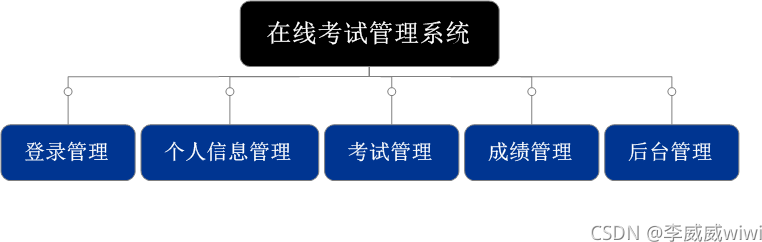
Python+Django 实现在线考试管理系统

本文最终实现一个 Web 在线考试管理系统，可作为 Python Web，Django 的练手项目，也可以作为计算机毕设参考项目。

# 系统功能需求分析

在线考试管理系统，主要包含如下几个功能模块：



1、**用户登录及**：实现简单的登录及验证

2**、个人信息展示**：展示考生个人信息

3、**考试管理**：展示可以考试的试卷，考试及自动计算考试成绩。

4、**考试成绩管理**：展示考试结果

5、**后台基础数据管理**：试卷，试题，考生等信息更新维护。

# 系统设计及实现思路

系统设计包括三部分：数据库设计，功能函数视图设计，前端页面设计

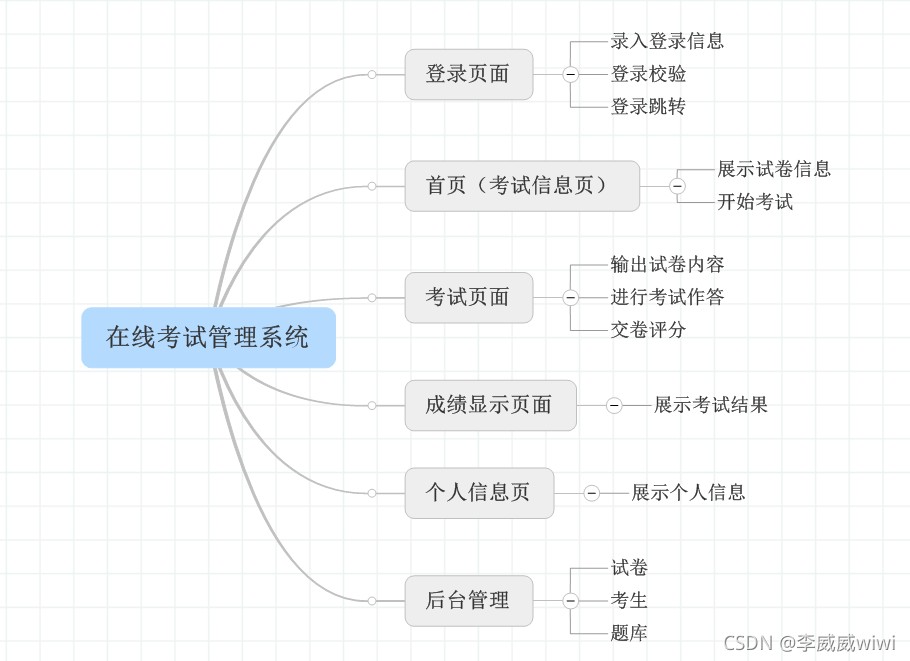
## 1、数据库设计

根据我们的的系统需求分析，我们预计需要如下几个数据表：

* 学院表：Academy
* 专业表：Major
* 课程表：Course
* 学生表：Student
* 题库表：QuestionBank
* 试卷表：TestPaper
* 学生成绩表：Record

## 2、页面及功能设计

为了实现我们前面的功能模块我们设计如下几个功能页面：



**1、登录页面：**

其中需要登录，校验，登录后同时需要存储用户信息在 Session 中，以备登录后的页面使用。

**2、首页（考试信息页）：**

页面需要显示当前用户可以参考的考试试卷信息，在此页面点击开始考试可以跳转到

考试页面。

**3、考试页面：**

展示对应试卷的题目和选项信息，同时可以进行答题，交卷后可以自动计算考试分数并存入数据库。

**4、成绩显示页面：**

展示对应考试的考试结果

**5、后台管理：**

用于管理我们的专业，考生，试卷，题库等基础信息，为了快速实现系统我们将直接启用 Django 自带的 Admin 管理功能。

**6、个人详情：**

用于展示个人详情信息。

# 系统实现过程及源码

大致理清了我们需要实现的功能模块之后，我们开始撸代码

## 1 、开发环境搭建及技术选型

服务端：Python 3.8

Web 框架：Django 3.2

数据库：MySQL mysql-8.0.13-winx64开发工具 IDE：Pycharm（社区版）

前端框架：Bootstrap 4

## 2、Django 项目创建及数据库连接配置

1.在任意盘符文件夹下新建一个空白 Pycharm 项目文件夹比如:PycharmProjects 2.打开 Pycharm,进入空白文件夹:PycharmProjects

