**在Intellij IDEA集成ssm+swagger环境**

Date:2018-09-27 Author：张翼龙

目录

[**O.准备好MySql下的数据库** 2](#_Toc525920424)

[**一.配置Maven环境** 2](#_Toc525920425)

[**二.通过spring boot建立 ssm项目** 5](#_Toc525920426)

[**三.修改配置** 8](#_Toc525920427)

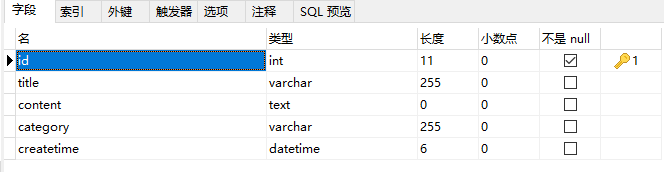
[**四.运行插件生成mapper、dao、和model** 14](#_Toc525920428)

[**五.运行ssm框架** 20](#_Toc525920429)

[**六.配置Swagger2完成api接口文档** 20](#_Toc525920430)

**O.准备好MySql下的数据库**

在MySql的test数据库下新建questions表用于环境搭建测试，表格结构如下,值可以自由填充



**一.配置Maven环境**

注Intellij IDEA默认集成了maven，本文档（2018-09-19）更新maven到最新版本3.5.4

1.下载apache-maven文件，选择自己需要的版本，地址：http://mirror.bit.edu.cn/apache/maven/maven-3/3.5.4/binaries/apache-maven-3.5.4-bin.zip

2.解压所下载文件，本人解压到：D:\maven\apache-maven-3.5.4

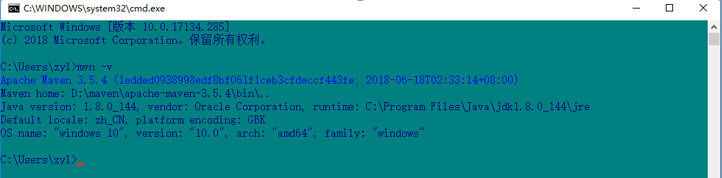
3.配置Maven系统环境变量

a. MAVEN\_HOME : D:\maven\apache-maven-3.5.4

b.PATH : %MAVEN\_HOME%\bin;

4.在CMD中输入mvn -v,如出现下列信息，表示配置成功。

其中会显示Java 配置环境以及Maven配置环境



5.创建D:\maven\中setting.xml文件，内容如下：

<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">

<pluginGroups />

<proxies />

<servers />

<localRepository>D:/maven/repository</localRepository>

<mirrors>

<mirror>

<id>alimaven</id>

<mirrorOf>central</mirrorOf>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/repositories/central/</url>

</mirror>

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

<mirrorOf>central</mirrorOf>

</mirror>

<mirror>

<id>central</id>

<name>Maven Repository Switchboard</name>

<url>http://repo1.maven.org/maven2/</url>

<mirrorOf>central</mirrorOf>

</mirror>

<mirror>

<id>repo2</id>

<mirrorOf>central</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://repo2.maven.org/maven2/</url>

</mirror>

<mirror>

<id>ibiblio</id>

<mirrorOf>central</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://mirrors.ibiblio.org/pub/mirrors/maven2/</url>

