# Django记录

## 1、安装Django

Win+ r 输入cmd

输入：

pip install django==4.1 -i https://pypi.tuna.tsinghua.edu.cn/simple

## 2、创建Django项目

方法有两种

### 第一种：使用Pycharm专业版直接创建Django项目即可

### 第二种：手动创建

1. 在任意位置创建一个文件夹，命名就是你项目的名字
2. 双击进入该文件夹，点击搜索框输入cmd，开启命令行
3. 输入命令，创建虚拟环境

* python -m venv 项目名称\_venv

1. 输入

* 项目名称\_venv\Scripts\activate
* 激活虚拟环境 deactivate:用来关闭虚拟环境

1. 输入以下命令创建项目

* django-admin startproject 项目名称 .
* 举例:django-admin startproject ll .
* 然后安装Django
* pip install django==4.1 -i https://pypi.tuna.tsinghua.edu.cn/simple
* 7、输入以下创建数据库
* py manage.py migrate
* 8、启动
* py manage.py runserver
* 9、浏览器输入127.0.0.1:8000 访问

## 3、创建App

### 1、在上面的启动之后，回到manage.py所在目录，在搜索框输入cmd再开一个命令行窗口

激活虚拟环境

项目名称\_venv\Scripts\activate

### 2、创建App

py manage.py startapp learning\_logs

### 3、定义数据模型Topic

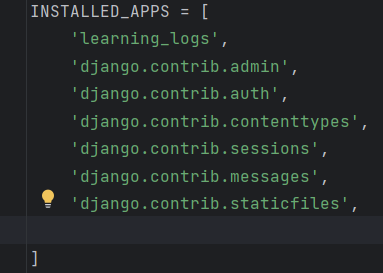
from django.db import models  
  
  
# Create your models here.  
  
class Topic(models.Model):  
 """用户学习的主题"""  
 text = models.CharField(max\_length=256)  
 data\_added = models.DateTimeField(auto\_now=True)  
  
  
 def \_\_str\_\_(self):  
 """返回模型的字符串显示"""  
 return self.text

### 4、更改语言时区、语言

LANGUAGE\_CODE = 'zh-Hans'  
TIME\_ZONE = 'Asia/Shanghai'

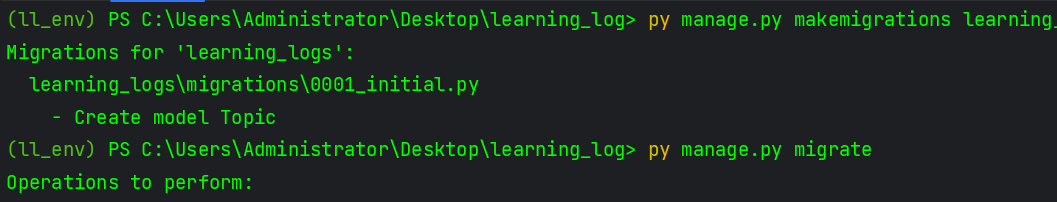
### 5、激活模型

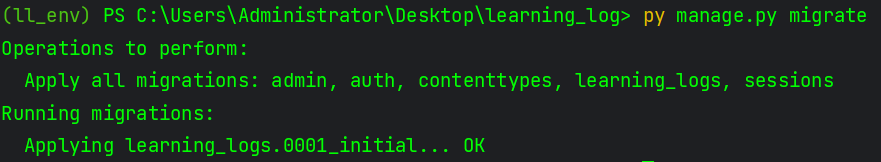
找到settings.py文件



由于创建了新的表（模型），需要迁移数据库中的数据。新建一个终端，输入以下命令

py manage.py makemigrations learning\_logs  
  
py manage.py migrate



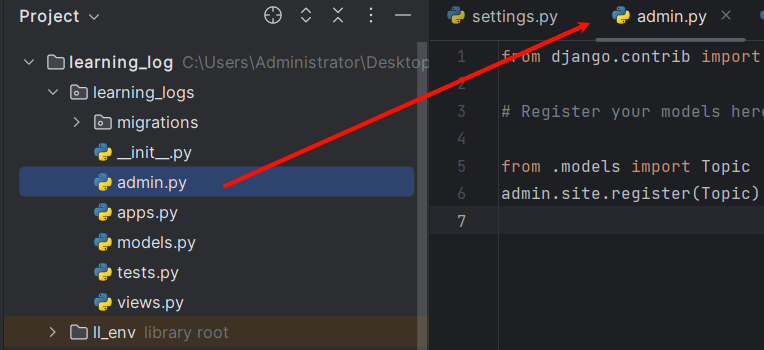


## 4、创建超级管理员（后台管理员）

py manage.py createsuperuser

向管理网站注册 Topic，请输入下面的代码：

from django.contrib import admin  
  
# Register your models here.  
  
from .models import Topic  
admin.site.register(Topic)



定义Entry模型

class Entry(models.Model):  
 """某个主题的具体知识"""  
 topic = models.ForeignKey(Topic, on\_delete=models.CASCADE)  
 text = models.TextField()  
 date\_added = models.DateTimeField(auto\_now=True)  
  
 class Meta:  
 verbose\_name\_plural = 'entries'  
  
 def \_\_str\_\_(self):  
 return f"{self.text[:50]}..."