3.通过 Pycharm 的 Windows 命令行输入界面输入创建 Django 项目的命令，创建一个新的项目: DjangoExam

django-admin startproject DjangoExam

## 3、 数据库创建和连接配置

Django 对各种数据库提供了很好的支持，包括：PostgreSQL、MySQL、SQLite、Oracle。 Django 为这些数据库提供了统一的调用 API。

我们可以根据自己业务需求选择不同的数据库。 MySQL 是 Web 应用中最常用的数据库。

本文采用 MySQL。

此步骤将数据库设置连接到自己的 MySQL 数据库,并完成数据库的创建.

### 数据库创建

Django 只能操作到数据表级别,不能操作到数据库级别,所以需要手工创建一个数据库:**djangoexam**

我们可以通过命令行创建一个数据库:

1. **进入 mysql 安装文件夹的 bin 子文件夹目录:**

比如:D:\Program Files\mysql-8.0.13-winx64\bin

1. **连接数据库:**

mysql -u root -p Enter password:\*\*\*\*\*\*

1. **连接登录成功后通过命令创建一个数据库:djangoexam**

CREATE DATABASE IF NOT EXISTS **djangoexam** DEFAULT CHARSET utf8;

或者通过 SQLlog 工具创建一个数据库: **djangoexam**

### Django 数据库连接配置

Django 使用 MySQL 需要 mysql 驱动,如果你没安装 mysql 驱动，可以执行以下命令安装：

pip install pymysql

安装好之后， 进入 DjangoExam 项目下的 DjangoExam 文件夹，打开 setting.py文件,找到 DATABASES 配置项，修改 DATABSES 配置项为如下内容:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql', # 数据库引擎

'NAME': 'djangoexam', # 数据库名称

'HOST': '127.0.0.1', # 数据库地址，本机 ip 地址 127.0.0.1 'PORT': 3306, # 端口

'USER': 'root', # 数据库用户名

'PASSWORD': '123456', # 数据库密码

}

}

然后使用 pymysql 模块连接 mysql 数据库：

在与 settings.py 同级目录下的 init .py 中引入模块和进行配置：

import pymysql pymysql.install\_as\_MySQLdb()

**至此，我们创建了一个 Django 项目 DjangoExam 用于我们后续的在线考试管理系统**

**开发的程序编写。**

**同时为此项目创建了一个 MySQL 数据库:djangoexam 用于我们程序开发过程中的数据存放和处理。**

## 4、 功能模块详细开发

一个 Django 项目框架搭建起来后，我们所有对系统的前后台所有的程序开发都可以在这个项目中进行了，一个典型的 Django 项目模块功能的开发包括如下几个步骤：

* **创建 app**
* **注册 app**
* **定义模型**
* **定义视图函数**
* **配置访问路由 URL**
* **静态资源准备及配置**
* **前端模板开发**
* **测试及运行**

### 创建 APP

在 Django 中的一个 app 代表一个功能模块，Django 规定，如果要使用模型，必须要 创建一个 app。

本来一般实际开发中方便管理一个功能模块需要单独新建一个app,我们为了快速开发，所有的功能和模型都在一个 app 中进行管理。

在命令行中输入 python manage.py startapp exam 指令，创建名为 exam 的 app：

python manage.py startapp exam

### 注册 APP

在 DjangoExam 文件夹目录下的 settings.py 中找到 INSTALLED\_APPS 配置项，将新创建的 exam 添加到项目的 app 列表,如下：

INSTALLED\_APPS = [

'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', 'exam', #添加此项

]

### 定义模型

我们之前创建了一个空白的数据库 djangoexam,这一步我们通过 Django 的模型来完成数据库表的创建.

