后台配置步骤

1. 阿里云服务器配置
2. JDK配置

参考文档<https://www.cnblogs.com/LiuChunfu/p/7302491.html>

1. Tomcat配置

参考文档<https://www.linuxidc.com/Linux/2016-09/134909.htm>

注意：阿里云服务器对端口有安全保护，需要手动配置开放端口,在实例->管理->本实例安全组->配置规则中配置开放端口。

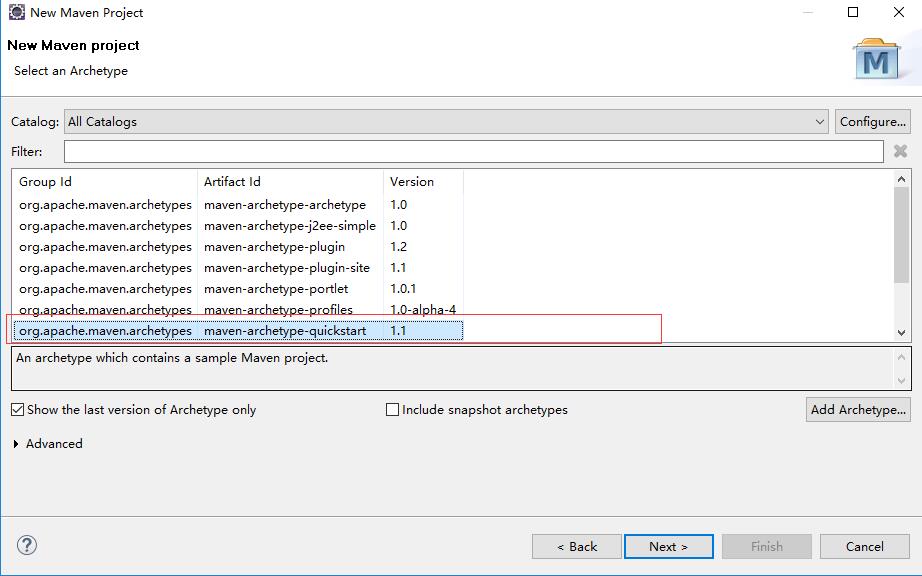
1. Mysql配置

参考文档<https://www.linuxidc.com/Linux/2016-09/134940.htm>

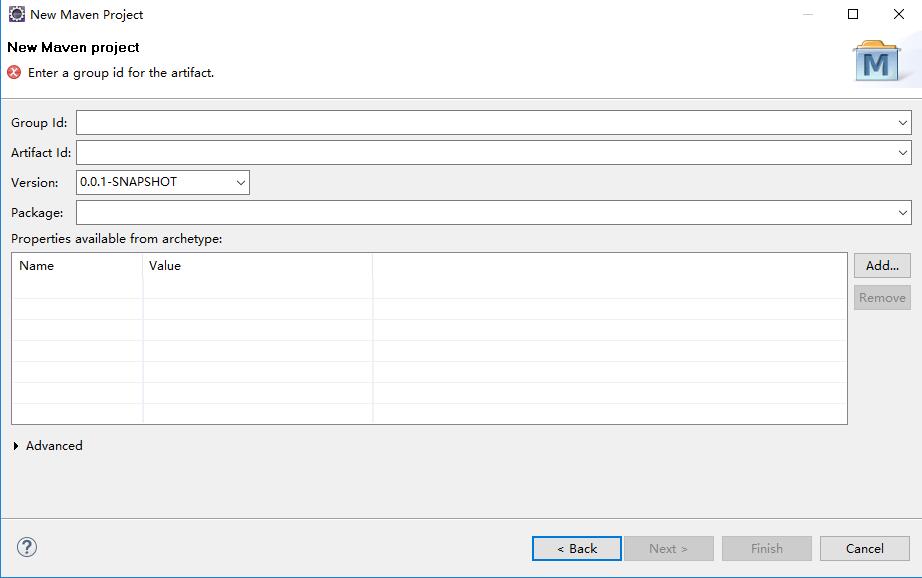
注意：需要开放3306端口，步骤同tomcat。

1. Spring-boot后台配置
2. Eclipse创建maven工程

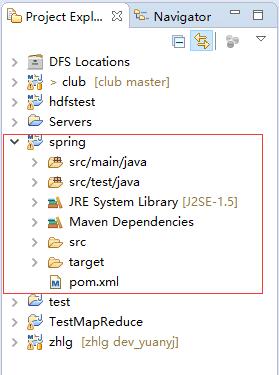
选择file->new->other->maven project 弹出一下页面，选中quickstart，