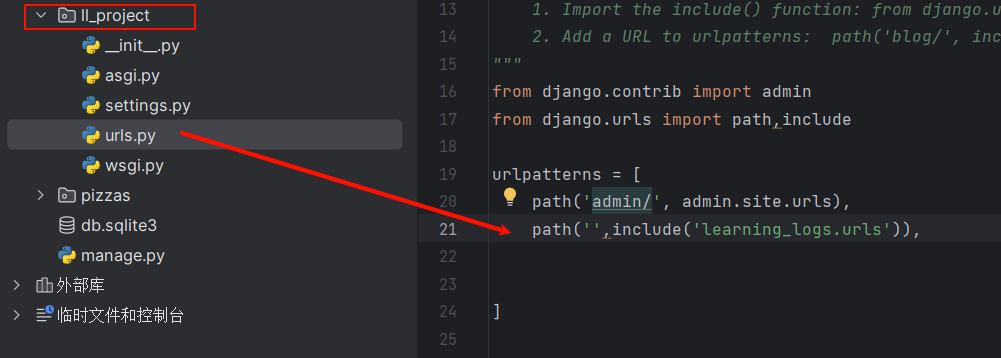
迁移数据库

py manage.py makemigrations learning\_logs  
  
py manage.py migrate

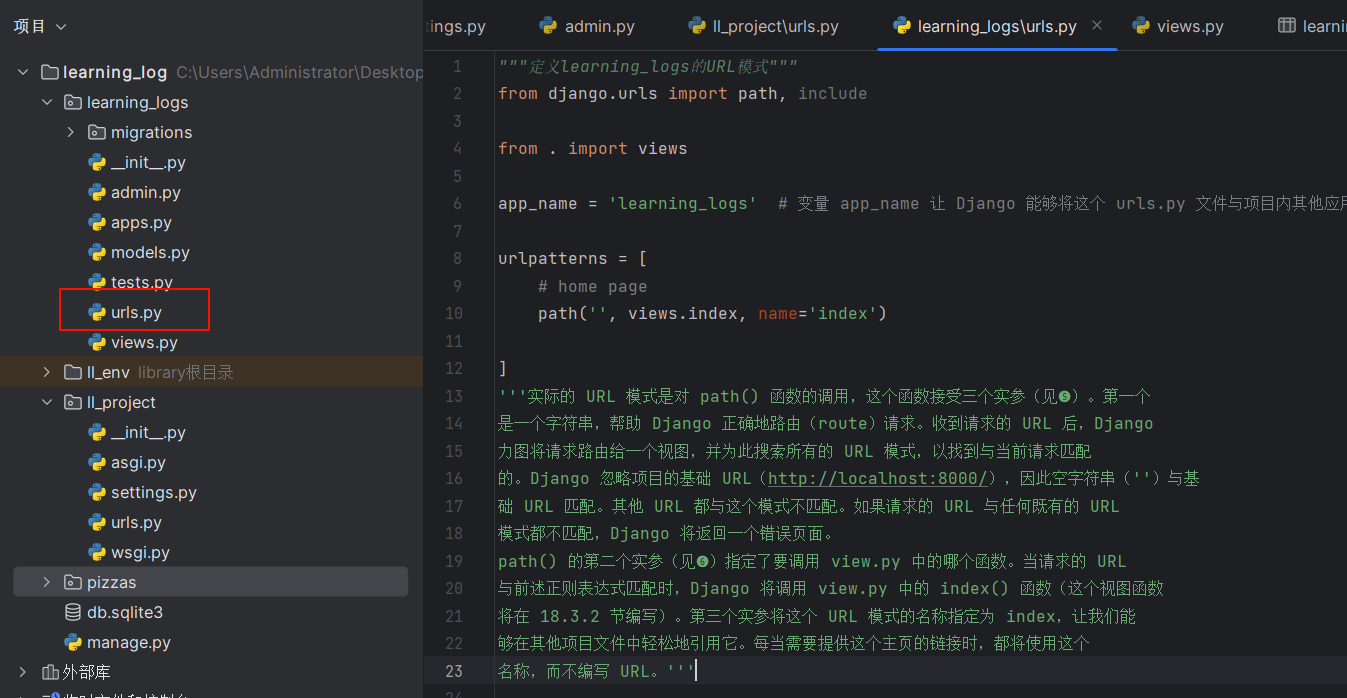
## 5、创建网页

**步骤：定义URL模式、编写视图函数、编写模板（html）**

1. 定义URL模式，在项目的urls文件中导入app的路由

* 

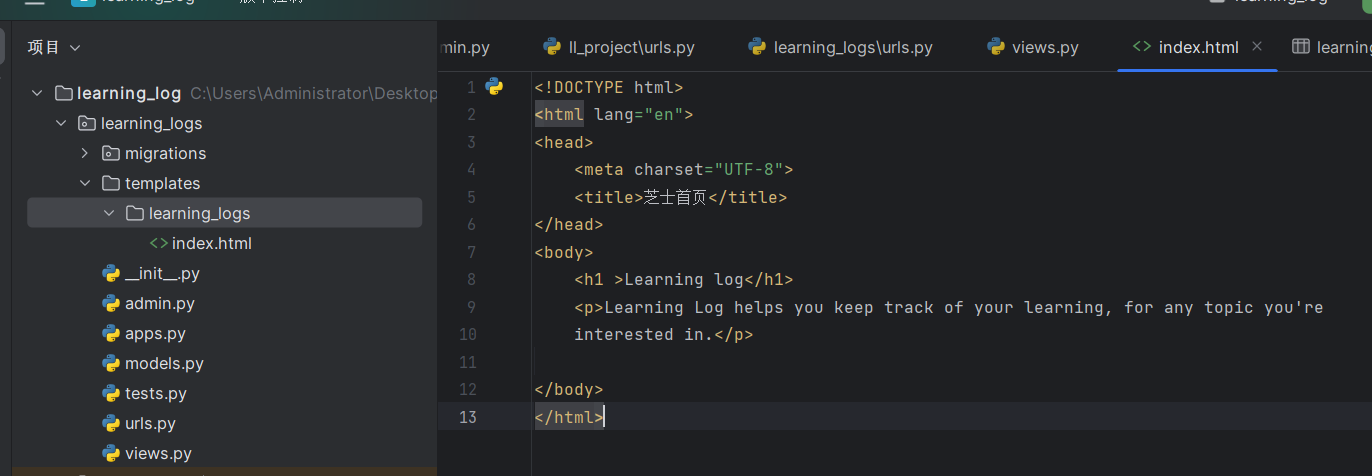
1. 在learning\_log app目录中新建urls文件，配置路由

* 

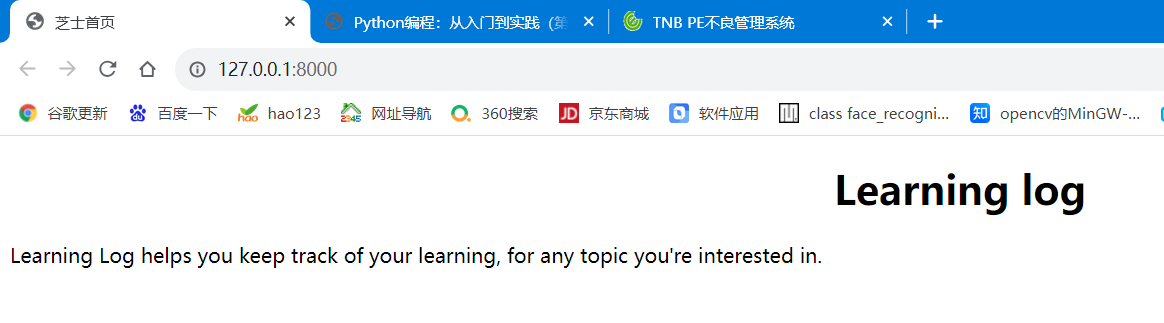
1. 编写index视图函数

* from django.shortcuts import render  
    
  # Create your views here.  
  def index(request):  
   '''学习笔记主页'''  
   return render(request, "learning\_logs/index.html")

1. 新建templates目录，在里面载创建一个learning\_logs目录，用于存放learning\_logs的前端界面

* 

1. 界面展示

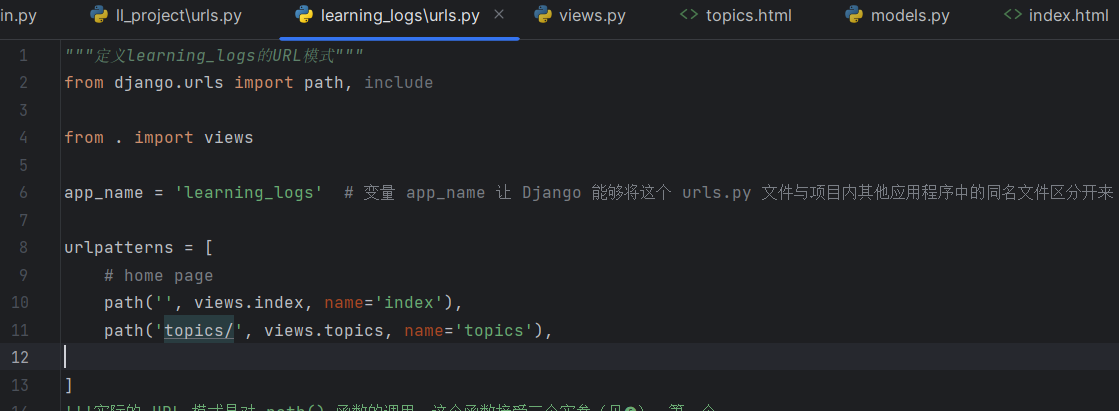
* 

## 6、创建其他网页

在index同一目录下，新建一个base作为模板，里面包含了许多网页共同的排版，节省开发时间，提高效率

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Base</title>  
</head>  
<body>  
 <p>  
 <a href="{% url 'learning\_logs:index' %}">learning Log</a>&nbsp;  
 <a href="{% url 'learning\_logs:topics' %}">topics</a>  
 </p>  
 {% block content %}{% endblock content %}  
 {#我们在最后一行插入了一对块标签这个块名为 content，是一个占位#}  
 {#符，其中包含的信息由子模板指定。#}  
 {#子模板并非必须定义父模板中的每个块，因此在父模板中，可以使用任意多个块#}  
 {#来预留空间，而子模板可根据需要定义相应数量的块。#}  
  
</body>  
</html>

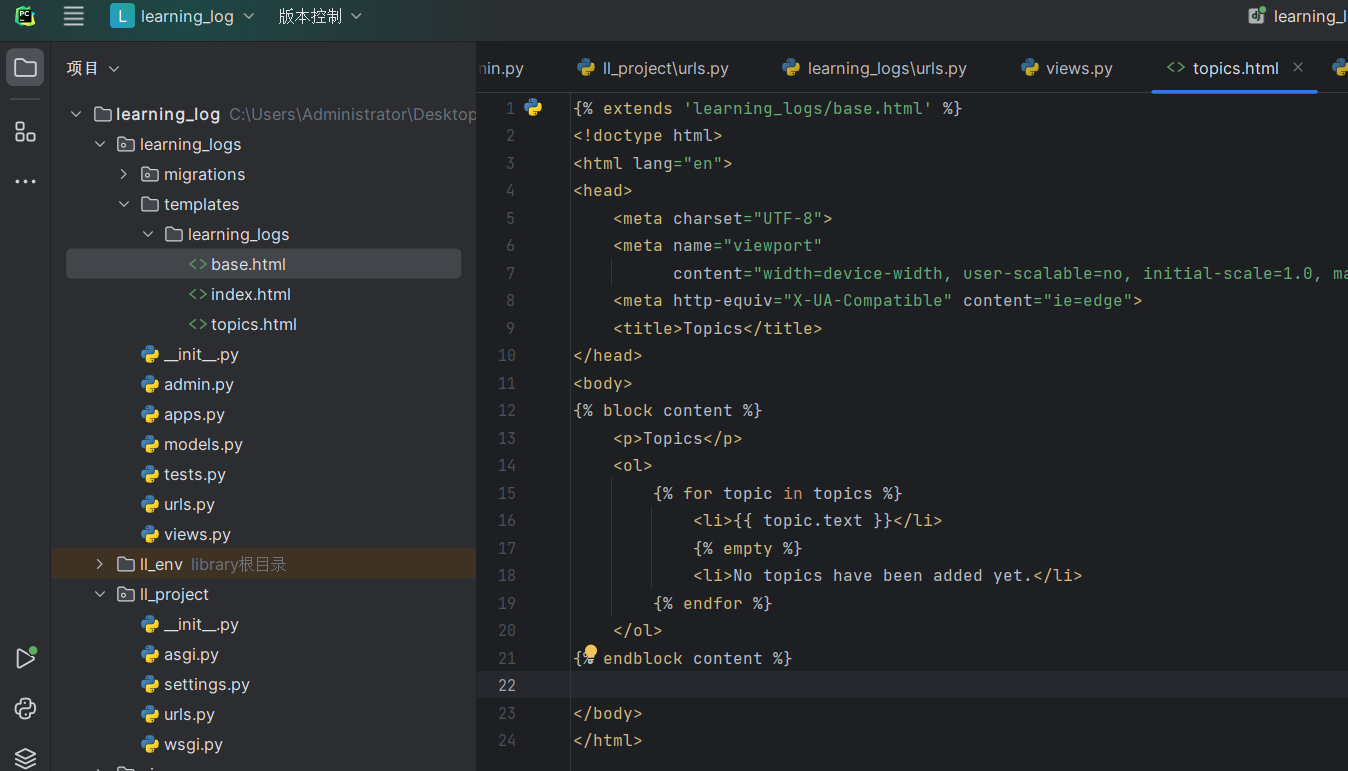
1、新建路由topics



2、编写topics视图函数

def topics(request):  
 '''学习笔记主题'''  
 topics = Topic.objects.order\_by("date\_added")  
 context = {"topics": topics}  
 return render(request, "learning\_logs/topics.html",context)

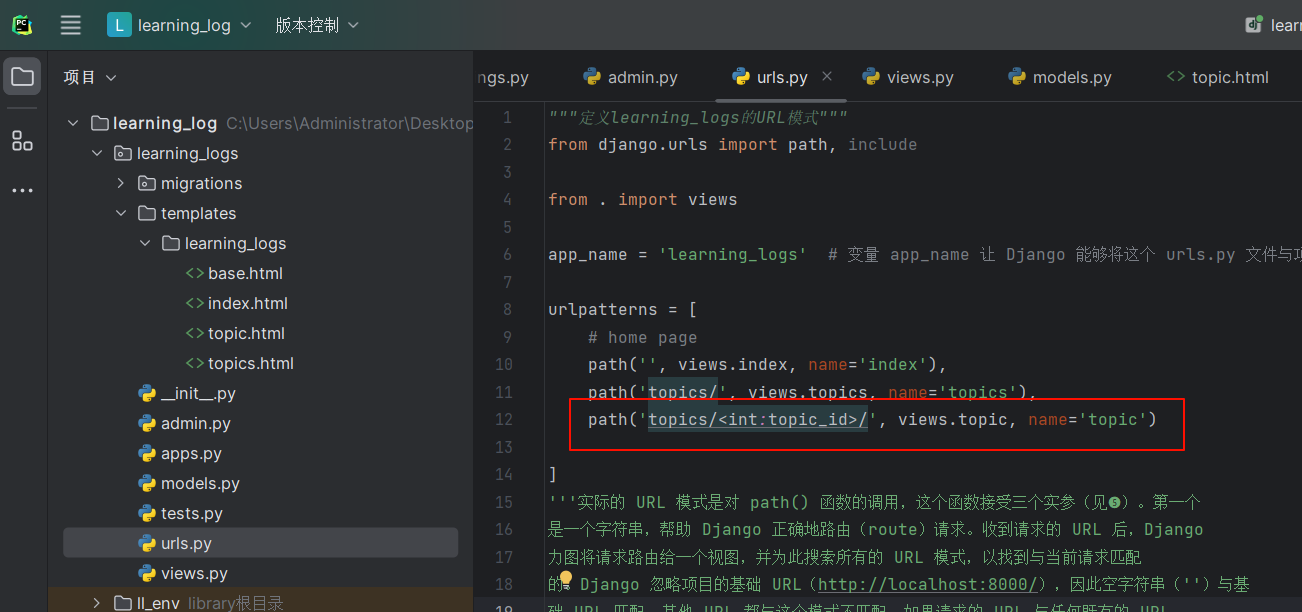
3、新建topics网页



{% extends 'learning\_logs/base.html' %}  
<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Topics</title>  
</head>  
<body>  
{% block content %}  
 <p>Topics</p>  
 <ol>  
 {% for topic in topics %}  
 <li>{{ topic.text }}</li>  
 {% empty %}  
 <li>No topics have been added yet.</li>  
 {% endfor %}  
 </ol>  
{% endblock content %}  
  
