产品名称 Product name	密级 Confidentiality level
	内部公开
产品版本 Product version	Total pages 共 6 页

Django 学习文档

拟制人：杜丹东　　时间：2023/05/18

评审人：　　时间：

批准人：　　时间：

软通动力技术服务有限公司

修订记录

版本	日期	修订内容	修订人
1.0	2023/05/18	初稿，模板制定	杜丹东
1.1	2023/05/18	首次基线化	杜丹东

目录

1. Django 初识

1.1. 简介

Python 下有许多款不同的 Web 框架。Django 是重量级选手中最有代表性的一位。许多成功的网站和 APP 都基于 Django。

Django 遵守 BSD 版权，初次发布于 2005 年 7 月，并于 2008 年 9 月发布了第一个正式版本 1.0 。

Django 是一个由 Python 编写的一个开放源代码的 Web 应用框架。

使用 Django，只要很少的代码，Python 的程序开发人员就可以轻松地完成一个正式网站所需要的大部分内容，并进一步开发出全功能的 Web 服务 Django 本身基于 MVC 模型，即 Model（模型）+ View（视图）+ Controller（控制器）设计模式，MVC 模式使后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。

1.2. MTV 模型

Django 的 MTV 模式本质上和 MVC 是一样的，也是为了各组件间保持松耦合关系，只是定义上有些许不同，Django 的 MTV 分别是指：

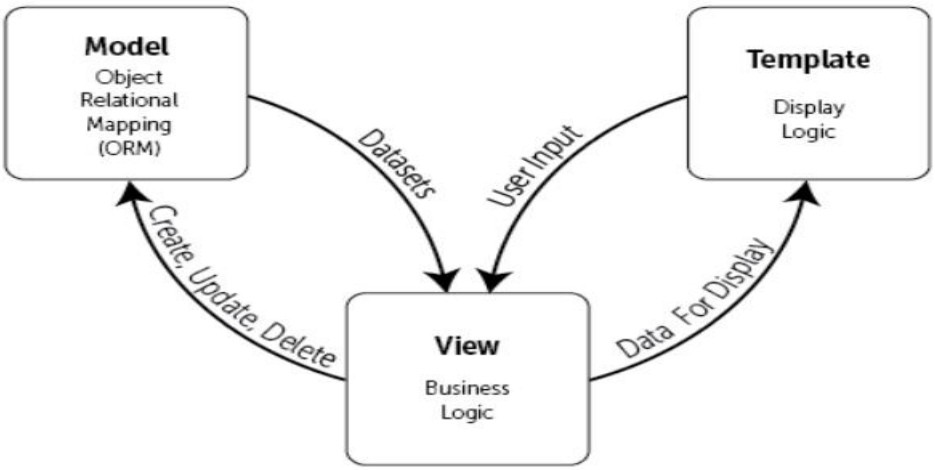
- M 表示模型（Model）：编写程序应有的功能，负责业务对象与数据库的映射(ORM)。
- T 表示模板 (Template)：负责如何把页面(html)展示给用户。
- V 表示视图（View）：负责业务逻辑，并在适当时候调用 Model 和 Template。

MVC（MTV）优势：

- 低耦合
- 开发快捷
- 部署方便
- 可重用性高
- 维护成本低

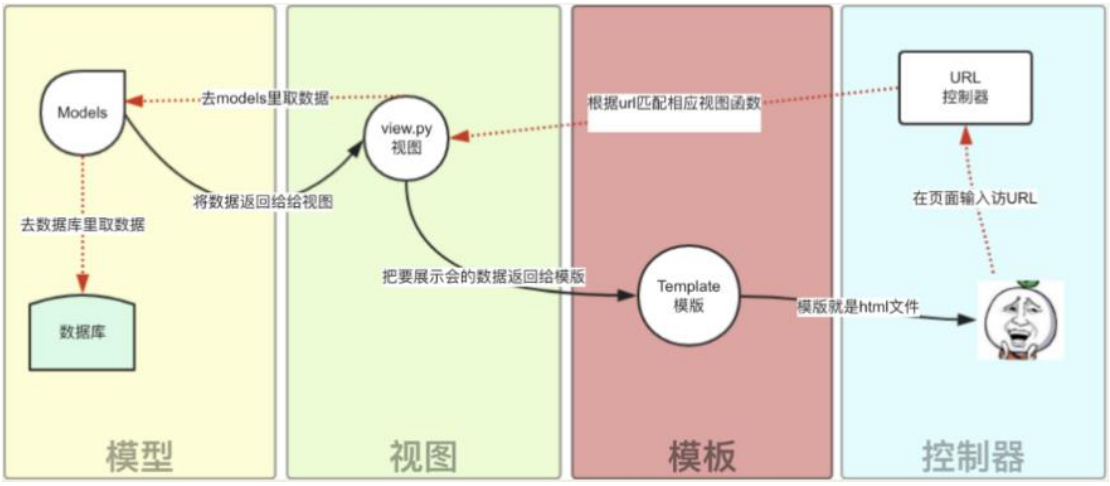
Python + Django 是快速开发、设计、部署网站的最佳组合。

除了以上三层之外，还需要一个 URL 分发器，它的作用是将一个个 URL 的页面请求分发给不同的 View 处理，View 再调用相应的 Model 和 Template，MTV 的响应模式如下所示：



