**使用SpringBoot集成SwaggerUi与MySql实现快速API**

1. **开发环境搭建**

开发环境：

Java 1.8

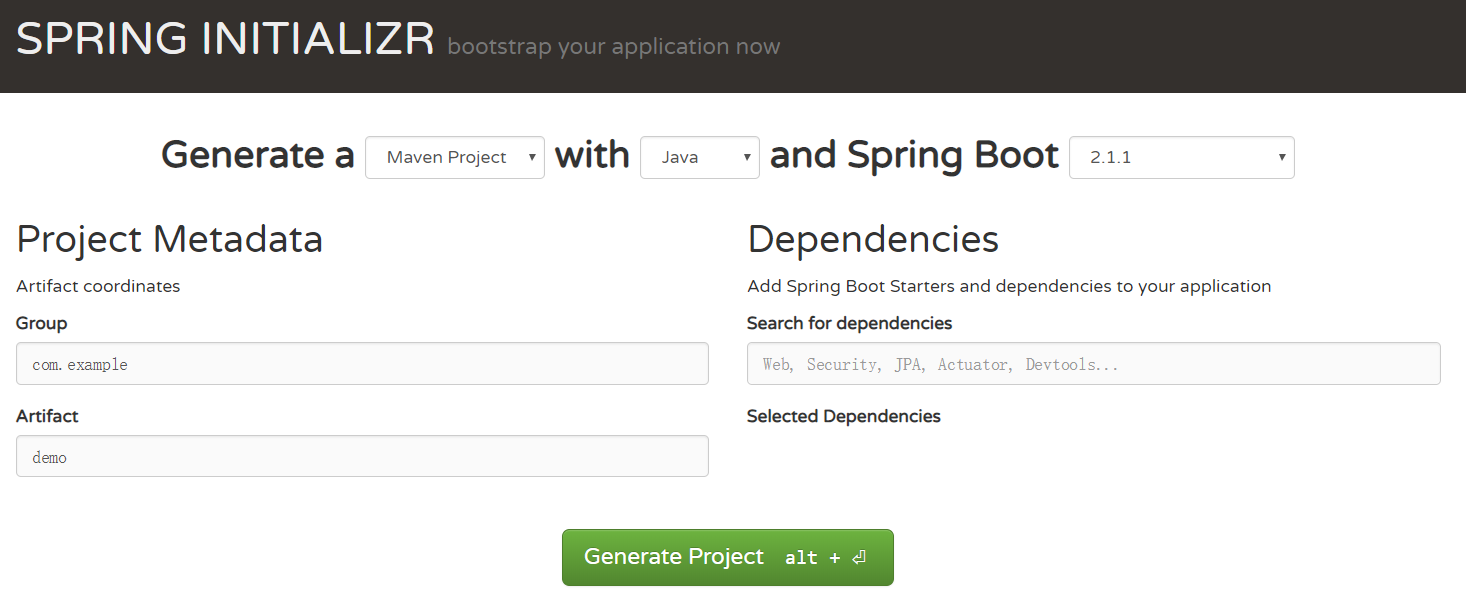
spring-2.0.0.RELEASE

mysql-8.0.13

Maven 3.6.0

1. **使用Maven构建项目**

访问：http://start.spring.io/，通过SPRING INITIALIZR工具产生基础项目



解压项目包，并用IDE以Maven项目导入，菜单中选择File–>New–>Project from Existing Sources，选择解压后的项目文件夹，点击OK，点击Import project from external model并选择Maven，点击Next到底为止。

通过上面步骤完成了基础项目的创建，如上图所示，Spring Boot的基础结构共三个文件

src/main/java下的程序入口：Application

src/main/resources下的配置文件：application.properties

src/test/下的测试入口：ApplicationTests

1. **引入依赖**

*<!--SpringBoot依赖-->*

<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-autoconfigure</**artifactId**>  
 <**version**>2.1.1.RELEASE</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-autoconfigure</**artifactId**>  
 <**version**>2.1.1.RELEASE</**version**>  
 </**dependency**>

*<!--MySQL数据库依赖-->*  
<**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 <**version**>5.1.47</**version**>  
</**dependency**>

*<!--Swagger依赖-->*

<**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger2</**artifactId**>  
 <**version**>2.9.2</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger-ui</**artifactId**>  
 <**version**>2.9.2</**version**>  
 </**dependency**>  
<**dependency**>  
 <**groupId**>com.github.xiaoymin</**groupId**>  
 <**artifactId**>swagger-bootstrap-ui</**artifactId**>  
 <**version**>1.8.8</**version**>  
</**dependency**>