</body>  
</html>

显示特定主题的详细页面

1、定义路由



2、定义视图函数 topic

def topic(request, topic\_id):  
 """显示单个主题及其所有条目"""  
 topic = Topic.objects.get(id=topic\_id)  
 entries = topic.entry\_set.order\_by("-date\_added") #- 表示降序  
 context = {"topic": topic, "entries": entries}  
 return render(request, "learning\_logs/topic.html", context)

3、编写模板topic.html

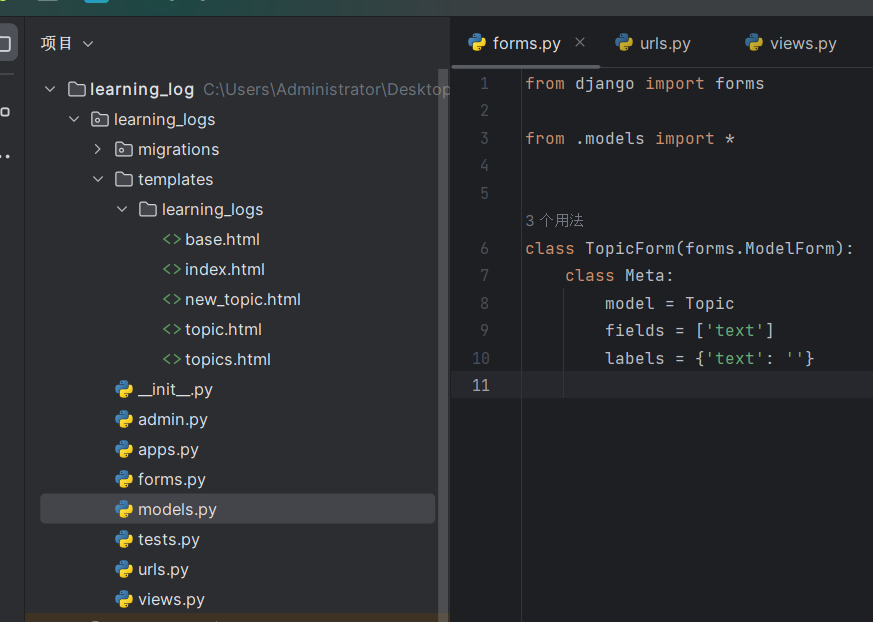
<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Topic</title>  
</head>  
<body>  
{% extends 'learning\_logs/base.html' %}  
{% block content%}  
 <p>Topics: {{ topic.text }}</p>  
 <p>Entries:</p>  
 <ol>  
 {% for entry in entries %}  
 <li>  
 <p>{{entry.date\_added|date:'M d, Y H:i'}}</p>  
 <p>{{entry.text|linebreaks}}</p>  
 </li>  
 {% empty %}  
 <li>There are no entries for this topic yet.</li>  
 {% endfor %}  
 </ol>  
{% endblock content%}  
</body>  
</html>

## 7、用户账户

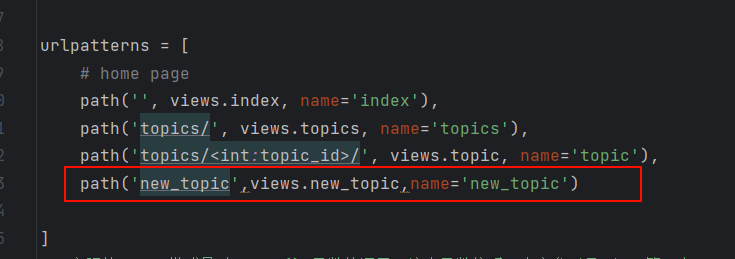
### 1、添加新主题（Topic）

#### 1、编辑forms.py文件

from django import forms  
  
from .models import \*  
  
  
class TopicForm(forms.ModelForm):  
 class Meta:  
 model = Topic  
 fields = ['text']  
 labels = {'text': ''}



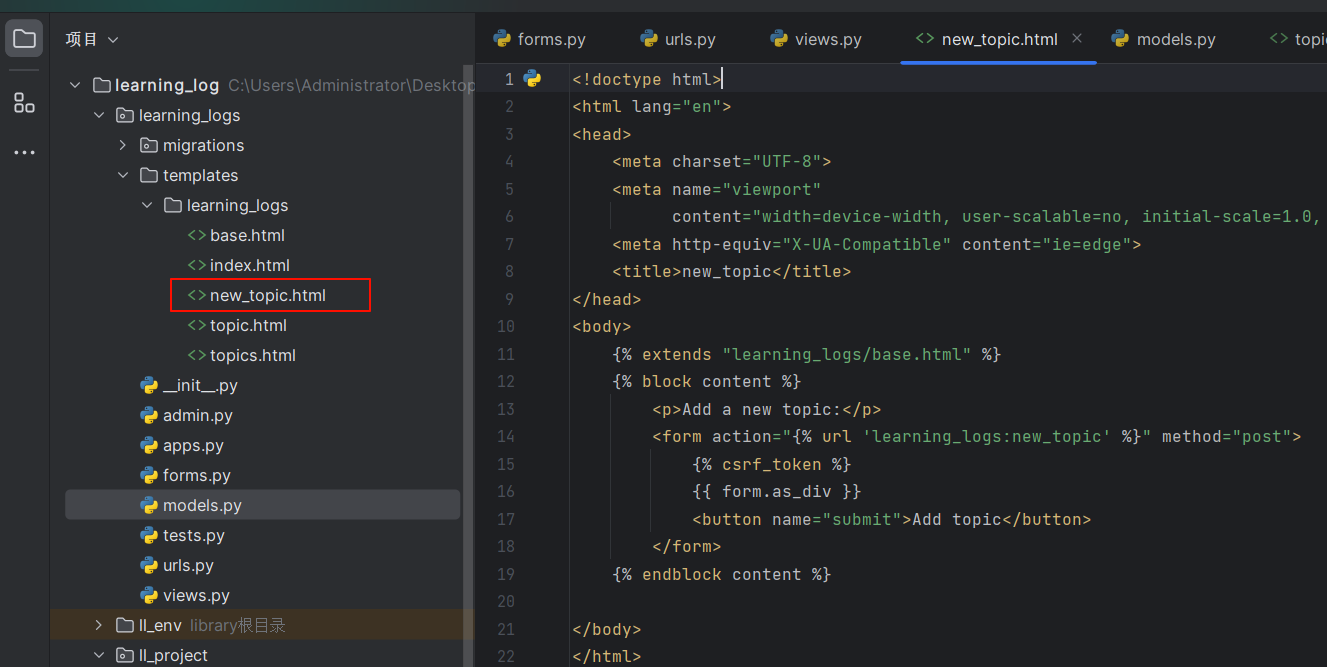
#### 2、定义URL



#### 3、定义view函数

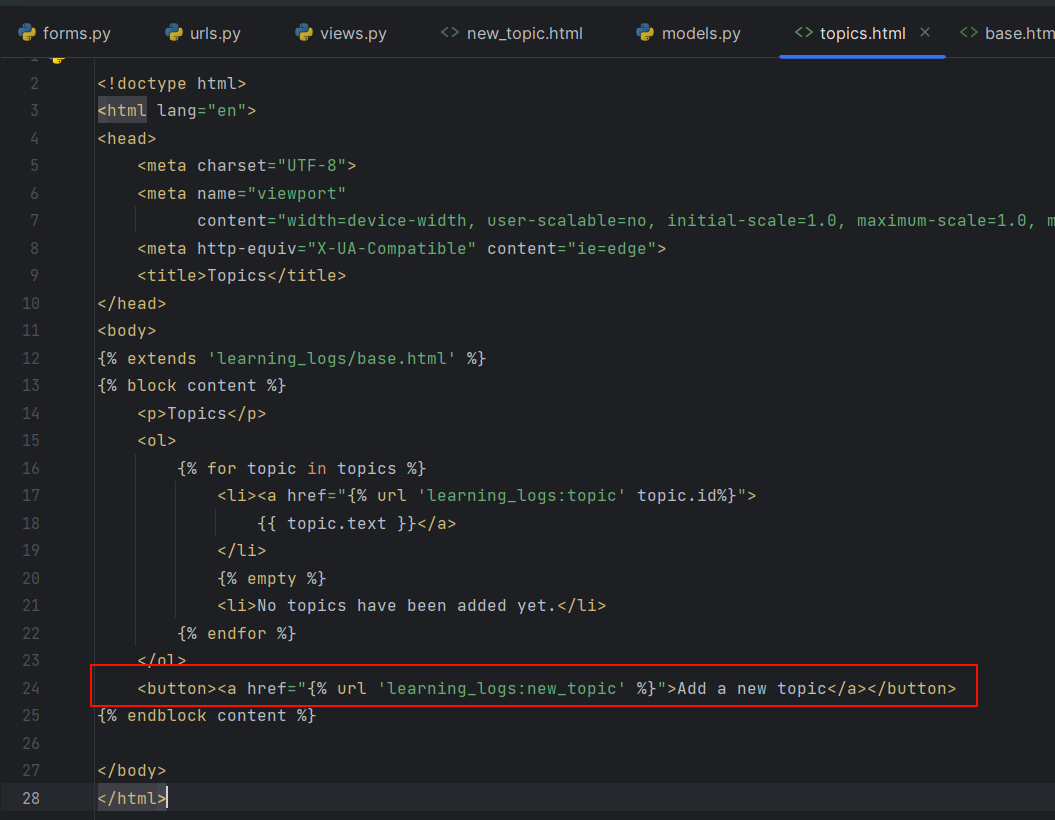
def new\_topic(request):  
 """添加新主题"""  
 if request.method != 'POST':  
 """未提交数据：创建一个新表单"""  
 form = TopicForm()  
 else:  
 form = TopicForm(data=request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('learning\_logs:topics')  
  
 context = {'form':form}  
 return render(request,'learning\_logs/new\_topic.html',context)

#### 4、定义new\_topic.html



<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>new\_topic</title>  
</head>  
<body>  
 {% extends "learning\_logs/base.html" %}  
 {% block content %}  
 <p>Add a new topic:</p>  
 <form action="{% url 'learning\_logs:new\_topic' %}" method="post">  
 {% csrf\_token %}  
 {{ form.as\_div }}  
 <button name="submit">Add topic</button>  
 </form>  
 {% endblock content %}  
  
</body>  
</html>

#### 5、在topics html文件中新建链接



### 2、添加新条目（Entry）

#### 1、用于添加新条目的表单

class EntryForm(forms.ModelForm):  
 class Meta:  
 model = Entry  
 fields = ['text']  
 labels = {'text': ''}  
 widgets = {'text': forms.Textarea(attrs={'cols': 80})}