因为 Django 对模型和目标数据库之间有自身的映射规则,如果自己在数据库中创建数据表,可能不一定符合 Django 的建表规则,从而导致模型和目标数据库无法建立通信联系。

所以最好我们在 Django 项目中还是通过 Django 模型来创建对应数据库表一个表对应一个模型，本步骤按照我们前面数据库设计需要设计 7 个模型：

打开 exam/models.py 文件，输入如下代码：

from django.db import models

# 学院表

class Academy(models.Model):

id = models.AutoField('序号',primary\_key=True) name = models.CharField('学院',max\_length=20)

# 修改显示的表的名字

class Meta:

verbose\_name = '学院'

verbose\_name\_plural = '学院'

def str (self):

return self.name

# 专业表

class Major(models.Model):

id = models.AutoField('序号',primary\_key=True)

academy = models.ForeignKey(Academy,on\_delete=models.CASCADE,verbose\_name='学院') major = models.CharField('专业',max\_length=30)

# 修改显示的表的名字

class Meta:

verbose\_name = '专业'

verbose\_name\_plural = '专业' def str (self):

return self.major

# 课程表

class Course(models.Model):

id = models.AutoField('序号',primary\_key=True)

course\_id = models.CharField('课程号',max\_length=10) course\_name = models.CharField('课程名称',max\_length=30)

class Meta:

verbose\_name = '课程'

verbose\_name\_plural = '课程'

def str (self):

return self.course\_name

# 学生表

class Student(models.Model):

sid = models.CharField('学号',max\_length=12,primary\_key=True) name = models.CharField('姓名',max\_length=20,unique=True) sex = models.BooleanField('性别',choices=((0,'女'),(1,'男')))

age = models.IntegerField('年龄')

academy = models.ForeignKey(Academy,on\_delete=models.CASCADE,verbose\_name='学院') major = models.ForeignKey(Major,on\_delete=models.CASCADE,verbose\_name='专业')

sclass = models.CharField('班级',max\_length=20,help\_text='例如: 17-03') email = models.EmailField('邮箱',default=None) # 默认为空 唯一值 pwd = models.CharField('密码',max\_length=20)

# 修改显示的表的名字

class Meta:

verbose\_name = '学生'

verbose\_name\_plural = '学生信息表' def str (self):

return self.sid

# 题库表

class QuestionBank(models.Model):

id = models.AutoField('序号',primary\_key=True)

major = models.ForeignKey(Major,on\_delete=models.CASCADE,verbose\_name='专业') course = models.ForeignKey(Course,on\_delete=models.CASCADE,verbose\_name='科目') title = models.TextField('题目')

qtype = models.CharField(' 题目类型',choices=((' 单选',' 单选'),(' 多选',' 多选'),(' 判断',' 判断

')),max\_length=40)

a = models.CharField('A 选项',max\_length=40) b = models.CharField('B 选项',max\_length=40) c = models.CharField('C 选项',max\_length=40) d = models.CharField('D 选项',max\_length=40)

answer = models.CharField('答案',choices=(('A','A'),('B','B'),('C','C'),('D','D')),max\_length=4)

difficulty = models.CharField(' 难度 ',choices=(('easy',' 简单 '),('middle',' 中等 '),('difficult',' 难

')),max\_length=10)

score = models.IntegerField('分值')

class Meta:

# 选择这个表之后显示的名字

verbose\_name = '题库' # 显示的表名

verbose\_name\_plural = '题库'

def str (self):

return '<%s:%s>' % (self.course, self.title)

# 试卷表

class TestPaper(models.Model):

id = models.AutoField('序号',primary\_key=True)

title = models.CharField('题目',max\_length=40,unique=True) pid = models.ManyToManyField(QuestionBank)

course = models.ForeignKey(Course,on\_delete=models.CASCADE,verbose\_name='科目')

major = models.ForeignKey(Major,on\_delete=models.CASCADE,verbose\_name='考卷适合专业') time = models.IntegerField('考试时长',help\_text='单位是分钟')

examtime = models.DateTimeField('上次考试时间')

class Meta:

# 选择这个表之后显示的名字

verbose\_name = '试卷'

verbose\_name\_plural = '试卷'

# # 学生成绩表

class Record(models.Model):

id = models.AutoField('序号',primary\_key=True)

sid = models.ForeignKey(Student,on\_delete=models.CASCADE,verbose\_name=' 学 号

',related\_name='stu\_xuehao')

course = models.ForeignKey(Course,on\_delete=models.CASCADE,verbose\_name=' 考 试 科 目

',related\_name='stu\_course')

grade = models.FloatField('成绩')

rtime = models.DateTimeField('考试时间',blank=True,null=True)

class Meta:

verbose\_name = '学生成绩'

verbose\_name\_plural = '学生成绩'

def str (self):

return '<%s:%s>' % (self.sid,self.grade)

