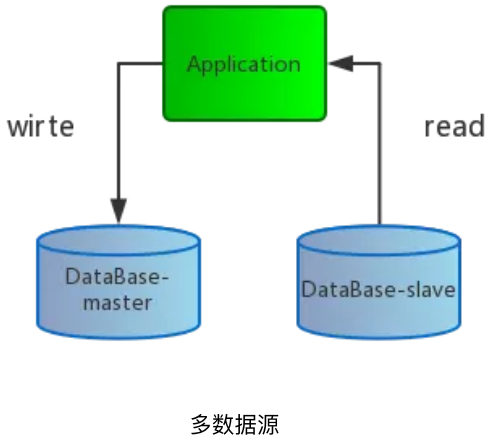
96 Winter_Chen (/u/a6fa02ee712b) +关注

(/u/a6fa02ee712b) 字数 1042 阅读 2935 评论 1 喜欢 50



个人开源项目

- springboot+mybatis+thymeleaf+docker构建的个人站点开源项目（集成了个人主页、个人作品、个人博客）
(https://blog.csdn.net/Winter_chen001/article/details/80266339)

推荐开源项目

- 开源的springboot接口文档组件swagger2
(https://blog.csdn.net/Winter_chen001/article/details/81335225)

springboot2.0正式版发布之后，很多的组件集成需要变更了，这次将多数据源的使用踩的坑给大家填一填。当前多数据源的主要为主从库，读写分离，动态切换数据源。使用的技术就是AOP进行dao方法的切面，所以大家的方法名开头都需要按照规范进行编写，如：get***、add*** 等等，

起步基础

本次的教程需要有springboot2.0集成mybatis 作为基础：

- 博客地址：springboot2.0 Mybatis 整合 (springboot2.0版本)
(<https://www.jianshu.com/p/5cd772c07041>)

- 基础项目源码：<https://github.com/WinterChenS/springboot2-mybatis-demo>
(<https://github.com/WinterChenS/springboot2-mybatis-demo>)

需要以上的步骤作为基础，运行成功之后可就可以开始配置多数据源了

(/apps/redi
utm_sourc
banner-clic

开始动手

添加依赖

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

修改启动类

修改之前：

```
@SpringBootApplication
@MapperScan("com.winterchen.dao")
public class SpringBootMybatisMutilDatabaseApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootMybatisMutilDatabaseApplication.class)
    }
}
```

修改之后：

```
@SpringBootApplication
public class SpringBootMybatisMutilDatabaseApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootMybatisMutilDatabaseApplication.class)
    }
}
```

因为改用多数据源，所以dao接口的扫描我们放在配置类中进行

修改项目配置

首先我们需要在配置文件中配置多数据源，看一下原本项目的配置：



```

spring:
  datasource:
    name: mysql_test
    #-----start-----# (1)
    type: com.alibaba.druid.pool.DruidDataSource
    #-----end-----#
    #druid相关配置
    druid:
      #监控统计拦截的filters
      filters: stat
      #-----start-----# (2)
      driver-class-name: com.mysql.jdbc.Driver
      #基本属性
      url: jdbc:mysql://127.0.0.1:3306/mytest?useUnicode=true&characterEn
      username: root
      password: root
      #-----end-----#
      #配置初始化大小/最小/最大
      initial-size: 1
      min-idle: 1
      max-active: 20
      #获取连接等待超时时间
      max-wait: 60000
      #间隔多久进行一次检测，检测需要关闭的空闲连接
      time-between-eviction-runs-millis: 60000
      #一个连接在池中最小生存的时间
      min-evictable-idle-time-millis: 300000
      validation-query: SELECT 'x'
      test-while-idle: true
      test-on-borrow: false
      test-on-return: false
      #打开PSCache，并指定每个连接上PSCache的大小。oracle设为true，mysql设为false
      pool-prepared-statements: false
      max-pool-prepared-statement-per-connection-size: 20

```

(/apps/redi
utm_sourc
banner-cli

****需要修改的地方:****

- (1) 需要将 type: com.alibaba.druid.pool.DruidDataSource 去除;
- (2) 将关于数据库的连接信息: driver-class-name、url、username、password 去除;