#### 2、定义URL

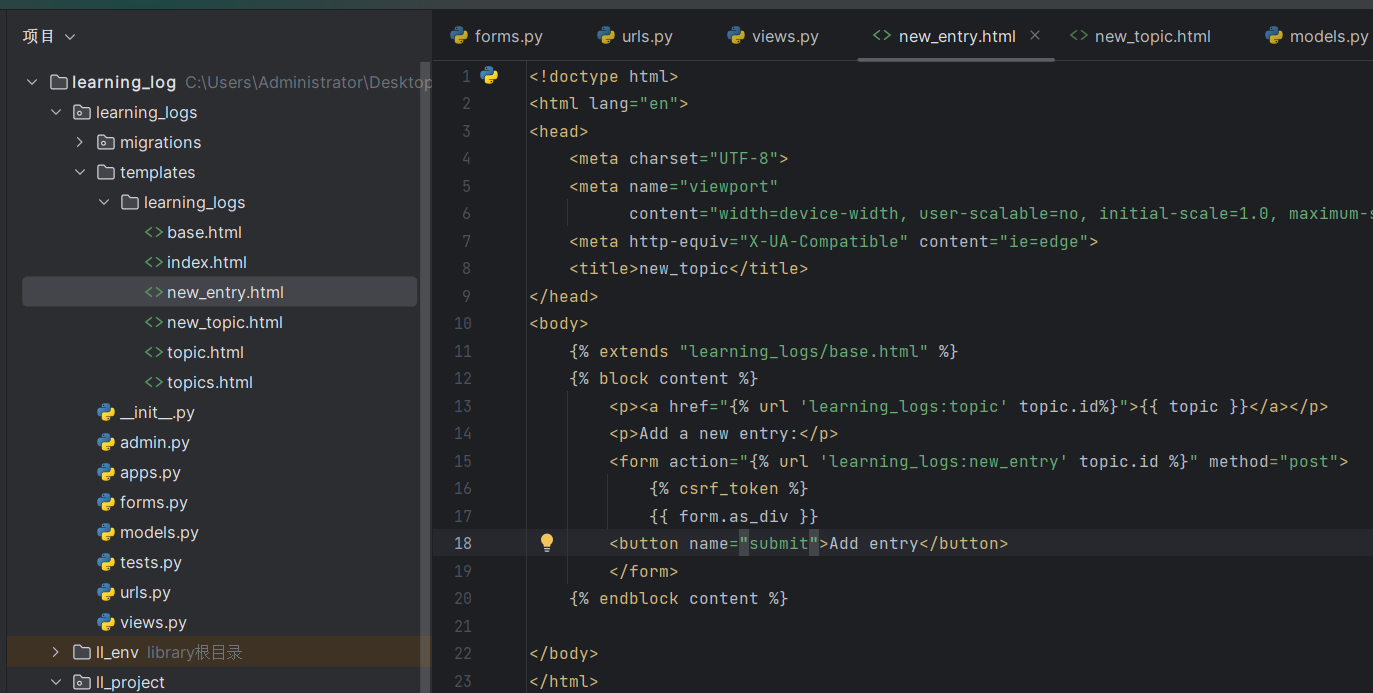


urlpatterns = [  
 # home page  
 path('', views.index, name='index'),  
 path('topics/', views.topics, name='topics'),  
 path('topics/<int:topic\_id>/', views.topic, name='topic'),  
 path('new\_topic', views.new\_topic, name='new\_topic'),  
 path('new\_entry/<int:topic\_id>/', views.new\_entry, name='new\_entry'),  
  
]

#### 3、定义view函数

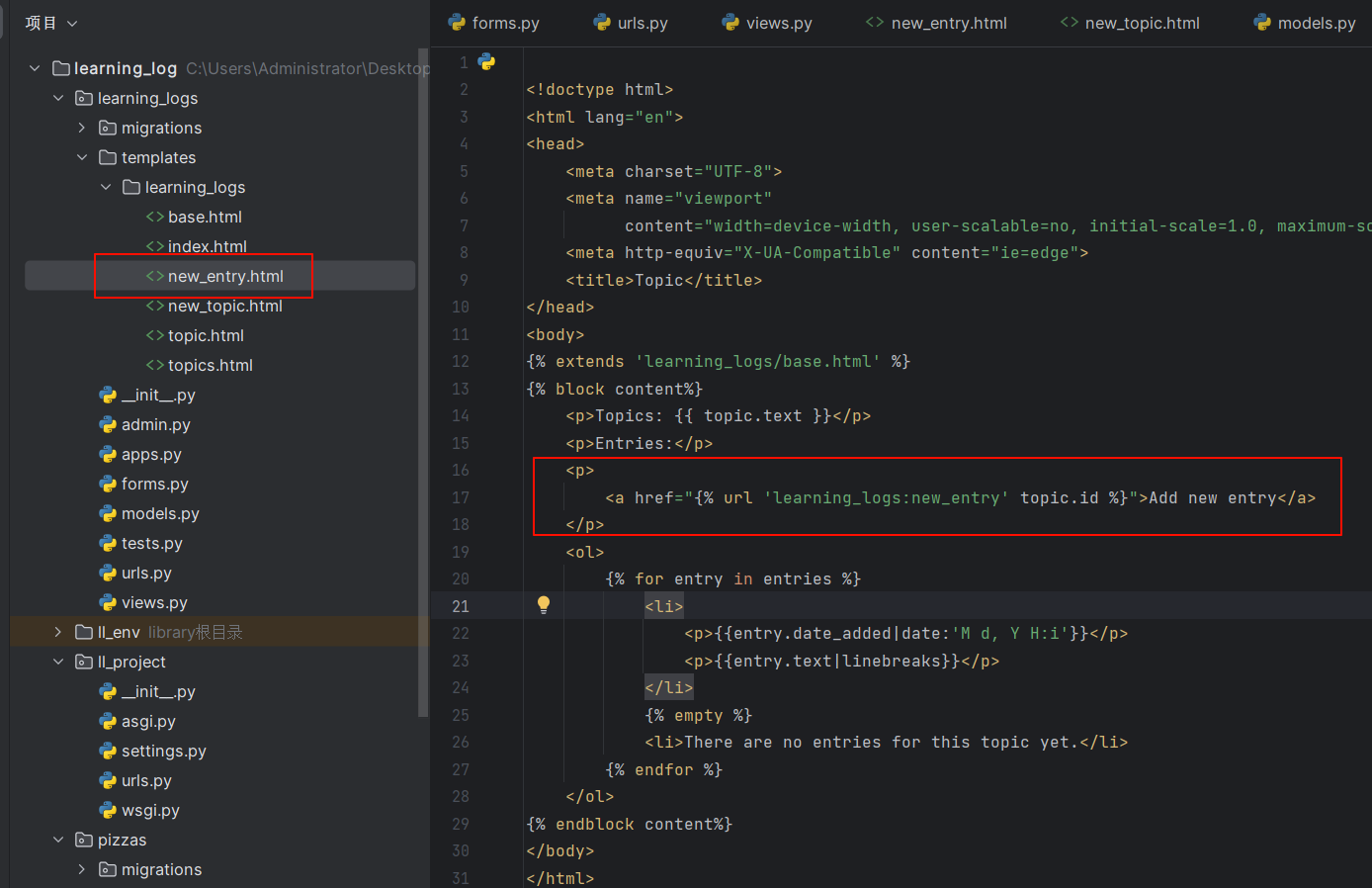
def new\_entry(request, topic\_id):  
 """在特定主题下添加新条目"""  
 topic = Topic.objects.get(id=topic\_id)  
 if request.method != 'POST':  
 form = EntryForm()  
 else:  
 form = EntryForm(data=request.POST)  
 if form.is\_valid():  
 new\_entry = form.save(commit=False)  
 new\_entry.topic = topic  
 new\_entry.save()  
 return redirect('learning\_logs:topic', topic\_id=topic\_id)  
  
 context = {'topic': topic, 'form': form}  
 return render(request, 'learning\_logs/new\_entry.html', context)

#### 4、定义new\_entry.html文件



<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>new\_topic</title>  
</head>  
<body>  
 {% extends "learning\_logs/base.html" %}  
 {% block content %}  
 <p><a href="{% url 'learning\_logs:topic' topic.id%}">{{ topic }}</a></p>  
 <p>Add a new entry:</p>  
 <form action="{% url 'learning\_logs:new\_entry' topic.id %}" method="post">  
 {% csrf\_token %}  
 {{ form.as\_div }}  
 <button name="submit">Add entry</button>  
 </form>  
 {% endblock content %}  
  
</body>  
</html>

#### 5、在topic html文件中新建添加新条目链接



### 3、编辑条目

#### 1、定义URL

urlpatterns = [  
 # home page  
 path('', views.index, name='index'),  
 path('topics/', views.topics, name='topics'),  
 path('topics/<int:topic\_id>/', views.topic, name='topic'),  
 path('new\_topic', views.new\_topic, name='new\_topic'),  
 path('new\_entry/<int:topic\_id>/', views.new\_entry, name='new\_entry'),  
 path('edit\_entry/<int:entry\_id>',views.edit\_entry,name='edit\_entry'),  
  
]

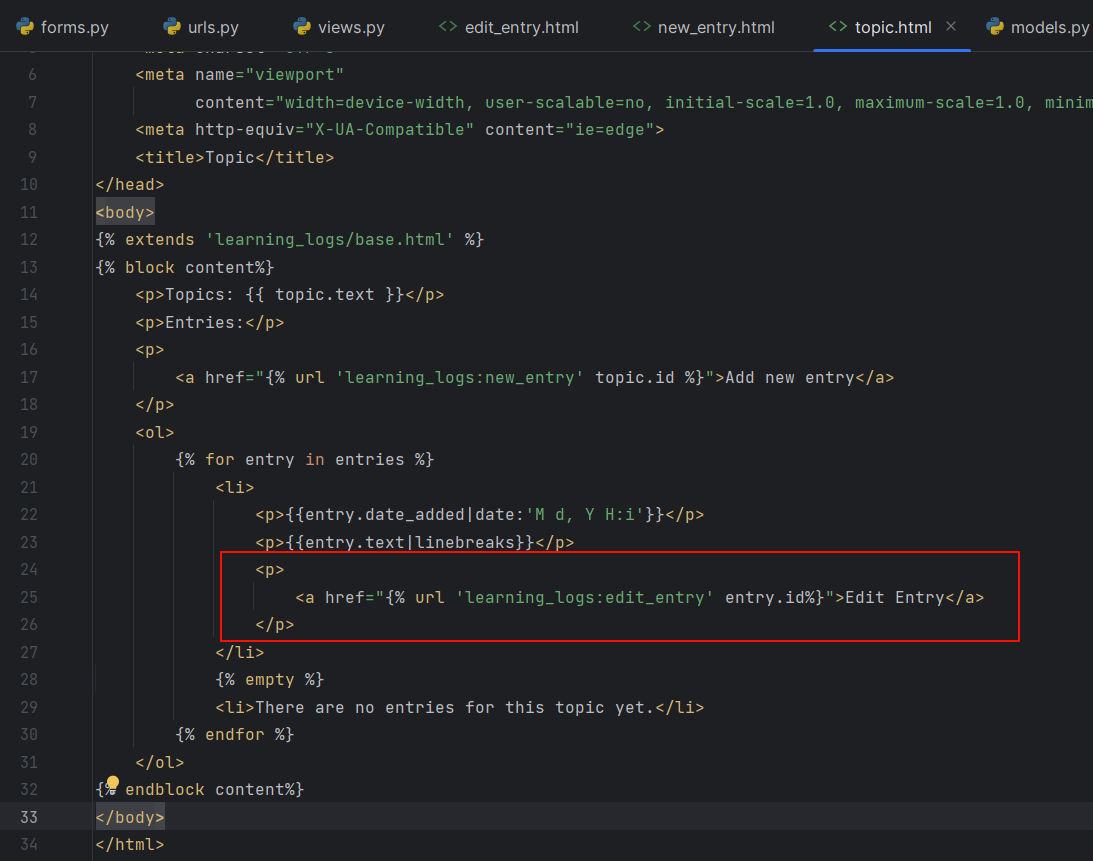
#### 2、定义view函数

def edit\_entry(request, entry\_id):  
 """编辑现有的条目"""  
 entry = Entry.objects.get(id=entry\_id)  
 topic = entry.topic  
 if request.method != 'POST':  
 # 初次请求：使用当前的条目填充表单  
 form = EntryForm(instance=entry)  
 else:  
 # post提交的数据，对数据进行处理  
 form = EntryForm(instance=entry, data=request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('learning\_logs:topic', topic\_id=topic.id)  
  
 context = {'entry': entry, 'topic': topic, 'form': form}  
 return render(request, 'learning\_logs/edit\_entry.html', context)