</mirror>

<mirror>

<id>jboss-public-repository-group</id>

<mirrorOf>central</mirrorOf>

<name>JBoss Public Repository Group</name>

<url>http://repository.jboss.org/nexus/content/groups/public</url>

</mirror>

<mirror>

<id>google-maven-central</id>

<name>Google Maven Central</name>

<url>https://maven-central.storage.googleapis.com

</url>

<mirrorOf>central</mirrorOf>

</mirror>

<!-- 中央仓库在中国的镜像 -->

<mirror>

<id>maven.net.cn</id>

<name>oneof the central mirrors in china</name>

<url>http://maven.net.cn/content/groups/public/</url>

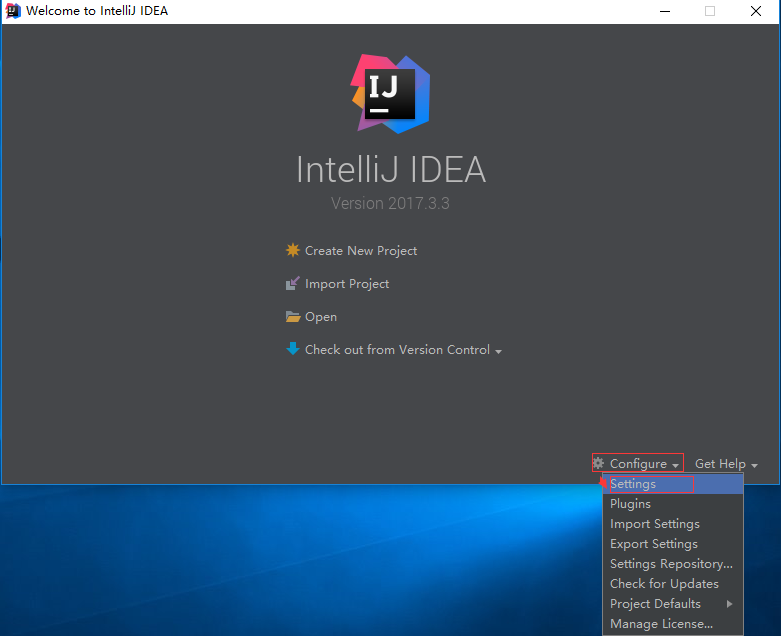
<mirrorOf>central</mirrorOf>

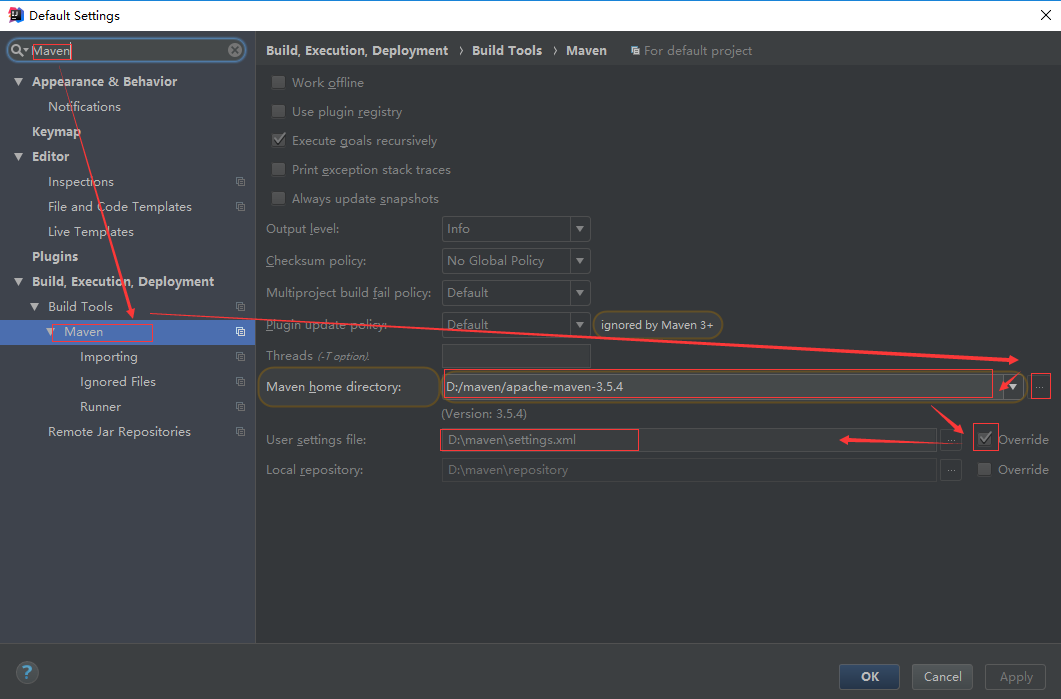
</mirror>

</mirrors>

</settings>

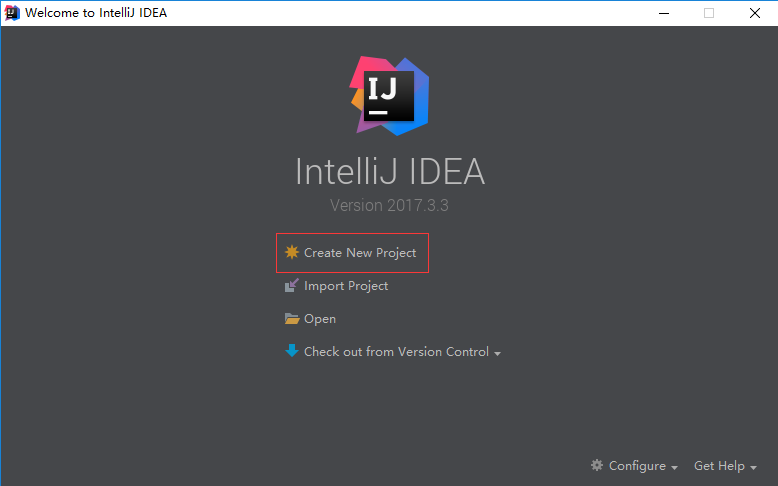
6 在IntellIJ IDEA 中配置Maven，搜索Maven按照如图配置



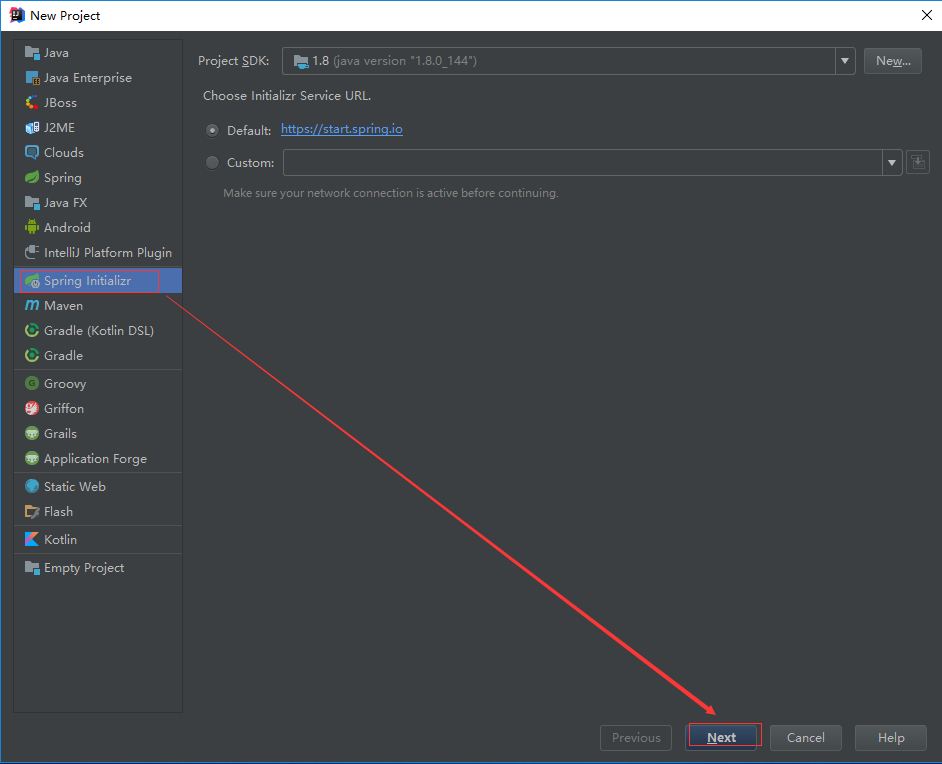


**二.通过spring boot建立 ssm项目**

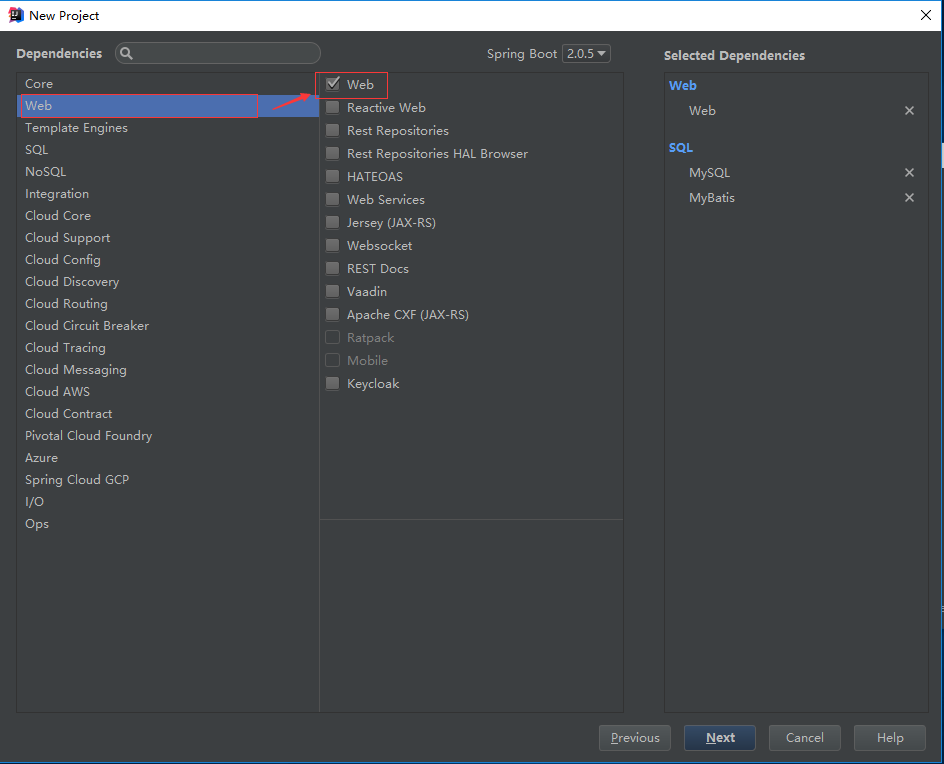
1. 创建一个maven的web工程。

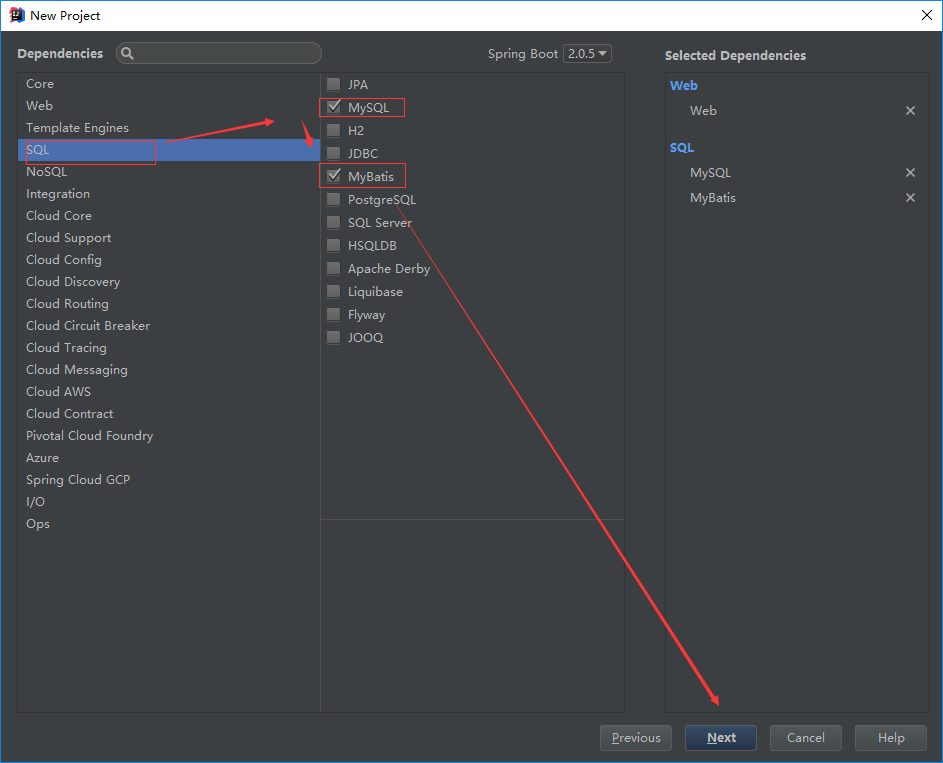


选择spring initializr

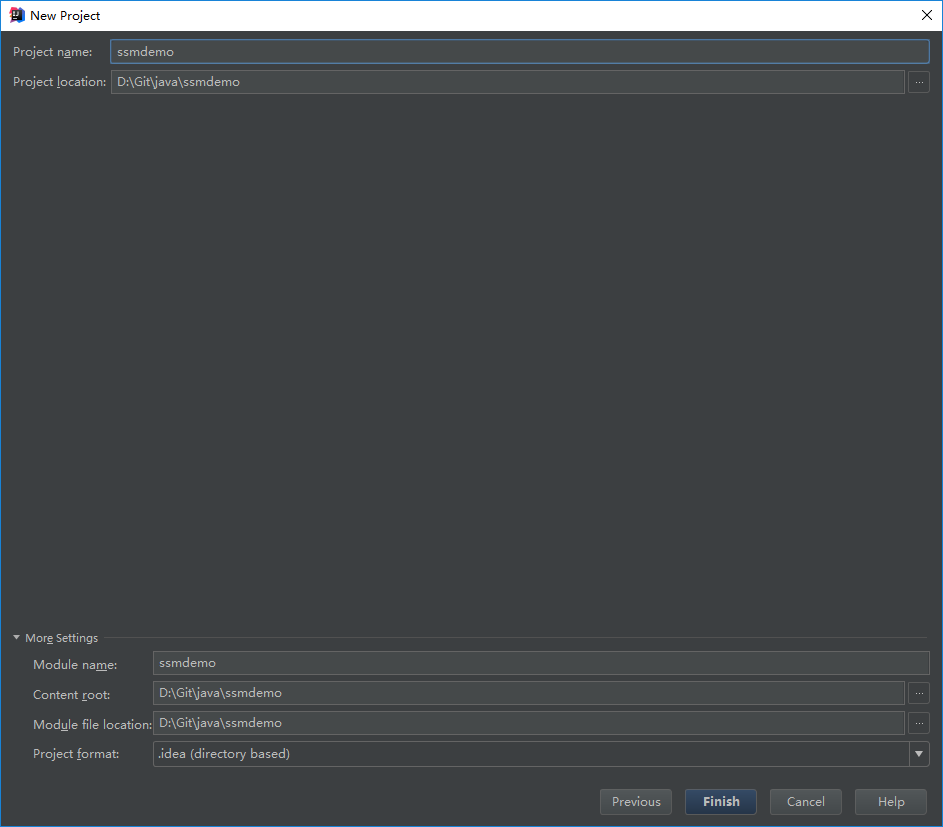


Web中选择Web,SQL中选择MySQL和MyBatis

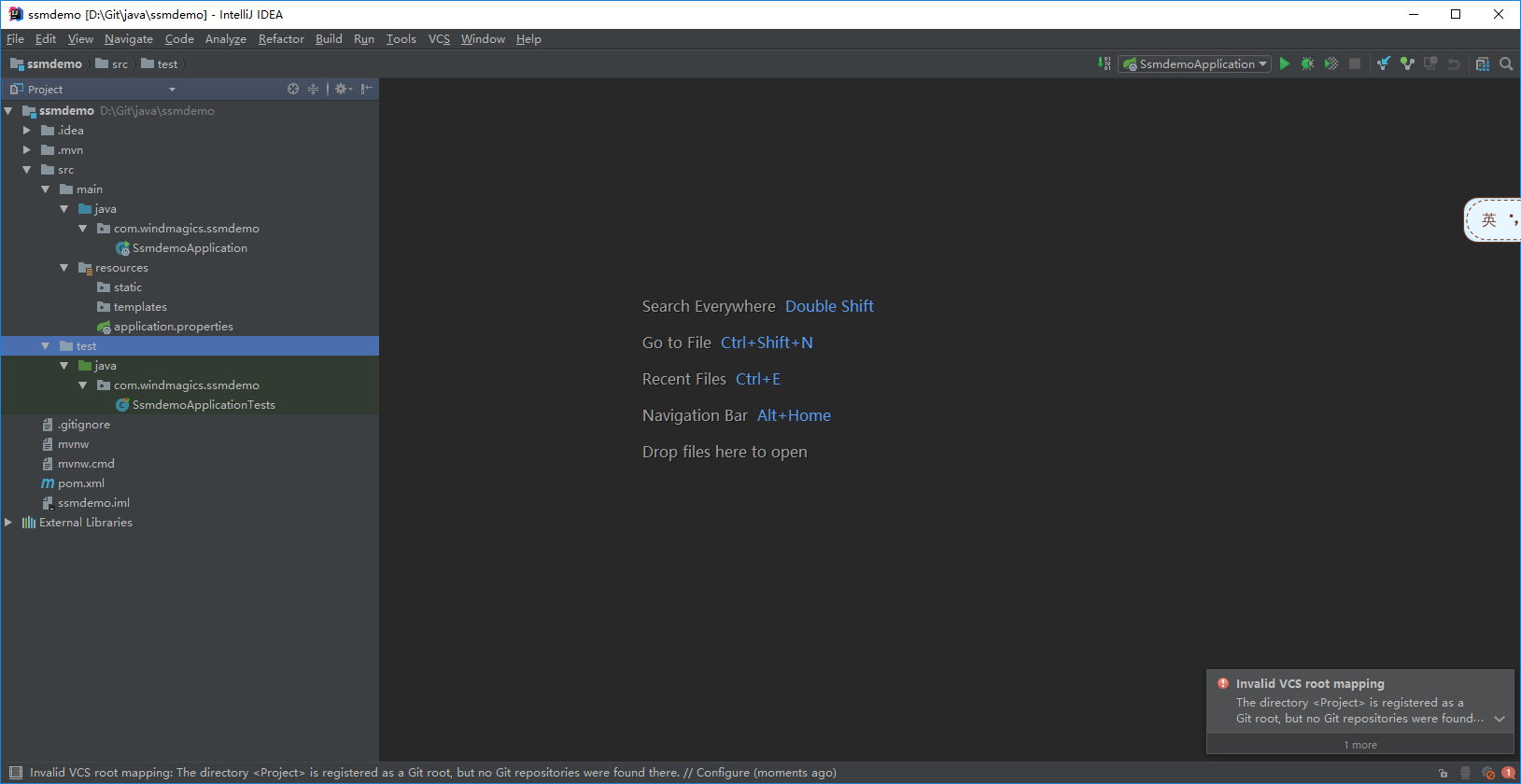




