第四节课

1.后台注册

后台的建立, 用户的登录.

在admin.py里面先写好字段

第一种注册后台的方式

```
    list_display = ["name", "price", "learn_time", "nums", "image", "describe"]
    list_filter = ["name", "price", "learn_time", "nums", "image", "describe"]
    search_fields  = ["name", "price", "learn_time", "nums", "image", "describe"]

    class Meta:
        verbose_name = u"课程"
        verbose_name_plural = verbose_name

    def __str__(self):
        return self.name


admin.site.register(Course, CourseAdmin)
```

第二种注册后台的方式

```
#使用装饰器的方式来注册
@admin.register(Course)
class CourseAdmin(admin.ModelAdmin):
    list_display = ["name", "price", "learn_time", "nums", "image", "describe"]
    list_filter = ["name", "price", "learn_time", "nums"]
    search_fields = ["name", "price", "learn_time", "nums"]

    class Meta:
        verbose_name = u"课程"
        verbose_name_plural = verbose_name

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

创建好超级用户:

tools -->run manage.py task

createsuperuser

账号>邮箱>密码

1. 用户的登录:

1. 配置好template的文件
2. 先在urls.py里面设置好页面的跳转,
3. 然后在view写好需要的逻辑,然后在对逻辑进行细化.
4. 登录的逻辑　验证成功返回首页并显示个人中心和课程

5. 验证不成功返回登录页面并有显示登录出错的原因

使用authenticate()方法来认证一个给定的用户名(username)和密码(password)。该函数以关键字参数的形式接受认证的凭证，默认配置的关键字是**username**和**password**。如果密码和用户名匹配，该函数返回一个**User**对象，否则，该函数返回**None**，

```
# 做登录和登出
from django.contrib.auth import authenticate, login, logout
# 引用一些模块做重定向 操作
from django.core.urlresolvers import reverse
from django.shortcuts import redirect


    return redirect(reverse('index'))
```

登录之后 只能使用用户名和密码登录

如果想过邮箱登录的话 需要重写authenticate方法,自定义auth方法 需要修改setting.py里面对于用户验证的自定义

```
#setting.py
AUTHENTICATION_BACKENDS = (
    'users.views.CustomBackend',
)

#user  view.py
from django.contrib.auth.backends import ModelBackend
from django.db.models import Q

from .models import UserProfile
class CustomBackend(ModelBackend):
    def authenticate(self, username=None, password=None, **kwargs):
        try:
            user = UserProfile.objects.get(Q(username=username) | Q(email=username))
                                            #不查询密码是密码存在数据库的时候是秘文.
            if user.check_password(password):  #可以使用check_password的方法
                return user
        except Exception as e:
            return None
```

使用form.py做登录验证

```
#froms.py

class LoginFrom(forms.Form):
    username = forms.CharField(required=True)
    password = forms.CharField(required=True, min_length=6)
```

```
#view.py


class LoginView(View):
```

```
    def get(self, request):
        return render(request, "login.html", {})
    def post(self, request):
        login_form = LoginFrom(request.POST)
        if login_form.is_valid():
            user_name = request.POST.get("username", "")
            pass_word = request.POST.get("password", "")
            user = authenticate(username=user_name, password=pass_word)
            if user is not None:
                login(request, user)
                return HttpResponseRedirect(reverse("index"))
            else:
                 return render(request, "login.html", {"msg": "用户名或密码错误！"##",
login_form":login_form})

            """ return HttpResponseRedirect(reverse("index"))
            else:
                 return render(request, "login.html", {"msg": "用户名或密码错误！)
        else:
            return render(request, "login.html", {login_form":login_form})

                        """
```

在index.html里面对错误是那个的进行提示 错误信息的回填

```
{% if login_form.errors.username %} errorput {% endif %}
{% if login_form.errors.password %} errorput {% endif %}
{% for error in login_form.errors.items %}{{ error }}{% endfor %}{{ msg }}
```

用户的登出 logout

```
# 登出
class LogoutView(View):
    def get(self, request):
        logout(request)
        return HttpResponseRedirect(reverse("index"))
```

额外内容

django-bootstrap4