jdbcUrl">
```

 $\label{lem:code} $$\sup_j dc: mysql: //192.168.3.28: 3306/test3 ? useUnicode=true& characterEnc oding=UTF-8& autoReconnect=true& failOverReadOnly=false</value>$

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"</pre>
value-ref="dataSourceOneSlaveOne"/>
        <entry key="dataSourceOneSlaveTwo"</pre>
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 -->
  </bean>
<!-- 配置路由策略 -->
<bean id="modMasterSlaveRoutingStrategy"</pre>
class="com.google.code.routing4db.strategy.impl.ModMasterSlaveRoutingStrateg
v">
   <!-- 分2集群组,采用Id属性进行分库路由 -->
   cproperty name="dataSourceNum" value="2">
    cproperty 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"</pre>
value="dataSourceOneMaster,dataSourceOneSlaveOne,dataSourceOneSlaveTwo"></en</pre>
try>
        <entry key="1"</pre>
value="dataSourceTwoMaster,dataSourceTwoSlaveOne,dataSourceTwoSlaveTwo"></en
try>
       </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 -->
```

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"</pre>
 class="org.mybatis.spring.mapper.MapperFactoryBean">
   property name="mapperInterface"
value="com.google.code.routing4db.dao.UserMapper" />
   cproperty 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"</pre>
class="com.google.code.routing4db.mybatis.RoutingMapperFactoryBean">
   property name="mapperInterface"
value="com.google.code.routing4db.dao.UserMapper" />
   cproperty name="sqlSessionFactory" ref="sqlSessionFactory" />
    <!-- 路由策略 -->
    property name="routingStrategy"
ref="masterSlaveStrategy">
</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">
```

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 或 gubaojian@163.com

QQ: 787277208