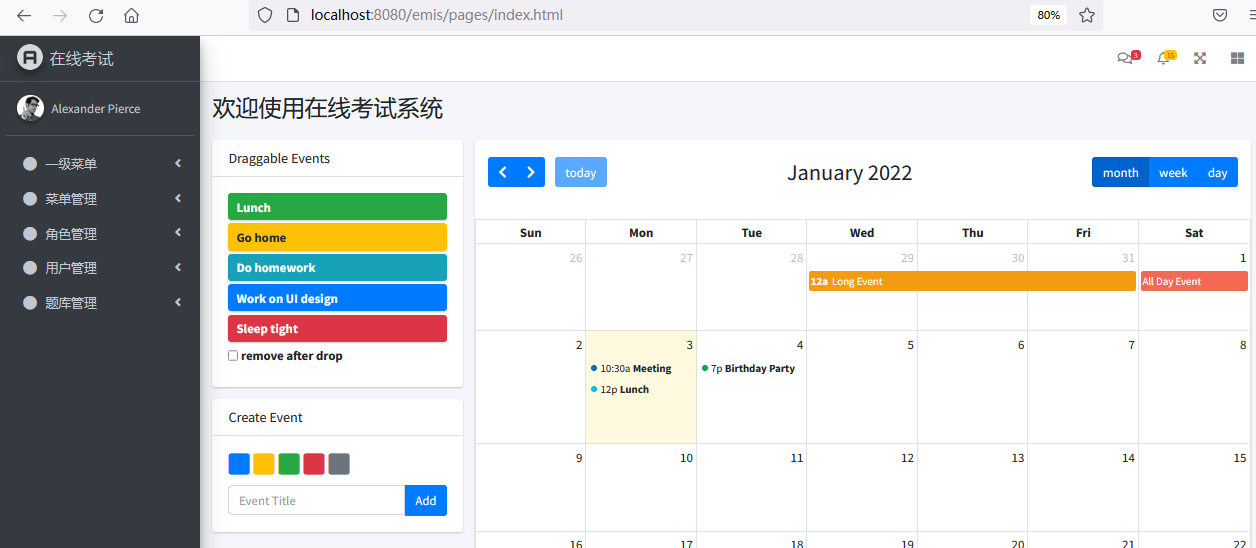
# 1 需求



**在线考试管理系统，模块如下：**

**登录模块**

**权限模块**

**学生管理模块**

**科目管理模块**

**考题管理模块**

**在线考试模块**

**统计模块**

# 2 技术选型

**前端：html+bootstrap模板，jQuery，ajax**

**后端：java，springboot，mybatis，mysql**

# 3 环境准备

**jdk1.8环境变量**

**mysql8**

**maven环境变量**

**idea社区版**

**开发相关材料和软件都放在“资料”文件夹中**

**链接：https://pan.baidu.com/s/1Ih0osapxjxHO6Ex47Q5-0w**

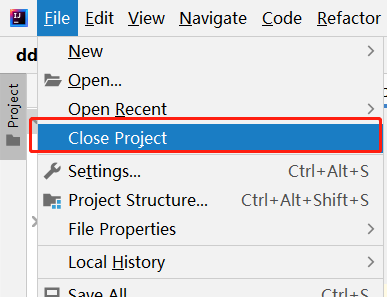
**提取码：1c63**



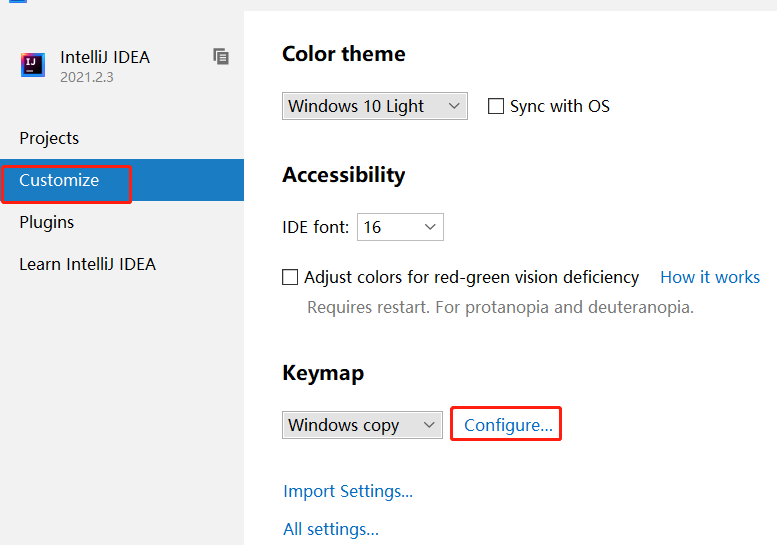
# 4 idea社区版创建springboot工程

## 4.1 设置自定义全局maven

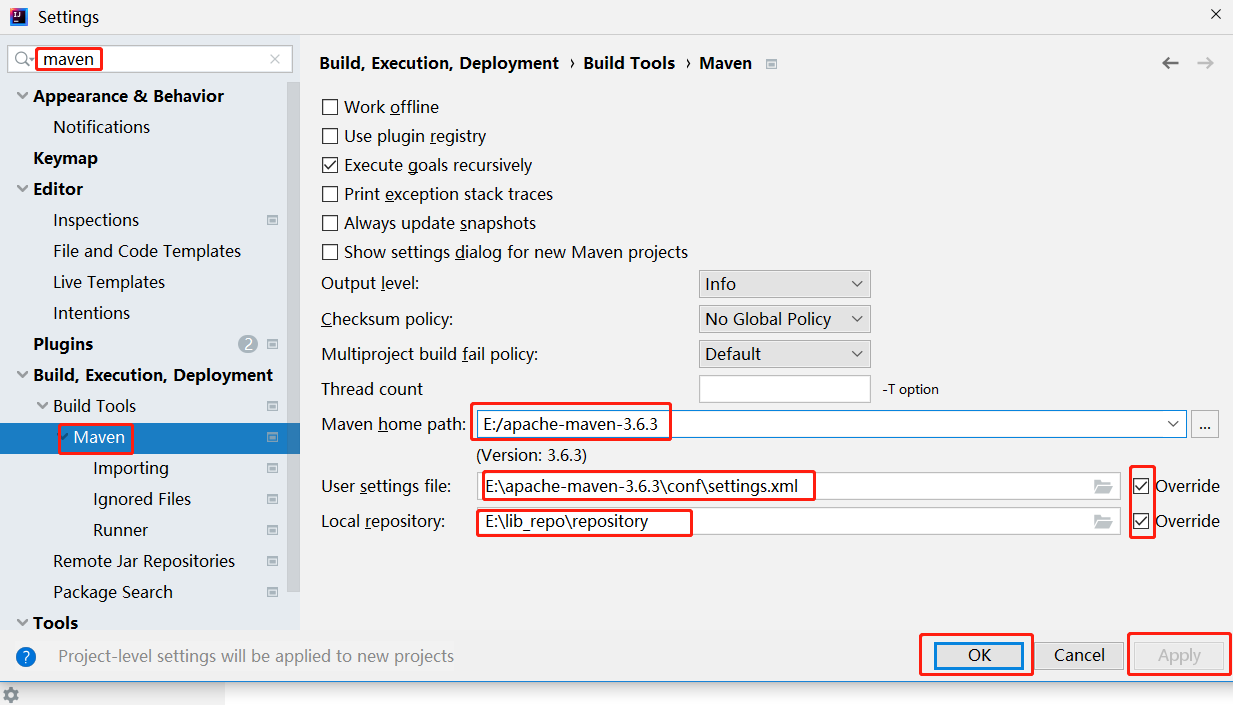
**1、选择关闭当前项目**



**2、选择Customize，点击Configure**



**3、查询maven，在左侧菜单点击选择Maven，填写外部maven的路径，填写自己的settings.xml文件，会自动填写本地仓库的路径，把2个Override的复选框勾选上**

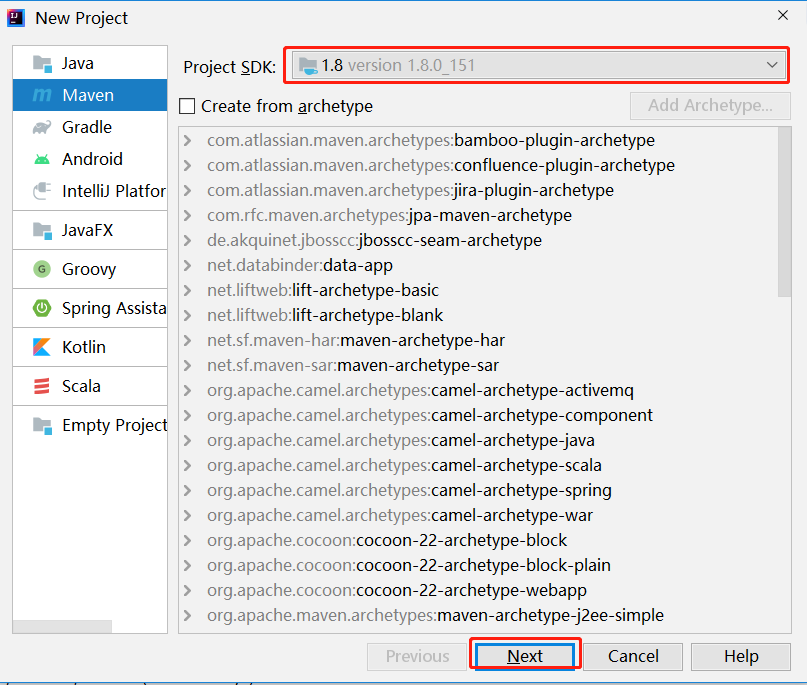


**4、点击Apply，点击OK**

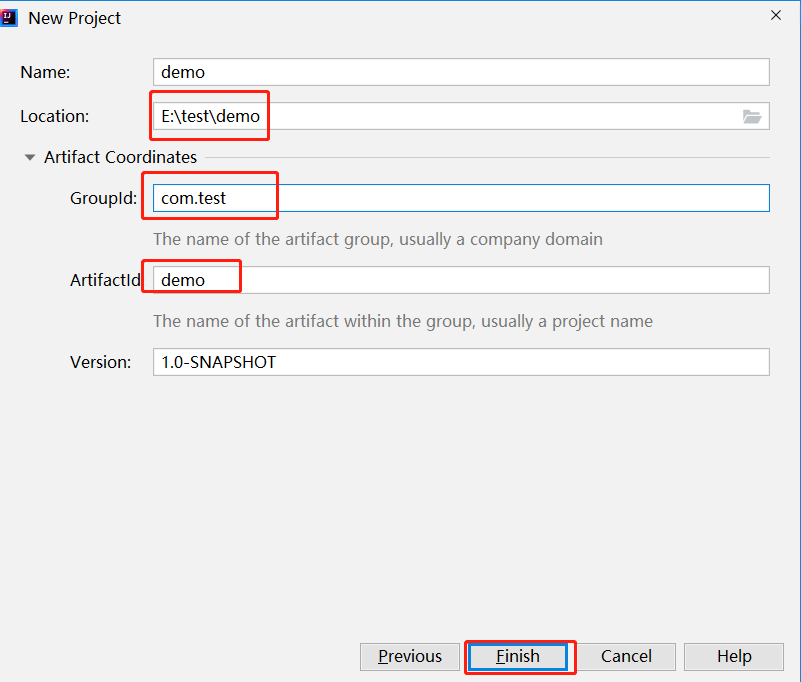
## 4.2 创建maven工程

### 4.2.1 idea2021之前的版本创建maven工程

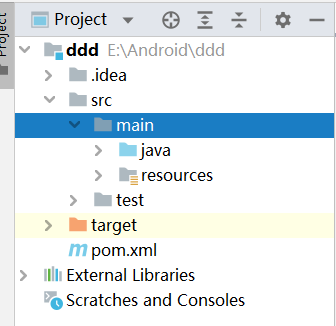
**new project，选择Maven ，选择jdk**



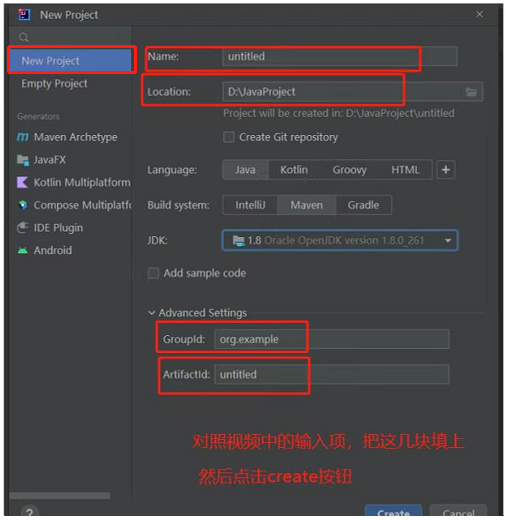
**点击Next按钮，工程路径（会自动带出工程名），填写GroupId和ArtifactId**



**点击Finish，会得到类似以下的项目结构**



### 4.2.2 idea2022版本创建maven工程



## 4.3 配置springboot+mybatis项目

### 4.3.1 配置pom.xml

**1、在**<**modelVersion**>4.0.0</**modelVersion**>**下添加以下配置**

<**parent**>

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

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

<**version**>2.6.2</**version**>

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

</**parent**>

**2、在**<**version**>1.0-SNAPSHOT</**version**>**下添加以下配置**

<**properties**>

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

</**properties**>

<**dependencies**>

<**dependency**>

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

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

</**dependency**>

<**dependency**>

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

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

<**version**>2.2.1</**version**>

</**dependency**>

<**dependency**>

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

<**artifactId**>spring-boot-devtools</**artifactId**>

<**scope**>runtime</**scope**>

<**optional**>true</**optional**>

</**dependency**>

<**dependency**>

<**groupId**>mysql</**groupId**>

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

<**scope**>runtime</**scope**>

*<!--*

*1、如果你的数据库是mysql8，这段<version>5.1.32</version>不用管*

*2、如果你的数据库是mysql5.x，放开<version>5.1.32</version>这段注释*

*-->*

*<!--<version>5.1.32</version>-->*

</**dependency**>

<**dependency**>

<**groupId**>com.alibaba</**groupId**>

<**artifactId**>druid</**artifactId**>

<**version**>1.1.23</**version**>

</**dependency**>

*<!-- json相关工具类 -->*

<**dependency**>

<**groupId**>com.alibaba</**groupId**>

<**artifactId**>fastjson</**artifactId**>

<**version**>1.2.73</**version**>

</**dependency**>

*<!-- MyBatis逆向工程依赖 -->*

<**dependency**>

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

<**artifactId**>mybatis-generator-core</**artifactId**>

<**version**>1.4.0</**version**>

</**dependency**>

</**dependencies**>

<**build**>

<**resources**>

<**resource**>

<**directory**>src/main/java</**directory**>

<**includes**>

<**include**>\*\*/\*.xml</**include**>

</**includes**>

<**filtering**>false</**filtering**>

</**resource**>