#### 3、定义editentryhtml文件

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>edit\_entry</title>  
</head>  
<body>  
 {% extends "learning\_logs/base.html" %}  
 {% block content %}  
 <p><a href="{% url 'learning\_logs:topic' topic.id%}">{{ topic }}</a></p>  
 <p>Edit entry:</p>  
 <form action="{% url 'learning\_logs:edit\_entry' entry.id %}" method="post">  
 {% csrf\_token %}  
 {{ form.as\_div }}  
 <button name="submit">Save changes</button>  
 </form>  
 {% endblock content %}  
  
</body>  
</html>

#### 4、在topichtml文件中新建编辑条目链接



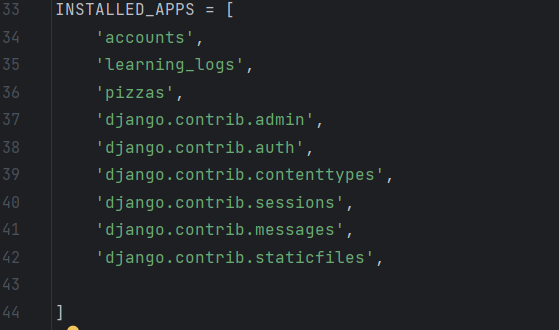
### 4、创建用户账户

#### 1、登录页面

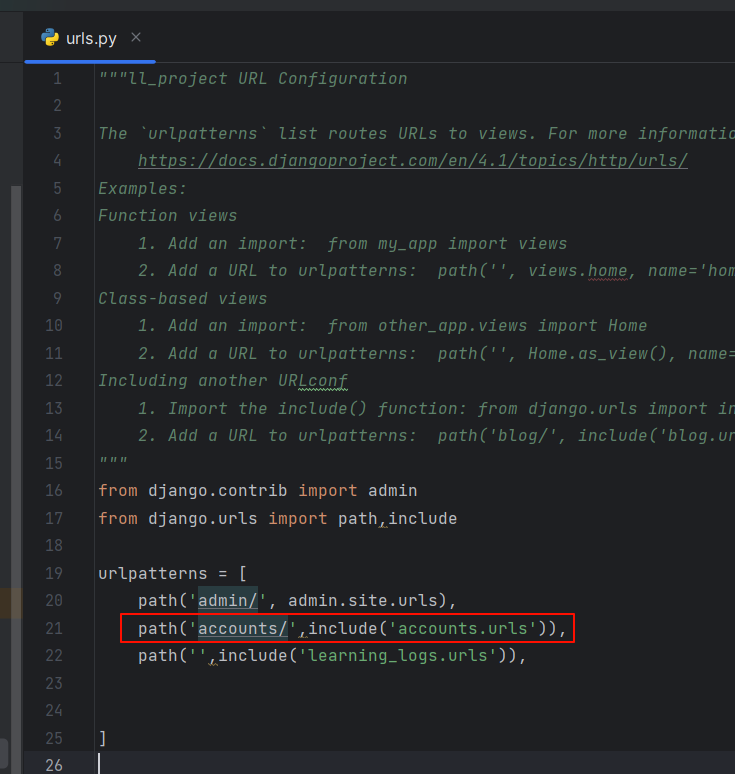
##### 创建管理用户账户的app：

py manage.py startapp accounts

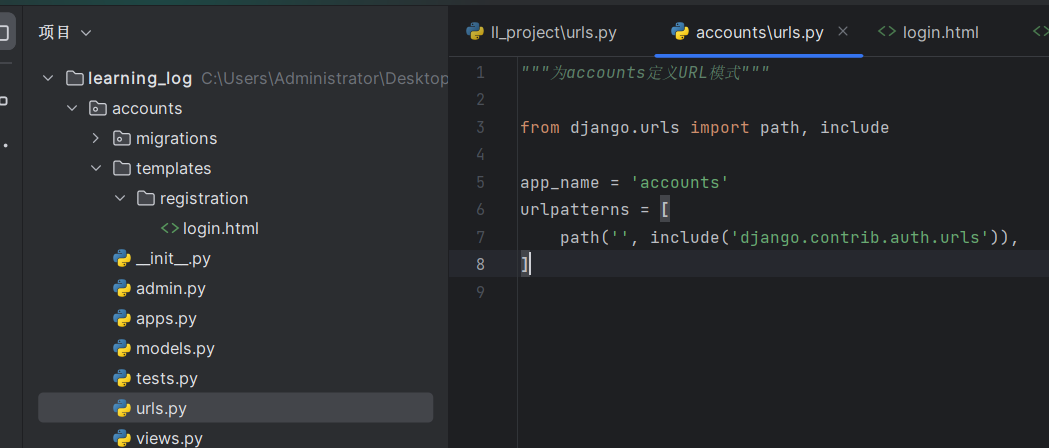
##### 在settings文件中注册app



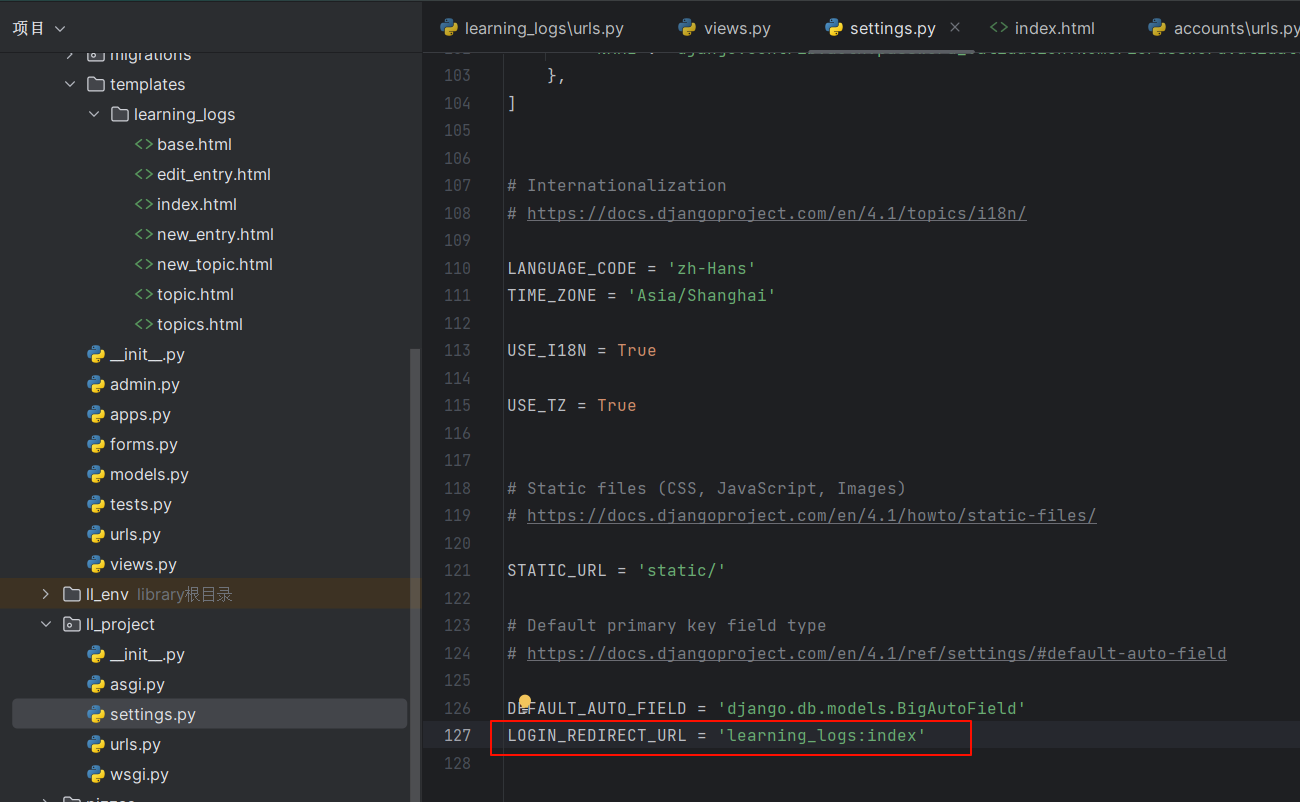
##### 在主路由中，添加accounts urls



##### 定义accounts urls

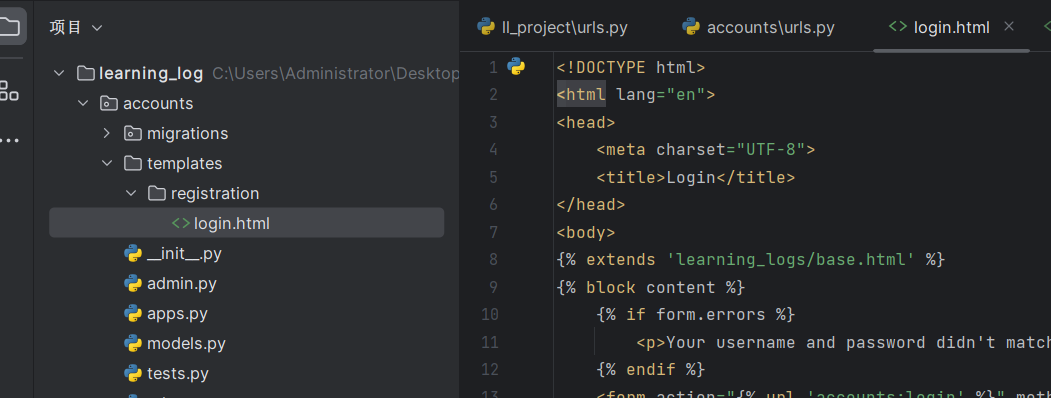


##### 在主settings中让djago知道用户成功登录后应该重定向到哪里

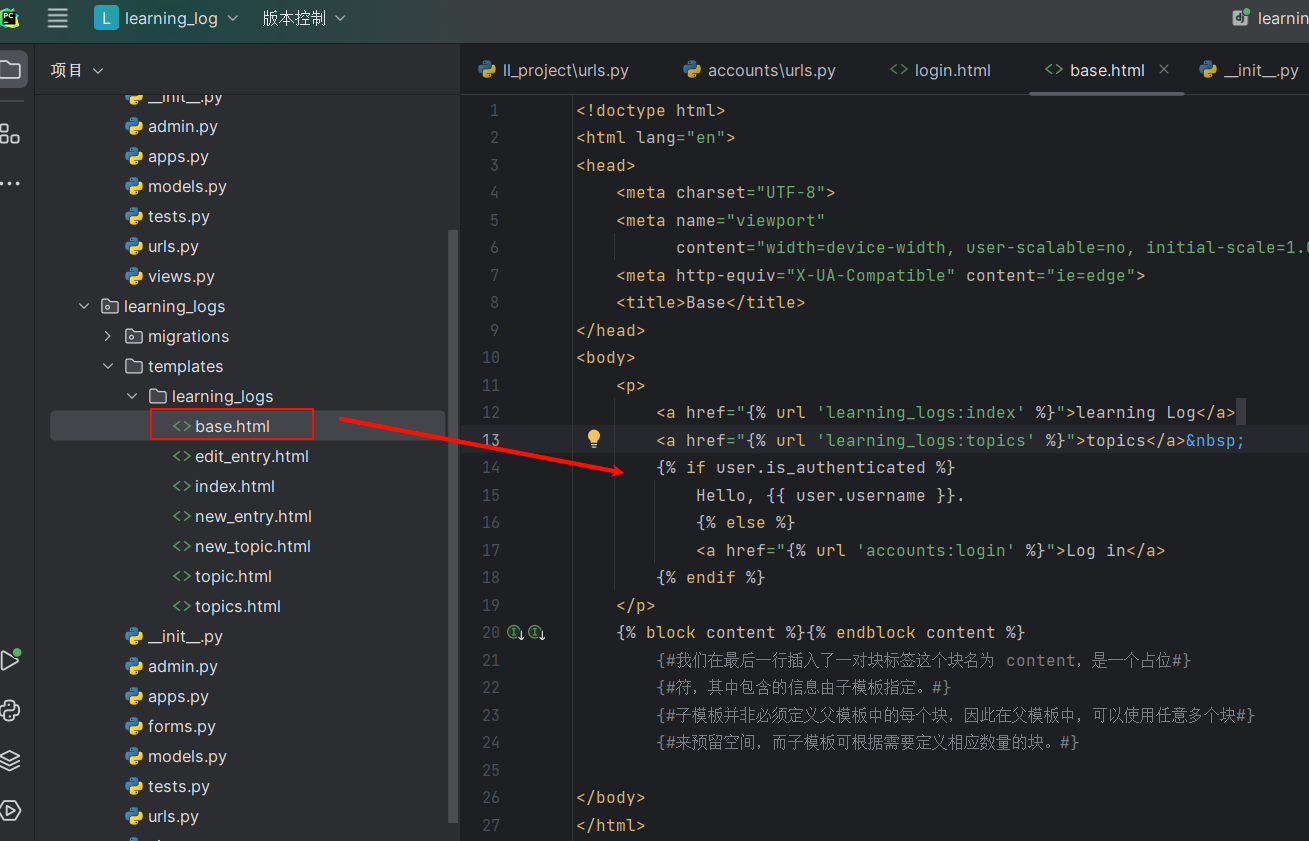


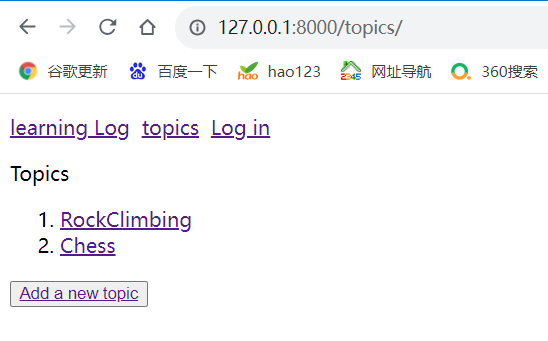
##### 编写login html页面

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Login</title>  
</head>  
<body>  
{% extends 'learning\_logs/base.html' %}  
{% block content %}  
 {% if form.errors %}  
 <p>Your username and password didn't match. Please try again.</p>  
 {% endif %}  
 <form action="{% url 'accounts:login' %}" method="post">  
 {% csrf\_token %}  