简易图

用户操作流程图：



解析：

用户通过浏览器向我们的服务器发起一个请求(request)，这个请求会去访问视图函数：

a.如果不涉及到数据调用，那么这个时候视图函数直接返回一个模板也就是一个网页给用户。

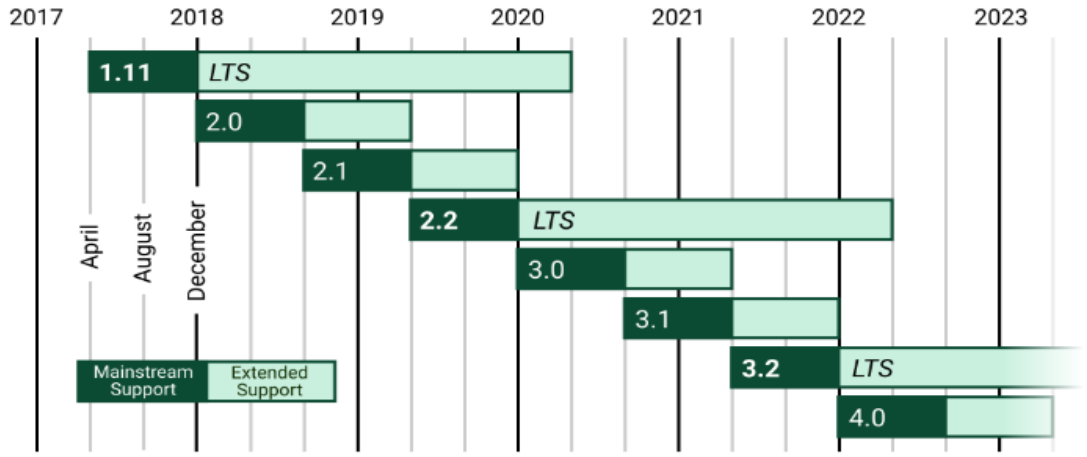
b.如果涉及到数据调用，那么视图函数调用模型，模型去数据库查找数据，然后逐级返回。

视图函数把返回的数据填充到模板中空格，最后返回网页给用户。

1.3. 对应关系

Django 版本对应的 Python 版本：

Django 版本	Python 版本
1.8	2.7, 3.2 , 3.3, 3.4, 3.5
1.9, 1.10	2.7, 3.4, 3.5
1.11	2.7, 3.4, 3.5, 3.6
2.0	3.4, 3.5, 3.6, 3.7
2.1, 2.2	3.5, 3.6, 3.7

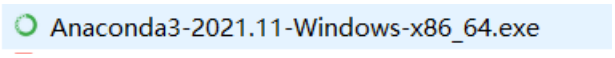


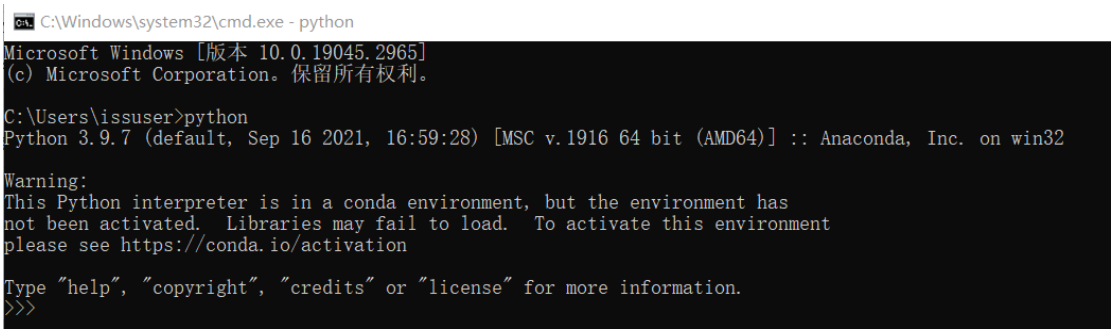
2. Django 安装

在安装 Django 前，系统需要已经安装了 Python 的开发环境。

2.1. Python 环境安装

推荐使用 anaconda 进行安装（只需把安装路径修改一下，然后一路下一步）





2.2. Django 安装

注意：尽量不要在 anaconda 中安装

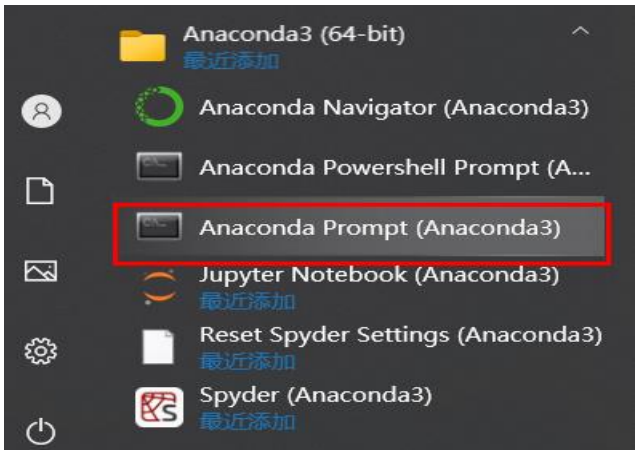
2.2.1. 离线安装

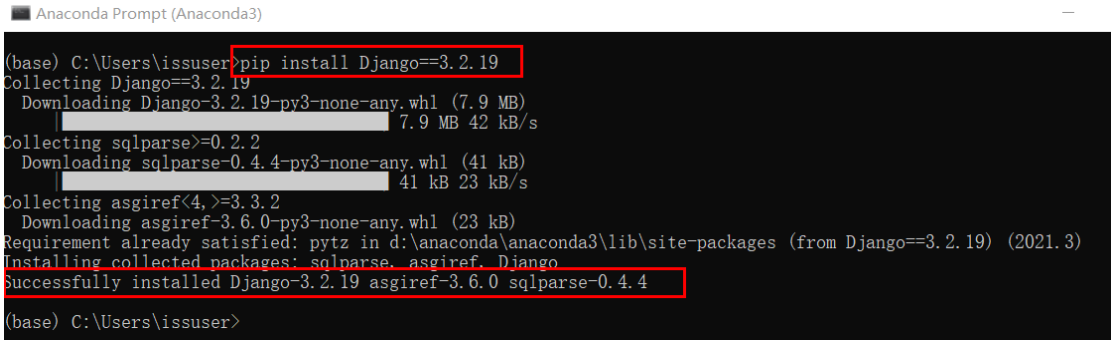
下载 Django 压缩包，解压并和 Python 安装目录放在同一个根目录，进入 Django 目录，执行 python setup.py install，然后开始安装，Django 将要被安装到 Python 的 Lib 下 site-packages。

