链滴

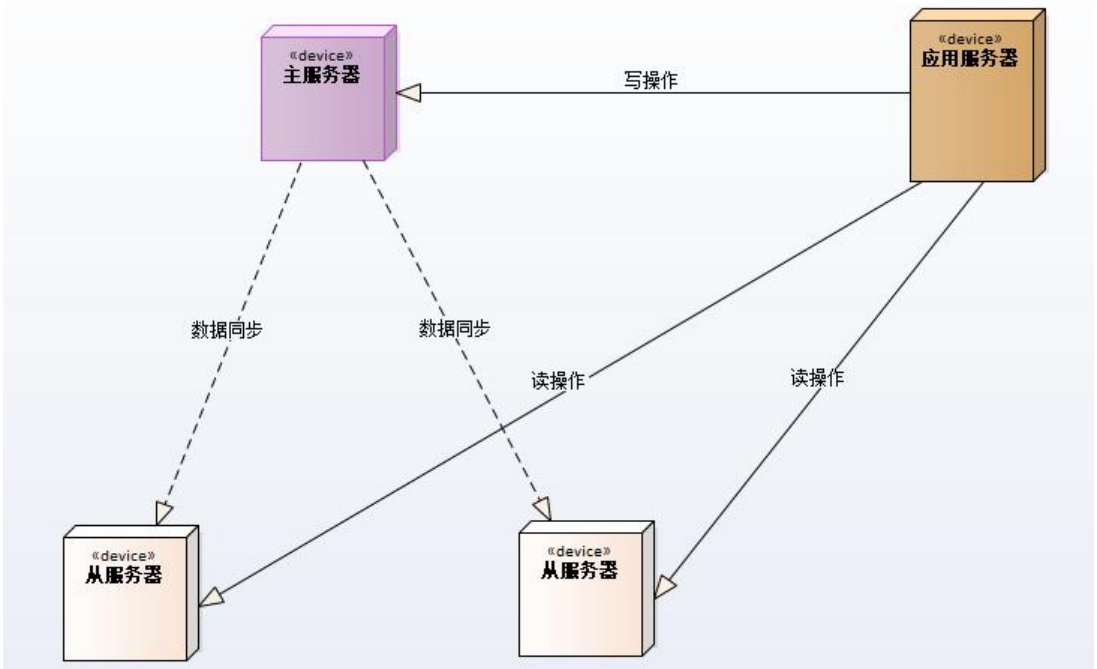
Springboot + Mysql8 实现读写分离

作者：jianzh5

原文链接：https://ld246.com/article/1574082736016

来源网站：链滴

在实际的生产环境中，为了确保数据库的稳定性，我们一般会给数据库配置双机热备机制，这样在master数据库崩溃后，slave数据库可以立即切换成主数据库，通过主从复制的方式将数据从主库同步至从，在业务代码中编写代码实现读写分离（让主数据库处理 事务性增、改、删操作，而从数据库处理查操作）来提升数据库的并发负载能力。



下面我们使用最新版本的Mysql数据库（8.0.16）结合SpringBoot实现这一完整步骤（一主一从）。

安装配置mysql

- 从 https://dev.mysql.com/downloads/mysql/页面下载mysql安装包，我这里下载的是mysql8.0.16 Linux-Generic.

- 准备两台虚拟机用作安装mysql，并将下载后的文件mysql-8.0.16-linux-glibc2.12-x86_64.tar.xz传至服务器/app/mysql
  - 192.168.249.131 CENTOS7 主
  - 192.168.249.129 CENTOS7 从
- 查看防火墙状态，如果启动需要先关闭防火墙

```
service firewalld status  ## 查看防火墙状态
service firewalld stop   ## 关闭防火墙
```

- 使用如下命令将xz文件解压成tar文件

```
xz -d mysql-8.0.16-linux-glibc2.12-x86_64.tar.xz
```

- 解压安装包

```
tar -xvf mysql-8.0.16-linux-gl-ibc2.12-x86_64.tar
```

- 在/app/mysql下建立data文件夹，用于存放数据
- 创建mysql用户组和mysql用户

```
groupadd mysql                          ## 创建用户组
useradd -g mysql -d /app/mysql mysql    ## 在用户组下创建mysql用户并授权相关目录
groupdel mysql                          ## 删除用户组名（若报已存在相关用户组）
userdel mysql                           ## 删除用户（若报已存在相关用户）
```