 {{ form.as\_div }}  
 <button name="submit">Log in</button>  
 </form>  
{% endblock content %}  
</body>  
</html>



##### 在base html页面添加login in选项



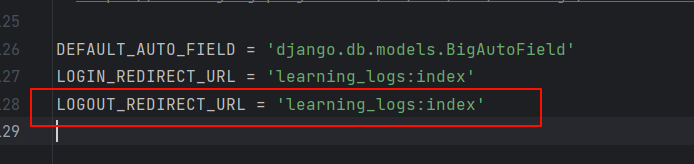


#### 2、注销登录

##### 在basehtml文件中添加注销

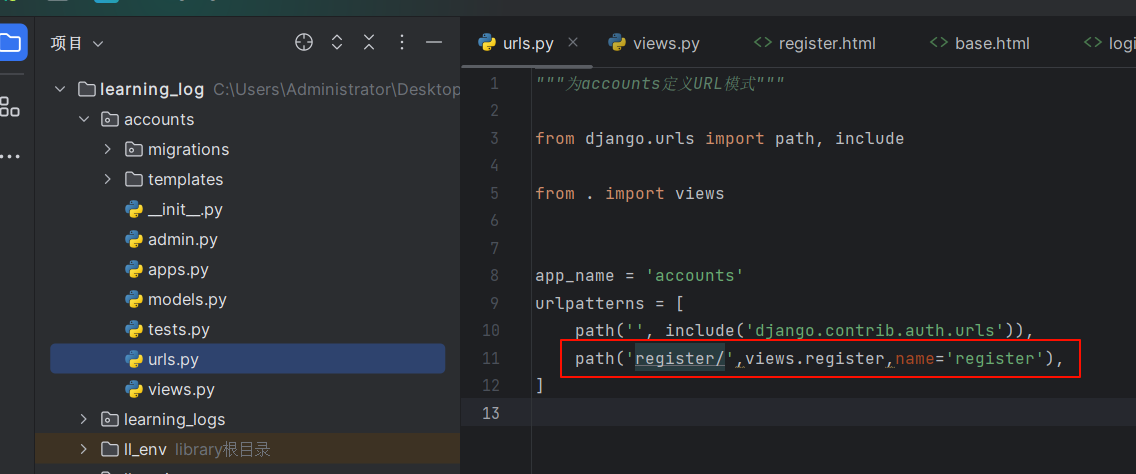


##### 用户点击注销后，在settings告诉django重定向跳转到index页面



#### 3、注册页面

##### 定义注册url



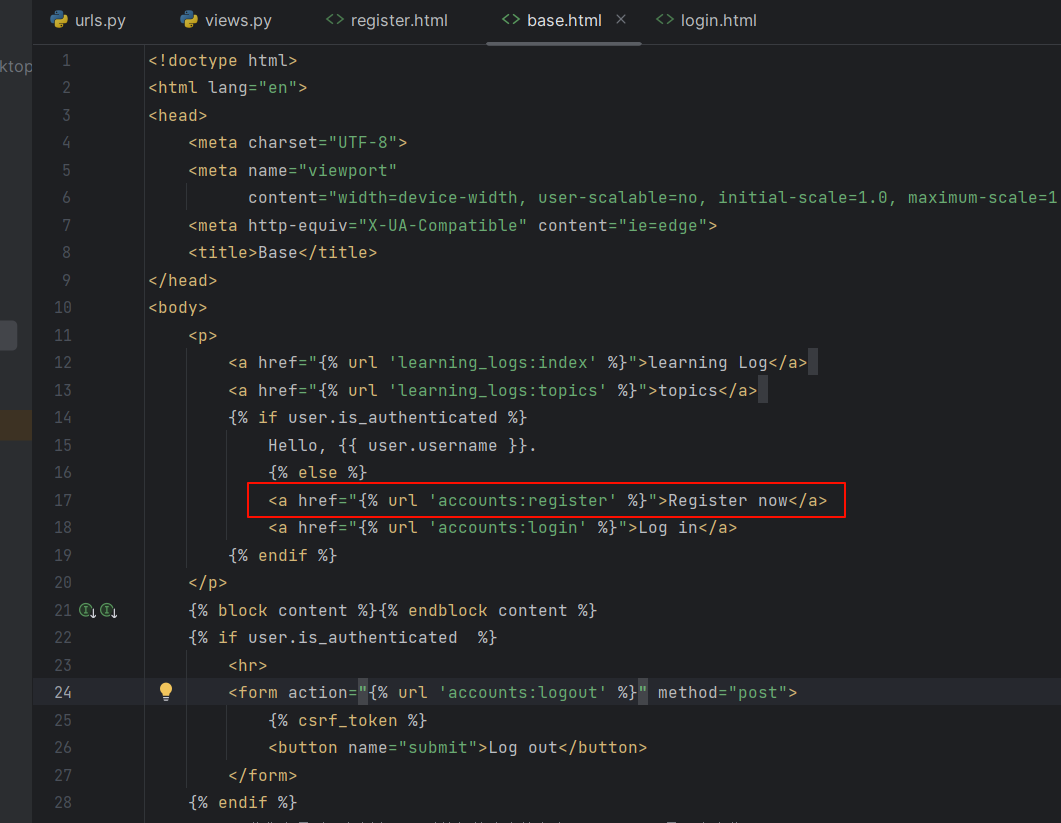
##### 实现view函数

from django.shortcuts import render, redirect  
from django.contrib.auth import login  
from django.contrib.auth.forms import UserCreationForm  
  
  
# Create your views here.  
def register(request):  
 """注册新用户"""  
 if request.method != "POST":  
 # 显示空的注册表单  
 form = UserCreationForm()  
 else:  
 # 处理填写好的表单  
 form = UserCreationForm(data=request.POST)  
 if form.is\_valid():  
 new\_user = form.save()  
 # 让用户自动登录，再重定向到主页  
 login(request, new\_user)  
 return redirect('learning\_logs:index')  
  
 # 显示空表单或指出表单无效  
 context = {'form': form}  
 return render(request, 'registration/register.html', context)

##### 实现register html页面

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Login</title>  
</head>  
<body>  
{% extends 'learning\_logs/base.html' %}  
{% block content %}  
 <form action="{% url 'accounts:register' %}" method="post">  
 {% csrf\_token %}  
 {{ form.as\_div }}  
 <button name="submit">Register now</button>  
 </form>  
{% endblock content %}  
</body>  
</html>

