

第一步、配置好 mysql 主从复制

第二步、在 spring 配置文件 application 中设置从库

The screenshot shows the IntelliJ IDEA interface with the file `application-dev.yml` open. The configuration section for the `druid` data source is highlighted:

```
master:
  type: com.alibaba.druid.pool.DruidDataSource
  driverClassName: com.mysql.cj.jdbc.Driver
  druid:
    # 主数据库
    master:
      # nullCatalogMeansCurrent=true 解决mysql-connector-mysql 8.0 (spring-boot-starter-parent 管理的版本) + Activiti 6.x 启动报错#
      url: jdbc:mysql://localhost:3306/ahrs_master_slave?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&useSSL=false&serverTimezone=Asia/Shanghai
      username: root
      password: root@123
    # 从库数据源
    slave:
      # 从数据源开关/默认关闭
      enabled: false
      url:
      username:
      password:
    # 主从复制读写分离从数据库
    slave:
      enabled: true
      url: jdbc:mysql://172.16.66.40:3306/ahrs_master_slave?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&useSSL=true&serverTimezone=GMT+8
      username: root
      password: root@123
    slaves:
      enabled: true
      url: jdbc:mysql://172.16.66.40:3306/ahrs_master_slave?useUnicode=true&characterEncoding=utf8&zeroDateTimeBehavior=convertToNull&useSSL=true&serverTimezone=GMT+8
      username: root
      password: root@123
  # 初始连接数
  initialSize: 5
  # 最小连接数
  minIdle: 10
```

Annotations in the code:

- A red arrow points to the `url` field of the `master` section with the note: "主数据库名应保持一致".
- A red arrow points to the `url` field of the first `slave` section with the note: "数据库地址实际应在不同服务器上，此处是为了测试方便".

第三步、注释多数据源配置文件

The screenshot shows the IntelliJ IDEA interface with the file `DruidConfig.java` open. A large red box highlights the commented-out code for the `DynamicDataSource`:

```
/*
 * @Configuration
 * public class DruidConfig {
 *
 *     // @Bean
 *     // @ConfigurationProperties("spring.datasource.druid.master")
 *     // public DataSource masterDataSource(DataSourceProperties druidProperties) {
 *     //     DruidDataSource dataSource = DruidDataSourceBuilder.create().build();
 *     //     return druidProperties.dataSource(dataSource);
 *     // }
 *
 *     // @Bean
 *     // @ConfigurationProperties("spring.datasource.druid.slave")
 *     // @ConditionalOnProperty(prefix = "spring.datasource.druid.slave", name = "enabled", havingValue = "true")
 *     // public DataSource slaveDataSource(DataSourceProperties druidProperties) {
 *     //     DruidDataSource dataSource = DruidDataSourceBuilder.create().build();
 *     //     return druidProperties.dataSource(dataSource);
 *     // }
 *
 *     // @Bean(name = "dynamicDataSource")
 *     // @Primary
 *     // public DynamicDataSource dataSource(DataSource masterDataSource, DataSource slaveDataSource) {
 *     //     Map<Object, Object> targetDataSources = new HashMap<>();
 *     //     targetDataSources.put(DataSourceType.MASTER.name(), masterDataSource);
 *     //     targetDataSources.put(DataSourceType.SLAVE.name(), slaveDataSource);
 *     //     return new DynamicDataSource(masterDataSource, targetDataSources);
 *     // }
 * }
 */
/* 去除监控页面底部的广告 */
```

