第十节课

课程回顾

1. 关于课程页面的显示,页面的挑战,课程的筛选.
2. web的安全问题,有两个网页

本节课程

1. 关于用户在后台的显示
2. 需要在user.admin里面注册相关的字段

```
# 使用类的方式来注册后台
from django.contrib import admin
from users.models import UserInfo, EmailVerify


class UserInfoAdmin(admin.ModelAdmin):
    list_display = ["username", "nick_name", "birthday", "gender", "image", "phone", "qq",
"summary"]
    list_filter = ["username", "nick_name", "birthday", "gender", "image", "phone", "qq",
"summary"]
    search_fields = ["username", "nick_name", "birthday", "gender", "image", "phone",
"qq", "summary"]

    class Meta():
        verbose_name = u"用户"
        verbose_name_plural = verbose_name

    def __str__(self):
        return self.username


# 使用装饰器的方式来注册后台
@admin.register(UserInfo)
class UserInfoAdmin(admin.ModelAdmin):
    list_display =["username", "nick_name", "birthday", "gender", "image", "phone", "qq",
"summary"]
    list_filter =["username", "nick_name", "birthday", "gender", "image", "phone", "qq",
"summary"]
    search_fields = ["username", "nick_name", "birthday", "gender", "image", "phone",
"qq", "summary"]

    class Meta():
        verbose_name = u"用户"
        verbose_name_plural = verbose_name

    def __str__(self):
        return self.username
```

3. 我们可以在后台把一些字段进行一个分类的处理

```
# 我们可以在 def自定义之后在相应的字段进行显示
```

```
fieldsets = (
        # 可以设置操作这个是分类的显示字段
        (None, {
            'fields': ('username', 'nick_name','email')
        }),
        ('Permissions', {
            'fields': ('groups', 'user_permissions')
        }),
        ('Status',{
            'fields': ( 'is_active', 'is_staff', 'is_superuser')
        }),
        # 显示不是隐藏    classes, 样式是不是
        (u'其他信息', {
            'classes': ('collapse',),
            'fields': ('birthday', 'gender', 'image', 'last_login', 'date_joined'),
        }),
    )
```

4. 不用的用户的登录时候的情况,可以设置几个中,在不用的组登录之后的情况

5. 把后台显示的信息进行修改

```
admin.site.site_header = '潭州课堂后台管理'
admin.site.site_title = '潭州课堂'
```

6. 利用字段is_staff 对用户是不是具有后台权限登录进行设置]

```
# 先配置一个url文件,用于激活is_staff的地方,

# 我们先判断这个用是不会已经存在,然后激活is_active 然后发送邮件, 然后配置发送邮件之后的url,然后进入url之后,显示一个页面.然后一个确认提交按钮. 配置一个确认url的配置 .在view里面修改user.is_staff的值.

# 提升权限
class UpdateUseView(View):
    def get(self, request):
        update_from = UpdateUseForm()
        return render(request, 'update_sub.html', {'update_from': update_from})

    def post(self, request):
        update_from = UpdateUseForm(request.POST)
        if update_from.is_valid():
            email = request.POST.get("email", "")
            # 与数据库做对比
            user = UserInfo.objects.get(email=email)
            if user:
                if user.is_active:
                    send_register_email(email, 'update')
                    return render(request, 'update_success.html', {'email': email})
                else:
                    return render(request, 'update_sub.html', {'msg': u"用户未激活!!",
                                                               'update_from': update_from
                                                               })

            return render(request, 'update_sub.html', {'msg': u"用户不存在!!",
                                                       'update_from': update_from})

        else:
```

```
            return render(request, 'update_sub.html', {'update_from': update_from})
```

7. 配置一个updatause的页面 在页面中做好的相关的配置,,

```
    # 用户升级
    url(r'^update_use/$', UpdateUseView.as_view(), name="update_use"),

    # 确认用户升级
    url(r'^update/(?P<update_code>.*)/$', UpdateView.as_view(), name='update'),

    # 提交升级
    url(r'^update_sure/$', ModifyUpdateView.as_view(), name="update_sure"),
```

8. 配置view.py

```
# 显示升级的页面
class UpdateView(View):

    def get(self, request, update_code):
        all_records = EmailVerify.objects.filter(code=update_code)
        if all_records:
            for record in all_records:
                email = record.email
                return render(request, "update_sure.html", {"email": email})


# 升级确认页面的提交
class ModifyUpdateView(View):
    def post(self, request):
        modify_form = UploadInfoForm(request.POST)
        if modify_form.is_valid():
            email = request.POST.get("email", "")

            # 把UserInfo的数据取到,然后把密码加密后修改密码并保存

            user = UserInfo.objects.get(email=email)
            user.is_staff = True
            user.save()
            # return render(request, 'reset_success.html')
            return HttpResponseRedirect(reverse("login"))
        else:
            email = request.POST.get("email", "")
            return render(request, "update_sure.html", {"email": email, "modify_form":
modify_form})
```

9. 再到course做也搜索用法, 利用keyword关键之来搜索

```
# 一点在前端页面做一个关键字来对于后端进行使用



# 回到courselise的view逻辑
```

```
        # 课程搜索
        search_keywords = request.GET.get('key', "")
        if search_keywords:
            #  准确查找
            all_course = all_course.filter(
                Q(name__icontains=search_keywords) |
Q(describe__icontains=search_keywords))
```