修改后:



```

spring:
  datasource:
    name: mysql_test
    #----- start -----# (1)
    master:
      #基本属性--注意, 这里的为【jdbcurl】-- 默认使用HikariPool作为数据库连接池
      jdbcurl: jdbc:mysql://127.0.0.1:3306/mytest?useUnicode=true&character
      username: root
      password: root
      driver-class-name: com.mysql.jdbc.Driver
    slave:
      #基本属性--注意, 这里为 【url】-- 使用 druid 作为数据库连接池
      url: jdbc:mysql://127.0.0.1:3306/mytest?useUnicode=true&characterEn
      username: root
      password: root
      driver-class-name: com.mysql.jdbc.Driver
    read: get,select,count,list,query,find
    write: add,create,update,delete,remove,insert
    #----- end -----#
    #druid相关配置
    druid:
      #监控统计拦截的filters
      filters: stat,wall
      #配置初始化大小/最小/最大
      initial-size: 1
      min-idle: 1
      max-active: 20
      #获取连接等待超时时间
      max-wait: 60000
      #间隔多久进行一次检测, 检测需要关闭的空闲连接
      time-between-eviction-runs-millis: 60000
      #一个连接在池中最小生存的时间
      min-evictable-idle-time-millis: 300000
      validation-query: SELECT 'x'
      test-while-idle: true
      test-on-borrow: false
      test-on-return: false
      #打开PSCache, 并指定每个连接上PSCache的大小。oracle设为true, mysql设为false
      pool-prepared-statements: false
      max-pool-prepared-statement-per-connection-size: 20

```

(/apps/redi
utm_sourc
banner-cli

需要修改地方:

- (1) 在如上的配置中添加 master 、 slave 两个数据源;

注意!! 两中数据源中有一处是不一样的, 原因是因为 master 数据源使用 Hikari 连接池, slave 使用的是 druid 作为数据库连接池, 所以两处的配置分别为:

```

master:
  jdbcurl: jdbc:mysql://127.0.0.1:3306/mytest?useUnicode=true&characterEnco

```

```

slave:
  url: jdbc:mysql://127.0.0.1:3306/mytest?useUnicode=true&characterEncoding

```

数据库的连接不一样的, 如果配置成一样的会在启动的时候报错。

注意！！

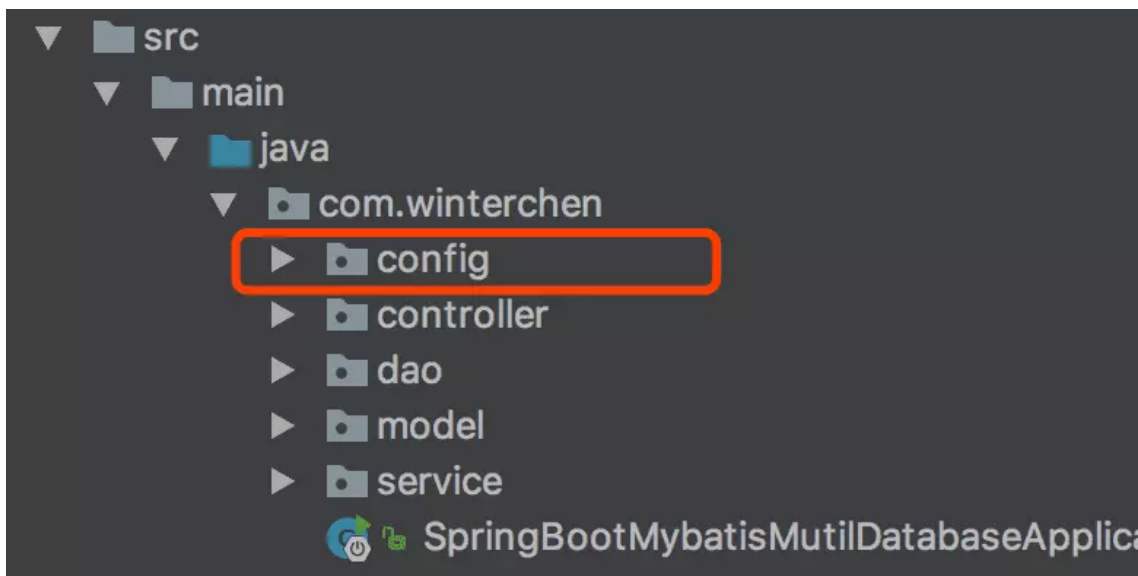
dao接口方法的方法名规则配置在这里了，当然可以自行更改：

```
read: get,select,count,list,query,find  
write: add,create,update,delete,remove,insert
```