- 初始化安装mysql数据库

```
./mysql-8.0.16-linux-glibc2.12-x86_64/bin/mysqld --user=mysql --basedir=/app/mysql --dat
dir=/app/mysql/data --initialize

2019-07-01T02:05:52.681626Z 0 [Warning] [MY-011070] [Server] 'Disabling symbolic links usi
g --skip-symbolic-links (or equivalent) is the default. Consider not using this option as it' is d
precated and will be removed in a future release.
2019-07-01T02:05:52.681694Z 0 [System] [MY-013169] [Server] /app/mysql/mysql-8.0.16-linu
-glibc2.12-x86_64/bin/mysqld (mysqld 8.0.16) initializing of server in progress as process 147

2019-07-01T02:05:52.681726Z 0 [ERROR] [MY-010338] [Server] Can't find error-message file '
app/mysql/share/errmsg.sys'. Check error-message file location and 'lc-messages-dir' configu
ation directive.
2019-07-01T02:05:55.713747Z 5 [Note] [MY-010454] [Server] A temporary password is genera
ed for root@localhost: xa6(H>rK/r<E
2019-07-01T02:05:57.303240Z 0 [System] [MY-013170] [Server] /app/mysql/mysql-8.0.16-linu
-glibc2.12-x86_64/bin/mysqld (mysqld 8.0.16) initializing of server has completed
```

注意，此时mysql会生成一个默认的临时密码，如上图所示，需要先保存下来然后修改

- 建立mysql服务并增加执行权限

```
cp mysql-8.0.16-linux-glibc2.12-x86_64/support-files/mysql.server /etc/init.d/mysqld
```

- 修改mysql配置文件 vi /etc/my.cnf 增加如下配置

```
[mysqld]
port=3306
basedir=/app/mysql/mysql-8.0.16-linux-glibc2.12-x86_64
```

```
datadir=/app/mysql/data
socket=/tmp/mysql.sock
symbolic-links=0

[mysqld_safe]
log-error=/app/mysql/data/log/error.log
pid-file=/app/mysql/data/mysql.pid
user=mysql
tmpdir=/tmp
character_set_server=utf8
default-storage-engine=INNODB
init_connect='SET NAMES utf8'

!includedir /etc/my.cnf.d
```

如果报日志权限相关错误，请先建立对应日志文件，并给mysql用户授权

```
chown -R mysql:mysql /app/mysql/data/log/error.log
```

- 启动mysql服务

```
service mysqld start
```

- 建立mysql客户端软连接

```
 ln -s /app/mysql/mysql-8.0.16-linux-glibc2.12-x86_64/bin/mysql /usr/local/bin/mysql
```

- 登录mysql修改密码

```
mysql -uroot -p密码       ## 登录
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '000000';
```

- 设置远程登录

```
use mysql;
update user set host='%' where user='root' limit 1;
flush privileges;
```

配置mysql主从同步（binlog）

复制原理

- Master将数据改变记录到二进制日志(binary log)中，也就是配置文件log-bin指定的文件，这些记叫做二进制日志事件(binary log events)
- Slave通过I/O线程读取Master中的binary log events并写入到它的中继日志(relay log)
- Slave重做中继日志中的事件，把中继日志中的事件信息一条一条的在本地执行一次，完成数据在地的存储，从而实现将改变反映到它自己的数据(数据重放)

复制要求

- 主从服务器操作系统版本和位数一致
- Master和Slave数据库的版本要一致

- Master和Slave数据库中的数据要一致
- Master开启二进制日志，Master和Slave的server_id在局域网内必须唯一

配置步骤

主数据库（192.168.249.131）

- 创建同步用户并授权

```
CREATE USER 'slave'@'192.168.249.129' IDENTIFIED WITH 'mysql_native_password' BY '00000
';
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'slave'@'192.168.249.129';
FLUSH PRIVILEGES;
```

**注意这里创建用户时需要选用mysql_native_password加密方式插件，否则默认会使用caching_sh
2_password加密方式，这样在同步的时候需要使用SSL的身份进行验证，为了方便简单，我们直接
用mysql_native_password方式**

- 修改配置/etc/my.cnf，新增如下配置，开启binlog,并重启mysql服务

```
[mysqld]
# 开启二进制日志功能
log-bin=mysql-bin
# 设置server_id，,注意在网段内要唯一
server-id=131
#(可选配置）要同步的数据库名，要同步多个数据库，就多加几个replicate-db-db=数据库名
binlog-do-db=mydb
#（可选配置）要忽略的数据库
binlog-ignore-db=mysql
```

- 查看主服务器状态

```
show master status
```

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------+
| mysql-bin.000001 |      155 | mydb         | mysql            |                   |
+------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)
```

注意看里面的参数，特别前面两个File和Position，在从服务器（Slave）配置主从关系会有用到的。

从数据库（192.168.249.129）

- 修改/etc/my.cnf，新增如下配置,并重启服务

```
[mysqld]
server-id=129
log-bin=mysql-bin
replicate-do-db=mydb
replicate-ignore-db=mysql
```

● 在slave中设置master信息，指定同步位置

```
stop slave;
change master to master_host='192.168.249.131',master_user='slave',master_password='000
00',master_log_file='mysql-bin.000001',master_log_pos=155;
start slave;
```

参数说明：

master_host='192.168.249.131' ## Master的IP地址

master_user='slave' ## 用于同步数据的用户（在Master中授权的用户）

master_password='000000' ## 同步数据用户的密码

master_port=3306 ## Master数据库服务的端口

masterlogfile='mysql-bin.000001' ##指定Slave从哪个日志文件开始读复制数据（Master上执行令的结果的File字段）

masterlogpos=155 ## 从哪个POSITION号开始读（Master上执行命令的结果的Position字段）

masterconnectretry=30 ##当重新建立主从连接时，如果连接建立失败，间隔多久后重试。单位为，默认设置为60秒，同步延迟调优参数。

● 查看从服务器状态