编写好了 Model 后，接下来就需要进行数据迁移。迁移是 Django 对模型所做的更改传递到数据库中的方式。

**注意，每当对数据库进行了更改（添加、修改、删除等）操作，都需要进行数据迁移。**

Django 的迁移代码是由模型文件自动生成的，它本质上只是个历史记录，Django 可以用它来进行数据库的滚动更新，通过这种方式使其能够和当前的模型匹配。

在命令行中输入命令让 Django 知道我们自定义模型有一些变更,并根据我们自定义 app 的模型生成创建数据表的脚本:

python manage.py makemigrations

最后通过命令创建 app 模型对应的数据库表:

python manage.py migrate

### 定义视图函数

Django 中视图的概念是「一类具有相同功能和模板的网页的集合」。比如，在一个考试系统中，我们可能需要如下几个视图：

**登录：输入用户和密码，根据校验结果进行登录处理。**

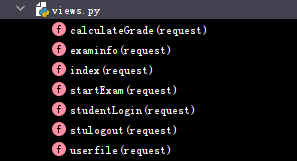
**考试：展示考试试题及选项，根据选择的结果记录考试成绩。**

这些需求都靠视图（View）来完成。

每一个视图表现为一个简单的 Python 函数，它需要要做的只有两件事：返回一个包含被请求页面内容的 HttpResponse 对象，或者抛出一个异常，比如 Http404 。

视图函数中的 request 与网页发来的请求有关，里面包含 get 或 post 的内容、用户浏览器、系统等信息。

根据系统设计过程中需要的功能，我们在 exam/views.py 文件中创建如下几个视图函数：



**studentLogin：**

此视图函数用户登录，调用此函数时系统将会检验输入的表单的用户名和密码是否匹配，如果匹配则将用户信息存入 session，同时跳转到首页（考试信息）。

# 学生登录

def studentLogin(request):

if request.method == 'POST': # 获取表单信息

sid = request.POST.get('sid')

password = request.POST.get('password') print("sid", sid, "password", password)

# 通过学号获取该学生实体

student = models.Student.objects.get(sid=sid) print(student)

if password == student.pwd: # 登录成功

request.session['username']=sid #user 的值发送给 session 里的 username request.session['is\_login']=True #认证为真

# 查询考试信息

paper = models.TestPaper.objects.filter(major=student.major) # 查询成绩信息

grade = models.Record.objects.filter(sid=student.sid)

# 渲染 index 模板

return render(request, 'index.html', {'student': student, 'paper': paper, 'grade': grade})

else:

return render(request,'login.html',{'message':'密码不正确'}) elif request.method == 'GET':

return render(request, 'login.html')

else:

return HttpResponse("请使用 GET 或 POST 请求数据")

**index：**

根据登录函数传的参数，显示该用户可以参加的考试试卷，在该页面可以点击开始考试按钮进行考试。

# 首页

def index(request):

if request.session.get('is\_login',None): #若 session 认证为真 username = request.session.get('username',None) print(username )

student = models.Student.objects.get(sid=username) # 查询考试信息

paper = models.TestPaper.objects.filter(major=student.major)

return render(request, 'index.html',{'student': student,'paper': paper})

else:

return render(request, 'index.html')

**userfile：**

显示当前用户的个人信息

def userfile(request):

if request.session.get('is\_login',None): #若 session 认证为真 username = request.session.get('username',None) print(username )

student = models.Student.objects.get(sid=username) # 查询考试信息

paper = models.TestPaper.objects.filter(major=student.major) return render(request, 'userfile.html',{'student': student})

**stulogout：**

退出登录，清楚 Session 信息

#学生退出登录

def stulogout(request): # logout(request)

request.session.clear()

url = reverse('exam:index') return redirect(url)

**startExam：**

开始考试，跳转到考试页面

# 考试信息

def startExam(request):

sid = request.GET.get('sid')

title = request.GET.get('title') # 试卷名字 唯一 subject1 = request.GET.get('subject') # 考试科目 # 获取学生信息

student = models.Student.objects.get(sid=sid) # 试卷信息

paper = models.TestPaper.objects.filter(title=title,course course\_name=subject1) context = {

'student': student, 'paper': paper,

'title': title, 'subject':subject1,

'count': paper.count() # 数据表中数据的条数

}

return render(request, 'exam.html', context=context)

**examinfo：**

显示考试成绩页面

def examinfo(request):

if request.session.get('is\_login',None): #若 session 认证为真

username = request.session.get('username',None) student = models.Student.objects.get(sid=username) # 查询成绩信息

grade = models.Record.objects.filter(sid=student.sid)

return render(request, 'examinfo.html',{'student': student,'grade': grade})

else:

return render(request, 'examinfo.html')

**calculateGrade：**

交卷时自动计算考试成绩，并存入数据库

# 计算考试成绩

def calculateGrade(request):

if request.method == 'POST':

sid = request.POST.get('sid')

subject1 = request.POST.get('subject')

student = models.Student.objects.get(sid=sid)

paper = models.TestPaper.objects.filter(major=student.major) grade = models.Record.objects.filter(sid=student.sid)

course = models.Course.objects.filter(course\_name=subject1).first() now = datetime.now()

