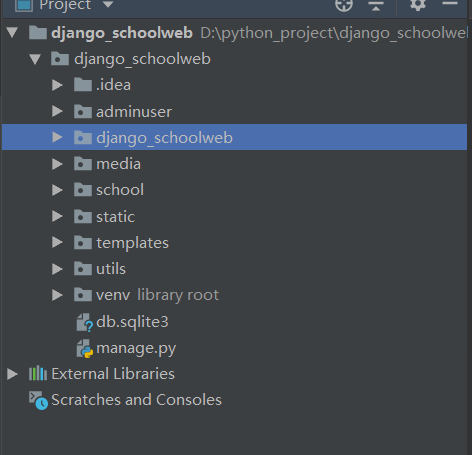
# 项目结构



--adminuser: 后台app，里面主要是关于管理员后台的逻辑（主要看view，url，models）

-- view.py 定义了项目的关于管理员操作的视图函数

-- models.py 定义了关于管理员和网站基本组成例如：学院等的模型类

-- urls.py 定义了关于管理员后台的路由

--django\_schoolweb:项目最基本最初始的app（只要看里面setting和url）

--setting.py （项目的配置文件，里面定义了数据库连接信息和静态目录，app注册）

--urls.py 项目总的路由

--media：用户上传和要下载的东西都存在这里面

--static 静态文件目录例如css，js都可以放里面

--school：普通用户使用的app，里面定义普通用户页面操作的逻辑

-- view.py 定义了项目的关于普通用户操作的视图函数

-- models.py 定义了关于普通用户相关联的一些模型类

-- urls.py 定义了关于普通用户前台的路由

--templates: 模板文件夹，里面放html文件

--admin:管理员后台页面

--school：前台页面

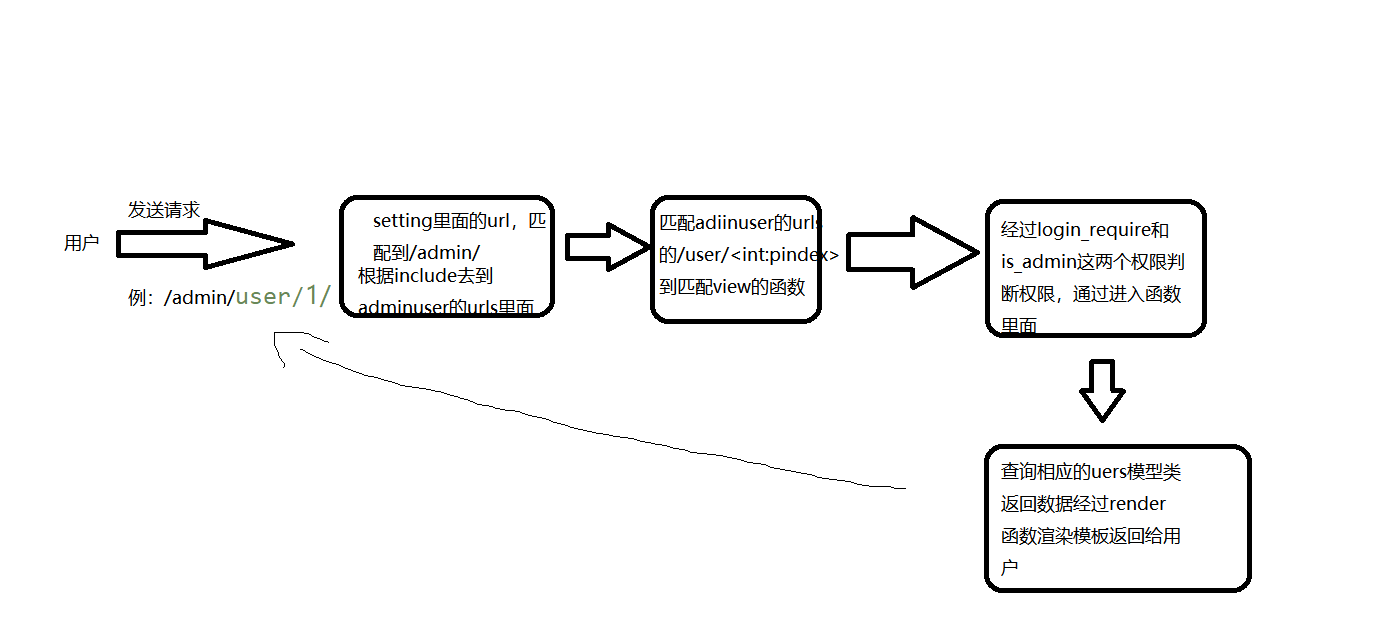
--utils:自己定义的东西放里面（我在里面放了读取excle的方法）

--venv 虚拟环境

。。。其他不用管

# 项目运行流程

以访问用户列表为例



# 数据库

# Django里面封装了orm（对象关系映射），就是可以不用去操作数据库写sql语言就可以实行对数据库的增删改查。把数据库中的每个表都映射成了models.py里面定义的实体类。

通过实体类里面的一些方法就可以对数据库进行增删改查，而不用写sql语句。

例如：对于ResearchPic(adminuser/models.py)这个实体类,实现对科研的增删改查

增 create()

ResearchPic.objects.create(research=research, url=pic, name=pic.name)

删：delete()

research = Research.objects.get(id=id)

research.delete()

改：通过查到这个实体类，对其进行修改然后save

research = Research.objects.get(id=id)

research.title = title

research.content = content

research.introduction = introduction

research.save()

查：

Research.objects.filter().all().order\_by("-publishtime") 返回多条数据

research = Research.objects.get(id=id) 返回一条数据

关于django orm操作可以百度一下。

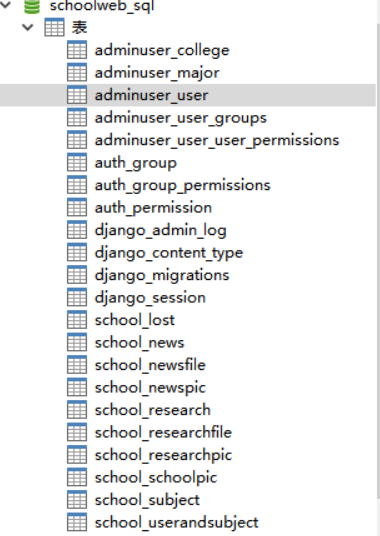
下面对数据库的一些表进行解释，这些表你在项目中的model都可以找到他们对应的model，但有些表他是系统自带的，这些表不用管他，只需要关注两个app对应的model类里面对应的表就行，而表的字段在model类里面的属性页一一对应。表名的命名方法：app名\_model类名