(/apps/redi
utm_sourc
banner-cli

创建配置包

首先在项目的 /src/main/java/com/winterchen/ 包下创建 config 包



创建config包

创建数据源类型的枚举DatabaseType

该枚举类主要用来区分读写



```
package com.winterchen.config;

/**
 * 列出数据源类型
 * Created by Donghua.Chen on 2018/5/29.
 */
public enum DatabaseType {

    master("write"), slave("read");

    DatabaseType(String name) {
        this.name = name;
    }

    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "DatabaseType{" +
            "name='" + name + '\'' +
            '}';
    }
}
```

(/apps/redi
utm_sourc
banner-clc

创建线程安全的DatabaseType容器

多数据源必须要保证数据源的线程安全的

```
package com.winterchen.config;

/**
 * 保存一个线程安全的DatabaseType容器
 * Created by Donghua.Chen on 2018/5/29.
 */
public class DatabaseContextHolder {

    //用于存放多线程环境下的成员变量
    private static final ThreadLocal<DatabaseType> contextHolder = new ThreadLocal<>();

    public static void setDatabaseType(DatabaseType type) {
        contextHolder.set(type);
    }

    public static DatabaseType getDatabaseType() {
        return contextHolder.get();
    }
}
```

创建动态数据源



实现数据源切换的功能就是自定义一个类扩展AbstractRoutingDataSource抽象类，其实该相当于数据源DataSource的路由中介，可以实现在项目运行时根据相应key值切换到对应的数据源DataSource上，有兴趣的同学可以看看它的源码。

(/apps/redi
utm_sourc
banner-clic

```
public class DynamicDataSource extends AbstractRoutingDataSource {

    static final Map<DatabaseType, List<String>> METHOD_TYPE_MAP = new HashMa

    @Nullable
    @Override
    protected Object determineCurrentLookupKey() {
        DatabaseType type = DatabaseContextHolder.getDatabaseType();
        logger.info("=====dataSource =====" + type);
        return type;
    }

    void setMethodType(DatabaseType type, String content) {
        List<String> list = Arrays.asList(content.split(","));
        METHOD_TYPE_MAP.put(type, list);
    }

}
```

创建数据源配置类DataSourceConfig



```
@Configuration
@MapperScan("com.winterchen.dao")
@EnableTransactionManagement
public class DataSourceConfig {

    private static Logger logger = LoggerFactory.getLogger(DataSourceConfig.class);

    @Autowired
    private Environment env; // (1)

    @Autowired
    private DataSourceProperties properties; // (2)

    @Value("${spring.datasource.druid.filters}") // (3)
    private String filters;

    @Value("${spring.datasource.druid.initial-size}")
    private Integer initialSize;

    @Value("${spring.datasource.druid.min-idle}")
    private Integer minIdle;

    @Value("${spring.datasource.druid.max-active}")
    private Integer maxActive;

    @Value("${spring.datasource.druid.max-wait}")
    private Integer maxWait;

    @Value("${spring.datasource.druid.time-between-eviction-runs-millis}")
    private Long timeBetweenEvictionRunsMillis;

    @Value("${spring.datasource.druid.min-evictable-idle-time-millis}")
    private Long minEvictableIdleTimeMillis;

    @Value("${spring.datasource.druid.validation-query}")
    private String validationQuery;

    @Value("${spring.datasource.druid.test-while-idle}")
    private Boolean testWhileIdle;

    @Value("${spring.datasource.druid.test-on-borrow}")
    private boolean testOnBorrow;

    @Value("${spring.datasource.druid.test-on-return}")
    private boolean testOnReturn;

    @Value("${spring.datasource.druid.pool-prepared-statements}")
    private boolean poolPreparedStatements;

    @Value("${spring.datasource.druid.max-pool-prepared-statement-per-connection-size}")
    private Integer maxPoolPreparedStatementPerConnectionSize;

    /**
     * 通过Spring JDBC 快速创建 DataSource
     * @return
     */
    @Bean(name = "masterDataSource")
    @Qualifier("masterDataSource")
    @ConfigurationProperties(prefix = "spring.datasource.master") // (4)
    public DataSource masterDataSource() {
        return DataSourceBuilder.create().build();
    }

    /**
     * 手动创建DruidDataSource,通过DataSourceProperties 读取配置
     */
}
```