# 计算考试成绩

questions = models.TestPaper.objects.filter(course course\_name=subject1).\ values('pid').values('pid id','pid answer','pid score')

stu\_grade = 0 # 初始化一个成绩

for p in questions:

qid = str(p['pid id'])

stu\_ans = request.POST.get(qid) cor\_ans = p['pid answer']

if stu\_ans == cor\_ans:

stu\_grade += p['pid score']

models.Record.objects.create(sid\_id=sid, course\_id=course.id, grade=stu\_grade,rtime=now) context = {

'student': student, 'paper': paper, 'grade': grade

}

return render(request, 'index.html', context=context)

### 配置访问路由 URL

有了视图后，我们需要将视图函数和 Web 网页链接对应起来。

url 可以理解为访问网站时输入的网址链接，配置好url 后Django 才知道怎样定位app。打开 DjangoExam/urls.py,输入如下代码：

from django.contrib import admin from django.urls import path, include from django.conf.urls import url

from exam import views

urlpatterns = [

path('admin/', admin.site.urls),

url(r'^$',views.index),#默认访问首页 url('index/',views.index,name='index'),

url('studentLogin/',views.studentLogin,name='studentLogin'),#学生登录 url('startExam/',views.startExam,name='startExam'),#开始考试

url('calculateGrade/',views.calculateGrade,name='calculateGrade'),#考试评分 path('stulogout/',views.stulogout,name='stulogout'), # 学生退出登录 path('userfile/',views.userfile,name='userfile'), # 个人信息

path('examinfo/',views.examinfo,name='examinfo'), # 考试信息

]

通过配置如上 URL，Django 将会根据用户请求的 URL 来选择使用哪个视图。

### 静态资源准备及配置

本系统前后端不分离,前端框架选用当前比较受欢迎的 Bootstrap4，为了快速开发，本系统所有的页面都使用原生 Bootstrap 进行开发，未采用第三方的模板和主题。

**下载 Bootstrap 中文文档**

<https://v4.bootcss.com/docs/getting-started/download/>

因为 bootstrap.js 依赖 **jquery.js** 和 **popper.js** 才能正常运行，因此我们也需要一并下载保存。

由于

popper.js

版本兼容问题，本系统采用 cdn 远程引入的形式。

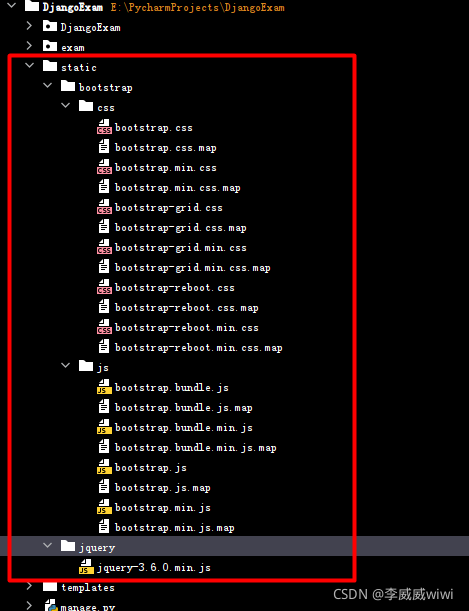
附上官网下载链接（进入下载页面，复制粘贴代码到新文件即可）：

<https://jquery.com/download/>

在项目根目录下新建一个文件夹 static 用于存放前端模板静态资源。

把刚才 Bootstrap 解压出来的 css 和 js 两个文件夹复制到 static 文件夹去。把刚刚解压出来的 jQuery 文件夹复制到 static 文件夹去。

完成后 Static 文件夹结构如下图：



准备好静态资源后，我们还需要在 Django 中指定静态文件的存放位置，这样才能够在模板中正确引用它们。

在 DjangoExam/settings.py 中进行如下配置：

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, 'static'), # 添加此项

]

### 模板创建

在创建模板之前，我们先在根目录下新建一个文件夹 templates 用于存放我们的所有的模板文件。

模板位置也同样需要进行配置指定模板的存放位置，在 DjangoExam/settings.py 中进行如下配置：

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS': [os.path.join(BASE\_DIR, 'templates')], # 添加此项

'APP\_DIRS': True, 'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug', 'django.template.context\_processors.request', 'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

接着我们在模板文件中新建三个文件：

* base.html：是整个项目的模板基础，所有的网页都从它继承；
* header.html：是网页顶部的导航栏；
* footer.html：是网页底部的注脚。

