1依赖包

<dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 <version>1.1.10</version>  
</dependency>  
<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.41</version>  
</dependency>  
<!--分页-->  
<!-- https://mvnrepository.com/artifact/com.github.pagehelper/pagehelper -->  
<dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper</artifactId>  
 <version>5.1.8</version>  
</dependency>  
<dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper-spring-boot-autoconfigure</artifactId>  
 <version>1.2.5</version>  
</dependency>  
<dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper-spring-boot-starter</artifactId>  
 <version>1.2.5</version>  
</dependency>  
<!--分页-->  
<!--必须加日志jar,否则druid报错 -->  
<dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.16</version>  
</dependency>

2.application.properties文件配置

注意:这两个文件夹的路径

mybatis.config-location=classpath:/mybatis/mybatis-config.xml  
mybatis.mapper-locations=classpath:/mybatis/mapper/\*.xml

server.port=8088  
  
spring.datasource.type=com.alibaba.druid.pool.DruidDataSource  
spring.datasource.driver-class-name=com.mysql.jdbc.Driver  
spring.datasource.url=jdbc:mysql://139.159.144.194:3307/db\_ncdp\_card\_product?useSSL=false&serverTimezone=GMT&characterEncoding=UTF-8  
spring.datasource.username=khdevuser  
spring.datasource.password=khdevdb\_0225  
# 下面为连接池的补充设置，应用到上面所有数据源中  
spring.datasource.initialSize=5  
spring.datasource.minIdle=5  
spring.datasource.maxActive=20  
# 配置获取连接等待超时的时间  
spring.datasource.maxWait=60000  
# 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒  
spring.datasource.timeBetweenEvictionRunsMillis=60000  
# 配置一个连接在池中最小生存的时间，单位是毫秒  
spring.datasource.minEvictableIdleTimeMillis=300000  
spring.datasource.validationQuery=SELECT 1 FROM DUAL  
spring.datasource.testWhileIdle=true  
spring.datasource.testOnBorrow=false  
spring.datasource.testOnReturn=false  
# 配置监控统计拦截的filters，去掉后监控界面sql无法统计，'wall'用于防火墙  
spring.datasource.filters=stat,wall,log4j  
spring.datasource.logSlowSql=true  
  
mybatis.config-location=classpath:/mybatis/mybatis-config.xml  
mybatis.mapper-locations=classpath:/mybatis/mapper/\*.xml  
#解决spring-boot-starter-actuator与datasource引起的循环依赖问题  
spring.cloud.refresh.refreshable=none

3.config配置类(数据库连接池的配置)

@Configuration  
public class DruidDataSourceConfig {  
 private Logger logger = LoggerFactory.*getLogger*(DruidDataSourceConfig.class);  
  
 @Value("${spring.datasource.url}")  
 private String dbUrl;  
  
 @Value("${spring.datasource.username}")  
 private String username;  
  
 @Value("${spring.datasource.password}")  
 private String password;  
  
 @Value("${spring.datasource.driver-class-name}")  
 private String driverClassName;  
  
 @Value("${spring.datasource.initialSize}")  
 private int initialSize;  
  
 @Value("${spring.datasource.minIdle}")  
 private int minIdle;  
  
 @Value("${spring.datasource.maxActive}")  
 private int maxActive;  
  
 @Value("${spring.datasource.maxWait}")  
 private int maxWait;  
  
 @Value("${spring.datasource.timeBetweenEvictionRunsMillis}")  
 private int timeBetweenEvictionRunsMillis;  
  
 @Value("${spring.datasource.minEvictableIdleTimeMillis}")  
 private int minEvictableIdleTimeMillis;  
  
 @Value("${spring.datasource.validationQuery}")  
 private String validationQuery;  
  
 @Value("${spring.datasource.testWhileIdle}")  
 private boolean testWhileIdle;  
  
 @Value("${spring.datasource.testOnBorrow}")  
 private boolean testOnBorrow;  
  
 @Value("${spring.datasource.testOnReturn}")  
 private boolean testOnReturn;  
  
 @Value("${spring.datasource.filters}")  
 private String filters;  
  
 @Value("${spring.datasource.logSlowSql}")  
 private String logSlowSql;  
  
 */\*\*  
 \* 配置访问页面信息  
 \*  
 \** ***@return*** *\*/* @Bean  
 public ServletRegistrationBean druidServlet() {  
 ServletRegistrationBean reg = new ServletRegistrationBean();  
 reg.setServlet(new StatViewServlet());  
 reg.addUrlMappings("/druid/\*");  
 reg.addInitParameter("loginUsername", username);  
 reg.addInitParameter("loginPassword", password);  
 reg.addInitParameter("logSlowSql", logSlowSql);  
 return reg;  
 }  
  
 @Bean  
 public FilterRegistrationBean filterRegistrationBean() {  
 FilterRegistrationBean filterRegistrationBean = new FilterRegistrationBean();  
 filterRegistrationBean.setFilter(new WebStatFilter());  
 filterRegistrationBean.addUrlPatterns("/\*");  
 filterRegistrationBean.addInitParameter("exclusions", "\*.js,\*.gif,\*.jpg,\*.png,\*.css,\*.ico,/druid/\*");  
 filterRegistrationBean.addInitParameter("profileEnable", "true");  
 return filterRegistrationBean;  
 }  
  