</**resources**>

<**plugins**>

<**plugin**>

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

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

</**plugin**>

</**plugins**>

</**build**>

### 4.3.2 在src/main/resources下添加application.yml配置文件

**内容如下**

**spring**:

**mvc**:

**view**:

*# 如果属性后面有值，类似prefix: /pages/，:后面需要有一个空格*

**prefix**: /pages/

**suffix**: .html

**datasource**:

*# mysql5.x，以下被注释的url和driver-class-name*

*#url: jdbc:mysql://master:3306/reclamation?useUnicode=true&characterEncoding=UTF-8*

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

*# mysql8，用以下的url和driver-class-name*

**url**: jdbc:mysql://master:3306/emis?serverTimezone=UTC&useUnicode=true&characterEncoding=utf-8&useSSL=true

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

*# 这里的用户名密码，不要使用之前逆向工程的那个用户名和密码（之前的用户只能查询），去使用正常的用户*

**username**: root

**password**: 123456

*# 数据库连接池*

**type**: com.alibaba.druid.pool.DruidDataSource

## 4.4 测试

**1、在src/main/java下创建com包**

**2、在com包下创建springboot启动引导java文件，我这里命名为App（可以自定义）。**

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public class** App {

**public static void** main(String[] args) {

SpringApplication.*run*(App.**class**);

}

}

**3、在springboot启动类所在的包的子包下创建controller类**

**我这里springboot启动类是com包中的App。**

**在com.test.emis包下新建controller包，在com.test.emis.controller包中创建测试controller类，代码如下。**

@Controller

@RequestMapping(**"/test"**)

**public class** TestController {

@RequestMapping(**"/index"**)

@ResponseBody

**public** String index(){

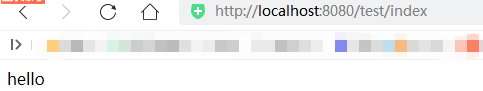
**return "hello"**;

}

}

**3、执行springboot启动类的main方法启动springboot项目，我这里是运行EmisApplication的main方法。springboot项目的默认端口是8080，默认上下文是""空字符串。**

**4、在浏览器访问controller，我这里的url是**[**http://localhost:8080/test/index**](http://localhost:8080/test/index)



# 5 springboot的静态资源

**这里的静态资源主要是html、js、css和图片等。**

## 5.1 静态资源存放

**springboot的静态资源是放在src/main/resources/static文件夹下，可以在static文件夹下创建多级子文件夹，用来存放静态资源。类似下图。**



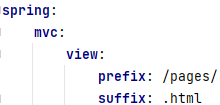
## 5.2 直接访问静态资源

**访问的时候，路径从static开始（不包括static文件夹），例如，在static文件夹下有一个pages文件夹，pages文件夹中有一个index.html，那么访问路径就是“ip:port/上下文/pages/index.html”，这是直接访问静态资源。**

**注意：如果发现在static文件夹下添加了静态资源之后无法访问，尝试maven clean，maven install，然后再次访问。**

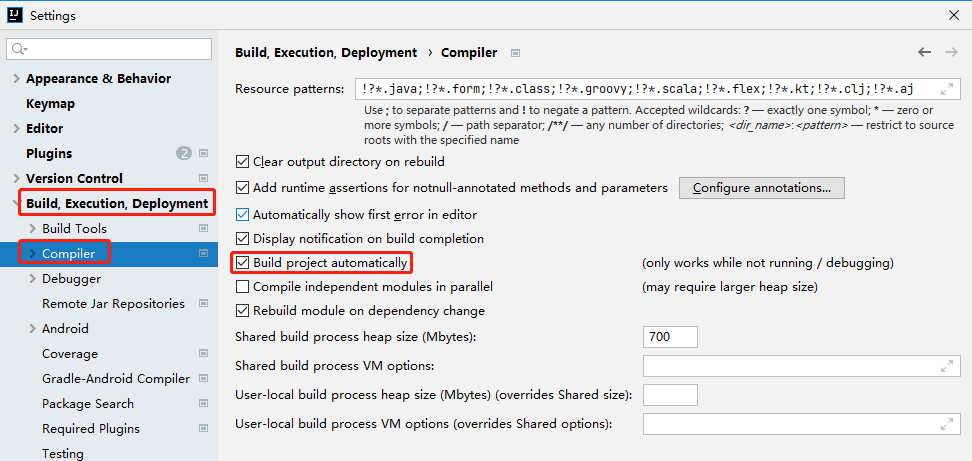
## 5.3 controller跳转前端页面

**如果想要通过controller跳转到html页面，那么需要在application.yml文件中配置html文件的前缀（html所属文件夹）和后缀（.html），配置如下。**



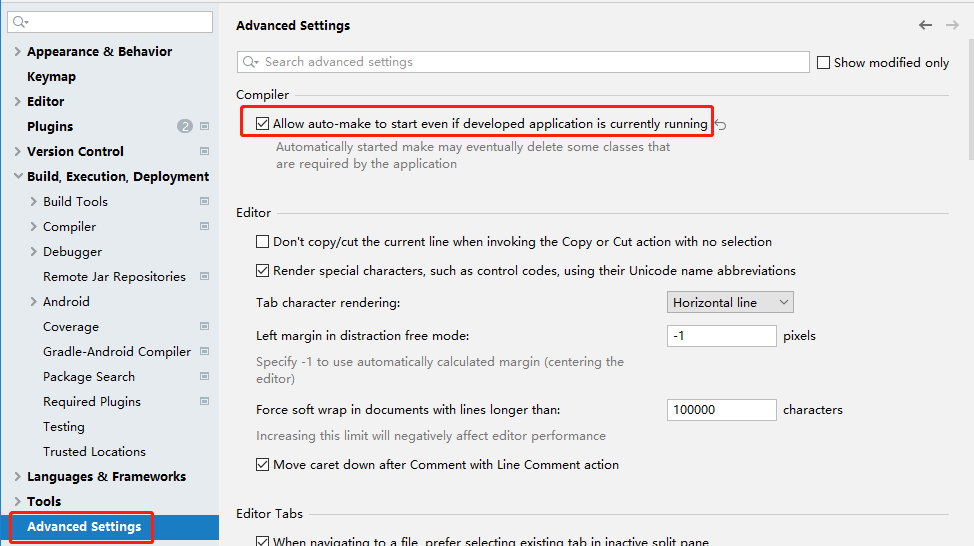
## 5.4 idea实时刷新前端代码

**1、“File” -> “Settings” -> “Build,Execution,Deplyment” -> “Compiler”，选中打勾 “Build project automatically” 。**



**2、开启compiler.automake.allow.when.app.running**

**“File” -> “Settings” -> “Advanced Settings” ，勾选下图所示选项**

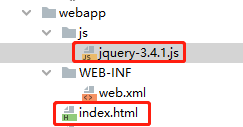


# 6 jQuery

**前端页面部分也有逻辑需要编写代码，这里使用jQuery。在需要使用jQuery的页面引入jQuery即可。**

## 6.1 页面中引用jQuery

**示例，我这里是在src/main/webapp下创建js文件夹，把jquery-3.4.1.js拷贝到js文件夹，在src/main/webapp/index.html中引用jquery-3.4.1.js。**



**引用代码如下。**

<**head**>

<**meta charset="UTF-8"**>

<**title**>Title</**title**>

<**script type="text/javascript" src="./js/jquery-3.4.1.js"**></**script**>

</**head**>

## 6.2 具体教程

**jQuery有很多在线文档，这里不再重复，我们以菜鸟教程的jQuery为例。地址**[**https://www.runoob.com/jquery/jquery-tutorial.html**](https://www.runoob.com/jquery/jquery-tutorial.html)



