**通过SpringBoot集成Graphiql连接MySql数据库**

1. **开发环境搭建**

开发环境：

Java 1.8

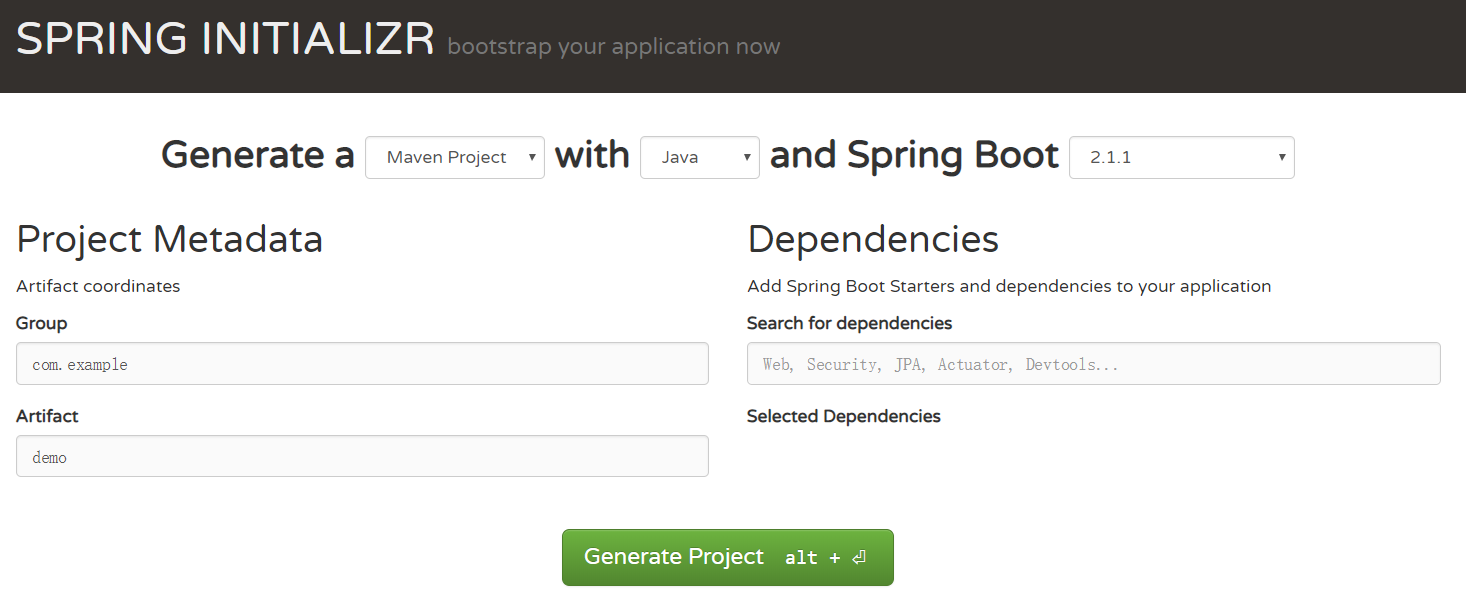
spring-2.0.0.RELEASE

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：DemoApplication

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

src/test/下的测试入口：DemoApplicationTests

1. **引入依赖**

*<!--SpringBoot依赖-->*<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-data-jpa</**artifactId**>  
</**dependency**>  
<**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**>  
*<!--H2数据库依赖-->*<**dependency**>  
 <**groupId**>com.h2database</**groupId**>  
 <**artifactId**>h2</**artifactId**>  
 <**scope**>runtime</**scope**>  
</**dependency**>  
*<!--MySQL数据库依赖-->*<**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 <**version**>8.0.13</**version**>  
</**dependency**>  
*<!--Graphql依赖-->*<**dependency**>  
 <**groupId**>io.leangen.graphql</**groupId**>  
 <**artifactId**>spqr</**artifactId**>  
 <**version**>0.9.9</**version**>  
</**dependency**>  
*<!--lombok依赖-->*<**dependency**>  
 <**groupId**>org.projectlombok</**groupId**>  
 <**artifactId**>lombok</**artifactId**>  
 <**version**>1.18.4</**version**>  
 <**scope**>provided</**scope**>  
</**dependency**>