*<!--lombok依赖-->*  
<**dependency**>  
 <**groupId**>org.projectlombok</**groupId**>  
 <**artifactId**>lombok</**artifactId**>  
 <**version**>1.16.18</**version**>  
 <**scope**>provided</**scope**>  
</**dependency**>

**4、创建Swagger2配置类**

在Application.java同级创建Swagger2的配置类Swagger2。  
@Configuration  
@EnableSwagger2  
**public class** Swagger2 {  
 @Bean  
 **public** Docket createRestApi() {  
 **return new** Docket(DocumentationType.***SWAGGER\_2***)  
 .apiInfo(apiInfo())  
 .select()  
 .apis(RequestHandlerSelectors.*basePackage*(**"com.example.controller"**))  
 .paths(PathSelectors.*any*())  
 .build();  
 }  
  
 **private** ApiInfo apiInfo() {  
 ApiInfo apiInfo = **new** ApiInfo(**"公司名"**,  
 **"更多详细信息请登录官方网站：url"**,  
 **"版本1.0"**,  
 **"http://www.google.com/"**,  
 **"联系人邮箱"**,  
 **""**,  
 **"http://www.google.com/"**);  
 **return** apiInfo;  
 }  
}

通过@Configuration注解，让Spring来加载该类配置。再通过@EnableSwagger2注解来启用Swagger2。通过createRestApi函数创建Docket的Bean之后，apiInfo()用来创建该Api的基本信息（这些基本信息会展现在文档页面中）

**5、配置数据库信息**

创建Database类，储存JDBC 驱动名及数据库 URL和数据库的用户名与密码

**package** com.example.web;  
*/\*  
数据库信息  
 \*/***public class** Database {  
 *// JDBC 驱动名及数据库 URL* **public static final** String ***JDBC\_DRIVER*** = **"com.mysql.jdbc.Driver"**;  
 **public static final** String ***DB\_URL*** = **"jdbc:mysql://localhost:3306/college?serverTimezone=GMT"**;  
  
 *// 数据库的用户名与密码，需要根据自己的设置* **public static final** String ***USER*** = **"root"**;  
 **public static final** String ***PASS*** = **"password"**;  
}

**6、创建实体类**

本例共5个实体类:Institute（学院），Major（专业），Course（课程），Student（学生），Teacher（教师）。

各实体类详细信息如下

**Institute（学院）：**

@Entity

@Getter  
@Setter

**public class** Institute {  
 **private** Long **instituteId**;  
 **private** String **instituteName**;  
 **private** Integer **numberOfMajor**;  
 **private** ArrayList<Major> **majorArrayList**;  
}

**Major（专业）：**

@Entity

@Getter  
@Setter

**public class** Major {  
 **private** Long **majorId**;  
 **private** String **majorName**;  
 **private** Long **instituteId**;  
 **private** String **instituteName**;  
 **private** ArrayList<Course> **courseArrayList**;

}

**Course（课程）：**

@Entity

@Getter  
@Setter

**public class** Course {  
 **private** Long **courseId**;  
 **private** String **courseName**;  
 **private** Long **majorId**;  
 **private** String **majorName**;  
 **private** Long **teacherId**;  
 **private** String **teacherName**;

}

**Student（学生）：**

@Entity

@Getter  
@Setter  
**public class** Student {  
 **private** Long **studentId**;  
 **private** String **studentName**;  
 **private** Long **majorId**;  
 **private** String **studentSex**;  
 **private** String **majorName**;

}

**Teacher（教师）：**

@Entity

@Getter  
@Setter  
**public class** Teacher {  
 **private** Long **teacherId**;  
 **private** String **teacherName**;  
 **private** String **teacherSex**;  
 **private** Long **instituteId**;  
 **private** String **instituteName**;