Django 下载地址：https://www.djangoproject.com/download/

注意：此方法不推荐（比较麻烦），有兴趣自行百度。

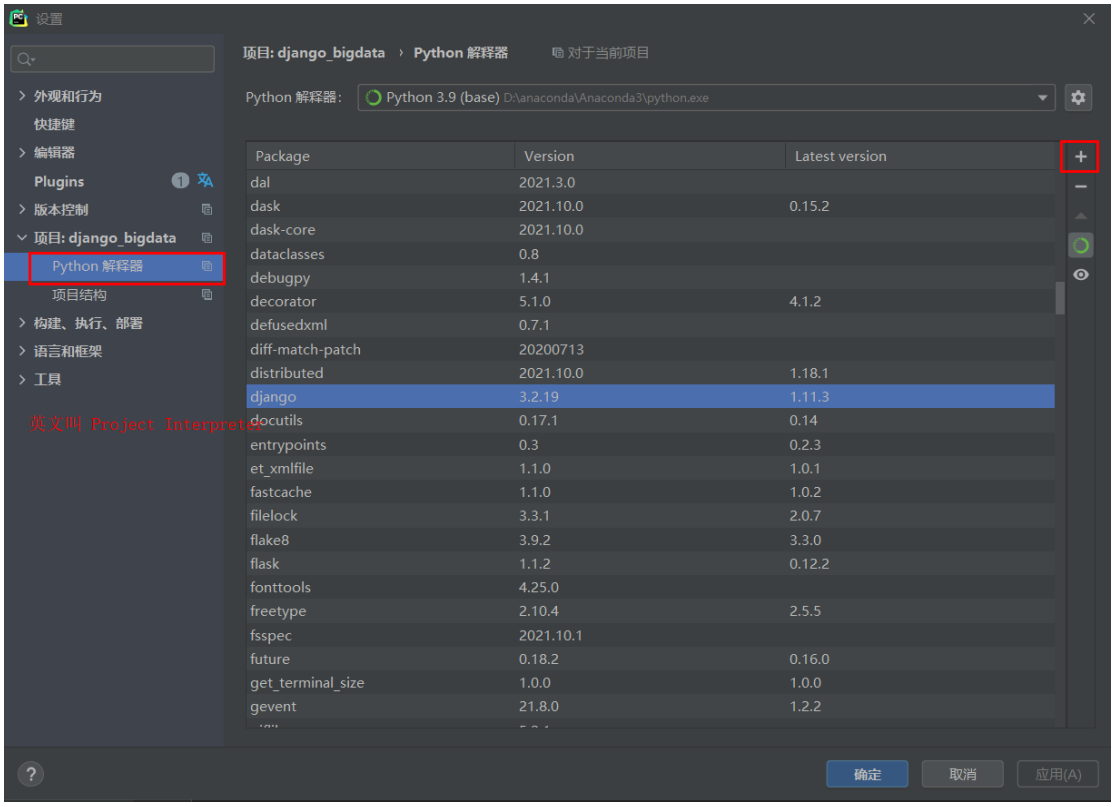
2.2.2. 命令行安装（推荐）

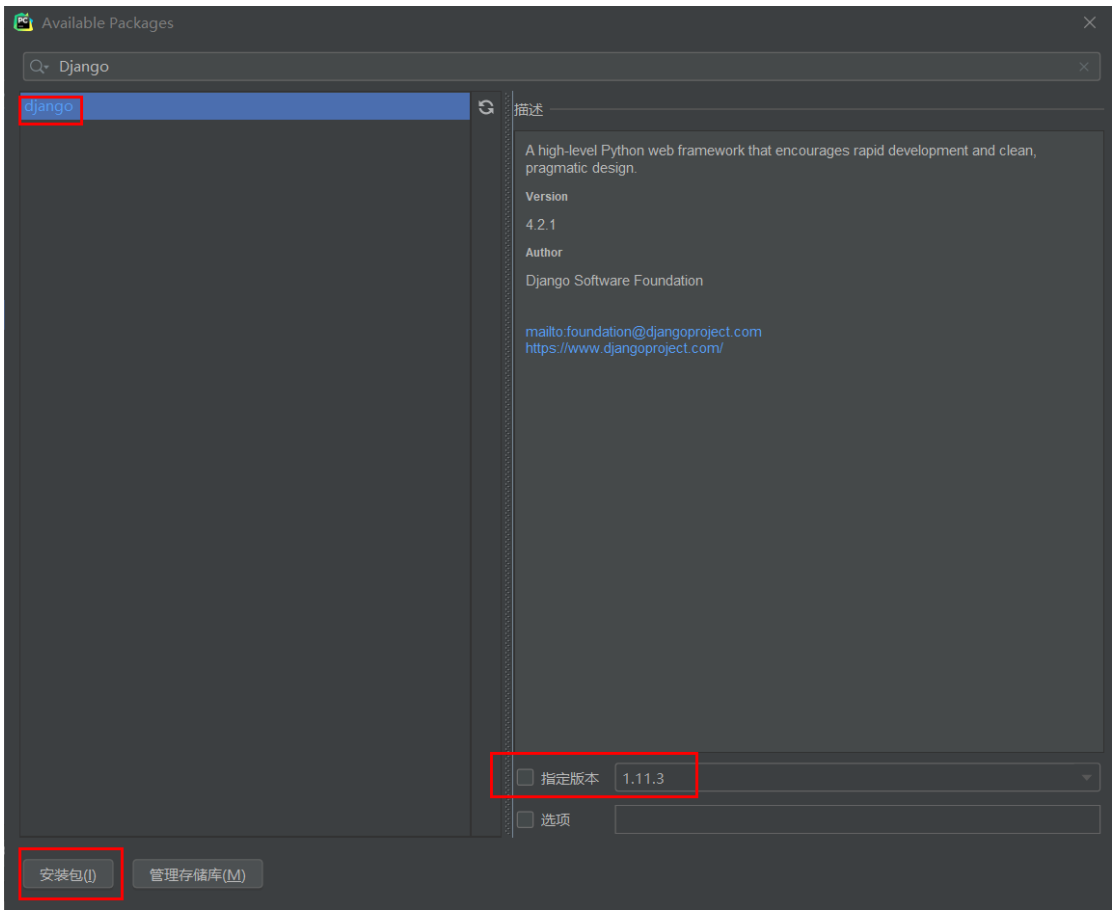




2.2.3. Pycharm 中安装

file ---> settings ----> project 解释器 ---> 点击+ ----> 输入 Django ---> install(如果选择版本时,选择 CheckBox ,选择指定版本)


设置
项目: django_bigdata › Python 解释器
对于当前项目
外观和行为
快捷键
编辑器
Plugins
版本控制
项目: django_bigdata
Python 解释器
项目结构
构建、执行、部署
语言和框架
工具
Python 解释器: Python 3.9 (base) D:\anaconda\Anaconda3\python.exe
Package
Version
Latest version
dal 2021.3.0
dask 2021.10.0 0.15.2
dask-core 2021.10.0
dataclasses 0.8
debugpy 1.4.1
decorator 5.1.0 4.1.2
defusedxml 0.7.1
diff-match-patch 20200713
distributed 2021.10.0 1.18.1
django 3.2.19 1.11.3
docutils 0.17.1 0.14
entrypoints 0.3 0.2.3
et_xmlfile 1.1.0 1.0.1
fastcache 1.1.0 1.0.2
filelock 3.3.1 2.0.7
flake8 3.9.2 3.3.0