1. **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"**;  
}

1. **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 {  
 @Id @GeneratedValue  
 @GraphQLQuery(name = **"instituteId"**, description = **"A Institute's instituteId"**)  
 **private** Long **instituteId**;  
 @GraphQLQuery(name = **"instituteName"**, description = **"A Institute's instituteName"**)  
 **private** String **instituteName**;  
 @GraphQLQuery( name = **"numberOfMajor"**, description = **"A Institute's number of major"**)  
 **private** Integer **numberOfMajor**;  
 @GraphQLQuery( name = **"majorArrayList"**, description = **"A list of major"**)  
 **private** ArrayList<Major> **majorArrayList**;  
}

**Major（专业）：**

@Entity

@Getter  
@Setter

**public class** Major {  
 @Id  
 @GeneratedValue  
 @GraphQLQuery(name = **"majorId"**, description = **"A major's id"**)  
 **private** Long **majorId**;  
 @GraphQLQuery(name = **"majorName"**, description = **"A major's majorName"**)  
 **private** String **majorName**;  
 @GraphQLQuery(name = **"instituteId"**, description = **"A instituteId of major"**)  
 **private** Long **instituteId**;  
 @GraphQLQuery(name = **"instituteName"**, description = **"A instituteName of major"**)  
 **private** String **instituteName**;  
 @GraphQLQuery(name = **"courseArrayList"**, description = **"A list of courseArrayList"**)  
 **private** ArrayList<Course> **courseArrayList**;

}

**Course（课程）：**

@Entity

@Getter  
@Setter

**public class** Course {  
 @Id @GeneratedValue  
 @GraphQLQuery(name = **"courseId"**, description = **"A course's courseId"**)  
 **private** Long **courseId**;  
 @GraphQLQuery(name = **"courseName"**, description = **"A course's courseName"**)  
 **private** String **courseName**;  
 @GraphQLQuery(name = **"majorId"**, description = **"A major's id"**)  
 **private** Long **majorId**;  
 @GraphQLQuery(name = **"majorName"**, description = **"A major's majorName"**)  
 **private** String **majorName**;  
 @GraphQLQuery(name = **"teacherId"**, description = **"A teacher's Id"**)  
 **private** Long **teacherId**;  
 @GraphQLQuery(name = **"teacherName"**, description = **"A teacher's Name"**)  
 **private** String **teacherName**;

}

**Student（学生）：**

@Entity

@Getter  
@Setter  
**public class** Student {  
 @Id @GeneratedValue  
 @GraphQLQuery(name = **"studentId"**, description = **"A student's studentId"**)  
 **private** Long **studentId**;  
 @GraphQLQuery(name = **"studentName"**, description = **"A student's studentName"**)  
 **private** String **studentName**;  
 @GraphQLQuery( name = **"majorId"**, description = **"A student's major"**)  
 **private** Long **majorId**;  
 @GraphQLQuery( name = **"studentSex"**, description = **"A student's sex"**)  
 **private** String **studentSex**;  
 @GraphQLQuery(name = **"majorName"**, description = **"A major's majorName"**)  
 **private** String **majorName**;

}

**Teacher（教师）：**

@Entity

@Getter  
@Setter

**public class** Teacher {  
 @Id @GeneratedValue  
 @GraphQLQuery(name = **"teacherId"**, description = **"A teacher's Id"**)  
 **private** Long **teacherId**;  
 @GraphQLQuery(name = **"teacherName"**, description = **"A teacher's Name"**)  
 **private** String **teacherName**;  
 @GraphQLQuery( name = **"teacherSex"**, description = **"A teacher's sex"**)  
 **private** String **teacherSex**;  
 @GraphQLQuery(name = **"instituteId"**, description = **"A Institute's instituteId"**)  
 **private** Long **instituteId**;  
 @GraphQLQuery(name = **"instituteName"**, description = **"A Institute's instituteName"**)  
 **private** String **instituteName**;