点击next弹出以下页面，group id和artifact id可以自由填写，group id可填写公司域名的反写，例如“com.xxx”，artifact id是工程名称，可自己填写。项目的package默认为group id. artifact id



信息填写完整后点击finish完成工程创建，在project explore视图中可看到刚创建的maven工程。



Src/main/java是工程包目录，用户自己编写的java创在此目录下；

Src/test/java是测试文件目录；

src是src/xxx/xxx的多级目录；

target目录是maven打包生成包的目录，如jar包、war包；

pom.xml是项目配置文件，项目依赖的jar包在改文件中配置。

1. Spring-boot框架maven配置

在pom.xml中配置spring-boot框架所依赖的jar包，配置如下：

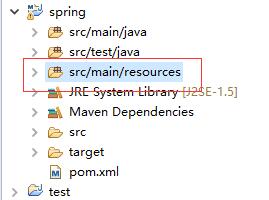
|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.6.RELEASE</version>  </parent>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>3.8.1</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>6.0.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.mybatis.spring.boot/mybatis-spring-boot-starter -->  <dependency>  <groupId>org.mybatis.spring.boot</groupId>  <artifactId>mybatis-spring-boot-starter</artifactId>  <version>1.3.0</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.mybatis.generator/mybatis-generator-core -->  <dependency>  <groupId>org.mybatis.generator</groupId>  <artifactId>mybatis-generator-core</artifactId>  <version>1.3.5</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.json/json -->  <dependency>  <groupId>org.json</groupId>  <artifactId>json</artifactId>  <version>20160810</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-devtools -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  <version>1.5.6.RELEASE</version>  </dependency>  https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->  <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  <version>4.5.2</version>  </dependency>  </dependencies> |

Maven工具会根据配置从远程maven库中自动下载依赖jar包

1. Spring-boot配置

为了配置spring-boot框架需要对maven项目做些修改，具体如下：

在项目根目录下创建src/main/resources 源目录:



在该目录下创建application.properties框架配置文件，spring-boot框架在启动默认从改目录下读取本配置文件，内容如下：

|  |
| --- |
| #内置服务器配置  #reserved properties / search 'COMMON SPRING BOOT PROPERTIES' in the doc.  server.port=8014  #server.address=192.168.3.2  #server.address=172.18.45.160  server.session.timeout=30  #数据库配置  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  spring.datasource.url=jdbc:mysql://127.0.0.1:3306/zhlg?useUnicode=true&characterEncoding=utf-8&serverTimezone=UTC&useSSL=true  spring.datasource.username=root  spring.datasource.password=123456  # MyBatis Generator包路径配置  mybatis.generator.model.package=com.zyhl.zhlg.model.gen  mybatis.generator.dao.package=com.zyhl.zhlg.dao.gen  mybatis.generator.xml.mapper.package=com.zyhl.zhlg.dao.gen  mybatis.generator.target.project=zhlg/src/main/java  mybatis.generator.target.project.xml=zhlg/src/main/resources  mybatis.generator.driver.classpath=C:\\Users\\yuanyj\\.m2\\repository\\mysql\\mysql-connector-java\\5.1.31\\mysql-connector-java-5.1.31.jar |

文件定义了框架启动端口和ip地址，其中ip地址可不配，默认为当前机器的ip;

通知也定义了mysql数据库的配置。

在App.java文件中添加以下两个注解

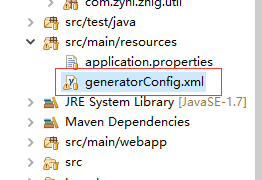


至此，spring-boot框架已搭建完成。

1. Spring-boot mybatis集成

Mybatis是现在常用的数据层框架，可自动生成数据库代码。

1. eclipse安装mybatis generator插件，在help->marketplace中搜索mybatis generator即可安装。
2. 在src/main/resources目录下创建mybatis配置文件，选中new->other->mybatis创建