**这里内容很多，我们暂时只关注参考手册的内容即可。**

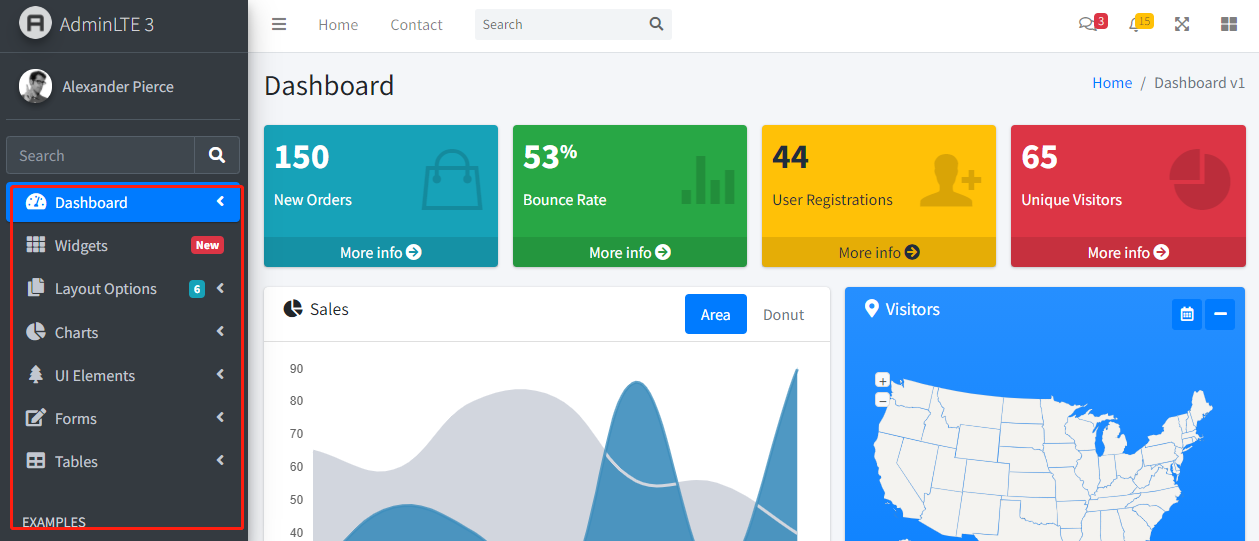


# 7 根据AdminLTE模板构建管理类系统的静态页模板

## 7.1 AdminLTE

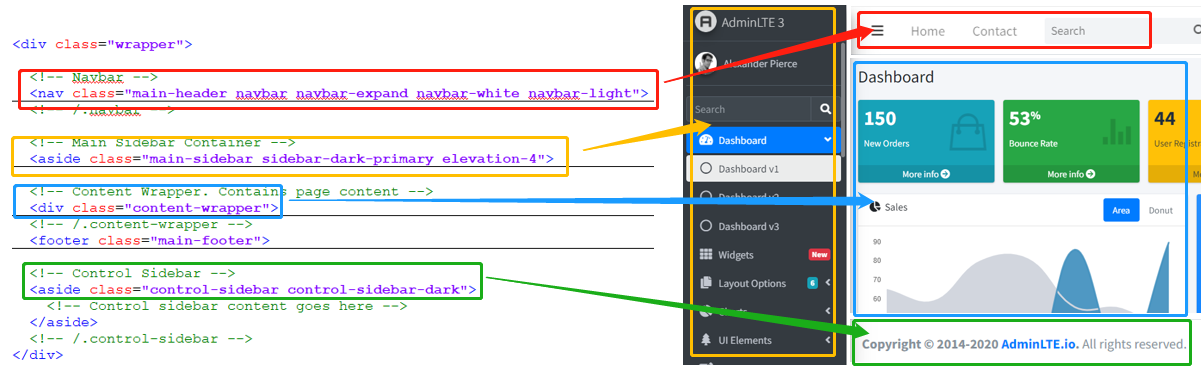
**我们使用的模板是AdminLTE，基于boostrap和jQuery开发的，其中包含大量前端组件，我们可以根据需要灵活组合使用。**

**AdminLTE模板组件在“在线考试系统/资料/管理系统前端html模板.zip”，解压，进入解压文件夹，鼠标双击index.html，就可以看到各种组件。**



**AdminLTE模板页面的基本结构是“左、上、下”页面三分的结构，这种结构是目前绝大多数管理类系统的页面结构。**

**以下是AdminLTE模板页面的“左、上、下”页面三分的结构与代码的对应。**



## 7.2 构建管理类系统的静态页模板

**管理类系统的页面类型大概有以下几类：**

**1、登录页（大多数的管理类系统都没有注册页，需要通过用户管理模块增加用户）。**

**2、欢迎页，登录成功后进入的首页。**

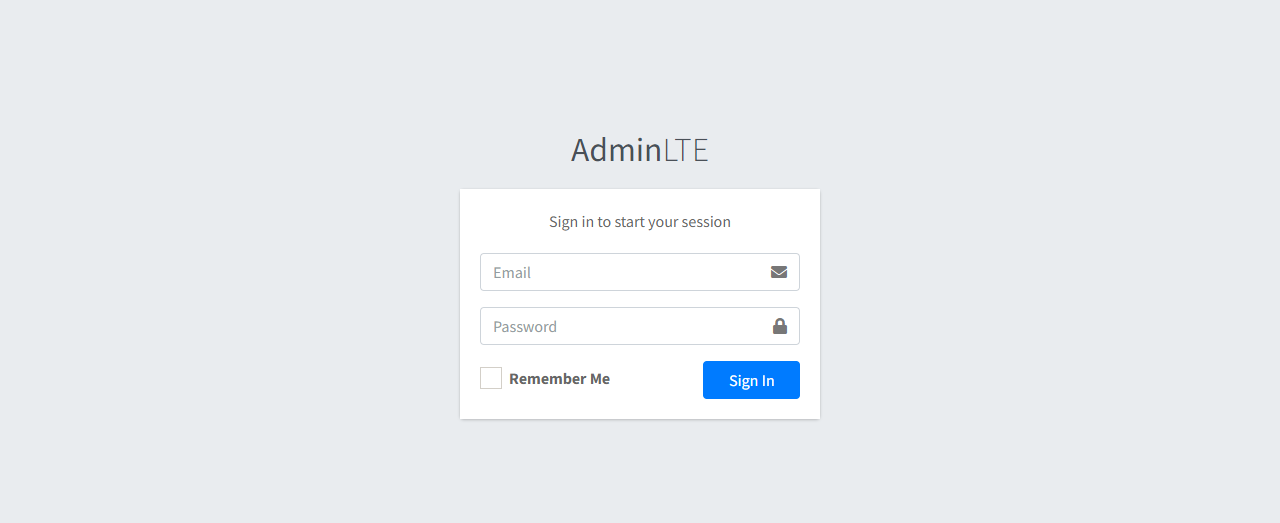
**3、列表页，展示数据集合的表格，一般带分页和查询功能，展示的每行数据的最后会有可以点击的“修改”和“删除”功能。**