分别编写三个静态 HTML 文件代码如下：

**templates/base.html：**

<!-- 载入静态文件-->

{% load static %}

<!-- 网站主语言 -->

<html lang="zh-cn">

<head>

<!-- 网站采用的字符编码 -->

<meta charset="utf-8">

<!-- 预留网站标题的位置 -->

<title>{% block title %}{% endblock %}</title>

<!-- 引入 bootstrap 的css 文件 -->

<link rel="stylesheet" href="{% static 'bootstrap/css/bootstrap.min.css' %}">

<link rel="stylesheet" href="[https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-](https://cdn.jsdelivr.net/npm/bootstrap-icons%401.5.0/font/bootstrap-) icons.css">

</head>

<body>

<!-- 引入导航栏 -->

{% include 'header.html' %}

<!-- 预留具体页面的位置 -->

{% block content %}{% endblock content %}

<!-- 引入注脚 -->

{% include 'footer.html' %}

<!-- bootstrap.js 依赖 jquery.js 和 popper.js，因此在这里引入 -->

<script src="https://cdn.staticfile.org/jquery/3.2.1/jquery.min.js"></script>

<script src="{% static 'jquery/jquery-3.6.0.js' %}"></script>

<!--

-->

popper.js 采用 cdn 远程引入，意思是你不需要把它下载到本地。在实际的开发中推荐静态文件尽量都使用 cdn 的形式。

教程采用本地引入是为了让读者了解静态文件本地部署的流程。

<script src[="h](https://cdn.jsdelivr.net/npm/popper.js%401.16.1-lts/dist/umd/popper.min.js)tt[ps://cdn.jsdelivr.net/npm/popper.js@1.16.1-lts/dist/umd/popper.min.js](https://cdn.jsdelivr.net/npm/popper.js%401.16.1-lts/dist/umd/popper.min.js)"></script>

<!-- 引入 bootstrap 的 js 文件 -->

<script src="{% static 'bootstrap/js/bootstrap.min.js' %}"></script>

</body>

</html>

**templates/header.html：**

<!-- 定义导航栏 -->

<nav class="navbar navbar-expand-lg navbar-dark bg-primary">

<div class="container">

<!-- 导航栏商标 -->

<a class="navbar-brand" href="#">在线考试</a>

<!-- 导航入口 -->

<div>

<ul class="navbar-nav">

<li class="nav-item">

<a class="nav-link" href="/index/">首页</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/examinfo/">考试记录</a>

</li>

<!-- Django 的 if 模板语句 -->

{% if request.session.username %}

<!-- 如果用户已经登录，则显示用户名下拉框 -->

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">

{{ request.session.username }}

</a>

</li>

<div class="dropdown-menu" aria-labelledby="navbarDropdown">

<a class="dropdown-item" href="/userfile/">个人信息</a>

<a class="dropdown-item" href="/stulogout/">退出登录</a>

</div>

<!-- 如果用户未登录，则显示 “登录” -->

{% else %}

<li class="nav-item">

<a class="nav-link" href="/studentLogin/">登录</a>

</li>

<!-- if 语句在这里结束 -->

{% endif %}

<li class="nav-item">

<a class="nav-link" href="/admin">管理员</a>

</li>

</ul>

</div>

</div>

</nav>

**templates/footer.html：**

{% load static %}

<!-- Footer -->

<div>

<br><br><br>

</div>

<footer class="py-3 bg-dark fixed-bottom">

<div class="container">

<p class="m-0 text-center text-white">Copyright &copy; DjangoExam 2021</p>

</div>

</footer>

上述三个文件是网站页面的通用组件模块，基本上每个页面都不会变，所以我们把他们独立出来。

我们编写 Django 后续的页面模板时可以直接继承对应的通用模板组件。接下来我们在三个通用组件的基础上分别编写各个视图对应的模板页面：限于篇幅，放一些关键代码供参考：

index.html



<!-- extends 表明此页面继承自 base.html 文件 -->

{% extends "base.html" %}

{% load static %}

<!-- 写入 base.html 中定义的 title -->

{% block title %}

在线考试系统

{% endblock title %}

<!-- 写入 base.html 中定义的 content -->

{% block content %}

<div class="container">

<div class="container">

<br>

<h3>考试信息</h3>

<div class="container">

<div class="row mt-4">

{% for paper1 in paper %}

<!-- 文章内容 -->

<div class="col-6 mb-6">

<!-- 卡片容器 -->

<div class="card">

<!-- 标题 -->

<h4 class="card-header">{{ paper1.title }}</h4>

<!-- 摘要 -->

<div class="card-body">

<h4 class="card-title">{{ paper1.course }}</h4>

<p class="card-text">{{ paper1.examtime }}</p>

<a

href="/startExam/?sid={{ student.sid }}&title={{ paper1.title }}&subject={{ paper1.course }}" class="card- link">开始考试</a>