flask 1.1.2 0.12.2
fonttools 4.25.0
freetype 2.10.4 2.5.5
fsspec 2021.10.1
future 0.18.2 0.16.0
get_terminal_size 1.0.0 1.0.0
gevent 21.8.0 1.2.2
英文叫 Project Interpreter
确定
取消
应用(A)


Available Packages
Django
django
描述
A high-level Python web framework that encourages rapid development and clean, pragmatic design.
Version
4.2.1
Author
Django Software Foundation
mailto:foundation@djangoproject.com
https://www.djangoproject.com/
指定版本
1.11.3
选项
安装包(I)
管理存储库(M)

2.2.4. 验证是否安装成功

```
import django
django.get_version() 或 print(django.VERSION)
```

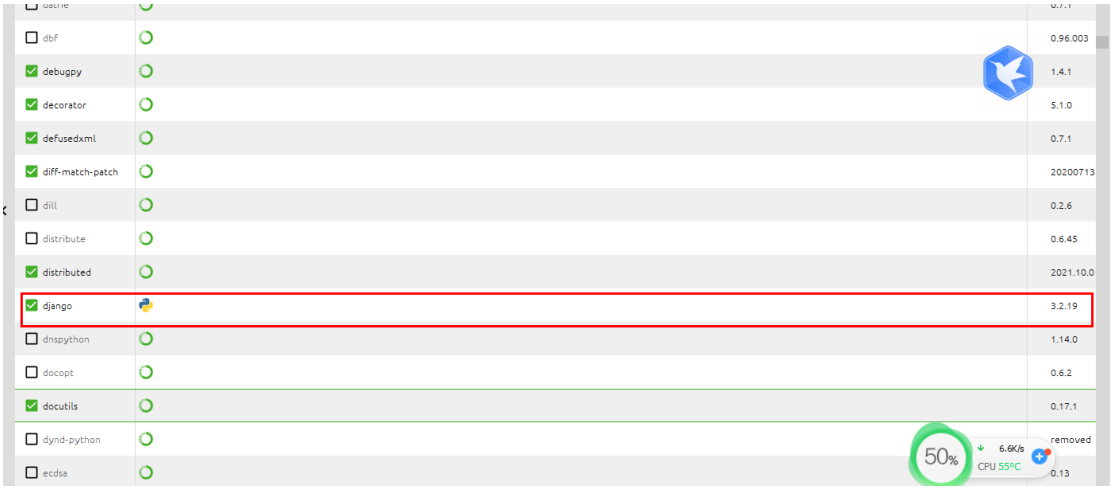
C:\Windows\system32\cmd.exe - python

```
Microsoft Windows [版本 10.0.19045.2965]
(c) Microsoft Corporation。保留所有权利。

C:\Users\issuser>python
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.get_version()
'3.2.19'
>>> print(django.VERSION)
(3, 2, 19, 'final', 0)
>>>
```