项目名称默认即可



完成后的项目结构



**三.修改配置**

1更改项目文件Pom.xml内容，在原来的配置基础上我指定了mysql的version，根据以后的需要加入了fastjson（本示例中没有用到fastjson），,druid,swagger,swagger-ui依赖，你可以自己决定添加或保留,配置了maven构建下的mybatis.generator插件和war包生成插件（本示例中生成war过程没有讲述）最终内容如下

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.windmagics</groupId>

<artifactId>ssmdemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>ssmdemo</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.5.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

<mysql.version>5.1.29</mysql.version>

</properties>

<dependencies>

<!-- web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- mybatis -->

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>1.3.2</version>

</dependency>

<!-- mysql -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<!-- test -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/com.alibaba/fastjson -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.2.49</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.alibaba/druid -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.1.11</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.9.2</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- maven -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

<!-- mybatis.generator -->

<plugin>

<groupId>org.mybatis.generator</groupId>

<artifactId>mybatis-generator-maven-plugin</artifactId>

<version>1.3.7</version>

<dependencies>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>${mysql.version}</version>

<scope>runtime</scope>

</dependency>

</dependencies>

<configuration>

<!-- 打印执行过程 -->

<verbose>true</verbose>

<!-- 允许覆盖生成的文件 -->

<overwrite>true</overwrite>

<!-- 配置文件路径 -->

<configurationFile>${basedir}/src/main/resources/generatorConfig.xml</configurationFile>

<!-- 生成部分mapper,指定table名字 -->

<!--<tableNames>-->

<!--t\_author-->

<!--</tableNames>-->

</configuration>

</plugin>

<!-- war -->

<plugin>

<artifactId>maven-war-plugin</artifactId>

<version>3.2.0</version>

</plugin>

</plugins>

</build>

</project>

2 新增mybatis-generator插件所用的配置文件generatorConfig.xml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">