(/apps/redi
utm_sourc
banner-cli




```

    * @return
    * @throws SQLException
    */
@Bean(name = "slaveDataSource")
@Qualifier("slaveDataSource")
@ConfigurationProperties(prefix = "spring.datasource.slave")
public DataSource slaveDataSource() throws SQLException {
    DruidDataSource dataSource = new DruidDataSource();
    dataSource.setFilters(filters);
    dataSource.setUrl(properties.getUrl());
    dataSource.setDriverClassName(properties.getDriverClassName());
    dataSource.setUsername(properties.getUsername());
    dataSource.setPassword(properties.getPassword());
    dataSource.setInitialSize(initialSize);
    dataSource.setMinIdle(minIdle);
    dataSource.setMaxActive(maxActive);
    dataSource.setMaxWait(maxWait);
    dataSource.setTimeBetweenEvictionRunsMillis(timeBetweenEvictionRunsMi
    dataSource.setMinEvictableIdleTimeMillis(minEvictableIdleTimeMillis);
    dataSource.setValidationQuery(validationQuery);
    dataSource.setTestWhileIdle(testWhileIdle);
    dataSource.setTestOnBorrow(testOnBorrow);
    dataSource.setTestOnReturn(testOnReturn);
    dataSource.setPoolPreparedStatements(poolPreparedStatements);
    dataSource.setMaxPoolPreparedStatementPerConnectionSize(maxPoolPrepar
    return dataSource;
}

/**
 * 构造多数据源连接池
 * Master 数据源连接池采用 HikariDataSource
 * Slave 数据源连接池采用 DruidDataSource
 * @param master
 * @param slave
 * @return
 */
@Bean
@Primary
public DynamicDataSource dataSource(@Qualifier("masterDataSource") DataSo
                                @Qualifier("slaveDataSource") DataSou
    Map<Object, Object> targetDataSources = new HashMap<>();
    targetDataSources.put(DatabaseType.master, master);
    targetDataSources.put(DatabaseType.slave, slave);

    DynamicDataSource dataSource = new DynamicDataSource();
    dataSource.setTargetDataSources(targetDataSources);// 该方法是AbstractF
    dataSource.setDefaultTargetDataSource(slave);// 默认的datasource设置为m

    String read = env.getProperty("spring.datasource.read");
    dataSource.setMethodType(DatabaseType.slave, read);

    String write = env.getProperty("spring.datasource.write");
    dataSource.setMethodType(DatabaseType.master, write);

    return dataSource;
}

@Bean
public SqlSessionFactory sqlSessionFactory(@Qualifier("masterDataSource")
                                           @Qualifier("slaveDataSource")
    SqlSessionFactoryBean fb = new SqlSessionFactoryBean();
    fb.setDataSource(this.dataSource(myTestDbDataSource, myTestDb2DataSou
    fb.setTypeAliasesPackage(env.getProperty("mybatis.type-aliases-packag
    fb.setMapperLocations(new PathMatchingResourcePatternResolver().getRe
    return fb.getObject();
}

```

(/apps/redi
utm_sourc
banner-clic



```
@Bean
public DataSourceTransactionManager transactionManager(DynamicDataSource
    return new DataSourceTransactionManager(dataSource);
}
```

(/apps/redi
utm_sourc
banner-cli

以上的代码中：

- (1) 注入类 `Environment` 可以很方便的获取配置文件中的参数
- (2) `DataSourceProperties` 和 (4) 中的 `@ConfigurationProperties(prefix = "spring.datasource.master")` 配合使用，将配置文件中的配置数据自动封装到实体类 `DataSourceProperties` 中
- (3) `@Value` 注解同样是指定获取配置文件中的配置；

