# 3 Sharding-JDBC读写分离

## 3.1 不使用Spring

### 引入Maven依赖

```
<dependency>
    <groupId>org.apache.shardingsphere</groupId>
    <artifactId>sharding-jdbc-core</artifactId>
    <version>${sharding-sphere.version}</version>
</dependency>
```

### 基于Java编码的规则配置

```
    // 配置真实数据源
    Map<String, DataSource> dataSourceMap = new HashMap<>();

    // 配置主库
    BasicDataSource masterDataSource = new BasicDataSource();
    masterDataSource.setDriverClassName("com.mysql.jdbc.Driver");
    masterDataSource.setUrl("jdbc:mysql://localhost:3306/ds_master");
    masterDataSource.setUsername("root");
    masterDataSource.setPassword("");
    dataSourceMap.put("ds_master", masterDataSource);

    // 配置第一个从库
    BasicDataSource slaveDataSource1 = new BasicDataSource();
    slaveDataSource1.setDriverClassName("com.mysql.jdbc.Driver");
    slaveDataSource1.setUrl("jdbc:mysql://localhost:3306/ds_slave0");
    slaveDataSource1.setUsername("root");
    slaveDataSource1.setPassword("");
    dataSourceMap.put("ds_slave0", slaveDataSource1);

    // 配置第二个从库
    BasicDataSource slaveDataSource2 = new BasicDataSource();
    slaveDataSource2.setDriverClassName("com.mysql.jdbc.Driver");
    slaveDataSource2.setUrl("jdbc:mysql://localhost:3306/ds_slave1");
```

```
    slaveDataSource2.setUsername("root");
    slaveDataSource2.setPassword("");
    dataSourceMap.put("ds_slave1", slaveDataSource2);

    // 配置读写分离规则
    MasterSlaveRuleConfiguration masterSlaveRuleConfig = new
MasterSlaveRuleConfiguration("ds_master_slave", "ds_master",
Arrays.asList("ds_slave0", "ds_slave1"));

    // 获取数据源对象
    DataSource dataSource =
MasterSlaveDataSourceFactory.createDataSource(createDataSourceMap(),
masterSlaveRuleConfig, new HashMap<String, Object>(), new Properties());
```

## 基于Yaml的规则配置

或通过Yaml方式配置，与以上配置等价：

```
dataSources:
  ds_master: !!org.apache.commons.dbcp.BasicDataSource
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/ds_master
    username: root
    password:
  ds_slave0: !!org.apache.commons.dbcp.BasicDataSource
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/ds_slave0
    username: root
    password:
  ds_slave1: !!org.apache.commons.dbcp.BasicDataSource
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/ds_slave1
    username: root
    password:

masterSlaveRule:
  name: ds_ms
  masterDataSourceName: ds_master
  slaveDataSourceNames: [ds_slave0, ds_slave1]

  props:
```

```
      sql.show: true
  configMap:
      key1: value1
    DataSource dataSource =
MasterSlaveDataSourceFactory.createDataSource(yamlFile);
```

## 使用原生JDBC

通过MasterSlaveDataSourceFactory工厂和规则配置对象获取MasterSlaveDataSource，MasterSlaveDataSource实现自JDBC的标准接口DataSource。然后可通过DataSource选择使用原生JDBC开发，或者使用JPA, MyBatis等ORM工具。 以JDBC原生实现为例：

```
DataSource dataSource =
MasterSlaveDataSourceFactory.createDataSource(yamlFile);
String sql = "SELECT i.* FROM t_order o JOIN t_order_item i ON
o.order_id=i.order_id WHERE o.user_id=? AND o.order_id=?";
try (
        Connection conn = dataSource.getConnection();
        PreparedStatement preparedStatement = conn.prepareStatement(sql))
{
    preparedStatement.setInt(1, 10);
    preparedStatement.setInt(2, 1001);
    try (ResultSet rs = preparedStatement.executeQuery()) {
        while(rs.next()) {
            System.out.println(rs.getInt(1));
            System.out.println(rs.getInt(2));
        }
    }
}
```

# 3.2 使用Spring

## 引入Maven依赖

```
<!-- for spring boot -->
<dependency>
    <groupId>io.shardingsphere</groupId>
    <artifactId>sharding-jdbc-spring-boot-starter</artifactId>
    <version>${sharding-sphere.version}</version>
</dependency>

<!-- for spring namespace -->
<dependency>
    <groupId>io.shardingsphere</groupId>
    <artifactId>sharding-jdbc-spring-namespace</artifactId>
    <version>${sharding-sphere.version}</version>
</dependency>
```