**4、新增/修改页，一个form表单，同一个的功能模块的新增/修改页的表单元素是一致的，只是修改页需要回填数据。**

**5、其他特殊页，例如在线答题页、数据统计页、文件上传页等等，这些在管理类系统中的占比很小。**

### 7.2.1 登录页

**根据“pages\examples”中的register.html构建登录页**



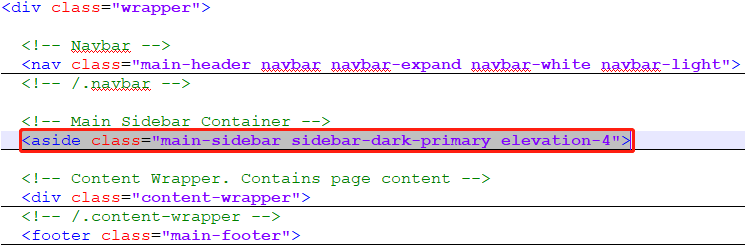
### 7.2.2 页面主体构建

**从AdminLTE的index.html中，拆出“左、上、下”主体结构，只要结构，其中具体的代码删除。**

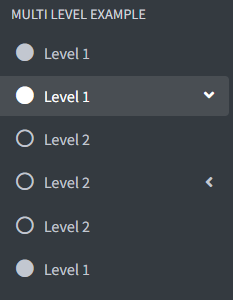
### 7.2.3 左侧菜单

**我们这里，菜单等组件是作为单独的html文件存在，主页面需要对于菜单等组件进行加载，这里是使用jQuery动态加载html页面，核心代码见“在线考试系统\资料”文件夹中的《jQuery动态加载html.js》。**

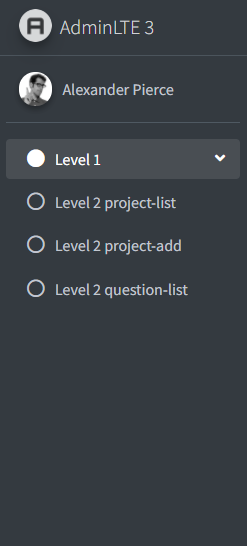
**左侧菜单代码在AdminLTE模板的多数页面中都有。**



**菜单的主体部分看下图对应的代码。**

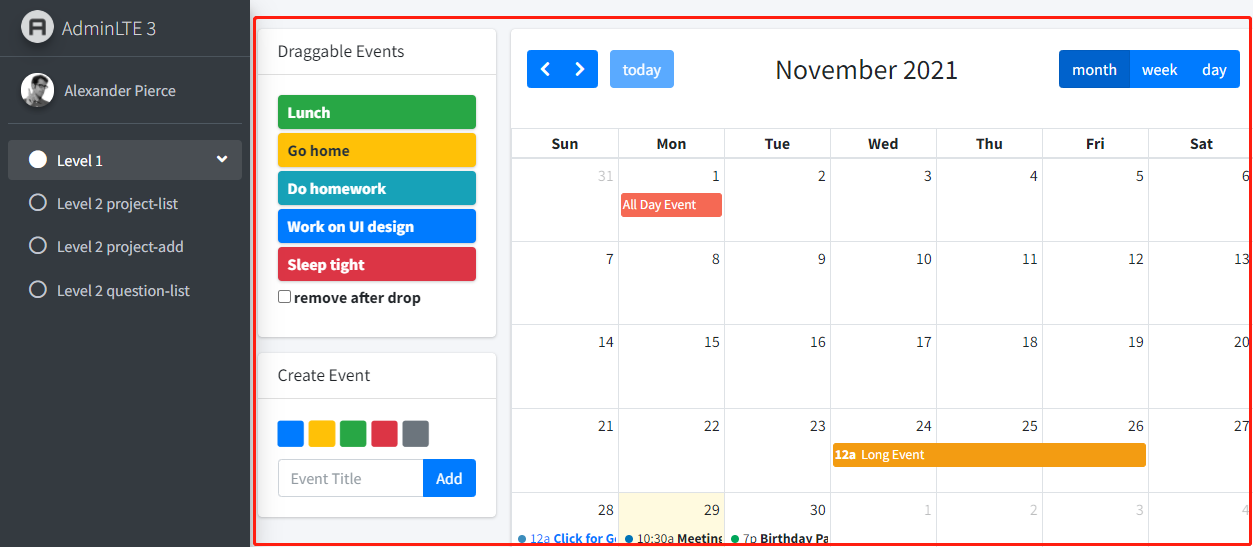


**最终构建的左侧菜单为下图所示。**



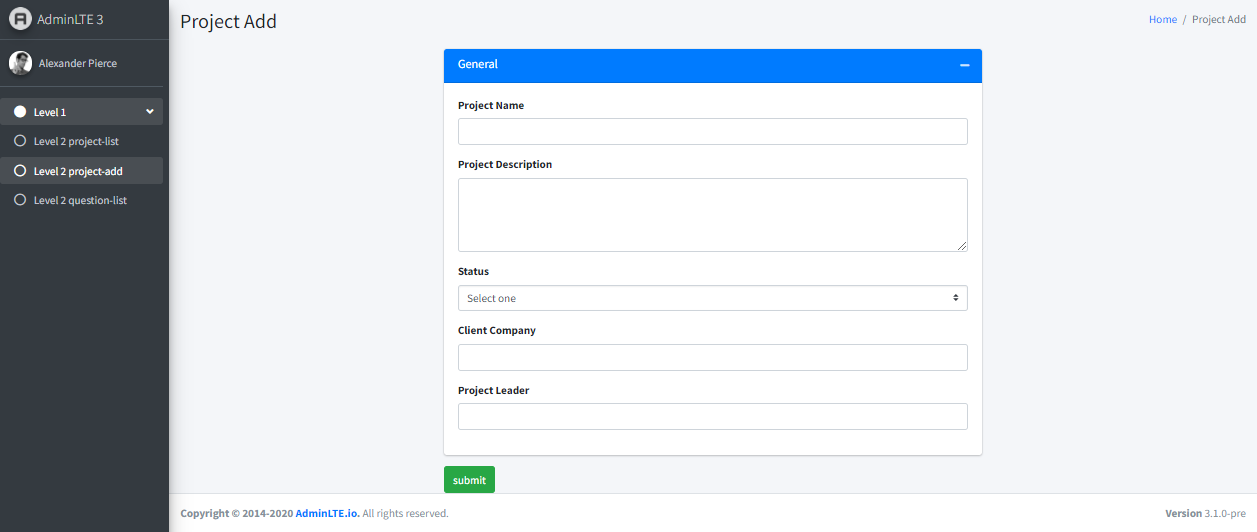
### 7.2.4 右侧欢迎页

**用日历作为欢迎页的右侧主体部分，代码见“/pages/calendar.html”。**



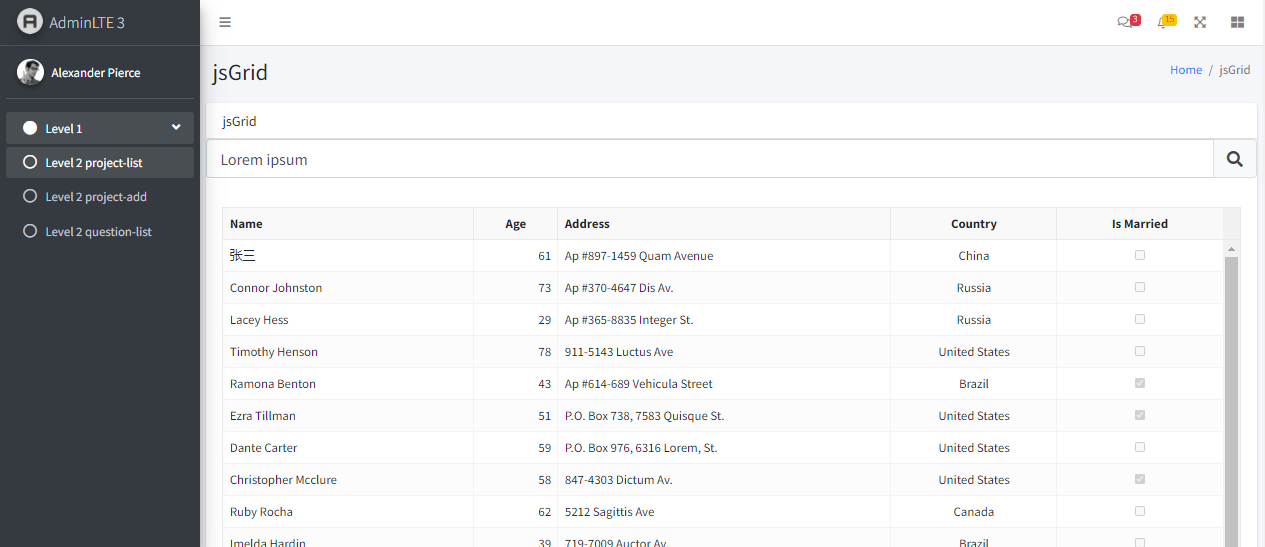
### 7.2.5 新增/修改数据页模板

**代码主体在“pages/examples/project-add.html”。**



### 7.2.6 数据列表页模板

**代码主体在“pages/tables/jsgrid.html”，查询框部分见“pages/search/simple.html”。**



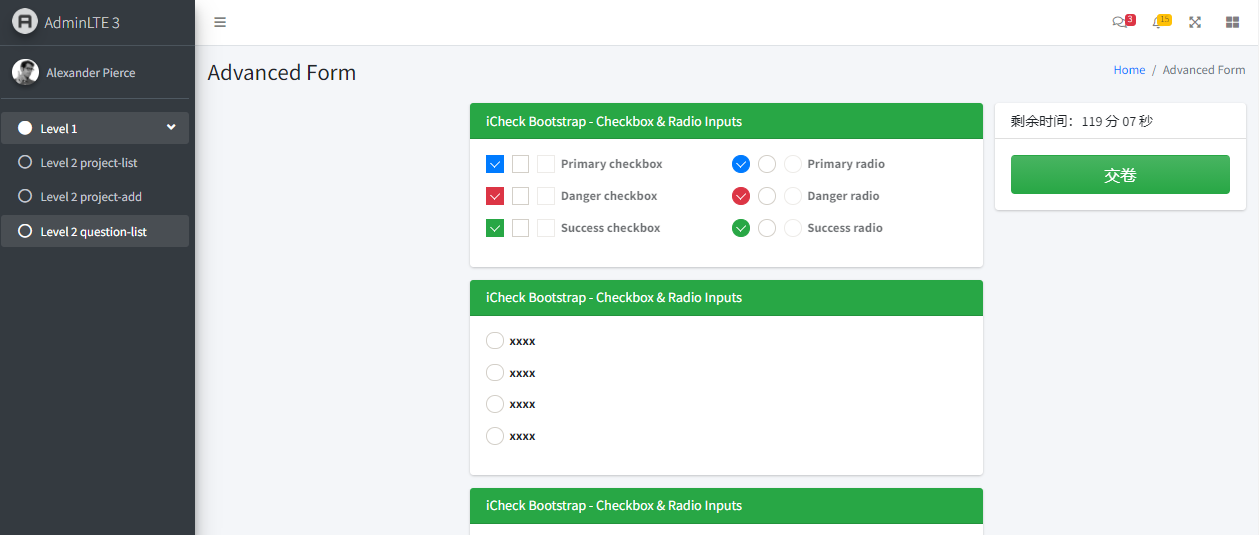
### 7.2.7 考试页模板

**考试页需要有倒计时和交卷按钮，倒计时和交卷按钮悬浮在页面上，倒计时结束后交卷按钮失效，变为不可点击状态。**

**悬浮框代码代码见“/pages/calendar.html”中的悬浮部分。**

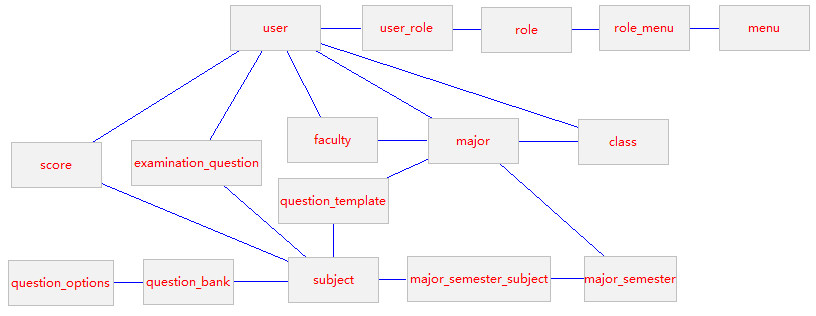
**复选框、单选钮部分代码见“pages/forms/advanced.html”。**