内容如下：

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <!DOCTYPE generatorConfiguration  PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"  "http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd">  <!-- 配置Run As Maven build : Goals 参数 : mybatis-generator:generate -Dmybatis.generator.overwrite=true -->  <!-- 配置 tableName,使用 Run As Maven build 生成 dao model 层 -->  <generatorConfiguration>  <!-- 配置文件路径 -->  <properties resource=*"application.properties"*/>  <!--数据库驱动包路径 -->  <classPathEntry location=*"${mybatis.generator.driver.classpath}"*/>  <context id=*"DB2Tables"* targetRuntime=*"MyBatis3"*>  <!--关闭注释 -->  <commentGenerator>  <property name=*"suppressDate"* value=*"true"*/>  <!-- 是否去除自动生成的注释 true：是 ： false:否 -->  <property name=*"suppressAllComments"* value=*"false"*/>  </commentGenerator>  <!--数据库连接信息 -->  <jdbcConnection  driverClass=*"${spring.datasource.driver-class-name}"*  connectionURL=*"${spring.datasource.url}"*  userId=*"${spring.datasource.username}"*  password=*"${spring.datasource.password}"*>  </jdbcConnection>  <!--生成的model 包路径 -->  <javaModelGenerator targetPackage=*"${mybatis.generator.model.package}"* targetProject=*"${mybatis.generator.target.project}"*>  <property name=*"enableSubPackages"* value=*"ture"*/>  <property name=*"trimStrings"* value=*"true"*/>  </javaModelGenerator>  <!--生成xml mapper文件 路径 -->  <sqlMapGenerator targetPackage=*"${mybatis.generator.xml.mapper.package}"* targetProject=*"${mybatis.generator.target.project}"*>  <property name=*"enableSubPackages"* value=*"ture"*/>  </sqlMapGenerator>  <!-- 生成的Dao接口 的包路径 -->  <javaClientGenerator type=*"XMLMAPPER"* targetPackage=*"${mybatis.generator.dao.package}"* targetProject=*"${mybatis.generator.target.project}"*>  <property name=*"enableSubPackages"* value=*"ture"*/>  </javaClientGenerator>  <!--对应数据库表名 -->    <table tableName=*"gis\_heatmap\_data\_copy"* domainObjectName=*"HeatMapEntity"*>  </table>    </context>  </generatorConfiguration> |

主要配置如数据库驱动包路径、数据库连接信息、生成的model 包路径、生成xml mapper文件路径、生成的Dao接口的包路径统一配置在*application.properties*文件中如下：

|  |
| --- |
| #数据库配置  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  spring.datasource.url=jdbc:mysql://127.0.0.1:3306/zhlg?useUnicode=true&characterEncoding=utf-8&serverTimezone=UTC&useSSL=true  spring.datasource.username=root  spring.datasource.password=123456  # MyBatis Generator包路径配置  mybatis.generator.model.package=com.zyhl.zhlg.model.gen  mybatis.generator.dao.package=com.zyhl.zhlg.dao.gen  mybatis.generator.xml.mapper.package=com.zyhl.zhlg.dao.gen  mybatis.generator.target.project=zhlg/src/main/java  mybatis.generator.target.project.xml=zhlg/src/main/resources  mybatis.generator.driver.classpath=C:\\Users\\yuanyj\\.m2\\repository\\mysql\\mysql-connector-java\\5.1.31\\mysql-connector-java-5.1.31.jar |

tableName和domainObjectName定义了需要生成的数据库表名和生成的实体类名称。

1. 在App.java文件中添加以下注解

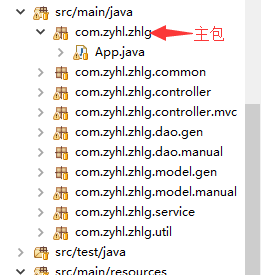


该注解表示框架在启动时会扫描定义的映射路径。

至此spring-boot已集成了mybatis框架，要生成数据库表实体类只需要右键generatorConfig.xml->generate mybatis/IBATIS artifact即可。