}

1. **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 = ?**

额外添加的方法：获取课程详细信息（包括选课学生与授课老师）

**public static** ArrayList<String> courseDetails() {  
 ArrayList<String> details = **new** ArrayList<>();  
 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 course.courseId,courseName,course.majorId,majorName,teacher.teacherId,teacherName,"** +  
 **"studentId,studentName from course inner join student,major,teacher "** +  
 **"where course.majorId=student.majorId and course.majorId=major.majorId "** +  
 **"and teacher.teacherId=course.teacherId"**;  
 ResultSet courseRs = stmt.executeQuery(sql);  
 **while** (courseRs.next()) {  
 **long** coursetId = courseRs.getLong(**"courseId"**);  
 String courseName = courseRs.getString(**"courseName"**);  
 Integer teacherId = courseRs.getInt(**"teacherId"**);  
 String teacherName = courseRs.getString(**"teacherName"**);  
 Integer majorId = courseRs.getInt(**"majorId"**);  
 String majorName = courseRs.getString(**"majorName"**);  
 Integer studentId = courseRs.getInt(**"studentId"**);  
 String studentName = courseRs.getString(**"studentName"**);  
 String s = **"课程编号:"** + coursetId + **","** + **" "** + **"课程名称："** + courseName + **","** +**" "** + **"教师编号："** + teacherId + **","** + **" "** + **"教师名称："** + teacherName + **","** + **" "** + **"专业编号："** + majorId + **","** + **" "** + **"专业名称："** + majorName + **","** + **" "** + **"学生编号："** + studentId + **","** + **" "** + **"学生名称："** + studentName;  
 details.add(s);  
 s=**""**;  
 details.add(s);  
 }  
 courseRs.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** details;  
}

1. **Controller配置**

GraphqlController类配置如下：

@RestController  
**public class** GraphQLController {  
  
 **private final** GraphQL **graphQL**;  
  
 **public** GraphQLController(CollegeService collegeService) {  
 GraphQLSchema schema = **new** GraphQLSchemaGenerator()  
 .withResolverBuilders(  
 *//Resolve by annotations* **new** AnnotatedResolverBuilder())  
 .withOperationsFromSingleton(collegeService)  
 .withValueMapperFactory(**new** JacksonValueMapperFactory())  
 .generate();  
 **graphQL** = GraphQL.*newGraphQL*(schema).build();  
 }  
  
 @PostMapping(value = **"/graphql"**, consumes = MediaType.***APPLICATION\_JSON\_UTF8\_VALUE***, produces = MediaType.***APPLICATION\_JSON\_UTF8\_VALUE***)  
 @ResponseBody  
 **public** Map<String, Object> graphql(@RequestBody Map<String, String> request, HttpServletRequest raw) {  
 ExecutionResult executionResult = **graphQL**.execute(ExecutionInput.*newExecutionInput*()  
 .query(request.get(**"query"**))  
 .operationName(request.get(**"operationName"**))  
 .context(raw)  
 .build());  
 **return** executionResult.toSpecification();  
 }  
}

1. **ApplicationWebConfig配置：**

ApplicationWebConfig类配置如下：

@Configuration  
**public class** ApplicationWebConfig **implements** WebMvcConfigurer {  
 @Override  
 **public void** addViewControllers(ViewControllerRegistry registry) {  
 registry.addViewController(**"/graphiql"**).setViewName(  
 **"forward:/graphiql/index.html"**);  
 }  
}

1. **CollegeService配置**

CollegeService类配置Graphiql的语句，包含学院等5个实体类的Graphql语句方法配置。

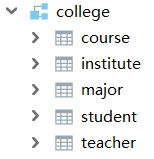
5个实体类结构类似，以下只给出Institute（学院）的方法配置。

@Service  
**public class** CollegeService {  
 **private** Map<Long, Institute> **instituteMap** = **new** HashMap<>();  
 **private** Map<Long, Major> **majorMap** = **new** HashMap<>();  
 **private** Map<Long, Course> **courseMap** = **new** HashMap<>();  
 **private** Map<Long, Student> **studentMap** = **new** HashMap<>();  
 **private** Map<Long, Teacher> **teacherMap** = **new** HashMap<>();  
  