## 基于Spring boot的规则配置

```
sharding.jdbc.datasource.names=master,slave0,slave1

sharding.jdbc.datasource.master.type=org.apache.commons.dbcp.BasicDataSour
ce
sharding.jdbc.datasource.master.driver-class-name=com.mysql.jdbc.Driver
sharding.jdbc.datasource.master.url=jdbc:mysql://localhost:3306/master
sharding.jdbc.datasource.master.username=root
sharding.jdbc.datasource.master.password=

sharding.jdbc.datasource.slave0.type=org.apache.commons.dbcp.BasicDataSour
ce
sharding.jdbc.datasource.slave0.driver-class-name=com.mysql.jdbc.Driver
sharding.jdbc.datasource.slave0.url=jdbc:mysql://localhost:3306/slave0
sharding.jdbc.datasource.slave0.username=root
sharding.jdbc.datasource.slave0.password=

sharding.jdbc.datasource.slave1.type=org.apache.commons.dbcp.BasicDataSour
ce
sharding.jdbc.datasource.slave1.driver-class-name=com.mysql.jdbc.Driver
sharding.jdbc.datasource.slave1.url=jdbc:mysql://localhost:3306/slave1
sharding.jdbc.datasource.slave1.username=root
sharding.jdbc.datasource.slave1.password=

sharding.jdbc.config.masterslave.name=ms
sharding.jdbc.config.masterslave.master-data-source-name=master
```

```
sharding.jdbc.config.masterslave.slave-data-source-names=slave0,slave1

sharding.jdbc.config.props.sql.show=true
```

## 基于Spring命名空间的规则配置

```
<?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:master-
slave="http://shardingsphere.io/schema/shardingsphere/masterslave"
    xsi:schemaLocation="http://www.springframework.org/schema/beans

 http://www.springframework.org/schema/beans/spring-beans.xsd

 http://shardingsphere.io/schema/shardingsphere/masterslave

 http://shardingsphere.io/schema/shardingsphere/masterslave/master-
slave.xsd
                        ">
    <bean id="ds_master" class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/ds_master"
/>
        <property name="username" value="root" />
        <property name="password" value="" />
    </bean>
    <bean id="ds_slave0" class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/ds_slave0"
/>
        <property name="username" value="root" />
        <property name="password" value="" />
    </bean>
    <bean id="ds_slave1" class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/ds_slave1"
/>
```

```
        <property name="username" value="root" />
        <property name="password" value="" />
    </bean>

    <master-slave:data-source id="masterSlaveDataSource" master-data-
source-name="ds_master" slave-data-source-names="ds_slave0, ds_slave1" >
        <master-slave:props>
                <prop key="sql.show">${sql_show}</prop>
                <prop key="executor.size">10</prop>
                <prop key="foo">bar</prop>
            </master-slave:props>
    </master-slave:data-source>
</beans>
```

## 在Spring中使用DataSource

直接通过注入的方式即可使用DataSource，或者将DataSource配置在JPA、Hibernate或MyBatis中使用。

```
@Resource
private DataSource dataSource;
```

# 3.3 配置项说明

读写分离

```
dataSources: #省略数据源配置，与数据分片一致

masterSlaveRule:
  name: #读写分离数据源名称，自定义
  masterDataSourceName: #主库数据源名称，数据源处定义的数据源名
  slaveDataSourceNames: #从库数据源名称列表，数据源处定义的数据源名
    - <data_source_name1>
    - <data_source_name2>
    - <data_source_name_x>
  loadBalanceAlgorithmClassName: #从库负载均衡算法类名称。该类需实现
MasterSlaveLoadBalanceAlgorithm接口且提供无参数构造器
  loadBalanceAlgorithmType: #从库负载均衡算法类型，可选值：ROUND_ROBIN，
RANDOM。若`loadBalanceAlgorithmClassName`存在则忽略该配置
```

```yaml
props: #属性配置
  sql.show: #是否开启SQL显示，默认值: false
  executor.size: #工作线程数量，默认值: CPU核数
  check.table.metadata.enabled: #是否在启动时检查分表元数据一致性，默认值: false

configMap: #用户自定义配置
  key1: value1
  key2: value2
  keyx: valuex
```