<generatorConfiguration>

<context id="DB2Tables" targetRuntime="MyBatis3">

<!-- 注释 规则-->

<commentGenerator>

<property name="suppressAllComments" value="true"/>

<property name="suppressDate" value="true"/>

</commentGenerator>

<!-- 数据库连接 -->

<jdbcConnection driverClass="com.mysql.jdbc.Driver"

connectionURL="jdbc:mysql://localhost:3306/test?useUnicode=true&amp;characterEncoding=UTF-8"

userId="root"

password="windmagics">

</jdbcConnection>

<!-- 数据表对应的model层,生成实体类 -->

<javaModelGenerator targetPackage="com.windmagics.ssmdemo.model" targetProject="src/main/java">

<!-- enableSubPackages:是否让schema作为包的后缀 -->

<property name="enableSubPackages" value="true" />

<!-- 从数据库返回的值被清理前后的空格 -->

<property name="trimStrings" value="true" />

</javaModelGenerator>

<!-- sql mapper 映射配置文件 -->

<sqlMapGenerator targetPackage="mapper" targetProject="src/main/resources">

<property name="enableSubPackages" value="true" />

</sqlMapGenerator>

<!-- mybatis3中的mapper接口 -->

<javaClientGenerator type="XMLMAPPER" targetPackage="com.windmagics.ssmdemo.dao" targetProject="src/main/java">