**倒计时js核心代码，详见“在线考试系统（servlet-html-jetty）IEDA版\资料”《倒计时js代码.txt》**



# 8 表设计

## 8.1 表关系



## 8.2 表结构

|  |  |  |
| --- | --- | --- |
| **用户表** | | |
| **user** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(11)** |
| **username** | **用户名** | **varchar(255)** |
| **password** | **密码** | **varchar(255)** |
| **real\_name** | **姓名** | **varchar(255)** |
| **sex** | **性别** | **varchar(255)** |
| **sno** | **学号** | **varchar(255)** |
| **admission\_date** | **入学时间** | **datetime** |
| **faculty\_id** | **院系id** | **int(2)** |
| **major\_id** | **专业id** | **int(5)** |
| **class\_id** | **班级id** | **int(5)** |

|  |  |  |
| --- | --- | --- |
| **角色表** | | |
| **role** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(2)** |
| **role\_code** | **角色编码** | **varchar(50)** |
| **role\_name** | **角色中文** | **varchar(255)** |

|  |  |  |
| --- | --- | --- |
| **用户与角色关联表** | | |
| **user\_role** | | |
| **字段** | **中文** | **数据类型** |
| **use\_id** | **用户id** | **int(10)** |
| **role\_id** | **角色id** | **int(2)** |

|  |  |  |
| --- | --- | --- |
| **菜单表** | | |
| **menu** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(2)** |
| **menu\_code** | **菜单编码** | **varchar(50)** |
| **menu\_name** | **菜单中文** | **varchar(255)** |
| **menu\_url** | **菜单url** | **varchar(255)** |
| **menu\_level** | **菜单级别** | **varchar(255)** |
| **parent\_id** | **父菜单id** | **int(5)** |
| **sort** | **排序** | **int(5)** |

|  |  |  |
| --- | --- | --- |
| **角色与菜单关联表** | | |
| **role\_menu** | | |
| **字段** | **中文** | **数据类型** |
| **role\_id** | **角色id** | **int(2)** |
| **menu\_id** | **菜单id** | **int(2)** |

|  |  |  |
| --- | --- | --- |
| **院系表** | | |
| **faculty** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(2)** |
| **faculty\_name** | **院系名称** | **varchar(50)** |

|  |  |  |
| --- | --- | --- |
| **专业表** | | |
| **major** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(5)** |
| **faculty\_id** | **院系id** | **int(2)** |
| **major\_name** | **专业名称** | **varchar(50)** |

|  |  |  |
| --- | --- | --- |
| **班级表** | | |
| **class** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(5)** |
| **faculty\_id** | **院系id** | **int(2)** |
| **major\_id** | **专业id** | **int(5)** |
| **class\_name** | **班级名称** | **varchar(50)** |

|  |  |  |
| --- | --- | --- |
| **科目表** | | |
| **subject** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(5)** |
| **subject\_name** | **科目名称** | **varchar(50)** |

|  |  |  |
| --- | --- | --- |
| **专业学期表** | | |
| **major\_semester** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(5)** |
| **major\_id** | **专业id** | **int(5)** |
| **school\_year** | **学年，1、2、3…** | **int(2)** |
| **semester** | **学期，上学期0下学期1** | **int(1)** |

|  |  |  |
| --- | --- | --- |
| **专业学期科目表** | | |
| **major\_semester\_subject** | | |
| **字段** | **中文** | **数据类型** |
| **major\_semester\_id** | **专业学期id** | **int(5)** |
| **subject\_id** | **科目id** | **int(5)** |

|  |  |  |
| --- | --- | --- |
| **题库表** | | |
| **question\_bank** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(11)** |
| **subject\_id** | **科目id** | **int(5)** |
| **question\_code** | **考题编码** | **varchar(255)** |
| **question\_name** | **题目** | **varchar(255)** |
| **type** | **类型，0单选1多选** | **int(1)** |
| **difficulty** | **难度，0易1中2难3很难** | **int(1)** |
| **answer** | **答案** | **varchar(50)** |
| **create\_time** | **入库时间** | **datetime** |

|  |  |  |
| --- | --- | --- |
| **题库选项表** | | |
| **question\_options** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(11)** |
| **question\_code** | **考题编码** | **varchar(255)** |
| **options\_text** | **选项内容** | **varchar(255)** |
| **options\_code** | **选项编码** | **varchar(10)** |

|  |  |  |
| --- | --- | --- |
| **考题模板表** | | |
| **question\_template** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(11)** |
| **template\_name** | **模板名称** | **varchar(50)** |
| **major\_id** | **专业id** | **int(5)** |
| **subject\_id** | **科目id** | **int(5)** |
| **single\_num** | **单选题数目** | **int(3)** |
| **single\_score** | **单选题分值** | **int(2)** |
| **single\_difficulty\_proportion** | **单选题难度比例，易:中:难:很难** | **varchar(10)** |
| **multiple\_num** | **多选题数目** | **int(3)** |
| **multiple\_score** | **多选题分值** | **int(2)** |
| **multiple\_difficulty\_proportion** | **多选题难度比例，易:中:难:很难** | **varchar(10)** |
| **aggregate\_score** | **总分** | **int(3)** |

|  |  |  |
| --- | --- | --- |
| **用户答题表** | | |
| **examination\_question** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(11)** |
| **user\_id** | **用户id** | **int(11)** |
| **subject\_id** | **科目id** | **int(5)** |
| **question\_name** | **题目** | **varchar(255)** |
| **answer** | **答案** | **varchar(50)** |
| **correct** | **是否正确，0否1是** | **int(1)** |

|  |  |  |
| --- | --- | --- |
| **用户成绩表** | | |
| **score** | | |
| **字段** | **中文** | **数据类型** |
| **id** | **主键** | **自增int(11)** |
| **user\_id** | **用户id** | **int(11)** |
| **subject\_id** | **科目id** | **int(5)** |
| **score** | **分数** | **int(3)** |
| **answer** | **答题时间** | **datetime** |

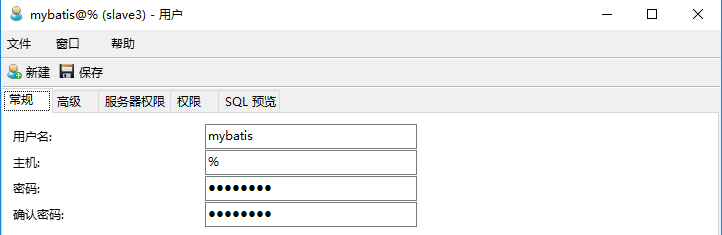
# 9 MyBatis逆向工程

**使用MyBatis根据数据库的表生成对应的实体类、mapper接口和mapper的xml文件。**

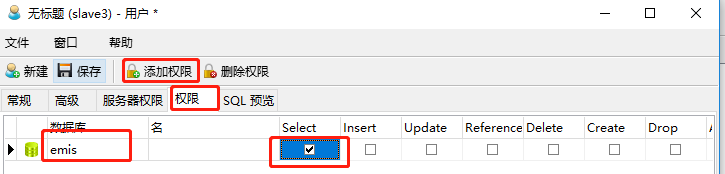
**1、创建一个新的数据库账户，只给这个账户指需要逆向工程的数据库的读的权限，用来完成逆向工程。**

**我这里账户是mybatis，密码是111111，需要逆向工程的数据库是emis，直接使用Navicat工具完成账户创建和授权。**

**1）添加账户信息**



**2）授权**



**也可以使用SQL语句完成添加用户并授权（mysql8版本）**

# 创建用户

CREATE USER 'test1'@'%' IDENTIFIED BY '123456';

# 更改密码加密规则

ALTER USER 'test1'@'%' IDENTIFIED WITH mysql\_native\_password BY '123456';

# 授权查询权限

GRANT SELECT ON `test`.\* TO test1@'%' WITH GRANT OPTION;

# 刷新权限

FLUSH PRIVILEGES;

**2、添加依赖**

**先放开之前的注释，除了我们已经添加的mybatis的依赖之外，我们需要再添加mybatis-generator-core的依赖**

<**dependency**>

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

<**artifactId**>mybatis-generator-core</**artifactId**>

<**version**>1.4.0</**version**>

</**dependency**>

**3、编写逆向工程的配置文件generatorConfig.xml**

*<?***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="testTables" targetRuntime="MyBatis3"**>

<**commentGenerator**>

*<!-- 是否去除自动生成的注释 true：是 ： false:否 -->*

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

</**commentGenerator**>

*<!--数据库连接的信息：驱动类、连接地址、用户名、密码 -->*

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

**connectionURL="jdbc:mysql://slave3:3306/emis" userId="mybatis" password="111111"**>

</**jdbcConnection**>

*<!-- <jdbcConnection driverClass="oracle.jdbc.OracleDriver"*

*connectionURL="jdbc:oracle:thin:@127.0.0.1:1521:yycg" userId="yycg" password="yycg">*

*</jdbcConnection> -->*

*<!-- 默认false，把JDBC DECIMAL 和 NUMERIC 类型解析为 Integer，为 true时把JDBC DECIMAL 和*

*NUMERIC 类型解析为java.math.BigDecimal -->*

<**javaTypeResolver**>

<**property name="forceBigDecimals" value="false"** />

</**javaTypeResolver**>

*<!-- targetProject:生成PO类的位置 -->*

<**javaModelGenerator targetPackage="com.test.emis.entity"**

**targetProject=".\src\main\java"**>

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

<**property name="enableSubPackages" value="false"** />

*<!-- 从数据库返回的值不清理前后的空格，直接返回 -->*

<**property name="trimStrings" value="false"** />

</**javaModelGenerator**>

*<!-- targetProject:mapper映射文件生成的位置 -->*

<**sqlMapGenerator targetPackage="com.test.emis.mapper"**

**targetProject=".\src\main\java"**>

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

<**property name="enableSubPackages" value="false"** />

</**sqlMapGenerator**>

*<!-- targetPackage：mapper接口生成的位置 -->*

<**javaClientGenerator type="XMLMAPPER" targetPackage="com.test.emis.mapper"**

**targetProject=".\src\main\java"**>

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

<**property name="enableSubPackages" value="false"** />

</**javaClientGenerator**>

*<!-- 指定数据库表 ，不生成Example相关的代码和配置-->*

<**table tableName="class" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="examination\_question" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="faculty" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="major" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="major\_semester" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="major\_semester\_subject" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="menu" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="question\_bank" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="question\_options" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="question\_template" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="role" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="role\_menu" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="score" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="subject" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="user" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