更详细的配置大家可以参考官方文档。

配置AOP

本章的开头已经说过，多数据源动态切换的原理是利用AOP切面进行动态的切换的，当调用 `dao` 接口方法时，根据接口方法的方法名开头进行区分读写。



处理数据源，根据命名区分

ated by Donghua.Chen on 2018/5/29.

```

t
nent
eAspectJAutoProxy(proxyTargetClass = true)
class DataSourceAspect {

private static Logger logger = LoggerFactory.getLogger(DataSourceAspect.class);

pointcut("execution(* com.winterchen.dao.*.*(..))")//切点
public void aspect() {

before("aspect()")
public void before(JoinPoint point) { //在指定切点的方法之前执行
    String className = point.getTarget().getClass().getName();
    String method = point.getSignature().getName();
    String args = StringUtils.join(point.getArgs(), ",");
    logger.info("className:{}, method:{}, args:{} ", className, method, args);
    try {
        for (DatabaseType type : DatabaseType.values()) {
            List<String> values = DynamicDataSource.METHOD_TYPE_MAP.get(type);
            for (String key : values) {
                if (method.startsWith(key)) {
                    logger.info(">>{} 方法使用的数据源为:{}<<", method, key);
                    DatabaseContextHolder.setDatabaseType(type);
                    DatabaseType types = DatabaseContextHolder.getDatabaseType();
                    logger.info(">>{}方法使用的数据源为:{}<<", method, types);
                }
            }
        }
    } catch (Exception e) {
        logger.error(e.getMessage(), e);
    }
}
}

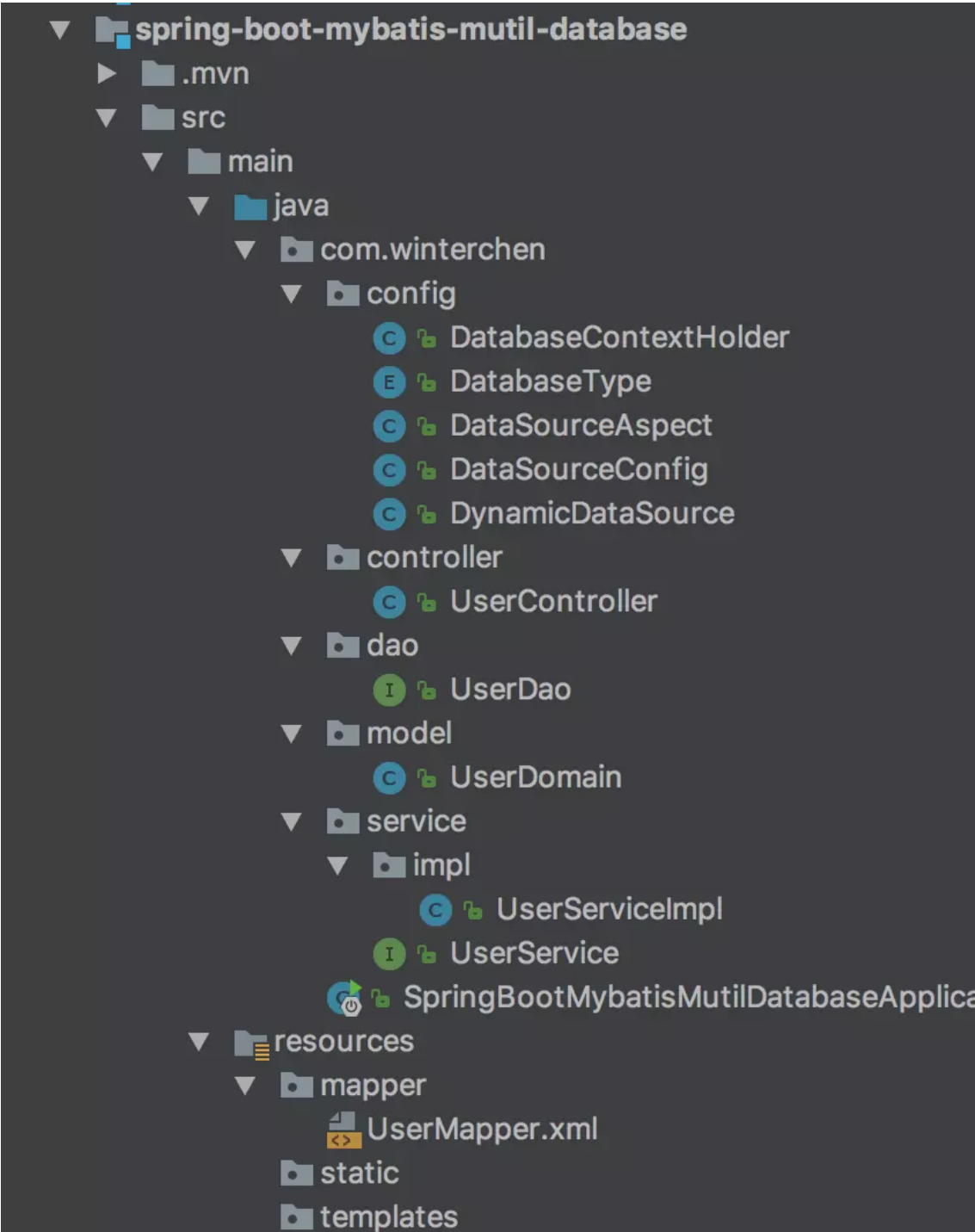
```