##### 在base html页面添加register选项

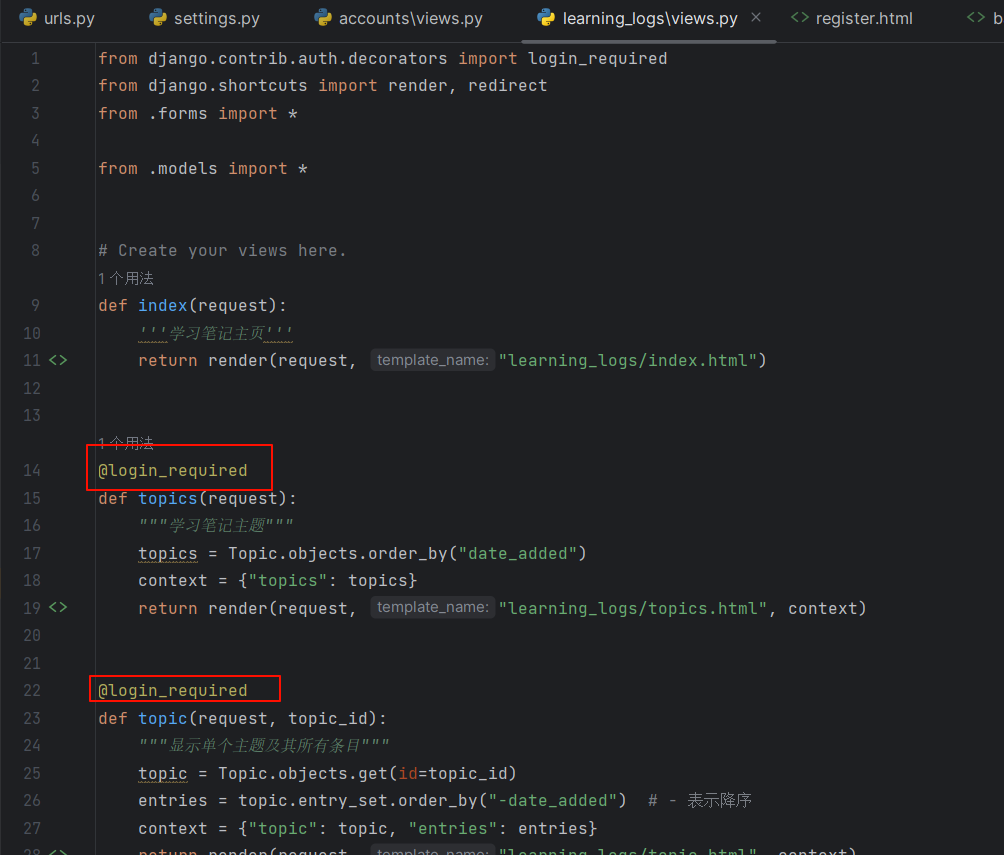


### 5、让用户拥有自己的数据

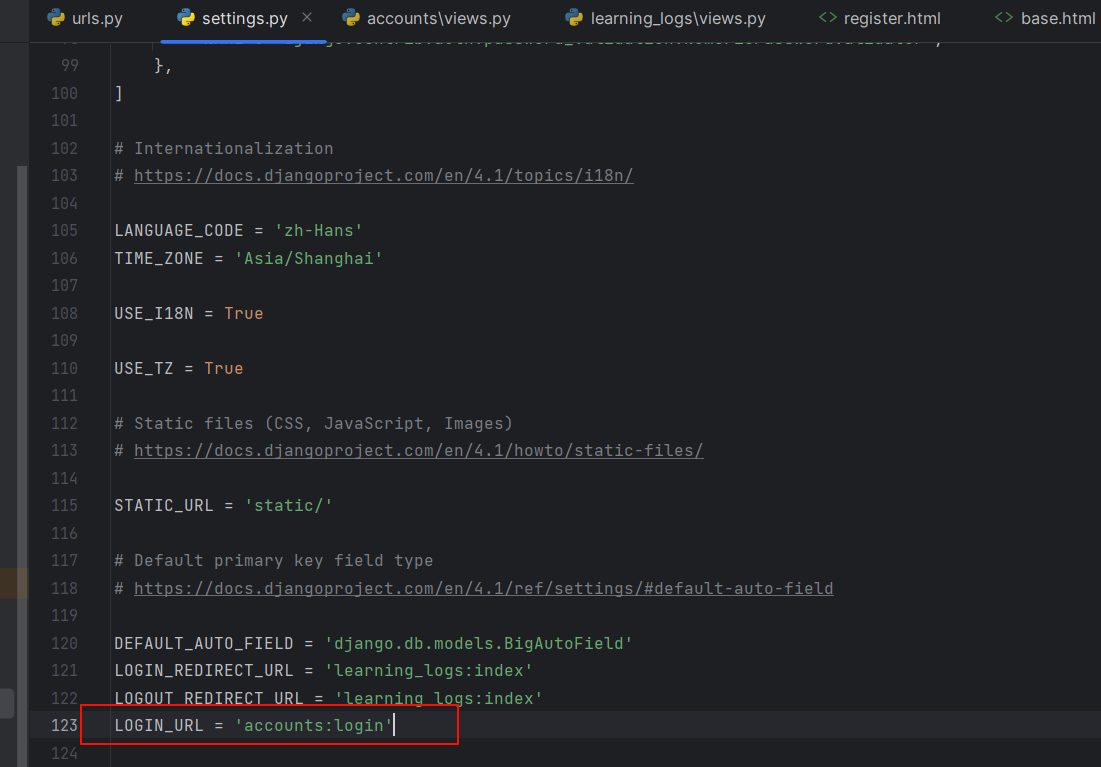
##### 1、使用@login\_required

在访问特定页面时，检查用户是否为登录状态

除了index函数外，其他函数都加上该装饰器

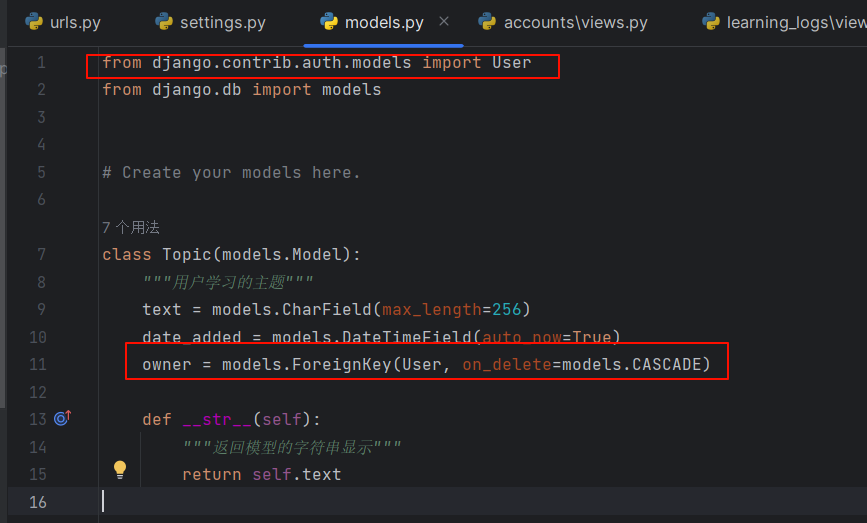


并且在settings文件中 配置当用户被此装饰器“拦截”后，系统将重定向到哪个页面



##### 2、将数据关联到用户

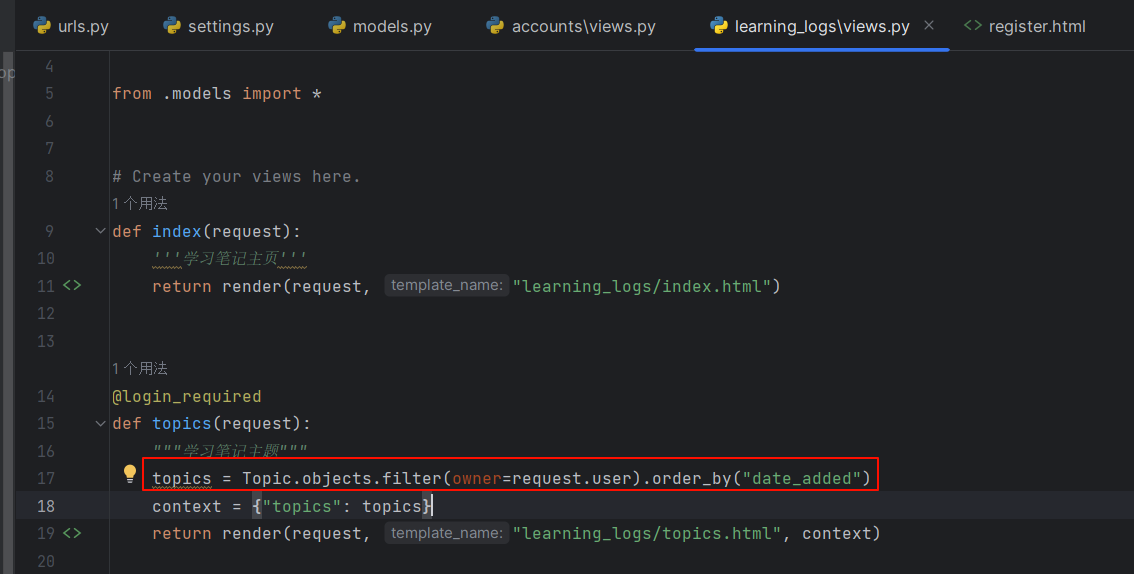
在models文件中添加两行代码



执行数据库迁移：py manage.py makemigrations

py manage.py migrate

##### 3、让用户只能看自己的主题

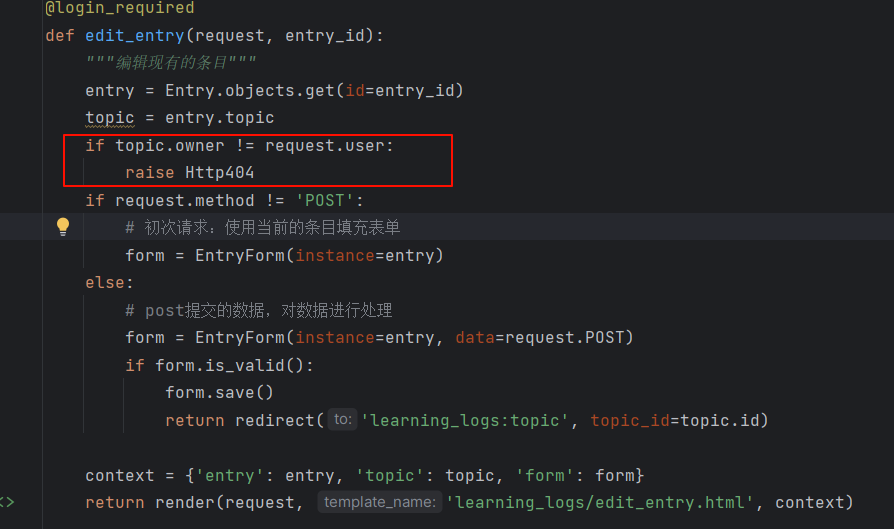


##### 4、保护用户的主题



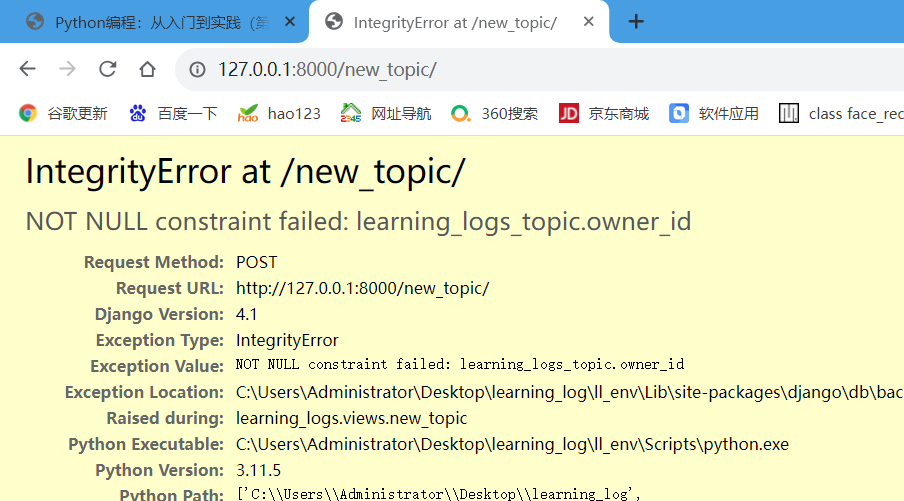
如果当前主题的所有者非登录用户，抛出404

##### 5、保护页面edit——entry

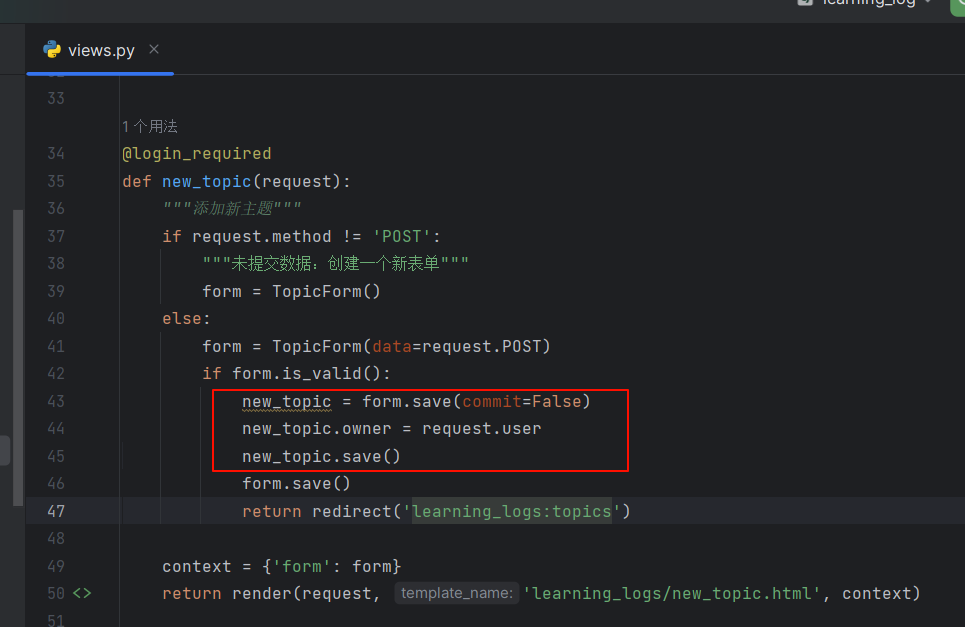


如果当前条目的所有者非登录用户，抛出404

##### 解决添加新主题时的报错



添加如下代码

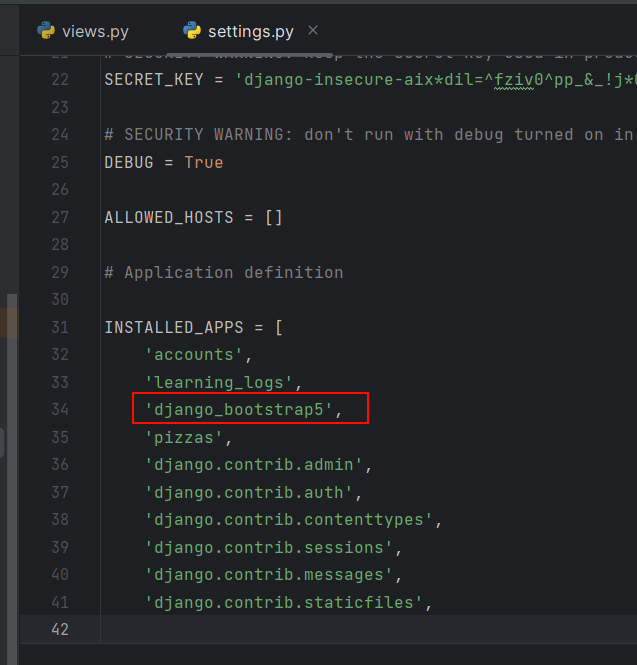


## 8、添加样式美化网站

### 1、安装bootstrap5

1、 pip install django-bootstrap5

2、在settings里添加代码



### 2、修改basehtml文件

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Base</title>  
 {% load django\_bootstrap5 %}  
 {% bootstrap\_css %}  
 {% bootstrap\_javascript %}  
</head>  
<body>  
<nav class="navbar navbar-expand-md navbar-light bg-light mb-4 border">  
 <div class="container-fluid">  
 <a class="navbar-brand" href="{% url 'learning\_logs:index' %}">  
 学习日志</a>  
 <button class="navbar-toggler" type="button" data-bs-toggle="collapse"  
 data-bs-target="#navbarCollapse" aria-controls="navbarCollapse"  
 aria-expanded="false" aria-label="Toggle navigation">  
 <span class="navbar-toggler-icon"></span>  
 </button>  
 <div class="collapse navbar-collapse" id="navbarCollapse">  
 <ul class="navbar-nav me-auto mb-2 mb-md-0">  
 <li class="nav-item">  
 <a class="nav-link" href="{% url 'learning\_logs:topics' %}">  
 主题</a>  
 </li>  
 </ul> <!-- 定义导航栏左侧链接的代码到此结束 -->  
 <ul class="navbar-nav ms-auto mb-2 mb-md-0">  
 {% if user.is\_authenticated %}  
 <li>  
 <span class="navbar-text me-2">你好啊,{{ user.username }}！</span>  
 </li>  
 {% else %}  
 <li class="nav-item"><a class="nav-link" href="{% url 'accounts:register' %}">注册账号</a></li>  
 <li class="nav-item"><a class="nav-link" href="{% url 'accounts:login' %}">登录</a></li>  
 {% endif %}  
 </ul><!--与账号相关的链接到此结束-->  
 {% if user.is\_authenticated %}  
 <form action="{% url 'accounts:logout' %}" method="post">  
 {% csrf\_token %}  
 <button name="submit" class="btn btn-outline-secondary btn-sm">退出当前账号</button>  
 </form>  
 {% endif %}  
 </div> <!-- 定义导航栏可折叠部分的代码到此结束 -->  
 </div> <!-- 定义导航栏容器的代码到此结束 -->  
 </nav> <!-- 定义导航栏的代码到此结束 -->  
  
<main class="container">  
 <div class="pb-2 mb-2 border-bottom">  
 {% block page\_header %}{% endblock page\_header %}  
 </div>  
 <div>  
 {% block content %}{% endblock content %}  
 </div>  
</main>  
 {#我们在最后一行插入了一对块标签这个块名为 content，是一个占位#}  
 {#符，其中包含的信息由子模板指定。#}  
 {#子模板并非必须定义父模板中的每个块，因此在父模板中，可以使用任意多个块#}  