<!-- enableSubPackages:是否让schema作为包的后缀 -->

<property name="enableSubPackages" value="true" />

</javaClientGenerator>

<!-- 数据表进行生成操作 schema:相当于库名; tableName:表名; domainObjectName:对应的DO -->

<table schema="test" tableName="questions" domainObjectName="Question"

enableCountByExample="false" enableUpdateByExample="false"

enableDeleteByExample="false" enableSelectByExample="false"

selectByExampleQueryId="false">

</table>

</context>

</generatorConfiguration>

3 配置数据库采用yaml文件格式进行配置，视觉上更简洁

application.properties 改名为application.yml，增加application-dev.yml文件

其中application-dev.yml文件内容如下（注意缩进）

spring:

datasource:

name: dev

url: jdbc:mysql://localhost:3306/test

username: root

password: windmagics

其中application.yml文件内容如下（注意缩进）

server:

port: 8480

#Spring

spring:

application:

name: test

#profile

profiles:

active: dev

#datasource

datasource:

# druid

type: com.alibaba.druid.pool.DruidDataSource

driver-class-name: com.mysql.jdbc.Driver

filters: stat

maxActive: 20

initialSize: 1

maxWait: 60000

minIdle: 1

timeBetweenEvictionRunsMillis: 60000

minEvictableIdleTimeMillis: 300000

validationQuery: select 'x'

testWhileIdle: true

testOnBorrow: false

testOnReturn: false

poolPreparedStatements: true

maxOpenPreparedStatements: 20

#mybatis

mybatis:

## 主要是为了让框架扫描到mapper文件

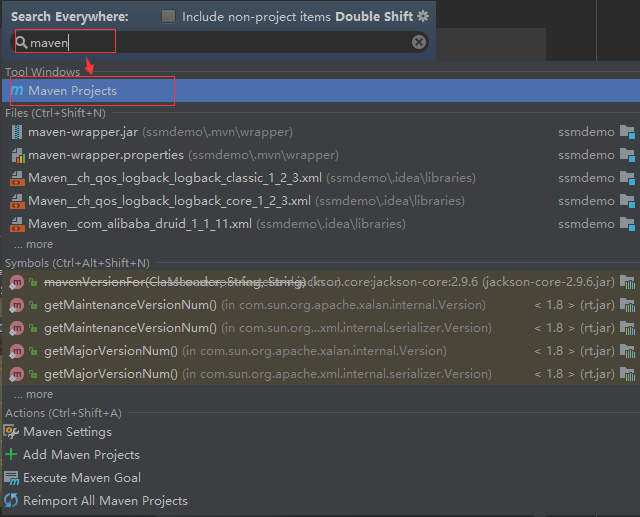
mapper-locations: classpath\*:/mapper/\*\*Mapper.xml

##起别名。可省略写mybatis的xml中的resultType的全路径

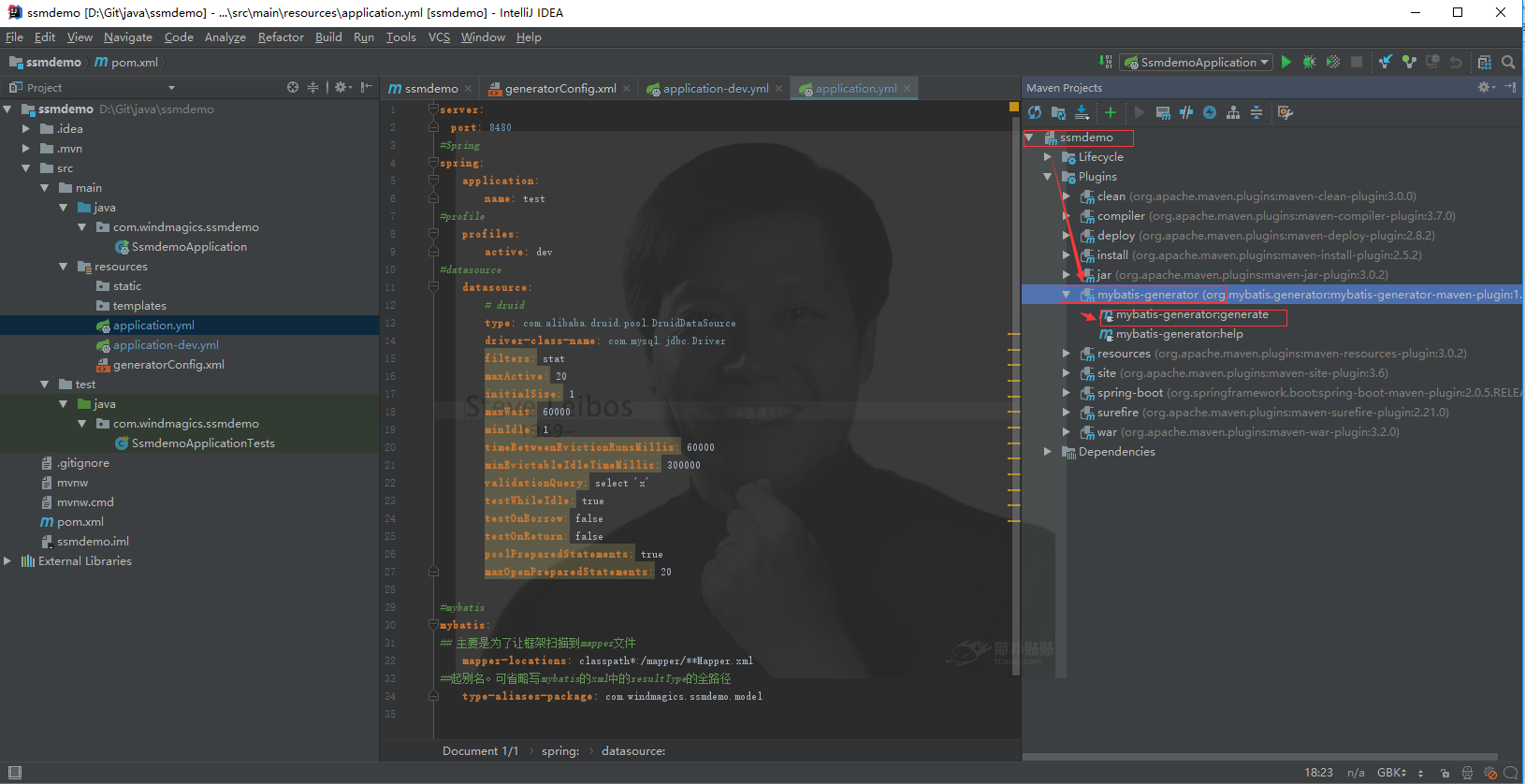
type-aliases-package: com.windmagics.ssmdemo.model