(/apps/redi
utm_sourc
banner-clic

如上可以看到，切点切在 dao 的接口方法中，根据接口方法的方法名进行匹配数据源，然后将数据源set到用于存放数据源线程安全的容器中；

完整的项目结构了解一下：





(/apps/redi
utm_sourc
banner-clic

完整项目结构

项目启动

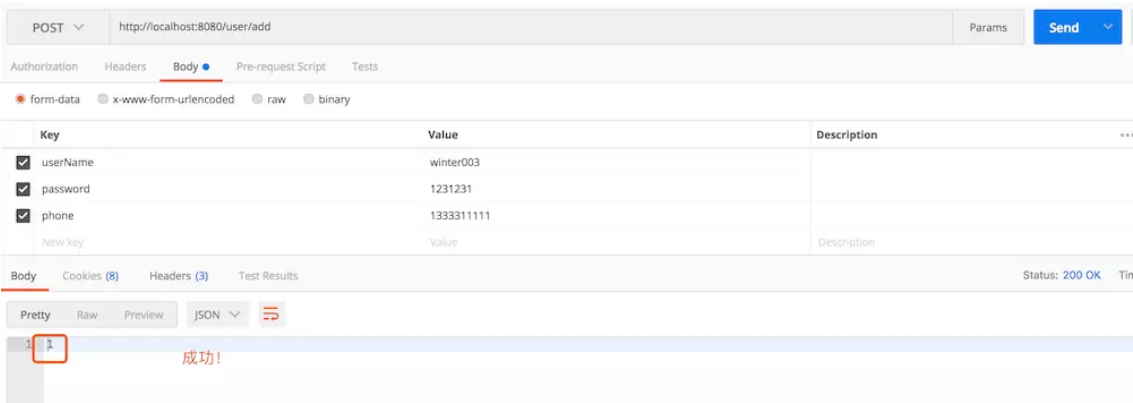
启动成功：

```
2018-05-30 17:27:16.492 INFO 35406 --- [main] o.s.j.e.a.AnnotationMethodProcessingHandler
2018-05-30 17:27:16.496 INFO 35406 --- [main] o.s.j.e.a.AnnotationMethodProcessingHandler
2018-05-30 17:27:16.498 INFO 35406 --- [main] o.s.j.e.a.AnnotationMethodProcessingHandler
2018-05-30 17:27:16.590 INFO 35406 --- [main] o.s.b.w.embedded.tomcat.EmbeddedTomcat
2018-05-30 17:27:16.598 INFO 35406 --- [main] pringBootMybatisMut
```

^

🔗

添加用户 (write) :