注意：可以直接通过数据库改数据，但不要直接去改表的结构，这样很可能会导致model类和数据库表不匹配，发生错误。正确的操作是修改model类里面的属性，然后执行下面两条命令让他自动迁移到表里面 python manage.py makemigrations

python manage.py migrate

另外最好也不要随便修改model，因为每个表相关联，修改错误可能会导致迁移失败

数据库表的解释（只对定义的进行解释，自带的不用管）



adminuser\_college: 对应了adminuser这个app里面的college这个类，存的是学院的数据

adminuser\_major:存的是专业的数据，和学院的关系是一对多

adminuser\_user:存的是用户数据

School\_lost:失物招领的数据

School\_news:发布消息的数据

School\_newsfile:发布消息的文件数据的链接 和上面消息是一对多的关系

School\_newspic:发布消息的图片数据的链接 和上面消息是一对多的关系

School\_research:存科研成果的数据

School\_researchfile:科研成果的文件数据的链接 和上面科研成果是一对多的关系

School\_researchpic:科研成果的图片数据的链接 和上面科研成果是一对多的关系

School\_schoolpic:学校轮播风景数据

School\_subject:课程表 其中和老师用户是一对多关系，和学生用户是多对多关系，和学院是一对多的关系

School\_userandsubject:学生和课程的多对多表用来记录一些关于两者的数据，例如学生这门课的成绩

# JS

Js这里写的请求就是ajax，大概都一样，关于ajax是什么有什么好处百度一下

ajax形式：

1. $.ajax({
2. async:true,
3. url:'/order/order\_handle/', //请求网址
4. type:'POST', //请求方式 GET或POST
5. traditional:true,
6. data:{ // 发送的数据 post才要填
7. "cart\_ids":cart\_ids,
8. "total":total,
9. }
10. }).done(function (data) {
11. //请求完运行的方法，data是后台返回的值
12. })

**College.html**

使用ajax发送GET请求到admin/college/delete/<int:id>/ 这个路由

$(function () {

$(".delete").click(function (event) {

let id = $(this).attr("name")

$.ajax({

async: true,

url: '/admin/college/delete/' + id + "/", //请求地址

type: 'GET', //请求类型

traditional: true,

}).done(function (data) {

console.log(data)

if (data.ret == "success") {

window.location.reload()

console.log("success")

} else {

console.log("error")

}

})