C:\Windows\system32\cmd.exe - python

```
Microsoft Windows [版本 10.0.19045.2965]
(c) Microsoft Corporation。保留所有权利。

C:\Users\issuser>python
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> django.get_version()
'3.2.19'
>>> print(django.VERSION)
(3, 2, 19, 'final', 0)
>>>
```

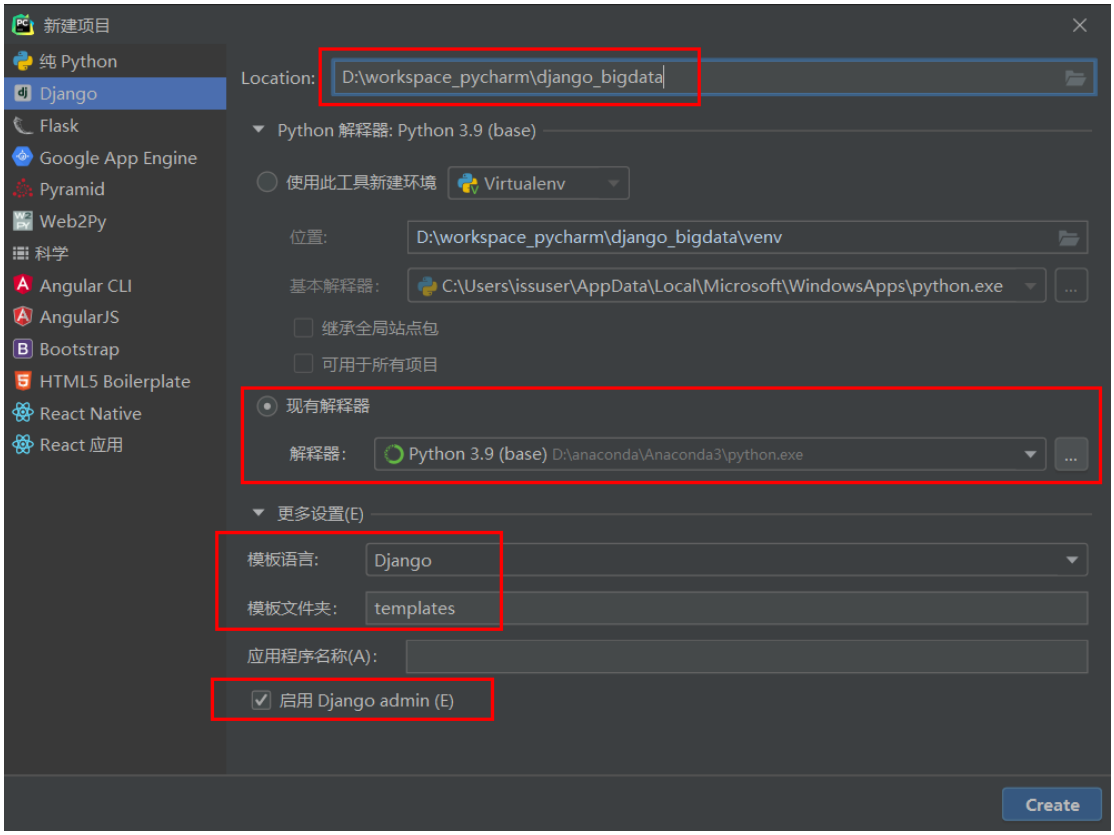
或

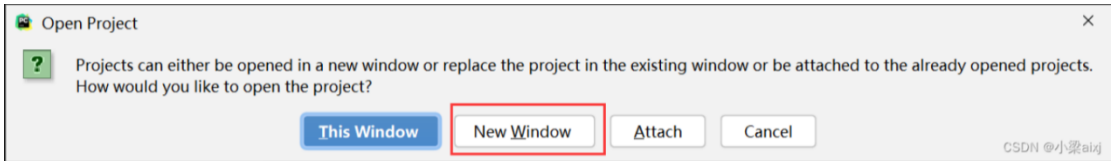


3. Django 项目介绍

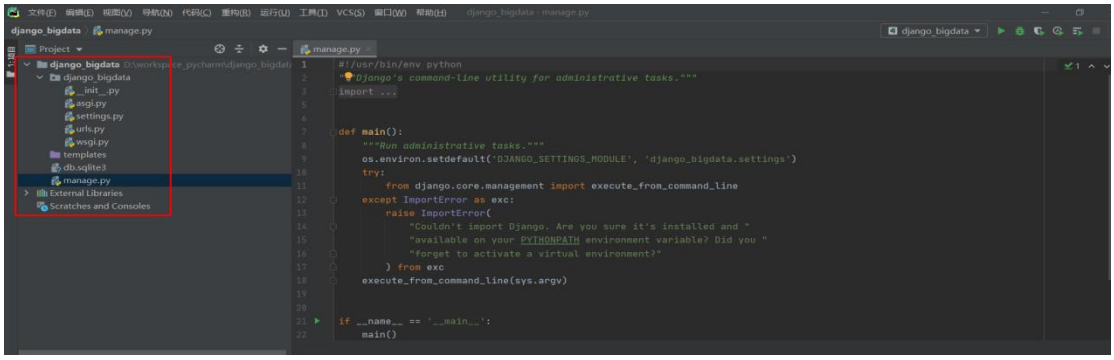
3.1. 命令行创建 Django 项目（略）

3.2. Pycharm 中创建 Django 项目

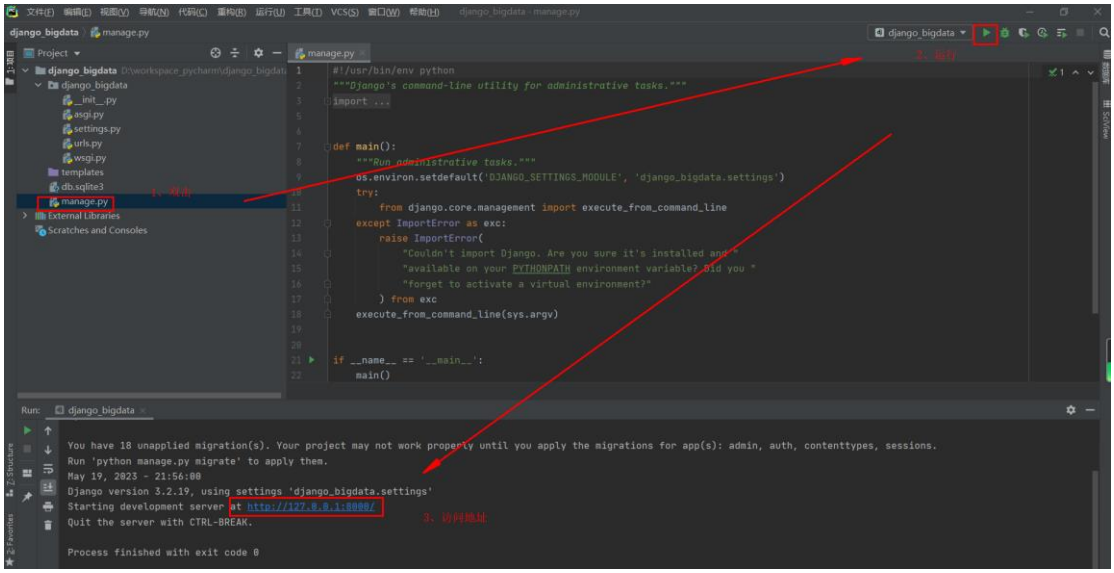




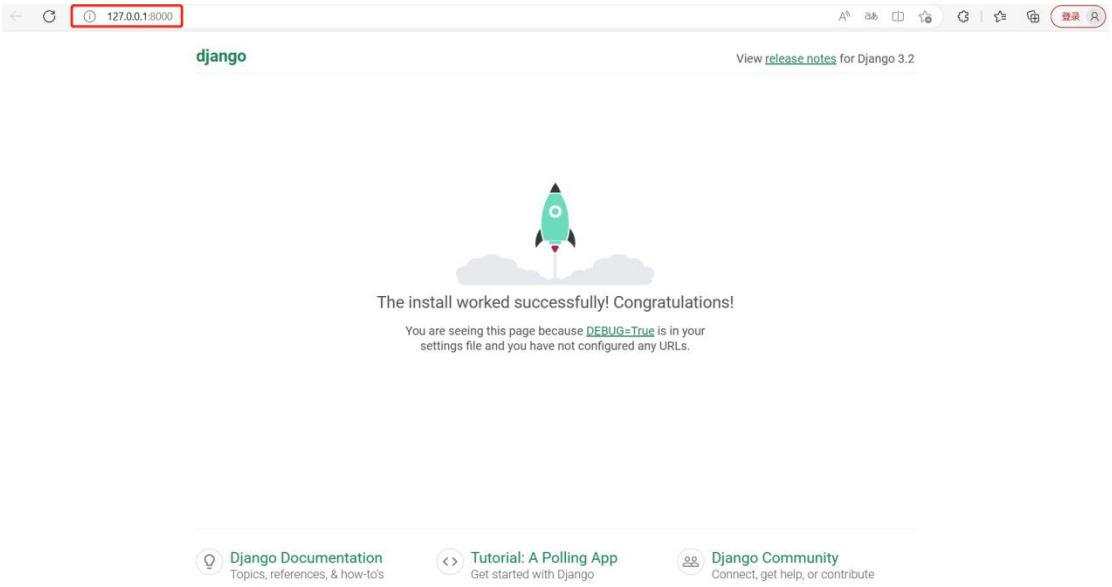
3.3. 完成后界面



3.4. 验证站点



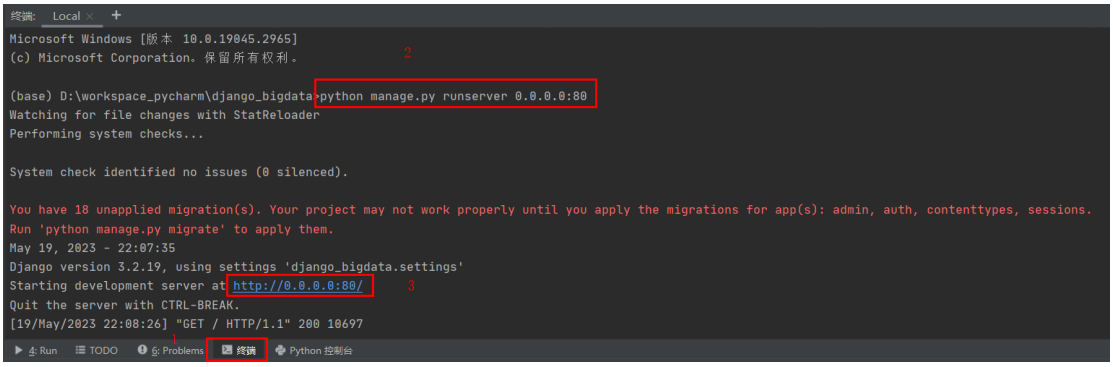