(/apps/redi
utm_sourc
banner-cli

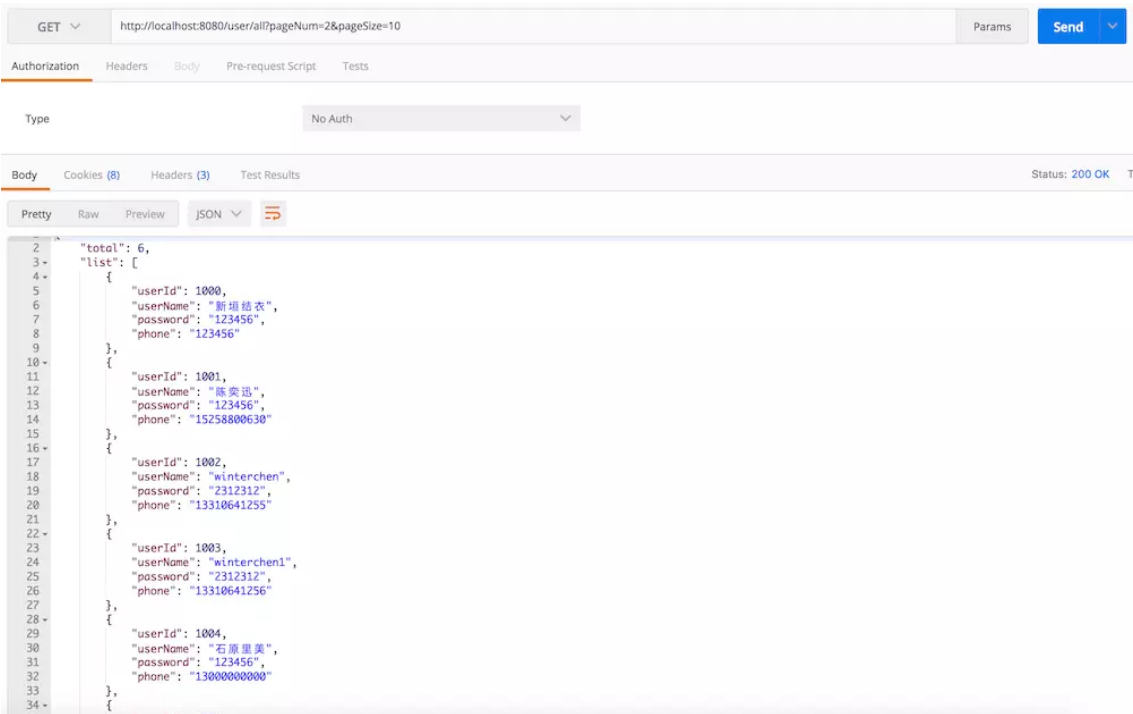
添加用户

日志:

```
2018-05-30 17:29:07.347 INFO 35406 --- [nio-8080-exec-1] com.winterchen.conf
2018-05-30 17:29:07.350 INFO 35406 --- [nio-8080-exec-1] com.winterchen.conf
2018-05-30 17:29:07.351 INFO 35406 --- [nio-8080-exec-1] com.winterchen.conf
2018-05-30 17:29:07.461 INFO 35406 --- [nio-8080-exec-1] com.winterchen.conf
2018-05-30 17:29:07.462 INFO 35406 --- [nio-8080-exec-1] com.zaxxer.hikari.H
2018-05-30 17:29:07.952 INFO 35406 --- [nio-8080-exec-1] com.zaxxer.hikari.H
```

可以看出使用的就是 write 数据源, 并且该数据源是使用 HikariPool 作为数据库连接池的

查询用户 (read) :



查询用户

日志：

```
2018-05-30 17:29:41.616 INFO 35406 --- [nio-8080-exec-2] com.winterchen.conf
2018-05-30 17:29:41.618 INFO 35406 --- [nio-8080-exec-2] com.winterchen.conf
2018-05-30 17:29:41.618 INFO 35406 --- [nio-8080-exec-2] com.winterchen.conf
2018-05-30 17:29:41.693 INFO 35406 --- [nio-8080-exec-2] com.winterchen.conf
2018-05-30 17:29:41.982 INFO 35406 --- [nio-8080-exec-2] com.alibaba.druid.p
```

(/apps/redi
utm_sourc
banner-clic

可以看出使用的是 read 数据源。

源码地址：戳[这里](https://github.com/WinterChenS/springboot-learning-experience/tree/master/spring-boot-mybatis-mutil-database) (https://github.com/WinterChenS/springboot-learning-experience/tree/master/spring-boot-mybatis-mutil-database)

springboot技术交流群：681513531

小礼物走一走，来简书关注我

赞赏支持

Spring Boot (/nb/19194215)

举报文章 © 著作权归作者所有

96

Winter_Chen (/u/a6fa02ee712b) ♂
(/u/a6fa02ee712b)
写了 8391 字，被 195 人关注，获得了 288 个喜欢

+ 关注

个人博客：<https://winterchen.com>

喜欢 | 50

微信

微博

图片

更多分享

下载简书 App ▶
随时随地发现和创作内容

(/apps/redirect?utm_source=note-bottom-click)





登录 (/sign-in?utm_source=desktop&utm_medium=not-signed-in-comr



(/apps/redi
utm_sourc
banner-clic

1条评论

只看作者

按时间倒序






按时间正序

 IT人故事会 (/u/46fb70e81d8c) 
2楼 · 2018.06.03 00:41
(/u/46fb70e81d8c)
老铁,顶, 互相多学习!

赞


回复

被以下专题收入，发现更多相似内容

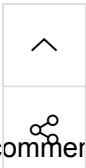
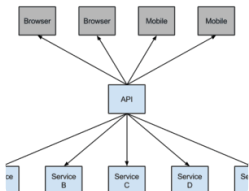
-  MyBatis... (/c/3f2879e1ad45?utm_source=desktop&utm_medium=notes-included-collection)
-  SpringB... (/c/c3fe8e7aeb09?utm_source=desktop&utm_medium=notes-included-collection)
-  架构社区 (/c/77c2623f32a6?utm_source=desktop&utm_medium=notes-included-collection)
-  SpringBoot (/c/7518b9ea714f?utm_source=desktop&utm_medium=notes-included-collection)
-  SpringB... (/c/edd22b195bd9?utm_source=desktop&utm_medium=notes-included-collection)

Java面试宝典Beta5.0 (/p/fb7d48083e5e?utm_campaign=maleskine&ut...

pdf下载地址：Java面试宝典 第一章内容介绍 20 第二章JavaSE基础 21 一、Java面向对象 21 1. 面向对象都有哪些特性以及你对这些特性的理解 21 2. 访问权限修饰符public、private、protected, 以及不写（默认）...