1. Spring-boot接口开发

在创建新包时，需要将包创建在主包的下方，否则可能会出错。

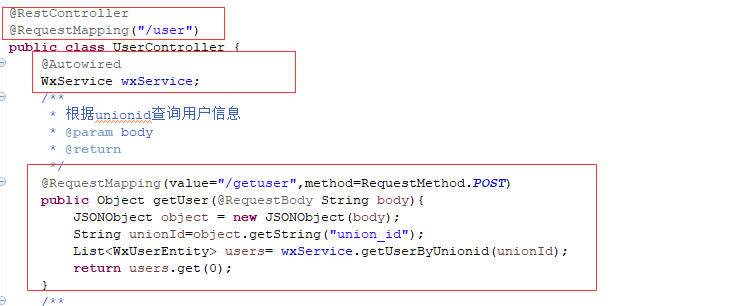


1. 创建service（名称可自定义）包，各数据库交互接口均在此编写。



@service表名本类是service类，可由其他类自动注入，@autowired注解表名自动注入对象。备注：XxxMapper是有mybatis自动生成的数据库映射接口。

1. 创建controller包，各个后台数据接口在此编写



@RestController表明本接口只提供restful服务

@RequestMapping("/user")表明访问路径

@Autowired

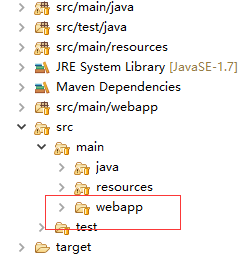
WxService wxService;表明自动注入数据库交互接口

@RequestMapping(value="/getuser",method=RequestMethod.POST) 表明本接口的访问路径为/user/getuser，类型为post.

至此一个基本的数据接口完成了，在前端可通过路径为ip:port/user/getuser，类型为post ajax请求访问本接口，并获取数据。

1. Spring-boot 静态资源访问

静态资源包含图片、文档、HTML网页等，spring-boot默认静态资源目录为/src/mian/webapp下，在/src/mian/目录下创建webapp目录



网站文件放置在该目录下，浏览器可通过url：ip:port/xxx.html访问网页。

1. Maven打包

Spring-boot项目可打成可执行的jar包，在打包之前需要在pom.xml文件中做些配置：

|  |
| --- |
| <build>  <finalName>zhlg</finalName>  <resources>  <resource>  <directory>${basedir}/src/main/webapp</directory>  <!-- 注意此次必须要放在此目录下才能被访问到 -->  <targetPath>META-INF/resources</targetPath>  <includes>  <include>\*\*/\*\*</include>  </includes>  </resource>  <resource>  <directory>${basedir}/src/main/resources</directory>  <includes>  <include>\*\*/\*\*</include>  </includes>  </resource>  </resources>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <configuration>  <source>1.7</source>  <target>1.7</target>  </configuration>  </plugin>  </plugins>  </build> |

finalName表示打成的jar包名称，且配置需要打进jar包的静态资源的目录。

然后右键项目，选择run as-> maven install即可在target目录下生成jar包。

1. Jar包发布
2. Nginx反向代理

使用nginx方向代理可将url路径映射到服务器端口，例如服务器域名为[www.example.com](http://www.example.com)，而spring-boot运行在8014端口，一般访问url为：[www.example.com:8014](http://www.example.com:8014)，但我们一般不这么访问，而是用路径[www.example.com/xxx](http://www.example.com/xxx)去访问8014端口，这就可用nginx方向代理经路径/xxx映射为8014端口。

1. 发布工具

Xshell和xftp，xshell用于连接服务器，并在本地打开服务器终端，xftp用于上传文件，如jar包。

1. Jar包发布

在将jar包上传到服务器后，可通过以下命令操作jar包

nohup java -jar xxx.jar >xxx.log & 命令可后台启动jar包(xxx.log是jar包的日志文件)

netstat -tulp|grep xxxx 命令查看占用端口的进程号

kill -9 xxxx 命令可杀死进程号对应的进程

tail -f xxx.log命令后台启动jar包可实时查看日志输出