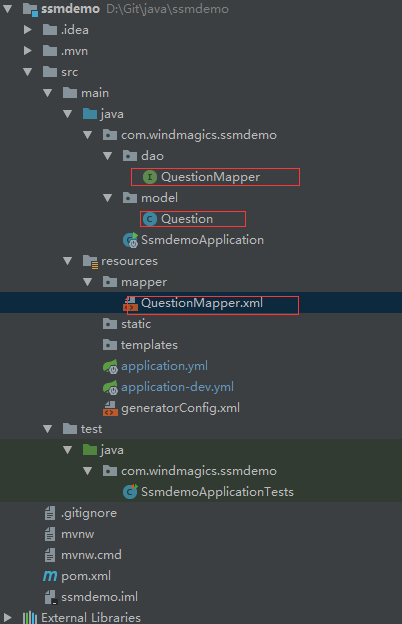
**四.运行插件生成mapper、dao、和model**

双击shift键调出快速搜索工具栏，输入maven找到maven projects工具

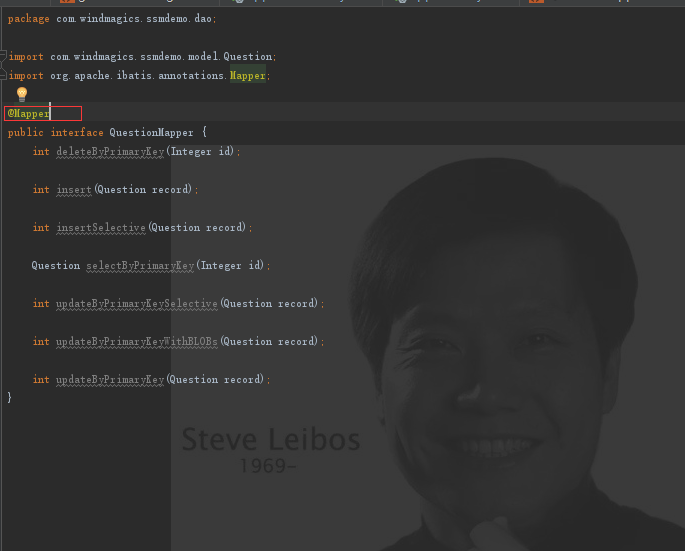


找到mybatis-generator插件下的mybatis-generator:generator命令双击进行执行



执行成功后会多出dao，model, resources/mapper文件夹和相应文件

打开QuestionMapper加入@Mapper注解



在com.windmagics.ssmdemo下建立controller和service文件夹  
service文件夹下分别新建QuestionService接口和QuestionServiceImpl实现

其中QuestionService接口文件内容如下

package com.windmagics.ssmdemo.service;  
import com.windmagics.ssmdemo.model.Question;  
import org.springframework.stereotype.Service;  
  
  
public interface QuestionService {  
 public Question getQuestions();  
}

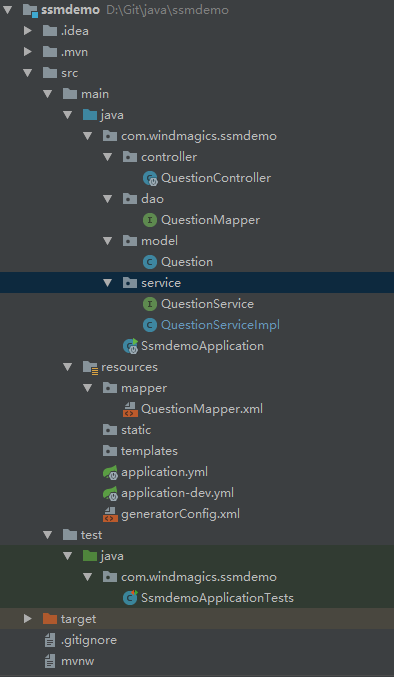
其中QuestionServiceImpl类文件内容如下（作为示例，我加入electByPrimaryKey()里id的值1）记得加入@service注解

package com.windmagics.ssmdemo.service;  
  