}

**7、数据库方法创建**

每个实体类有四种方法，getMap：从数据库获取信息delete：从数据库删除信息，save：保存新的信息，update:更新信息。

以下仅给出Institute（学院）的数据库方法及其他实体类的Sql语句。

**Institute（学院）：**

**public class** InstituteDatabase {

*/\*  
获得数据库内容  
 \*/*

**public static** Map<Long,Institute> getInstituteMap() {  
 Map<Long,Institute> instituteMap=**new** HashMap<>();  
 Map<Long,Major> majorMap=**new** HashMap<>();  
 Connection conn = **null**;  
 Statement stmt = **null**;  
 **try** {  
 *// 注册 JDBC 驱动* Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 *// 打开链接* conn = DriverManager.*getConnection*(Database.***DB\_URL***, Database.***USER***, Database.***PASS***);  
 *// 执行查询* stmt = conn.createStatement();  
 String sql;  
 sql = **"SELECT instituteId,instituteName,numberOfMajor FROM institute "**;  
 ResultSet instituteRs = stmt.executeQuery(sql);  
 *// 展开结果集数据库* **while** (instituteRs.next()) {  
 **long** instituteId = instituteRs.getLong(**"instituteId"**);  
 String instituteName = instituteRs.getString(**"instituteName"**);  
 **int** numberOfMajor = instituteRs.getInt(**"numberOfMajor"**);  
 Institute i = **new** Institute();  
 i.setInstituteName(instituteName);  
 i.setInstituteId((instituteId));  
 i.setNumberOfMajor(numberOfMajor);  
 i.setMajorArrayList(**new** ArrayList<>());  
 instituteMap.put(instituteId,i);  
 }  
 sql = **"SELECT majorId,instituteId FROM major"**;  
 ResultSet majorRs = stmt.executeQuery(sql);  
 **while** (majorRs.next()) {  
 **long** majorId = majorRs.getLong(**"majorId"**);  
 **long** instituteId = majorRs.getLong(**"instituteId"**);  
 instituteMap.get(instituteId).getMajorArrayList().add(MajorDatabase.*getMajorMap*().get(majorId));  
 }  
 majorRs.close();  
 instituteRs.close();  
 stmt.close();  
 conn.close();  
 } **catch** (SQLException se) {  
 *// 处理 JDBC 错误* se.printStackTrace();  
 } **catch** (Exception e) {  
 *// 处理 Class.forName 错误* e.printStackTrace();  
 } **finally** {  
 *// 关闭资源* **try** {  
 **if** (stmt != **null**) stmt.close();  
 } **catch** (SQLException se2) {  
 }  
 **try** {  
 **if** (conn != **null**) conn.close();  
 } **catch** (SQLException se) {  
 se.printStackTrace();  
 }  
 }  
 **return** instituteMap;  
 }  
*/\*  
保存数据库内容  
 \*/*

**public static** String saveInstitute(Institute Institute) {  
 Map<Long, Institute> userMap=**new** HashMap<>();  
 Connection conn = **null**;  
 PreparedStatement stmt = **null**;  
 **try** {  
 *// 注册 JDBC 驱动* Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 *// 打开链接* conn = DriverManager.*getConnection*(Database.***DB\_URL***, Database.***USER***, Database.***PASS***);  
 *// 执行查询* String sql;  
 sql = **"insert into institute values (?,?,?)"**;  
 stmt = conn.prepareStatement(sql);  
 stmt.setLong(1, Institute.getInstituteId());  
 stmt.setString(2, Institute.getInstituteName());  
 stmt.setInt(3, Institute.getNumberOfMajor());  
 **try** {  
 stmt.executeUpdate();  
 }  
 **catch** (SQLIntegrityConstraintViolationException e){  
 **return "保存失败"**;  
 }  
 stmt.close();  
 conn.close();  
 } **catch** (SQLException se) {  
 *// 处理 JDBC 错误* se.printStackTrace();  
 } **catch** (Exception e) {  
 *// 处理 Class.forName 错误* e.printStackTrace();  
 **return "失败"**;  
 } **finally** {  
 *// 关闭资源* **try** {  
 **if** (stmt != **null**) stmt.close();  
 } **catch** (SQLException se2) {  
 }*// 什么都不做* **try** {  
 **if** (conn != **null**) conn.close();  
 } **catch** (SQLException se) {  
 se.printStackTrace();  
 }  
 }  
 **return "保持成功"**;  
 }