<**table tableName="user\_role" enableCountByExample="false"**

**enableUpdateByExample="false" enableDeleteByExample="false"**

**enableSelectByExample="false" selectByExampleQueryId="false"**></**table**>

</**context**>

</**generatorConfiguration**>

**4、编写逆向工程的java类**

**import** org.mybatis.generator.api.MyBatisGenerator;

**import** org.mybatis.generator.config.Configuration;

**import** org.mybatis.generator.config.xml.ConfigurationParser;

**import** org.mybatis.generator.internal.DefaultShellCallback;

**import** java.io.File;

**import** java.util.ArrayList;

**import** java.util.List;

**public class** Generator {

**public static void** main(String[] args) **throws** Exception{

List<String> warnings = **new** ArrayList<String>();

**boolean** overwrite = **true**;

File configFile = **new** File(**"E:\\shixun\\emis\\src\\main\\java\\com\\mybatis\\generator\\generatorConfig.xml"**);

ConfigurationParser cp = **new** ConfigurationParser(warnings);

Configuration config = cp.parseConfiguration(configFile);

DefaultShellCallback callback = **new** DefaultShellCallback(overwrite);

MyBatisGenerator myBatisGenerator = **new** MyBatisGenerator(config, callback, warnings);

myBatisGenerator.generate(**null**);

}

}

**5、运行Generator类的main方法生成实体类、mapper接口和mapper的xml文件**

# 10 springboot项目中使用MyBatis

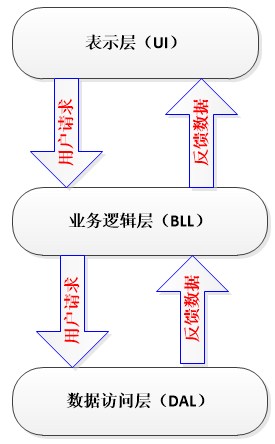
## 10.1 三层架构

**1、什么是三层？**

**UI(表现层): 主要是指与用户直接交互的部分。用于接收用户输入的数据和显示处理后用户需要的数据。**

**BLL:(业务逻辑层): UI层和DAL层之间的桥梁。实现业务逻辑。业务逻辑具体包含：验证、计算、业务规则等等。**

**DAL:(数据访问层): 与数据库打交道。主要实现对数据的增、删、改、查。将存储在数据库中的数据提交给业务层，同时将业务层处理的数据保存到数据库。（当然这些操作都是基于UI层的。用户的需求反映给界面（UI），UI反映给BLL，BLL反映给DAL，DAL进行数据的操作，操作后再一一返回，直到将用户所需数据反馈给用户）**

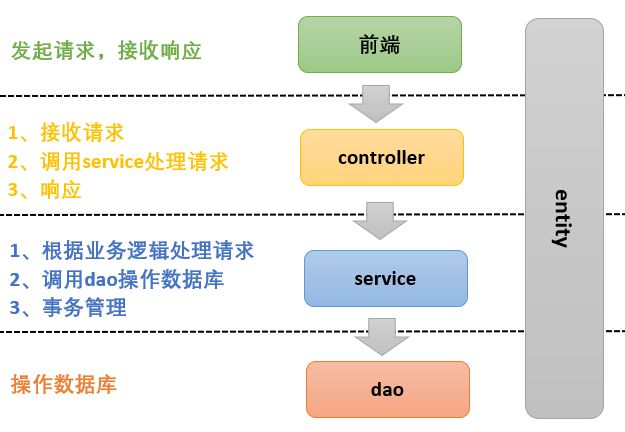


**2、为什么使用三层架构**

**代码解耦。**

**便于开发和维护，发生在哪一层的变化，只需更改该层，不需要更改整个系统。层次清晰，分工明确，每层之间耦合度低——提高了效率，适应需求变化，可维护性高，可扩展性高**

**3、根据三层架构划分代码**



**三层架构是一种架构思想，目的是代码的解耦，不是说前端就必须是使用html+js，后端必须使用java、Python、PHP等等这些。假设一个项目完全使用js开发，也是可以按照三层架构思想完成分层，划分出controller、service、dao这些。**

**4、本项目中三层结构的应用**

**1）前端，html+css+js。**

**2）controller，接收请求和响应，使用springboot的controller。**

**3）service，业务逻辑处理，面向接口编程（解耦）。**

**4）dao，操作数据库，使用MyBatis框架，mapper接口+xml。**

## 10.2 实操

**实现登录**

### 10.2.1 准备工作

**1、pom.xml配置**

*<?***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**>

<**parent**>

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

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

<**version**>2.6.2</**version**>

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

</**parent**>

<**groupId**>com.shixun</**groupId**>

<**artifactId**>tmis</**artifactId**>

<**version**>1.0-SNAPSHOT</**version**>

<**name**>tmis</**name**>

<**properties**>

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

<**maven.compiler.source**>1.8</**maven.compiler.source**>

<**maven.compiler.target**>1.8</**maven.compiler.target**>

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

</**properties**>

<**dependencies**>

*<!-- springboot的web模块 -->*

<**dependency**>

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

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

</**dependency**>

*<!-- springboot的MyBatis模块 -->*

<**dependency**>

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

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

<**version**>2.2.1</**version**>

</**dependency**>

*<!-- MySQL驱动 -->*

<**dependency**>

<**groupId**>mysql</**groupId**>

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

<**scope**>runtime</**scope**>

*<!--*

*1、如果你的数据库是mysql8，这段<version>5.1.32</version>不用管*

*2、如果你的数据库是mysql5.x，放开<version>5.1.32</version>这段注释*

*-->*

*<!--<version>5.1.32</version>-->*

</**dependency**>

*<!-- 数据库连接池 -->*

<**dependency**>

<**groupId**>com.alibaba</**groupId**>

<**artifactId**>druid</**artifactId**>

<**version**>1.1.23</**version**>

</**dependency**>

*<!-- json相关工具类 -->*

<**dependency**>

<**groupId**>com.alibaba</**groupId**>

<**artifactId**>fastjson</**artifactId**>

<**version**>1.2.73</**version**>

</**dependency**>

*<!-- MyBatis逆向工程依赖 -->*

<**dependency**>

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

<**artifactId**>mybatis-generator-core</**artifactId**>

<**version**>1.4.0</**version**>

</**dependency**>

</**dependencies**>

<**build**>

<**resources**>

*<!--*

*MyBatis的xml映射文件是放在src/mian/java下的包中的，*

*默认下springboot会过滤掉src/mian/java下的所有非java文件，*

*需要进行以下配置，让springboot不要过滤掉rc/mian/java下xml映射文件*

*-->*

<**resource**>

<**directory**>src/main/java</**directory**>

<**includes**>

<**include**>\*\*/\*.xml</**include**>

</**includes**>

<**filtering**>false</**filtering**>

</**resource**>

</**resources**>

</**build**>

</**project**>

**2、application.yml配置**



### 10.2.2 前端实现

**1、编写登录页面**

**2、编写js**

**使用jQuery获取输入框的值（包含前端校验），使用ajax向controller发起登录请求，根据响应数据判断登录是否成功，如果成功跳转到index页面，如果失败提示“用户名密码错误！”并且停留在登录页面。**

### 10.2.3 实现dao

**我们使用MyBatis框架，逆向工程已经生成数据库表对应的Entity、mapper接口和xml映射文件。**

**1、UserMapper接口上加上@Mapper注解**

clipboard.png

**2、在UserMapper接口中，添加查询user的方法，参数user对象，返回值是user对象。**

clipboard.png

**3、在UserMapper.xml文件中，添加查询配置，具体如下**

<**select id="login" parameterType="com.shixun.entity.User" resultMap="BaseResultMap"**>

select

<**include refid="Base\_Column\_List"** />

from user

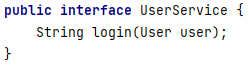
where username = #{username} and password = #{password}

</**select**>

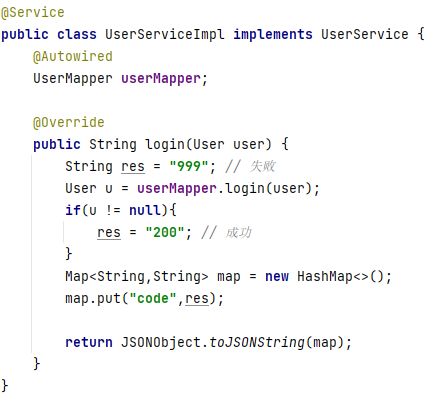
**注意，xml映射文件中的配置的id与mapper接口的方法名一一对应。**

### 10.2.4 实现service

**1、编写UserService接口，代码如下。**



**2、编写实现类UserServiceImpl，类上加@Service注解，依赖注入UserMapper属性，代码如下。**



### 10.2.5 实现controller



# 11 功能实现

## 11.1 菜单管理

### 11.1.1 新增菜单

**与登陆实现方式类似，关键逻辑如下。**

**1、菜单总共2级，页面需要做选择菜单级别的下拉列表。**

**2、当选择菜单级别为二级菜单时，需要显示一级菜单下拉列表（下拉选项从数据库查询），此处需要下拉列表联动处理。**

### 11.1.2 菜单列表

#### 11.1.2.1 前端部分

**1、html页面中必须有以下代码，这个div就是列表加载的div**

<**div id="jsGrid1"**></**div**>

**2、编写js**

$(**function** () {

$(**"#jsGrid1"**).jsGrid({

*// height: "100%",*

width: **"100%"**,

height: **"auto"**,

*// width: "auto",*

sorting: **true**, *// 排序*

paging: **true**, *// 分页计算，底部上一页下一页展示*

pageLoading: **true**, *//启动后台加载分页数据*

autoload: **true**, *//自动加载*

controller:{

loadData: **function**(filter){

*// alert(filter.pageIndex);*

**return** $.ajax({

type: **"post"**,

url: **"/menu/list"**,

dataType: **"json"**,

data: filter

});