  
  
import com.windmagics.ssmdemo.dao.QuestionMapper;  
import com.windmagics.ssmdemo.model.Question;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
  
@Service  
public class QuestionServiceImpl implements QuestionService {  
 @Autowired  
 private QuestionMapper questionMapper;  
 @Override  
 public Question getQuestions() {  
  
 Question question = questionMapper.selectByPrimaryKey(1);  
   
 return question;  
 }  
}

在controller目录下建立QuestionController控制器,内容如下，同样，加入@RestController注解及其他相应注解

package com.windmagics.ssmdemo.controller;  
  
  
  
   
import com.windmagics.ssmdemo.model.Question;   
import com.windmagics.ssmdemo.service.QuestionService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
  
  
   
@RestController  
@EnableAutoConfiguration  
@RequestMapping("/question")  
public class QuestionController {  
 @Autowired  
 private QuestionService questionService;  
  
  
 @GetMapping("getquestions")  
 public Question getQuestions()  
 {  
 return questionService.getQuestions();  
  
 }  
  
  
}

完成后项目目录如下



**五.运行ssm框架**

在浏览器中输入localhost:8480/question/getquestions结果如下，完成配置

{

* **id**: 1,
* **title**: "我是title",
* **category**: null,
* **createtime**: null,
* **content**: "我是内容"

}

**六.配置Swagger2完成api接口文档**

注：Swagger2可自动生成(前面的pom.xml中我们已经加入了swagger依赖)

在com.windmagics.ssmdemo下建立config文件夹，在config文件夹下建立SwaggerConfig类，文件内容如下

package com.windmagics.ssmdemo.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.ApiInfoBuilder;

import springfox.documentation.builders.PathSelectors;

import springfox.documentation.builders.RequestHandlerSelectors;

import springfox.documentation.service.ApiInfo;

import springfox.documentation.service.Contact;

import springfox.documentation.spi.DocumentationType;

import springfox.documentation.spring.web.plugins.Docket;

import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration

@EnableSwagger2

public class SwaggerConfig {

@Bean

public Docket createRestApi() {

return new Docket(DocumentationType.SWAGGER\_2)

.apiInfo(apiInfo())

.select()

.apis(RequestHandlerSelectors.basePackage("com.windmagics.ssmdemo.controller"))//暴露指定包下的接口说明，可以指定多个包

.paths(PathSelectors.any())

.build();

}

private ApiInfo apiInfo() {

return new ApiInfoBuilder()

.title("webapp接口文档")

.description("webapp开放接口相关文档，仅供学习，研究使用！")

.termsOfServiceUrl("http://github.com/lufever")

.contact(new Contact("zyl", "http://github.com/lufever", "xxx@xx.com"))

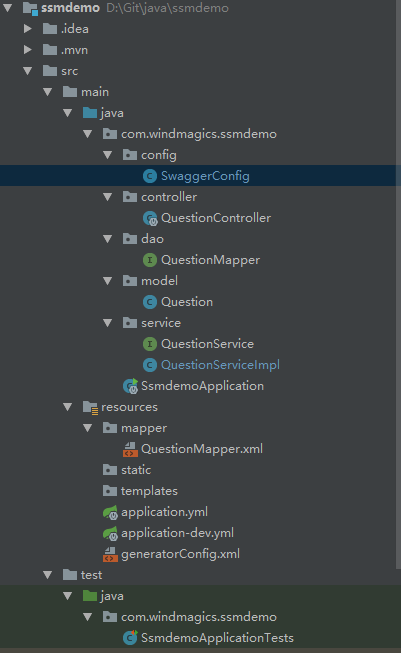
.version("1.0.0")

.build();

}

}

最终目录如下



在QuestionController中加入swagger api接口注解，加入后QuestionController文件内容如下

package com.windmagics.ssmdemo.controller;  
  
import com.windmagics.ssmdemo.model.Question;  
import com.windmagics.ssmdemo.service.QuestionService;  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiOperation;  
import io.swagger.annotations.ApiResponse;  
import io.swagger.annotations.ApiResponses;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;  
import org.springframework.http.MediaType;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import javax.xml.ws.Response;  
  
  
@RestController  
@EnableAutoConfiguration  
@RequestMapping("/question")  
@Api(tags = "测试", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
public class QuestionController {  
 @Autowired  
 private QuestionService questionService;  
  
 @ApiOperation(value = "获取id为1的数据", produces = MediaType.*APPLICATION\_JSON\_VALUE*)  
 @ApiResponses(value = {@ApiResponse(code = 0, message = "ok", response = Response.class)})  
 @GetMapping("getquestions")  
 public Question getQuestions()  
 {  
 return questionService.getQuestions();  
  
 }  
  
  
}

打开<http://localhost:8480/swagger-ui.html>

出现界面如下，完成配置