 **public** CollegeService() {*//构造函数*  
 **this**.**majorMap** = MajorDatabase.*getMajorMap*();  
 **this**.**instituteMap** = InstituteDatabase.*getInstituteMap*();  
 **this**.**courseMap** = CourseDatabase.*getCourseMap*();  
 **this**.**studentMap** = StudentDatabase.*getStudentMap*();  
 **this**.**teacherMap** = TeacherDatabase.*getTeacherMap*();  
 }  
*/\*  
Queries for institutes  
 \*/* @GraphQLQuery(name = **"institutes"**)*//获取学院信息* **public** Collection<Institute> getInstitutes() {  
 **return** InstituteDatabase.*getInstituteMap*().values();  
 }  
  
 @GraphQLQuery(name = **"Institute"**)*//根据Id获取学院信息* **public** Institute getInstituteById(@GraphQLArgument(name = **"instituteId"**) Long id) {  
 **return instituteMap**.get(id);  
 }  
  
 @GraphQLMutation(name = **"saveInstitute"**)*//储存学院信息* **public** String saveInstitute(@GraphQLArgument(name = **"institutedId"**) Long institutedId,  
 @GraphQLArgument(name = **"instituteName"**) String instituteName,  
 @GraphQLArgument(name = **"numberOfMajor"**) **int** numberOfMajor) {  
 Institute institute = **new** Institute();  
 institute.setInstituteId(institutedId);  
 institute.setInstituteName(instituteName);  
 institute.setNumberOfMajor(numberOfMajor);  
 **instituteMap**.put(institutedId, institute);  
 **return** InstituteDatabase.*saveInstitute*(institute);  
 }  
  
 @GraphQLMutation(name = **"updateInstitute"**)*//更新学院信息* **public** String updateInstitute(@GraphQLArgument(name = **"instituteId"**) Long instituteID,  
 @GraphQLArgument(name = **"instituteName"**) String instituteName,  
 @GraphQLArgument(name = **"numberOfMajor"**) **int** numberOfMajor  
 ) {  
 **try** {  
 **instituteMap**.get(instituteID).setInstituteName(instituteName);  
 **instituteMap**.get(instituteID).setNumberOfMajor(numberOfMajor);  
 } **catch** (Exception E) {  
 **return "更新信息失败，请检查是否存在该id"**;  
 }  
 **return** InstituteDatabase.*updateInstitute*(instituteID, **instituteMap**.get(instituteID));  
 }  
  
 @GraphQLMutation(name = **"deleteInstitute"**)*//删除学院信息* **public** String deleteInstitute(@GraphQLArgument(name = **"instituteId"**) Long id) {  
 **instituteMap**.remove(id);  
 **return** InstituteDatabase.*deleteInstitute*(id);  
 }

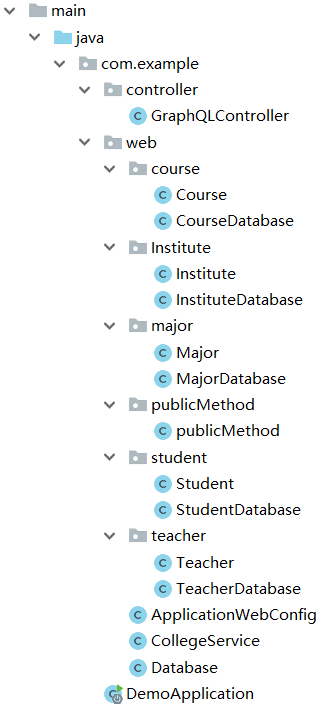
}

1. **结构说明**

最终形成的数据库Schema结构如下图：



java目录如下图：



1. **访问页面**

完成上述代码，启动Spring Boot程序，访问：http://localhost:8080/graphiql.html

。就能看到下图所展示的Garphiql的页面。可根据Graphql语法获得信息。