```
show slave status\G;
```

```
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.249.131
                  Master_User: slave
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000003
          Read_Master_Log_Pos: 155
               Relay_Log_File: localhost-relay-bin.000006
                Relay_Log_Pos: 369
        Relay_Master_Log_File: mysql-bin.000003
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB: mydb
          Replicate_Ignore_DB: mysql
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 155
              Relay_Log_Space: 795
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
```

至此数据库层面主从配置完成。

SpringBoot中配置主从读写分离

在主从模式下请遵守如下规则：

主数据库 只执行 INSERT,UPDATE,DELETE 操作

从数据库 只执行SELECT操作

我们这里使用开源项目[dynamic-datasource-spring-boot-starter]（https://gitee.com/baomidou/dynamic-datasource-spring-boot-starter/wikis/）作为读写分离的工具包

使用方法

1. 在mydb主数据库中建立一个简单数据表user，建好后从数据库会自动同步

```
DROP TABLE IF EXISTS `user`;
CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `account` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `position` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

2. 引入相关依赖

```
<dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.mybatis.spring.boot</groupId>
      <artifactId>mybatis-spring-boot-starter</artifactId>
      <version>2.0.1</version>
    </dependency>

    <dependency>
      <groupId>com.baomidou</groupId>
      <artifactId>dynamic-datasource-spring-boot-starter</artifactId>
      <version>2.5.5</version>
    </dependency>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.15</version>
    </dependency>

    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
```

```
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

    </dependencies>
```

3. 配置数据源

```
spring:
  datasource:
    dynamic:
      primary: master #设置默认的数据源或者数据源组,默认值即为master
      strict: false #设置严格模式,默认false不启动. 启动后再为匹配到指定数据源时候回抛出异常,不启
会使用默认数据源.
      datasource:
        master:
          type: com.zaxxer.hikari.HikariDataSource
          url: jdbc:mysql://192.168.249.131:3306/mydb?characterEncoding=utf8&zeroDateTimeB
havior=convertToNull&useSSL=false
          username: root
          password: '000000'
          driver-class-name: com.mysql.cj.jdbc.Driver
        slave_1:
          type: com.zaxxer.hikari.HikariDataSource
          url: jdbc:mysql://192.168.249.129:3306/mydb?characterEncoding=utf8&zeroDateTimeB
havior=convertToNull&useSSL=false
          username: root
          password: '000000'
          driver-class-name: com.mysql.cj.jdbc.Driver
```

4. 在启动类入口加入mybatis扫描包

```
@SpringBootApplication@MapperScan("com.jianzh5.dynamic.mapper")
public class DynamicDatsourceBootstrap {
    public static void main(String[] args) {
        SpringApplication.run(DynamicDatsourceBootstrap.class, args);
    }
}
```

5. 建立实体类User

```
@Data
public class User {
    private int id;
    private String account;
    private String name;
    private String position;
}
```

6.建立mapper接口文件，新增两个方法addUser(User user),getById(int id)

```
public interface UserDao {
    @Insert("INSERT INTO user(account, name, position) VALUES(#{account}, #{name}, #{positi
n})")
    @Options(useGeneratedKeys = true,keyProperty = "id")
    int addUser(User user);

    @Select("SELECT * FROM user WHERE id = #{id}")
    User getById(int id);
}
```

7. 建立Service相关实现

```
public interface UserService {
    int addUser(User user);

    User getById(int id);
}

@Service
public class UserServiceImpl implements UserService {
    @Resource
    private UserDao userDao;

    @Override
    public int addUser(User user) {
        return userDao.addUser(user);
    }


    @DS("slave")
    @Override
    public User getById(int id) {
        return userDao.getById(id);
    }
}
```

由于在数据源中配置了primary: master,默认操作都会从主库执行，使用注解@DS切换数据源，此注也可直接用于类文件上，同时存在方法注解优先于类上注解。

8. 编写单元测试进行测试

```
public class UserServiceTest extends DynamicDatsourceBootstrapTests {
    @Autowired
    private UserService userService;

    @Test
    public void testAddUser(){
        User user = new User();
        user.setName("李四");
        user.setAccount("sili");
        user.setPosition("JAVA开发工程师");
        int i = userService.addUser(user);
        System.out.println(user);
    }
```

```
    @Test
    public void testGetById(){
       int id = 4;
       User user = userService.getById(id);
        Assert.assertEquals("sanzhang",user.getAccount());
    }

}
```

9. 通过观察执行日志，发现读写数据库会根据@DS注解进行切换，至此Springboot集成数据库主从写分离完成。