3.5. 大功告成

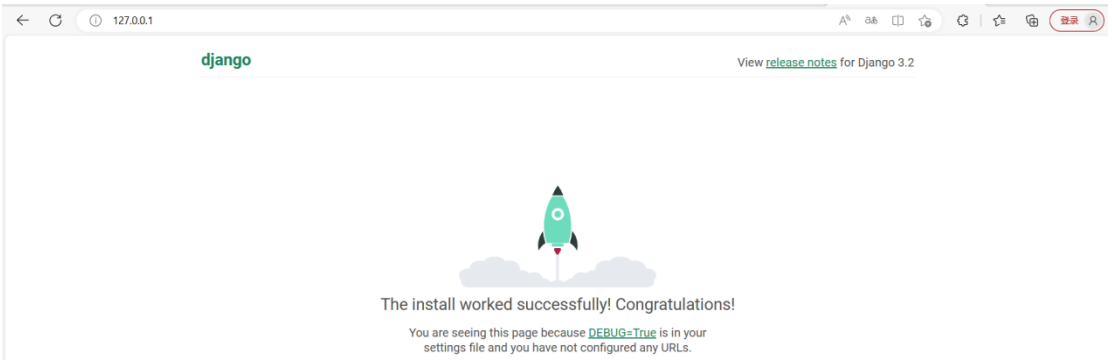


默认端口为 8000

3.6. 指定端口运行

```
python manage.py runserver 0.0.0.0:80
```





3.7. 扩展知识

0.0.0.0：这个地址事实上表示不确定地址，或“所有地址”、“任意地址”。

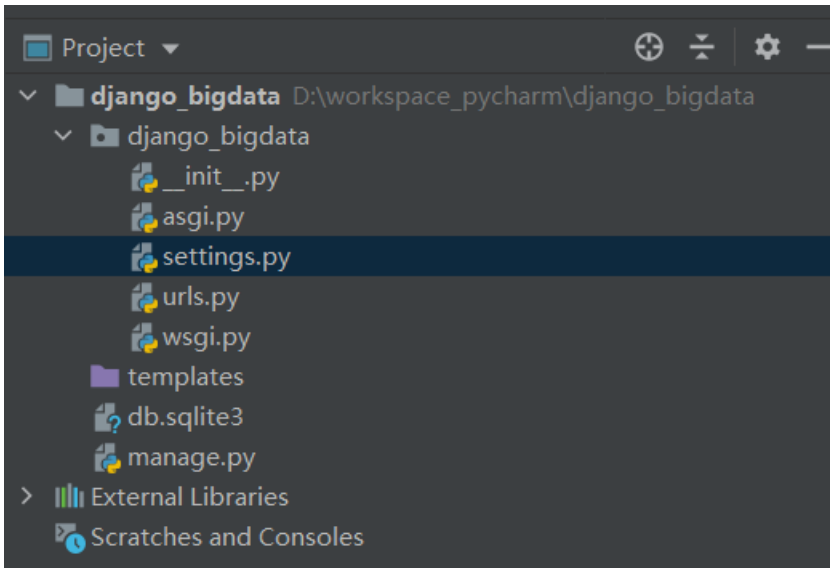
1、在服务器中，0.0.0.0 指的是本机上的所有 IPV4 地址，如果一个主机有两个 IP 地址，192.168.1.1 和 10.1.2.1，并且该主机上的一个服务监听的地址是 0.0.0.0 和端口 80,那么通过这两个<ip 地址:80>都能够访问该服务。

2、在路由中，0.0.0.0 表示的是默认路由，即当路由表中没有找到完全匹配的路由的时候所对应的路由。

3、runserver 127.0.0.1：表示将 socket 绑定到本机回环地址，只能监听本机对此服务的请求。

通俗点讲：通过本机上的任何一个 IP 地址：端口号都能访问的 Django 服务。

3.8. 项目结构介绍



- django_bigdata: 项目的容器。
- django_bigdata/__init__.py: 一个空文件，告诉 Python 该目录是一个 Python 包（python 包初始化文件）。