</div>

</div>

</div>

{% endfor %}

</div>

</div>

<p></p>

</div>

</div>

{% endblock content %}

login.html



{% extends "base.html" %} {% load static %}

{% block title %} 登录 {% endblock title %}

{% block content %}

<div class="container">

<div class="row justify-content-md-center">

<div class="col-4">

<br>

<form method="post" action="/studentLogin/">

<!-- {% csrf\_token %}-->

<!-- 账号 -->

<div class="form-group">

<label >学生学号</label>

<input type="text" class="form-control" name="sid" placeholder="输入学号">

</div>

<!-- 密码 -->

<div class="form-group">

<label for="password">密码</label>

<input type="password" class="form-control" id="password" name="password" placeholder="

输入密码">

</div>

<!-- 提交按钮 -->

<button type="submit" class="btn btn-primary">登录</button>

<div class="form-group">

<br>

<br>

</div>

</form>

</div>

</div>

</div>

{% endblock content %}

exam.html



<div class="container">

{% for paper1 in paper %}

{% for test in paper1.pid.all %}

<div class="row bg-light">

<div class="col-12">

<!-- <div class="card">-->

<!-- <div class="card-body h-10">-->

<div id="{{ forloop.counter }}">

<b>{{ forloop.counter}}.</b><span>({{ test.score }}分)</span>

<b>{{ test.title }}</b>

<ul>

name="{{ test.id }}"

<li class="option">

<input type="radio" class="radioOrCheck"

value="A"/>

<label>A.

<p class="ue" style="display: inline;">{{ test.a }}</p>

</label>

name="{{ test.id }}"

</li>

<li class="option">

<input type="radio" class="radioOrCheck"

value="B"/>

<label>

1. <p class="ue" style="display: inline;">{{ test.b }}</p>

</label>

name="{{ test.id }}"

</li>

<li class="option">

<input type="radio" class="radioOrCheck"

value="C"/>

<label>

1. <p class="ue" style="display: inline;">{{ test.c}}</p>

</label>

name="{{ test.id }}"

inline;">{{ test.d }}</p>

</li>

<li class="option">

<input type="radio" class="radioOrCheck"

value="D"/>

<label>

1. <p class="ue" style="display:

</label>

</li>

</ul>

<!-- </div>-->

<!-- </div>-->

</div>

</div>

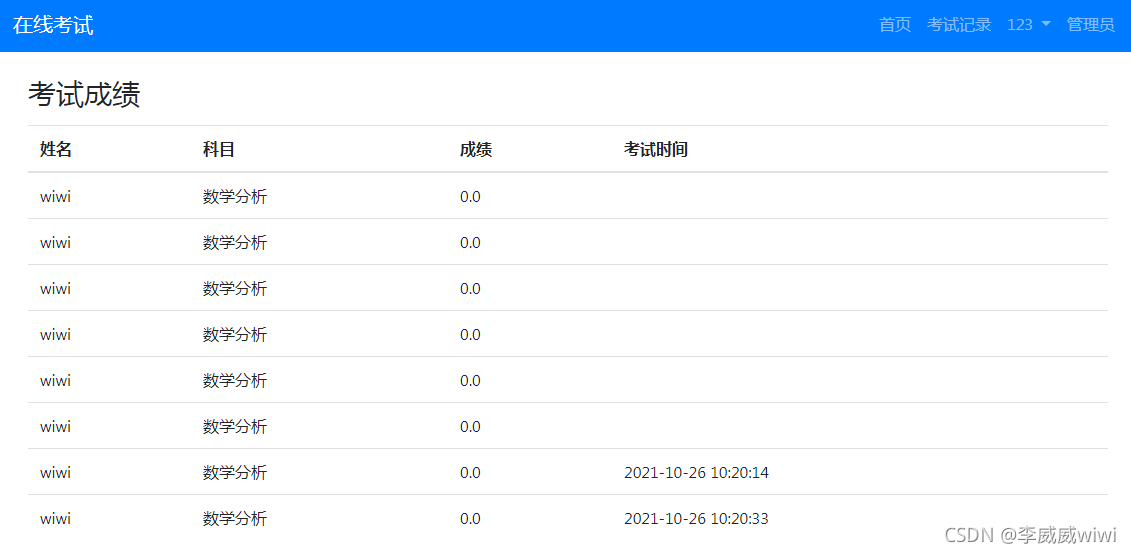
</div>

{% endfor %}

{% endfor %}

</div>

examinfo.html



<div class="container">

<div class="container">

<br>

<h3>考试成绩</h3>

<p></p>

<table class="table">

<thead>

<tr>

<th>姓名</th>

<th>科目</th>

<th>成绩</th>

<th>考试时间</th>

</tr>

</thead>

<tbody>

{% for grade1 in grade %}

<tr class="table">

<td>{{ student.name }}</td>

<td>{{ grade1.course }}</td>

<td>{{ grade1.grade }}</td>

<td>{{ grade1.rtime|date:"Y-m-d H:i:s"}}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</div>