第四步、放开 ShardingDataSourceConfig 配置文件注释并修改对应从库信息

```
1  package com.ruoyi.framework.config;
2
3  import com.alibaba.druid.spring.boot.autoconfigure.DruidProperties;
4  import org.springframework.context.annotation.Bean;
5  import org.springframework.context.annotation.Configuration;
6  import org.springframework.context.annotation.Primary;
7  import org.springframework.jdbc.datasource.DataSourceBuilder;
8
9  /**
10  * 分库分表配置
11  */
12 @Configuration
13 public class ShardingDataSourceConfig {
14
15     @Bean
16     @ConfigurationProperties("spring.datasource.druid.master")
17     public DataSource masterDataSource(DruidProperties druidProperties) {
18         DruidDataSource dataSource = DruidDataSourceBuilder.create().build();
19         return druidProperties.dataSource(dataSource);
20     }
21
22     @Bean
23     @ConfigurationProperties("spring.datasource.druid.slave1")
24     @ConditionalOnProperty(prefix = "spring.datasource.druid.slave1", name = "enabled", havingValue = "true")
25     public DataSource slave1DataSource(DruidProperties druidProperties) {
26
27         DruidDataSource dataSource = DruidDataSourceBuilder.create().build();
28         return druidProperties.dataSource(dataSource);
29     }
30
31     @Bean
32     @ConfigurationProperties("spring.datasource.druid.slave2")
33     @ConditionalOnProperty(prefix = "spring.datasource.druid.slave2", name = "enabled", havingValue = "true")
34     public DataSource slave2DataSource(DruidProperties druidProperties) {
35
36         DruidDataSource dataSource = DruidDataSourceBuilder.create().build();
37         return druidProperties.dataSource(dataSource);
38     }
39
40     @Bean(name = "shardingDataSource")
41     @Primary
42     public DataSource shardingDataSource(@Qualifier("masterDataSource") DataSource masterDataSource, @Qualifier("slave1DataSource") Data
43 ShardingDataSourceConfig slave1DataSource,
```

The screenshot shows two instances of an IDE interface. Both instances display the same Java code for a `ShardingDataSourceConfig` class. The code defines three beans: `masterDataSource`, `slave1DataSource`, and `slave2DataSource`. The `slave1DataSource` bean is highlighted with a red rectangular selection. The code uses `DruidProperties` from `com.alibaba.druid.spring.boot.autoconfigure` and `DataSourceBuilder` from `org.springframework.jdbc.datasource`. The `shardingDataSource` bean is annotated with `@Primary`. The IDE interface includes toolbars, a project tree, and a terminal window at the bottom.

```
61     "e = "shardingDataSource"
62     taSource shardingDataSource(@Qualifier("masterDataSource") DataSource masterDataSource, @Qualifier("slave1DataSource") DataSource slave1DataSource, @Qualifier("slave2DataSource") DataSource slave2DataSource)
63
64     Map<String, DataSource> dataSourceMap = new HashMap<>();
65     dataSourceMap.put("order1", masterDataSource);
66     dataSourceMap.put("order2", slave1DataSource);
67     dataSourceMap.put("order3", slave2DataSource);
68
69     sys_order 表规则配置
70     TableRuleConfiguration orderTableRuleConfig = new TableRuleConfiguration("sys_order", "order$->(1..2).sys_order_$->(0..1)");
71     配置分片策略
72     derTableRuleConfig.setDatabaseShardingStrategyConfig(new InlineShardingStrategyConfiguration("user_id", "order$->(user_id % 2 + 1)"));
73     配置分表策略
74     derTableRuleConfig.setTableShardingStrategyConfig(new InlineShardingStrategyConfiguration("order_id", "sys_order_$->(order_id % 2)"));
75     分布式ID
76     derTableRuleConfig.setKeyGeneratorConfig(new KeyGeneratorConfiguration("SNOWFLAKE", "order_id"));
77
78     配置分片规则
79     ardingRuleConfiguration shardingRuleConfig = new ShardingRuleConfiguration();
80     ardingRuleConfig.getTableRuleConfigs().add(orderTableRuleConfig);
81
82     dataSource dataSource = ShardingDataSourceFactory.createDataSource(dataSourceMap, shardingRuleConfig, getProperties());
83
84     dataSourceMap.put("master", masterDataSource);
85     dataSourceMap.put("slave1", slave1DataSource);
86     dataSourceMap.put("slave2", slave2DataSource);
87     rSlaveRuleConfiguration masterSlaveRuleConfig = new MasterSlaveRuleConfiguration( name: "master_slave", masterDataSourceName: "master", Arrays.asList("slave1", "slave2"));
88
89     联数据源对象
90     source dataSource = MasterSlaveDataSourceFactory.createDataSource(dataSourceMap, masterSlaveRuleConfig, getProperties());
91
92     return dataSource;
93 }
```