 @Bean  
 public DataSource druidDataSource() {  
 DruidDataSource datasource = new DruidDataSource();  
 datasource.setUrl(dbUrl);  
 datasource.setUsername(username);  
 datasource.setPassword(password);  
 datasource.setDriverClassName(driverClassName);  
 datasource.setInitialSize(initialSize);  
 datasource.setMinIdle(minIdle);  
 datasource.setMaxActive(maxActive);  
 datasource.setMaxWait(maxWait);  
 datasource.setTimeBetweenEvictionRunsMillis(timeBetweenEvictionRunsMillis);  
 datasource.setMinEvictableIdleTimeMillis(minEvictableIdleTimeMillis);  
 datasource.setValidationQuery(validationQuery);  
 datasource.setTestWhileIdle(testWhileIdle);  
 datasource.setTestOnBorrow(testOnBorrow);  
 datasource.setTestOnReturn(testOnReturn);  
 try {  
 datasource.setFilters(filters);  
 } catch (SQLException e) {  
 logger.error("druid configuration initialization filter", e);  
 }  
 return datasource;  
 }  
}

4mybatis的配置文件

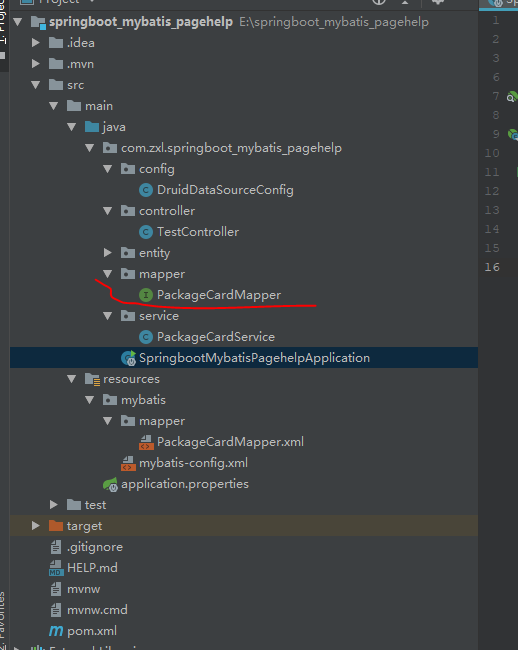
mybatis-config.xml

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE configuration  
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
  
  
 <settings>  
 <!-- 使全局的映射器启用或禁用缓存。 -->  
 <setting name="cacheEnabled" value="true"/>  
 <!-- 全局启用或禁用延迟加载。当禁用时，所有关联对象都会即时加载。 -->  
 <setting name="lazyLoadingEnabled" value="true"/>  
 <!-- 当启用时，有延迟加载属性的对象在被调用时将会完全加载任意属性。否则，每种属性将会按需要加载。 -->  
 <setting name="aggressiveLazyLoading" value="true"/>  
 <!-- 是否允许单条sql 返回多个数据集 (取决于驱动的兼容性) default:true -->  
 <setting name="multipleResultSetsEnabled" value="true"/>  
 <!-- 是否可以使用列的别名 (取决于驱动的兼容性) default:true -->  
 <setting name="useColumnLabel" value="true"/>  
 <!-- 允许JDBC 生成主键。需要驱动器支持。如果设为了true，这个设置将强制使用被生成的主键，有一些驱动器不兼容不过仍然可以执行。default:false -->  
 <setting name="useGeneratedKeys" value="false"/>  
 <!-- 指定 MyBatis 如何自动映射 数据基表的列 NONE：不隐射　PARTIAL:部分 FULL:全部 -->  
 <setting name="autoMappingBehavior" value="PARTIAL"/>  
 <!-- 这是默认的执行类型（SIMPLE: 简单； REUSE: 执行器可能重复使用prepared statements语句；BATCH: 执行器可以重复执行语句和批量更新） -->  
 <setting name="defaultExecutorType" value="SIMPLE"/>  
 <!-- 使用驼峰命名法转换字段。 -->  
 <setting name="mapUnderscoreToCamelCase" value="true"/>  
 <!-- 设置本地缓存范围 session:就会有数据的共享 statement:语句范围 (这样就不会有数据的共享 ) defalut:session -->  
 <setting name="localCacheScope" value="SESSION"/>  
 <!-- 设置但JDBC类型为空时,某些驱动程序 要指定值,default:OTHER，插入空值时不需要指定类型 -->  
 <setting name="jdbcTypeForNull" value="NULL"/>  
 <!-- 打印查询语句 -->  
 <setting name="logImpl" value="STDOUT\_LOGGING"/>  
 </settings>  
  
  
</configuration>

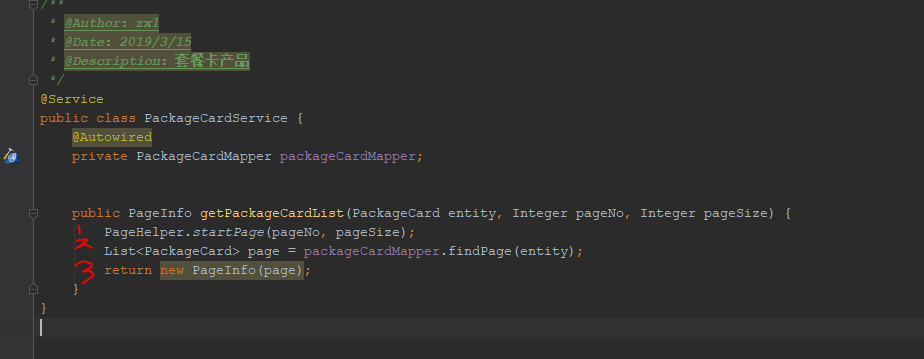
5启动类

注意要扫描map路径

@SpringBootApplication  
@MapperScan(basePackages = {"com.zxl.springboot\_mybatis\_pagehelp.mapper"})  
public class SpringbootMybatisPagehelpApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringbootMybatisPagehelpApplication.class, args);  
 }  
  
}



7service的分页代码只需要3步就可以



8controller层更简单了