}

},

pageIndex: 1, *// 当前页数，是第几页*

pageSize: 3, *// 每页数据条数*

pageButtonCount: 10, *// 展示可选页码数量*

pagePrevText: **"上一页"**,

pageNextText: **"下一页"**,

pageFirstText: **"首页"**,

pageLastText: **"尾页"**,

*// data: userList,*

fields: [

{

title: **""**, name: **"id"**, type: **"text"**, width: 20, align: **"center"**,

itemTemplate: **function**(value, item){

**return "<input type='checkbox' value='"** + value + **"'>"**;

}

},

{ title: **"菜单名"**, name: **"menuName"**, type: **"text"**, width: 100 },

{ title: **"菜单编码"**, name: **"menuCode"**, type: **"text"**, width: 80 },

{ title: **"菜单url"**, name: **"menuUrl"**, type: **"text"**, width: 80 },

{ title: **"上级菜单"**, name: **"parentMenu"**, type: **"text"**, width: 80 },

{

title: **"操作"**, name: **"id"**, type: **"text"**, width: 60, align: **"center"**,

itemTemplate: **function**(value, item){

**return "<a href='javascript:void(0)' onclick='alert("**+value+**");'>修改</a>"**

+ **"&nbsp;&nbsp;&nbsp;&nbsp;<a href='javascript:void(0)' onclick='alert("**+value+**");'>删除</a>"**;

}

}

]

});

});

#### 11.1.2.2 后端部分

**1、controller接收JSGrid请求参数**

String pageSize = **request**.getParameter(**"pageSize"**);

String pageIndex = **request**.getParameter(**"pageIndex"**);

**2、controller返回json格式**

**{**

**"data" : 数据集合,**

**"itemsCount" : 总条数**

**}**

**3、dao查询逻辑**

**1）根据pageIndex 和pageSize 做limit分页查询**

**SQL语句：**

String sql = **"SELECT \* FROM user1 limit ?,?"**;

**limit参数：**

**偏移量 =（pageIndex - 1）\* pageSize**

**步长 = pageSize**

**2）查询总条数**

**select count(\*) from user;**

### 11.1.3 菜单删除

**ajax请求，根据菜单id删除数据。**

### 11.1.4 菜单修改

**1、与菜单新增共用页面，在新增页添加一个隐藏域，用来存放被修改菜单的id值。**

**2、在菜单加载的时候判断隐藏域有没有值，有值就是修改页，没有值就是新增页。**

**3、是修改页的时候，ajax请求，根据菜单id查询数据，回填到表单元素中。其中注意回填父菜单，判断被修改的菜单的是不是二级菜单，如果是二级菜单，需要给父菜单的下拉列表加载所有的一级菜单选项，并且设置被修改的菜单的原一级菜单的选项是选中的。**

**4、提交修改到controller，与add基本一致，这里要注意需要把菜单的id值作为参数传递。**

## 11.2 角色管理

### 11.2.1 新增角色

**新增角色功能与新增菜单差不多，其中有些特殊处理：**

**给角色关联菜单，一个角色是可以关联多个菜单的，菜单本身也有层级关系（父子菜单结构），这个在页面展示选择菜单的时候，使用Tree结构更合适。**

**Tree结构的实现方式很多，我们这里使用zTree来实现。前端代码实例见“资料”中的“ztree示例.html”。**

**ztree所需的文件在“资料”中的“zTree\_v3-master.zip”。**

**百度网盘地址**

**链接：https://pan.baidu.com/s/1cwp9uGQ4NINsQIbfz2yYRA**

**提取码：9rao**

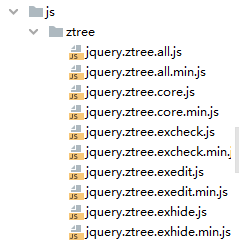
**新增角色，需要新增角色本身以及角色菜单关联表，要保证两个写的操作同时成功，否则就全部失败，这就需要使用数据库事务来实现。**

**数据库事务( transaction)是访问并可能操作各种数据项的一个数据库操作序列，这些操作要么全部执行,要么全部不执行，是一个不可分割的工作单位。事务由事务开始与事务结束之间执行的全部数据库操作组成。**

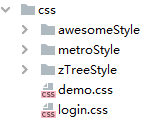
**关键代码：**

**1、ztree的js和css放到项目中**

**zTree\_v3-master\js下的文件都放到webapp/js/ztree下**



**zTree\_v3-master\css下的文件都放到webapp/css下**



**2、role-add.html页面引入ztree的js和css**

<**link rel="stylesheet" href="../css/zTreeStyle/zTreeStyle.css" type="text/css"**>

<**script src="../js/ztree/jquery.ztree.all.min.js"**></**script**>

**3、role-add.html页面，去掉下拉列表部分，增加以下代码**

<**div class="form-group"**>

<**label for="menuTree"**>菜单列表</**label**>

<**ul id="menuTree" class="ztree"**></**ul**>

<**input type="hidden" id="menuIds" class="form-control"**>

</**div**>

**4、role-add.js中新增以下代码**

$(**function**(){

$.ajax({

**"type"** : **"POST"**,

**"url"** : **"/emis/menu/tree.do"**,

**"dataType"** : **"json"**,

**"success"** : **function**(data){

loadMenuTree(data);

}

});

});

**var** loadMenuTree = **function**(zNodes){

**var** zTreeObj;*//zTree对象*

*// zTree 的参数配置*

**var** setting = {

view: {

selectedMulti: **true**, *//设置是否能够同时选中多个节点*

showIcon: **true**, *//设置是否显示节点图标*

showLine: **true**, *//设置是否显示节点与节点之间的连线*

showTitle: **true**, *//设置是否显示节点的title提示信息*

},

data: {

simpleData: {

enable: **true**, *//设置是否启用简单数据格式（zTree支持标准数据格式跟简单数据格式）*

idKey: **"id"**, *//设置启用简单数据格式时id对应的属性名称*

pIdKey: **"parentId"**,*//设置启用简单数据格式时parentId对应的属性名称,ztree根据id及pId层级关系构建树结构*

rootPId: **"0"**

},

key: { *// 当要组装树的数据的json结构与当前设置的不相符的时候，可以通过这里设置映射关系*

*// 在ztree节点展示中，节点的name属性的值从数据中的menuName对应的值获取*

name: **"menuName"**

}

},

check:{

enable: **true** *//设置是否显示checkbox复选框*

}

};

zTreeObj = $.fn.zTree.init($(**"#menuTree"**), setting, zNodes);

}

**var** getCheckedMenuIds = **function**(){

**var** treeObj = $.fn.zTree.getZTreeObj(**"menuTree"**);

**var** nodes = treeObj.getCheckedNodes(**true**);

**var** ids = **""**;

$.each(nodes, **function**(index, node){

ids += **","** + node.id;

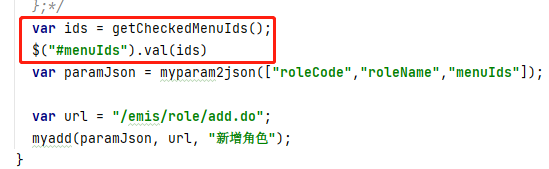
});

ids = ids.substring(1, ids.length);

**return** ids;

}

**5、role-add.js中的roleadd方法，添加如下代码**



**6、后端在新增角色菜单关联表时，需要使用新增角色的主键id，这个可以在mybatis的xml中的insert标签配置以下属性，就可以把id值赋值到实参对象的id属性上。**

<**insert id="xxx" parameterType="xxxxx" useGeneratedKeys="true" keyProperty="id"**>

### 11.2.2 角色列表

**无特殊处理，仿照菜单列表实现。**

### 11.2.3 删除角色

**删除角色需要同时删除角色表和角色菜单关联表的数据，service方法设置事务。**

### 11.2.4 修改角色

**1、数据回填到页面时，需要查询角色下属的菜单id，设置菜单ztree节点选中，js关键代码如下。**

**var** arr = [**'11'**,**'12'**];

**var** zTree = $.fn.zTree.getZTreeObj(**"menuTree"**);

$.each(arr,**function**(index,v){

**var** node = zTree.getNodeByParam(**"id"**,v);

**if**(node.parentId != 0){ *// 判断是二级节点*

zTreeObj.checkNode(node, **true**, **true**); *// 设置节点选中*

}

});

**2、提交数据到后台，在修改角色时，数据库操作如下。**

**1）修改角色表数据**

**2）删除角色对应的角色菜单关联表数据，新增角色菜单关联表数据**

## 11.3 用户管理

### 11.3.1 新增用户

**根据新增菜单实现，特殊部分如下。**

**1、页面部分表单除了用户本身数据，还有角色数据。我们这里只考虑用户和角色一对一关联，ajax加载一个角色的下拉列表即可。**

**2、后台除了保存用户本身数据，还要保存用户角色关联数据，一个事务中完成。**

### 11.3.2 用户列表

**根据角色列表完成即可，后台SQL的查询条件是按照用户名查询即可。**

### 11.3.3 修改用户

**1、回填表单数据时，除了回填用户本身数据，还要设置角色下拉列表的选项的选中。**

**2、后台保存修改用户的数据和保存修改角色类似，需要完成3个操作，在一个事务中。**

**1）保存用户数据本身。**

**2）根据根据用户id删除用户角色关联表数据。**

**3）新增用户角色关联表数据。**

### 11.3.4 删除用户

**需要完成2个操作，在一个事务中。**

**1、删除用户数据。**

**2、根据根据用户id删除用户角色关联表数据。**

## 11.4 模板管理

### 11.4.1 新建模板

<div class="card-body">

*<!--*

*院系、专业、学期、科目四个下拉列表，是一个三级联动*

*1、院系的下拉列表的选项是在页面加载的是自动去加载出来*

*2、专业的下拉列表选项加载，在选择院系的某个选项的时候触发，需要在院系的下拉列表onchange*

*3、学期的下拉列表选项加载，在选择专业的某个选项的时候触发，需要在专业的下拉列表onchange*

*4、科目的下拉列表选项加载，在选择学期的某个选项的时候触发，需要在学期的下拉列表onchange*

*-->*

<div class="form-group">

<label for="faculty\_id">院系</label>

<select id="faculty\_id" class="form-control custom-select" onchange="load\_major\_options()">

</select>

</div>

<div class="form-group">

<label for="major\_id">专业</label>

<select id="major\_id" class="form-control custom-select" onchange="load\_class\_options()">

</select>

</div>

<div class="form-group">

<label for="major\_semester\_id">学期</label>

<select id="major\_semester\_id" class="form-control custom-select" onchange="load\_class\_options()">

</select>

</div>

<div class="form-group">

<label for="subject\_id ">科目</label>

<select id="subject\_id " class="form-control custom-select" onchange="load\_class\_options()">