- django_bigdata/asgi.py: 异步网关协议接口，能够处理多种通用的协议类型，包括 HTTP，HTTP2 和 WebSocket，可以看成 ASGI 是 WSGI 的扩展。

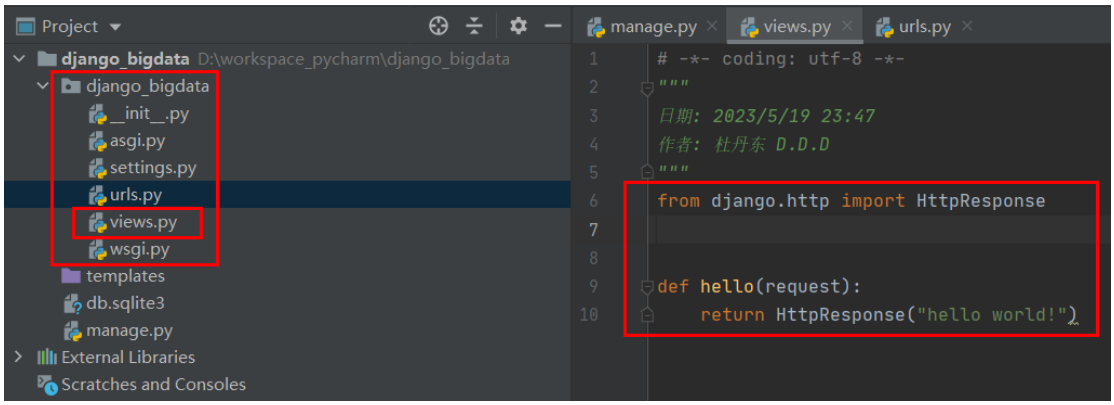
  (异步服务器网关接口规范，项目正式上线时需要使用）
- django_bigdata/settings.py: 该 Django 项目的设置/配置（项目配置文件），配置项名称必须是大写的。
- django_bigdata/urls.py: 该 Django 项目的 URL 声明; 一份由 Django 驱动的网站"目录"（项目的主路由配置，当 http 使用时优先调用的文件）
- django_bigdata/wsgi.py: Web 服务器网关接口（Python Web Server Gateway Interface 的缩写），Python 应用和 Web 服务器之间的一种接口，可以看成是一种协议、规范。它是基于 Http 协议的，不支持 WebSoket(Web 服务网关配置文件，项目正式上线时需要使用)
- templates 为空，用以包含响应的模板文件（存放 HTML 页面）。
- manage.py: 命令行工具(所有子命令的入口)，可让你以各种方式与 Django 进行交互。