*/\*  
删除数据库内容  
 \*/*  
 **public static** String deleteInstitute(Long id) {  
 Map<Long,Institute> userMap=**new** HashMap<>();  
 Connection conn = **null**;  
 PreparedStatement stmt = **null**;  
 **try** {  
 *// 注册 JDBC 驱动* Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 *// 打开链接* conn = DriverManager.*getConnection*(Database.***DB\_URL***, Database.***USER***, Database.***PASS***);  
 *// 执行查询* String sql;  
 sql = **"delete from institute where instituteId=?"**;  
 stmt = conn.prepareStatement(sql);  
 stmt.setLong(1,id);  
 stmt.executeUpdate();  
 stmt.close();  
 conn.close();  
 } **catch** (SQLException se) {  
 *// 处理 JDBC 错误* se.printStackTrace();  
 } **catch** (Exception e) {  
 *// 处理 Class.forName 错误* e.printStackTrace();  
 **return "失败"**;  
 } **finally** {  
 *// 关闭资源* **try** {  
 **if** (stmt != **null**) stmt.close();  
 } **catch** (SQLException se2) {  
 }*// 什么都不做* **try** {  
 **if** (conn != **null**) conn.close();  
 } **catch** (SQLException se) {  
 se.printStackTrace();  
 }  
 }  
 **return "删除成功"**;  
 }  
*/\*  
更新数据库内容  
 \*/*  
 **public static** String updateInstitute( Long id, Institute institute) {  
 Connection conn = **null**;  
 PreparedStatement stmt = **null**;  
 **try**{  
 *// 注册 JDBC 驱动* Class.*forName*(Database.***JDBC\_DRIVER***);  
 *// 打开链接* conn = DriverManager.*getConnection*(Database.***DB\_URL***,Database.***USER***,Database.***PASS***);  
 *// 执行* System.***out***.println(**"执行更新功能"**);  
 String sql;  
 sql = **"update institute set instituteId=?,instituteName=?,numberOfMajor=? where instituteId = ?"**;  
 stmt = conn.prepareStatement(sql);  
 stmt.setLong(1,institute.getInstituteId());  
 stmt.setString(2,institute.getInstituteName());  
 stmt.setLong(3,institute.getNumberOfMajor());  
 stmt.setLong(4,id);  
 stmt.executeUpdate();  
 *// 完成后关闭* stmt.close();  
 conn.close();  
 }**catch**(SQLException se){  
 *// 处理 JDBC 错误* se.printStackTrace();  
 }**catch**(Exception e){  
 *// 处理 Class.forName 错误* e.printStackTrace();  
 **return "失败"**;  
 }**finally**{  
 *// 关闭资源* **try**{  
 **if**(stmt!=**null**) stmt.close();  
 }**catch**(SQLException se2){  
 }  
 **try**{  
 **if**(conn!=**null**) conn.close();  
 }**catch**(SQLException se){  
 se.printStackTrace();  
 }  
 }  
 **return "更新详细信息成功"**;  
 }  
}

**Major(专业):**

getMap Sql语句：

**SELECT majorId,majorName,major.instituteId,instituteName FROM major inner join institute where major.instituteId=institute.instituteId**

**SELECT courseId,majorId FROM course**

Delete Sql语句：

**delete from major where majorId=?**

Insert Sql 语句：

**insert into major values (?,?,?)**

Update Sql 语句：

**update major set majorId=?,majorName=?,instituteId=? where majorId = ?**

**Course（课程）：**

getMap Sql语句：

**SELECT courseId,courseName,course.majorId,majorName,course.teacherId,teacher.teacherName from course inner join major,teacher where major.majorId=course.majorId and course.teacherId=teacher.teacherId**

Delete Sql语句：

**delete from course where courseId=?**

Insert Sql 语句：

**insert into course values (?,?,?,?)**

Update Sql 语句：

**update course set courseId=?,courseName=?,majorId=?,teacherId=? where courseId =?**

**Student（学生）：**

getMap Sql语句：

**SELECT studentId,studentName,studentSex,student.majorId,majorName FROM student inner join major WHERE major.majorId=student.majorId**

Delete Sql语句：

**delete from student where studentId=?**

Insert Sql 语句：

**insert into student values (?,?,?,?)**

Update Sql 语句：

**update student set studentId=?,studentName=?,studentSex=?,majorId=? where studentId = ?**

**Teacher（教师）：**

getMap Sql语句：

**SELECT teacherId,teacherName,teacherSex,teacher.instituteId,instituteName from teacher inner join institute where teacher.instituteId=institute.instituteId**