 {#来预留空间，而子模板可根据需要定义相应数量的块。#}  
  
</body>  
</html>

### 3、设置index文件

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>芝士首页</title>  
</head>  
<body>  
 {% extends 'learning\_logs/base.html' %}  
 {% block page\_header %}  
 <div class="p-3 mb-4 bg-light border rounded-3">  
 <div class="container-fluid py-4">  
 <h1 class="display-3">记录你的学习成果</h1>  
 <p class="lead">Make your own Learning Log,\_and keep a list of thetopicsyou'relearning about.  
 Whenever you learn something newabout a topic, make an entry summarizing what you've learned.  
 </p>  
 <p class="lead">制作你自己的学习日志，并把你正在学习的主题列在一张清单上。  
每当你对一个话题有了新的了解，就做一个总结。  
 </p>  
 <a class="btn btn-primary btn-lg mt-1" href="{% url 'accounts:register' %}">点我注册 &raquo</a>  
 </div>  
 </div>  
 {% endblock page\_header %}  
  
</body>  
</html>

### 4、设置login文件

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>登录</title>  
</head>  
<body>  
{% extends 'learning\_logs/base.html' %}  
{% load django\_bootstrap5 %}  
  
{% block page\_header %}  
 <h2>登录您的账号.</h2>  
{% endblock page\_header %}  
{% block content %}  
 {% if form.errors %}  
 <p>您的用户名和密码不匹配。请再试一次。</p>  
 {% endif %}  
 <form action="{% url 'accounts:login' %}" method="post">  
 {% csrf\_token %}  
 {% bootstrap\_form form %}  
 {% bootstrap\_button button\_type='submit' content='Log in'%}  
 </form>  
{% endblock content %}  
</body>  
</html>

### 5、修改topics文件

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>主题</title>  
</head>  
<body>  
{% extends 'learning\_logs/base.html' %}  
{% block page\_header %}  
 <h1>你的主题</h1>  
{% endblock page\_header %}  
{% block content %}  
 <ul class="list-group border-bottom pb-2 mb-4">  
 {% for topic in topics %}  
 <li class="list-group-item border-0"><a href="{% url 'learning\_logs:topic' topic.id%}">  
 {{ topic.text }}</a>  
 </li>  
 {% empty %}  
 <li class="list-group-item border-0">当前您还没有添加任何主题。</li>  
 {% endfor %}  
 </ul>  
 <a href="{% url 'learning\_logs:new\_topic' %}">点击添加新主题</a>  
{% endblock content %}  
  
</body>  
</html>

### 6、修改topic文件

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>主题详情</title>  
</head>  
<body>  
{% extends 'learning\_logs/base.html' %}  
{% block page\_header %}  
 <h1>{{ topic.text }}</h1>  
{% endblock page\_header %}  
{% block content%}  
 <p>  
 <a href="{% url 'learning\_logs:new\_entry' topic.id %}">添加新条目</a>  
 </p>  
 {% for entry in entries %}  
 <div class="card mb-3">  
 <h4 class="card-header">  
 {{ entry.date\_added|date:'M d, Y H:i' }}  
 <small>  
 <a href="{% url 'learning\_logs:edit\_entry' entry.id%}">  
 编辑条目  
 </a>  
 </small>  
 </h4>  
 <div class="card-body">  
 {{ entry.text|linebreaks }}  
 </div>  
 </div>  
 {% empty %}  
 <li>当前主题还没有条目.</li>  
 {% endfor %}  
{% endblock content%}  
</body>  
</html>

## 9、效果展示