});

})

**Collegedetail.html**

这里一共有两个ajax，一个是get（删除专业）一个是post（修改专业名），get和上面一样，但post请求要加上geCookie()和csrfSafeMethod（）和ajaxSetup（）这三个函数，否则会报403csrf错误,请求会被拦截。具体这三个函数是怎么使得csrf通过我也是百度直接拿下来的，没看里面怎么实现，应该就是在客户端cookie里存了csrf的token然后请求的时候再从token拿出来。关于什么是csrf百度一下。备注（在有的网页上我可能发送的ajax只有get请求，但我还是写了这三个方法，这个时候这三个方法可以去掉的。我只是摆在那里以备不时之需）

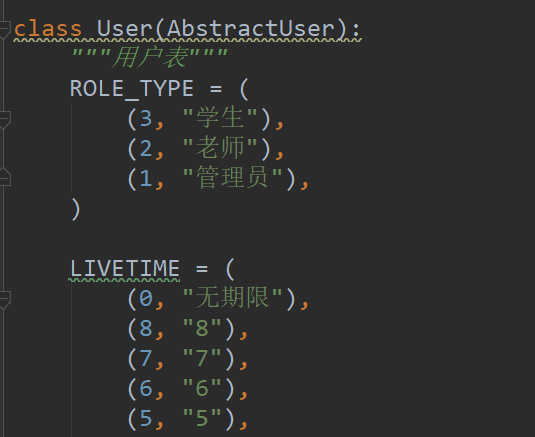
//使用ajax的post提交要加上geCookie()和csrfSafeMethod（）和ajaxSetup（）这三个函数，不然会报403,csrf  
function getCookie(name) {  
 var cookieValue = null;  
 if (document.cookie && document.cookie != '') {  
 var cookies = document.cookie.split(';');  
 for (var i = 0; i < cookies.length; i++) {  
 var cookie = jQuery.trim(cookies[i]);  
 // Does this cookie string begin with the name we want?  
 if (cookie.substring(0, name.length + 1) == (name + '=')) {  
 cookieValue = decodeURIComponent(cookie.substring(name.length + 1));  
 break;  
 }  
 }  
 }  
 return cookieValue;  
}  
function csrfSafeMethod(method) {  
 // these HTTP methods do not require CSRF protection  
 return (/^(GET|HEAD|OPTIONS|TRACE)$/.test(method));  
}  
$.ajaxSetup({  
 beforeSend: function (xhr, settings) {  
 var csrftoken = getCookie('csrftoken');  
 if (!csrfSafeMethod(settings.type) && !this.crossDomain) {  
 xhr.setRequestHeader("X-CSRFToken", csrftoken);  
 }  
 }  
});  
$(function () {  
 $(".delete").click(function (event) { //点击事件  
 let id = $(this).attr("name")  
 $.ajax({ //发送ajax  
 async: true,  
 url: '/admin/major/delete/' + id + "/",  
 type: 'GET',  
 traditional: true,  
 }).done(function (data) {  
 console.log(data)  
 if (data.ret == "success") { //返回成功  
 window.location.reload() // 刷新   
 console.log("success")  
 } else {  
 console.log("error")  
 }  
 })  
 });  
 $(".update").click(function (event) {  
 let bid = $(this).attr("name")  
 let name = $("#" + bid).val()  
 $.ajax({  
 async: true,  
 url: '/admin/major/update/' + bid + "/",  
 type: 'POST',  
 traditional: true,  
 data: {  
 "name": name  
 }  
 }).done(function (data) {  
 console.log(data)  
 if (data.ret == "success") {  
 window.location.reload()  
 console.log("success")  
 } else {  
 console.log("error")  
 }  
 })  
 });  
})