Delete Sql语句：

**delete from teacher where teacherId=?**

Insert Sql 语句：

**insert into teacher values (?,?,?,?)**

Update Sql 语句：

**update teacher set teacherId=?,teacherName=?,teacherSex=?,instituteId=? where teacherId = ?**

1. **Controller配置**

Controller包下共有5个类：CourseController，MajorController，InstituteController，StudentController，TeacherController。

以InstituteController为例说明类的配置：

@RestController  
@RequestMapping(value=**"/学院"**)  
**public class** InstituteController {  
 **private** Map<Long, Institute> **instituteMap** = **new** HashMap<>();  
 **public** InstituteController() {  
 **this**.**instituteMap** = InstituteDatabase.*getInstituteMap*();  
 }  
 */\*  
 institute  
 \*/* @ApiOperation(value = **"获取全部学院信息"**, notes = **"根据数据库信息获得信息"**, position = 1)  
 @RequestMapping(value = **""**, method = RequestMethod.***GET***)  
 **public** List<Institute> importInstitute() {  
 System.***out***.println(**"执行获取全部学院信息成功"**);  
 List<Institute> institutes = **new** ArrayList<Institute>(**instituteMap**.values());  
 **return** institutes;  
 }  
  
 @ApiOperation(value = **"创建学院"**, notes = **"根据数据库信息创建学院"**, position = 2)  
 @ApiImplicitParam(name = **"institute"**, value = **"学院详细实体"**, required = **true**, dataType = **"Institute"**)  
 @ApiResponses({  
 @ApiResponse(code = 404, message = **"无请求路径或页面跳转路径不对"**)  
 })  
 @RequestMapping(value = **""**, method = RequestMethod.***POST***)  
 **public** String postInstitute(@RequestBody Institute institute) {  
 **instituteMap**.put(institute.getInstituteId(), institute);  
 **return** InstituteDatabase.*saveInstitute*(institute);  
 }  
  
 @ApiOperation(value = **"通过id获取学院详细信息"**, notes = **"根据用户的id来获取用户详细信息"**)  
 @ApiImplicitParam(name = **"instituteId"**, value = **"学院id"**, required = **true**, dataType = **"Long"**, paramType = **"path"**)  
 @RequestMapping(value = **"/{instituteId}"**, method = RequestMethod.***GET***)  
 **public** Institute getInstitute(@PathVariable Long instituteId) {  
 System.***out***.println(**"执行查询功能"**);  
 **return instituteMap**.get(instituteId);  
 }  
  
 @ApiOperation(value = **"更新学院详细信息"**, notes = **"根据学院的id来指定更新对象"**)  
 @ApiImplicitParams({  
 @ApiImplicitParam(name = **"instituteId"**, value = **"学院id"**, required = **true**, dataType = **"Long"**, paramType = **"path"**),  
 @ApiImplicitParam(name = **"institute"**, value = **"学院信息"**, required = **true**, dataType = **"Institute"**)  
 })  
 @ApiResponses({  
 @ApiResponse(code = 500, message = **"错误，可能为不存在数据库的用户id"**)  
 })  
 @RequestMapping(value = **"/{instituteId}"**, method = RequestMethod.***PUT***)  
 **public** String putUser(@PathVariable Long instituteId, @RequestBody Institute institute) {  
 Institute t = **instituteMap**.get(instituteId);  
 t.setInstituteId(institute.getInstituteId());  
 t.setInstituteName(institute.getInstituteName());  
 t.setNumberOfMajor(institute.getNumberOfMajor());  
 t.setMajorArrayList(institute.getMajorArrayList());  
 **return** InstituteDatabase.*updateInstitute*(instituteId,institute);  
 }  
  
 @ApiOperation(value = **"删除学院"**, notes = **"根据学院的id来指定删除对象"**)  
 @ApiImplicitParam(name = **"instituteId"**, value = **"学院id"**, required = **true**, dataType = **"Long"**, paramType = **"path"**)  
 @RequestMapping(value = **"/{instituteId}"**, method = RequestMethod.***DELETE***)  
 **public** String deleteUser(@PathVariable Long instituteId) {  
 **instituteMap**.remove(instituteId);  
 **return** InstituteDatabase.*deleteInstitute*(instituteId);  
 }  
}

**9、访问页面**

本例在pom.xml 中引入了swagger-ui-layer 的依赖（来源：https://github.com/xiao

ymin/Swagger-Bootstrap-UI）。

完成上述代码，启动Spring Boot程序，访问：<http://localhost:8080/doc.html>。就能看到下图所展示的RESTful API的页面。