userfile.html

<div class="container">

<br>

<div class="row justify-content-md-center">

<div class="col-8">

<div class="panel panel-info">

<div class="panel-heading">

<h3 class="panel-title">个人信息</h3>

</div>

<div class="panel-body">

<table class="table table-borderless">

<tbody>

<tr>

<td>学号</td>

<td>{{ student.sid }}</td>

</tr>

<tr class="table">

<td>姓名</td>

<td>{{ student.name }}</td>

</tr>

<tr class="table">

<td>性别</td>

{% if student.sex%}

<td>男</td>

{% else %}

<td>女</td>

{% endif %}

</tr>

<tr class="table">

<td>学院</td>

<td>{{ student.academy }}</td>

</tr>

<tr class="table">

<td>专业</td>

<td>{{ student.major }}</td>

</tr>

<tr class="table">

<td>邮箱地址</td>

<td>{{ student.email }}</td>

</tr>

<tr class="table">

<td>出生日期</td>

<td>{{ student.birth }}</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

</div>

</div>

### Django 后台启用配置

Django 自带后台功能，我们只需要两步就可以使用后台功能直接更新我们的后台数据表信息。

1、创建超级用户

在命令行输入：python manage.py createsuperuser

根据提示信息输入用户名，邮箱和密码后系统就创建了一个超级用户。

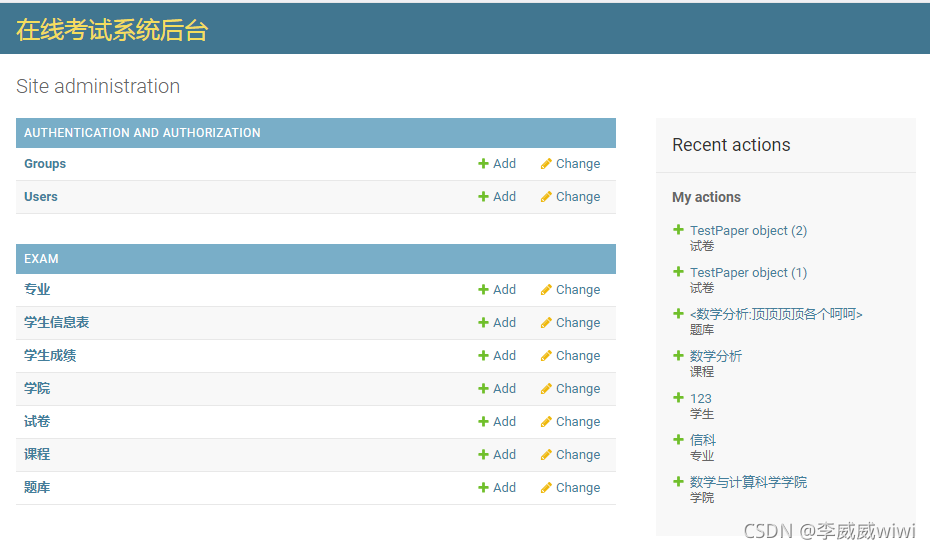
2、注册模型到 admin 中

打开 exam/admin.py 输入如下代码将所有的模型都注册到 admin 中，我们就可以在 Django 自带的后台中更新我们的数据表了。

from django.contrib import admin # Register your models here.

# 修改名称

admin.site.site\_header='在线考试系统后台' admin.site.site\_title='在线考试系统'

from exam.models import Academy,Major, Course,Student,QuestionBank,TestPaper,Record admin.site.register([Academy,Major,Course,Student,QuestionBank,TestPaper,Record])

### 运行服务器测试效果

在 Windows 命令行输入：

python manage.py runserver

运行服务器后在浏览器中输入我们之前配置的首页对应的 URL 地址： http://127.0.0.1:8000/index



## 结语：

本系统仅仅实现了考试管理系统的部分功能，还有很多的功能可以添加到此系统中。

比如：

可以引入教师管理模块，针对教师和学生端实现不同的功能需求。针对成绩部分，可以引入图表，进行可视化的数据分析等等。