</select>

</div>

<div class="form-group">

<label for="template\_name">模板名称</label>

<input type="text" id="template\_name" class="form-control">

</div>

<div class="form-group">

<label for="single\_num">单选题数目</label>

<input type="text" id="single\_num" class="form-control">

</div>

<div class="form-group">

<label for="single\_score">单选题分值</label>

<input type="text" id="single\_score" class="form-control">

</div>

<div class="form-group">

<label for="single\_difficulty\_proportion">单选题难度比例(易:中:难:很难)</label>

<input type="text" id="single\_difficulty\_proportion" class="form-control"

placeholder="例如5:3:1:1">

</div>

<div class="form-group">

<label for="multiple\_num">多选题数目</label>

<input type="text" id="multiple\_num" class="form-control">

</div>

<div class="form-group">

<label for="multiple\_score">多选题分值</label>

<input type="text" id="multiple\_score" class="form-control">

</div>

<div class="form-group">

<label for="multiple\_difficulty\_proportion">多选题难度比例(易:中:难:很难)</label>

<input type="text" id="multiple\_difficulty\_proportion" class="form-control"

placeholder="例如5:3:1:1">

</div>

*<!--*

*aggregate\_score，总分，由程序计算，不需要手填*

*总分 = 单选题分值 \* 单选题数目 + 多选题分值 \* 多选题数目*

*-->*

</div>

### 11.4.2 模板列表

### 11.4.3 修改模板

### 11.4.4 删除模板

## 11.5 题库管理

### 11.5.1 上传题库

**我们使用Excel上传，Excel模板在“资料”文件夹中。**

#### 11.5.1.1 前端关键代码

**1、页面html代码**

<**section class="content"**>

<**form action="/test/upload" enctype="multipart/form-data" method="post"**>

<**div class="row"**>

<**div class="col-md-3"**></**div**>

<**div class="col-md-6"**>

<**div class="card card-primary"**>

<**div class="card-header"**>

<**h3 class="card-title"**>试题上传</**h3**>

<**div class="card-tools"**>

<**button type="button" class="btn btn-tool" data-card-widget="collapse" title="Collapse"**>

<**i class="fas fa-minus"**></**i**>

</**button**>

</**div**>

</**div**>

<**div class="card-body"**>

<**div class="form-group"**>

<**label for="qType"**>类型</**label**>

<**select id="qType" name="qType" class="form-control custom-select"**>

<**option value=""**>--请选择--</**option**>

<**option value="0"**>单选</**option**>

<**option value="1"**>多选</**option**>

</**select**>

</**div**>

</**div**>

<**div class="card-body"**>

<**div class="form-group"**>

<**label for="subject"**>科目</**label**>

<**select id="subject" name="subject" class="form-control custom-select"**>

<**option value=""**>--请选择--</**option**>

<**option value="1"**>毛泽东思想概论</**option**>

<**option value="2"**>大学英语</**option**>

</**select**>

</**div**>

</**div**>

<**div class="card-body"**>

<**div class="form-group"**>

<**label for="inputName"**>选择文件</**label**>

<**input type="file" id="inputName" name="questonFile" class="form-control"**>

</**div**>

</**div**>

*<!-- /.card-body -->*

</**div**>

*<!-- /.card -->*

</**div**>

</**div**>

<**div class="row"**>

<**div class="col-9"**>

<**input type="submit" value="提交" class="btn btn-success float-right"**>

</**div**>

</**div**>

</**form**>

</**section**>

**2、左侧菜单的连接设置**

<**li class="nav-item"**>

<**a href="javascript:void(0);" class="nav-link" onclick="window.open('./file-upload.html')"**>

<**i class="far fa-circle nav-icon"**></**i**>

<**p**>试题上传</**p**>

</**a**>

</**li**>

#### 11.5.1.2 后台关键代码

**1、pom依赖**

<**dependency**>

<**groupId**>org.apache.poi</**groupId**>

<**artifactId**>poi</**artifactId**>

<**version**>3.14</**version**>

</**dependency**>

<**dependency**>

<**groupId**>org.apache.poi</**groupId**>

<**artifactId**>poi-ooxml</**artifactId**>

<**version**>3.14</**version**>

</**dependency**>

**2、poi解析Excel**

**public class** PoiUtil {

**public static** List<Map<String, Object>> readExcel(InputStream in) **throws** Exception{

*//in = new FileInputStream(new File(""));*

List<Map<String, Object>> list = **new** ArrayList<>();

*// 3、读取输入流，构建Excel文件对象，根据Excel版本不同构建对象有差别*

Workbook workbook = **null**;

workbook = **new** HSSFWorkbook(in); *// Excel2003*

*//workbook = new XSSFWorkbook(in); // Excel2007*

*// 4、得到文件中的sheet页，可以根据index（从0开），可以根据sheet的名称获取*

Sheet sheet = workbook.getSheetAt(1); *// 读取第二个页*

*// 5、从sheet中得到行对象,先获取行号，开始行和结束行*

**int** firstRowNum = sheet.getFirstRowNum(); *// 不一定是sheet页第1行，第一次开始有内容的行就是首行*

**int** lastRowNum = sheet.getLastRowNum(); *// 最后一行，最后有内容那一行*

*// 设置map集合中列名对应的key*

String[] tiles = **new** String[]{**"q\_name"**,**"options"**,**"answer"**,**"diff"**};

*/\**

*6、根据首行和尾行的行号，遍历所有行*

*通常来说，首行一般是标题行，标题行一般不解析，遍历的时候从首行的下一行开始*

*\*/*

**for**(**int** rowNum = firstRowNum + 1; rowNum <= lastRowNum; rowNum++){

*// 一行是一道题，一个map对象*

Map<String, Object> map = **new** HashMap<>();

*// 7、得到每行的对象*

Row row = sheet.getRow(rowNum);

*// 8、获取一行的开始列和列数，再去遍历*

**int** firstCellNum = row.getFirstCellNum(); *// 开始列的列号*

**int** cellsCount = row.getPhysicalNumberOfCells(); *// 列数*

*// 遍历*

**for**(**int** cellNum = firstCellNum; cellNum < cellsCount; cellNum++){

*// 9、得到列对象，cell对象，晶格对象*

Cell cell = row.getCell(cellNum);

*// 10、取列的值*

String cellValue = *getCellValue*(cell);

*// 对于选项来说，一个cell里有多个选项，规定一个选项写完必须换行，需要把一个cell里面的选项解析出来*

*// 从打印的情况来看，本身选项就是换行打印，意味着一个cell的选项之间是有一个分隔符（换行符\n）*

**if**(cellNum == 1){ *// 选项的列*

String[] options = cellValue.split(**"\n"**);

**for**(String option : options){

System.***out***.println(**"选项："** + option);

}

map.put(tiles[cellNum], options);

}**else** {

map.put(tiles[cellNum], cellValue);

System.***out***.println(cellValue);

}

System.***out***.println(**"--------------------------------------"**);

}

String questionCode = UUID.*randomUUID*().toString();

map.put(**"questionCode"**, questionCode);

list.add(map);

}

**return** list;

}

*/\*\**

*\* 获取cell的值*

*\** ***@param cell***

*\** ***@return***

*\*/*

**public static** String getCellValue(Cell cell){

String cellValue = **""**;

*// 如果cell是数字类型，把它按字符串类型来处理一下*

**if**(cell.getCellType() == Cell.***CELL\_TYPE\_NUMERIC***){

cell.setCellType(Cell.***CELL\_TYPE\_STRING***);

}

**switch** (cell.getCellType()){

**case** Cell.***CELL\_TYPE\_STRING***: *// 字符串*

cellValue = String.*valueOf*(cell.getStringCellValue());

**break**;

**case** Cell.***CELL\_TYPE\_BOOLEAN***: *// boolean*

cellValue = String.*valueOf*(cell.getBooleanCellValue());

**break**;

**case** Cell.***CELL\_TYPE\_FORMULA***: *// 公式*

cellValue = String.*valueOf*(cell.getCellFormula());

**break**;

**case** Cell.***CELL\_TYPE\_BLANK***: *// 空格*

cellValue = **""**;

**break**;

**case** Cell.***CELL\_TYPE\_ERROR***: *// 错误*

cellValue = **"非法字符"**;

**break**;

**default**:

cellValue = **"未知类型"**;

**break**;

}

**return** cellValue;

}

}

**2、springboot实现上传**

@RequestMapping(**"/upload"**)

@ResponseBody

**public** String upload(MultipartFile questonFile) **throws** Exception{

*// 上传文件*

String fileName = questonFile.getOriginalFilename();

System.***out***.println(**"------"** + fileName);

*// 得到上传文件的流*

InputStream in = questonFile.getInputStream();

*// 读取Excel，返回list集合*

List<Map<String, Object>> list = PoiUtil.*readExcel*(in);

**for**(Map<String, Object> map : list){

System.***out***.println(map);

}

System.***out***.println(**"qType: "** + **request**.getParameter(**"qType"**));

System.***out***.println(**"subject: "** + **request**.getParameter(**"subject"**));

**return "<script>"** +

**"alert('上传成功');"** +

**"window.opener=null;"** +

**"window.open('','\_self');"** +

**"window.close();"** +

**"</script>"**;

}

**3、保存数据库**

**我们最终得到的是试题对象与选项list集合，如下。**

Question question = map.get(**"questions"**);

List<QuestionOption> questionOptions = map.get(**"questionOptions"**);

**我们需要给mapper传递list集合参数的，可以在xml中使用类似如下代码，实现批量插入。**

<**foreach collection="list" item="item" index="index" open="(" close=")" separator=","**>

#{item.id},#{item.subjectId},#{item.questionCode},#{item.questionName}

</**foreach**>

## 11.6 院系管理

### 11.6.1 新建院系

### 11.6.2 院系列表

### 11.6.3 修改院系

### 11.6.4 删除院系

## 11.7 专业管理

**专业增删改查**

## 11.8 班级管理

**班级增删改查**

## 11.9 科目管理

**科目增删改查**

## 11.10 专业学期管理

**专业学期增删改查**

## 11.11 试卷管理

### 11.11.1 生成试卷

**根据院系、专业、学期、科目、生成考卷存放在内存中，每个科目可以多生成几份，在考试时随机抽取一份试卷。**

**关键逻辑：考卷的考题来自查询题库，这里需要根据考卷模板随机抽取考题，随机抽取算法详见《随机抽题-Java》。**

## 11.12 考试管理

### 11.12.1 在线考试

**从内存中获取考卷数据，在页面组装成考卷，供学生考试使用。**

### 11.12.2 成绩查询

**根据院系、专业、学期、科目、当前登录的用户id，查询当前登录用户的指定科目的成绩。**

**注意：院系、专业、学期、科目属于可选条件，如果都不选，就是查询当前用户所有已经考完的科目的成绩。**

## 11.13 统计信息

### 11.13.1 统计成绩

**汇总学生成绩，进行相应的统计，通过图表展示。**