 王震阳 (/u/773a782d9d83?)
48
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

(/p/46fd0faecac1?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
Spring Cloud (/p/46fd0faecac1?utm_campaign=maleskine&utm_conte...


Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智能路由，微代理，控制总线）。分布式系统的协调导致了样板模式,使用Spring Cloud开发人员可...

 卡卡罗2017 (/u/d90908cb0d85?)
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

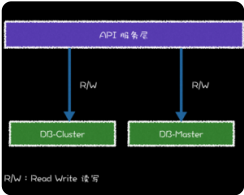
mapsto
utm_sour
banner-clic

Spring boot参考指南 (/p/67a0e41dfe05?utm_campaign=maleskine&ut...

Spring Boot 参考指南 介绍 转载自:https://www.gitbook.com/book/qbgbook/spring-boot-reference-guide-zh/details带目录浏览地址:http://www.maoyupeng.com/sprin...


 毛宇鹏 (/u/d3ea915e1e0f?)
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

(/p/c6770b1466b8?




utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
Spring Boot 整合 Mybatis 实现 Druid 多数据源详解 (/p/c6770b1466b8?u...

摘要: 原创出处:www.bysocket.com 泥瓦匠BYSocket 希望转载，保留摘要，谢谢！“清醒时做事，糊涂时跑步，大怒时睡觉，独处时思考” 本文提纲 一、多数据源的应用场景 二、运行 springboot-mybatis-mutil-...

 BYSocket泥瓦匠 (/u/08493ba9392a?)
48
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

面向对象的用电信息数据交换协议 (/p/94caedb70f65?utm_campaign=mal...

国家电网公司企业标准（Q/GDW）- 面向对象的用电信息数据交换协议 - 报批稿：20170802 前言： 排版 by Dr_Ting公众号：庭说移步 tingtalk.me 获得更友好的阅读体验 Q/GDW XXXX-201X《面向对象的用电信息...


 庭说 (/u/a0d04c114c89?)
48
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

(/p/378c2462c9c6?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
云起时，悲欢舒卷于心 (/p/378c2462c9c6?utm_campaign=maleskine&ut...


此刻云起，压着屋顶、山岗、树梢，在无人抵达的内心舒卷，开始酝酿大雨滂沱，悲欢离合。

 胡剑廷 (/u/a8a5822f713a?)
48
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

相处的不同层次 (/p/41eb4e60dcd6?utm_campaign=maleskine&utm_co...

共修功课第十四天：结婚后要学会转化自己的个性适应对方，与对方融合，借助日常发生的事情，尝试着去


练习。引导：现在我们都学习了，成长了，我们要有一个“家”的概念，要学会融入，不要让别人来适应你...

 wangjb_a9e9 (/u/50967fd1e971?
48utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

(/apps/redi
utm_sourc
banner-clic

虚了 (/p/72a767501688?utm_campaign=maleskine&utm_content=note...

我真心虚掉了。中午没午休，下午考了个该死的体育理论网上试卷。结果未卜。不能放纵，要坚持！坚持个毛线啊！虚弱无力，困困困。中午好好艳羡了一把，不要仇富。两个女教授，拼生活质量。让我这种穷...


 豪ssman (/u/b5d44a8da9cf?
48utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

(/p/01833f4b15fc?




utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio
妻不再相伴，但爱永远相随 (/p/01833f4b15fc?utm_campaign=maleskine...

文|水聽 我今年38岁，出生在江南水乡的一普通家庭，父母以种田为生。听母亲说，我在她肚子里四五个月时，就找人做了个B超，那时就知道是个男孩。出生后也把家里人乐坏了，农村人难免重男轻女，尤其是爷...

 水聽 (/u/7f8028a9af33?
48utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

河南永城市龙岗一叶子面膜微商代理政策 (/p/e0d3ef3ee7e3?utm_campaig...

现在，一叶子已经成为了很多宝妈，大学生和上班族获取收入的一个不错的途径了。一个好的产品，会让自己在经营的过程中变得非常的轻松，而且也会很容易成功。如果选择的产品不好，那么就很难有客户愿意...

 一櫻花 (/u/96c8364641c4?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