Debug: RuoyiApplication

2022-04-12 16:53:38.724 [activiti-acquire-timer-jobs] INFO o.a.e.i.a.AcquireTimerJobsRunnable - [run_115] - {} stopped async job due acquisition
2022-04-12 16:53:38.730 [thread-35] INFO c.a.d.p.DruidDataSource - [close_1928] - [dataSource-1] closed

Process finished with exit code 130

第五步、启动后确认成功

```
37
38     slave:
39         # 从数据库开关默认关闭
40         enabled: false
41         url: "mhz, 2020/6/11 16:42 + employee done"
42         username:
43         password:
44
45     # 主从复制模式分离从数据库
46     slave:
47         enabled: true
48         url: "mhz, 2020/6/11 16:42 + employee done"
49         username:
50         password:
```

Document 1/1 spring: datasource | druid: slave | url

Debug: RuoyiApplication

2022-04-12 16:53:38.724 [activiti-acquire-timer-jobs] INFO ShardingSphere-SQL - [log,74] - Logic SQL: SELECT COUNT(TRIGGER_NAME) FROM QRTZ_TRIGGERS WHERE SCHED_NAME = 'RuoyiScheduler-flh1649755381817_MisfireHandler' INFO ShardingSphere-SQL - [log,74] - SQLStatement: SelectStatementContext(super=CommonSQLStatementContext(sqlStatementContext))

2022-04-12 16:53:38.724 [activiti-acquire-timer-jobs] INFO ShardingSphere-SQL - [log,74] - Actual SQL: slave1::: SELECT COUNT(TRIGGER_NAME) FROM QRTZ_TRIGGERS WHERE SCHED_NAME = 'RuoyiScheduler-flh1649755381817_MisfireHandler' INFO ShardingSphere-SQL - [log,74] - SQLStatement: SelectStatementContext(super=CommonSQLStatementContext(sqlStatementContext))

2022-04-12 16:53:38.724 [activiti-acquire-timer-jobs] INFO ShardingSphere-SQL - [log,74] - Logic SQL: SELECT * FROM QRTZ_SCHEDULER_STATE WHERE SCHED_NAME = 'RuoyiScheduler-flh1649755381817_ClusterManager' INFO ShardingSphere-SQL - [log,74] - SQLStatement: SelectStatementContext(super=CommonSQLStatementContext(sqlStatementContext))

2022-04-12 16:53:38.724 [activiti-acquire-timer-jobs] INFO ShardingSphere-SQL - [log,74] - Actual SQL: slave2::: SELECT * FROM QRTZ_SCHEDULER_STATE WHERE SCHED_NAME = 'RuoyiScheduler-flh1649755381817_ClusterManager' INFO ShardingSphere-SQL - [log,74] - SQLStatement: SelectStatementContext(super=CommonSQLStatementContext(sqlStatementContext))

2022-04-12 16:53:38.724 [activiti-acquire-timer-jobs] INFO ShardingSphere-SQL - [log,74] - Logic SQL: UPDATE QRTZ_SCHEDULER_STATE SET LAST_CHECKIN_TIME = ? WHERE SCHED_NAME = 'RuoyiScheduler-flh1649755381817_ClusterManager' INFO ShardingSphere-SQL - [log,74] - SQLStatement: UpdateStatementContext(super=CommonSQLStatementContext(sqlStatementContext))

2022-04-12 16:53:38.724 [activiti-acquire-timer-jobs] INFO ShardingSphere-SQL - [log,74] - Actual SQL: master::: UPDATE QRTZ_SCHEDULER_STATE SET LAST_CHECKIN_TIME = ? WHERE SCHED_NAME = 'RuoyiScheduler-flh1649755381817_ClusterManager' INFO ShardingSphere-SQL - [log,74] - Logic SQL: select

2022-04-12 16:53:38.724 [activiti-acquire-timer-jobs] INFO ShardingSphere-SQL - [log,74] - Logic SQL: select

2022-04-12 16:53:38.724 [activiti-acquire-timer-jobs] INFO ShardingSphere-SQL - [log,74] - SQLStatement: CommonSQLStatementContext(sqlStatement=org.apache.shardingsphere.sql.parser.sql.statement.dml.SelectStatement@7d4333c)

2022-04-12 16:53:38.724 [activiti-acquire-timer-jobs] INFO ShardingSphere-SQL - [log,74] - Actual SQL: slave2 :::: select

All files are up-to-date (memory: 400)