**User.html**

<script>  
 function submitform() {  
 document.form.submit();  
 }  
 //使用ajax的post提交要加上geCookie()和csrfSafeMethod（）和ajaxSetup（）这三个函数，不然会报403,csrf  
 function getCookie(name) {  
 var cookieValue = null;  
 if (document.cookie && document.cookie != '') {  
 var cookies = document.cookie.split(';');  
 for (var i = 0; i < cookies.length; i++) {  
 var cookie = jQuery.trim(cookies[i]);  
 // Does this cookie string begin with the name we want?  
 if (cookie.substring(0, name.length + 1) == (name + '=')) {  
 cookieValue = decodeURIComponent(cookie.substring(name.length + 1));  
 break;  
 }  
 }  
 }  
 return cookieValue;  
 }  
 function csrfSafeMethod(method) {  
 // these HTTP methods do not require CSRF protection  
 return (/^(GET|HEAD|OPTIONS|TRACE)$/.test(method));  
 }  
 $.ajaxSetup({  
 beforeSend: function (xhr, settings) {  
 var csrftoken = getCookie('csrftoken');  
 if (!csrfSafeMethod(settings.type) && !this.crossDomain) {  
 xhr.setRequestHeader("X-CSRFToken", csrftoken);  
 }  
 }  
 });  
 $(function () {  
 $("#delete").click(function (event) { // 删除用户  
 let users = ""  
 $.each($('input:checkbox:checked'), function () {  
 users = $(this).val() + "," + users;  
 });  
 $.ajax({  
 async: true,  
 url: '/admin/user/delete/',  
 type: 'POST',  
 traditional: true,  
 data: {  
 "users": users  
 }  
 }).done(function (data) {  
 console.log(data)  
 if (data.ret == "success") {  
 window.location.reload()  
 console.log("success")  
 } else {  
 console.log("error")  
 }  
 })  
 });  
 $("#find").click(function (event) { //查找用户  
 let sex = $('#sex').val();  
 let takeschoolyear = $('#takeschoolyear').val();  
 let status = $('#status').val();  
 let username = $('#username').val();  
 let role = $('#role').val();  
 let last\_name = $('#last\_name').val();  
 let search = "?sex=" + sex + "&takeschoolyear=" + takeschoolyear + "&status=" + status + "&username=" + username + "&role=" + role + "&last\_name=" + last\_name  
 window.location.href = "http://127.0.0.1:8007/admin/user/1/" + search  
 });  
 $("#updaterole").click(function (event) { //跟新权限  
 let users = ""  
 $.each($('input:checkbox:checked'), function () { //获取选中的用户  
 users = $(this).val() + "," + users;  
 });  
 let role = $('#role').val();  
 $.ajax({  
 async: true,  
 url: '/admin/user/role/',  
 type: 'POST',  
 traditional: true,  
 data: {  
 "users": users,  
 "role": role,  
 }  
 }).done(function (data) {  
 console.log(data)  
 if (data.ret == "success") {  
 window.location.reload()  
 console.log("success")  
 } else {  
 console.log("error")  
 }  
 })  
 });  
 $("#update").click(function (event) { //跟新用户  
 let users = ""  
 $.each($('input:checkbox:checked'), function () {  
 users = $(this).val() + "," + users;  
 });  
 let sex = $('#sex').val();  
 let takeschoolyear = $('#takeschoolyear').val();  
 let status = $('#status').val();  
 let username = $('#username').val();  
 let role = $('#role').val();  
 let last\_name = $('#last\_name').val();  
 $.ajax({  
 async: true,  
 url: '/admin/user/update/',  
 type: 'POST',  
 traditional: true,  
 data: {  
 "sex": sex,  
 "takeschoolyear": takeschoolyear,  
 "status": status,  
 "username": username,  
 "role": role,  
 "last\_name": last\_name,  
 "users": users  
 }  
 }).done(function (data) {  
 console.log(data)  
 if (data.ret == "success") {  
 window.location.reload()  
 console.log("success")  
 } else {  
 console.log("error")  
 }  
 })  
 })  
 $("#create").click(function (event) { //创建用户  
 let sex = $('#sex').val();  
 let takeschoolyear = $('#takeschoolyear').val();  
 let status = $('#status').val();  
 let username = $('#username').val();  
 let role = $('#role').val();  
 let last\_name = $('#last\_name').val();  
 $.ajax({  
 async: true,  
 url: '/admin/user/create/',  
 type: 'POST',  
 traditional: true,  
 data: { //定义要发生的数据  
 "sex": sex,  
 "takeschoolyear": takeschoolyear,  
 "status": status,  
 "username": username,  
 "role": role,  
 "last\_name": last\_name,  
 }  
 }).done(function (data) {  
 console.log(data)  
 if (data.ret == "success") {  
 window.location.reload()  
 console.log("success")  
 } else {  
 console.log("error")  
 }  
 })  
 })  
 });  
</script>

# 其他：

1. 这个项目的用户类是使用django自带的用户类（AbstractUser）好处：这样许多字段就不用自己定义。并且可以使用自带的登录权限校验装饰器login\_required



1. 整个项目有app（除去基本的那个），一个前台一个后台。app里面关注view，models,url

s三个文件就行。Django MVT架构,就是主要由model，view，template组成。url相当于一个桥梁，连接前后端。