4. Hello World

4.1. 创建 views.py

在第二个 django 目录下创建 views.py,并添加如下代码

```
from django.http import HttpResponse
def hello(request):
    return HttpResponse("hello world!")
```



4.2. 绑定 URL 与视图函数

4.2.1. 导包

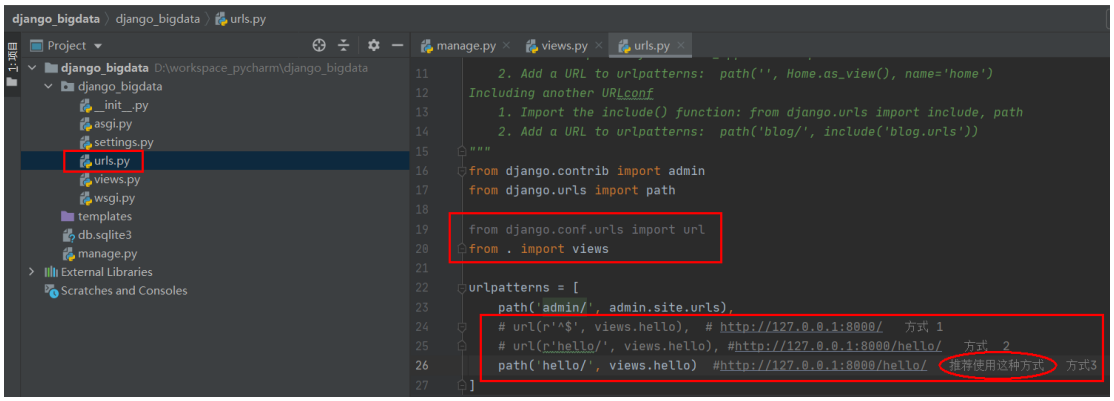
在 urls.py 中，添加如下代码

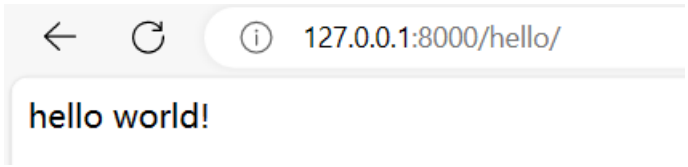
```
from django.conf.urls import url
from . import views
```

4.2.2. URL 映射

在 urls.py 中的 urlpatterns 中添加如下代码

```
# url(r'^$', views.hello), # http://127.0.0.1:8000/ 方式 1
# url(r'hello/', views.hello), # http://127.0.0.1:8000/hello/ 方式 2
path('hello/', views.hello) # http://127.0.0.1:8000/hello/ 推荐使用 方式 3
```





path(route, view, kwargs=None, name=None)

route: 字符串，表示 URL 规则（请求地址）。

view: 用于执行与正则表达式匹配的 URL 请求（处理方法）。

kwargs: 视图使用的字典类型的参数（携带的参数）。

name: 用来反向获取 URL，常用于 url 的反向解析（url 规则的名字）。

注意： 1、当 url 参数和 kwargs 参数冲突以 kwargs 参数为准。

2、项目中如果代码有改动，服务器会自动监测代码的改动并自动重新载入，所以如果你已经启动了服务器则不需手动重启。俗称“热部署”

至此，恭喜您已基本会使用 Django 框架了 √

5. 模板语法

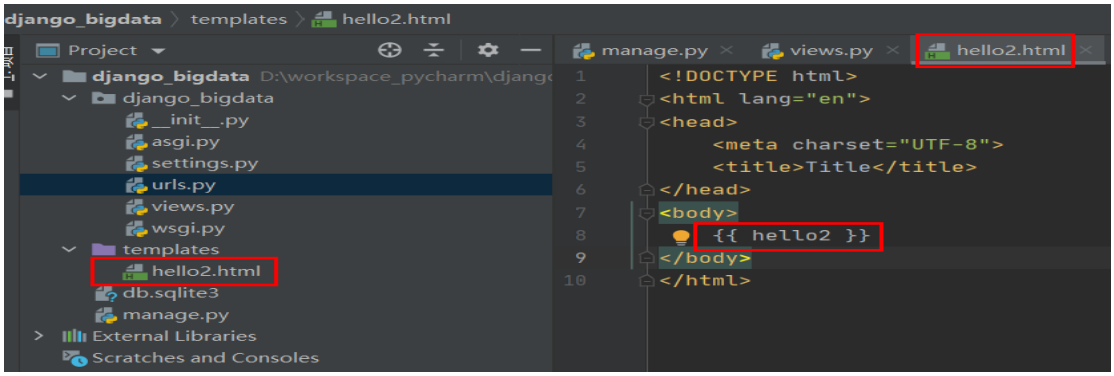
前面使用 django.http.HttpResponse()来输出"Hello World！"，将数据与视图混合在了一起，不符合 Django 的 MVC 思想。

5.1. 模板初识

5.1.1. 创建模板

在 templates 文件夹下创建 hello2.html，并写入如下代码：

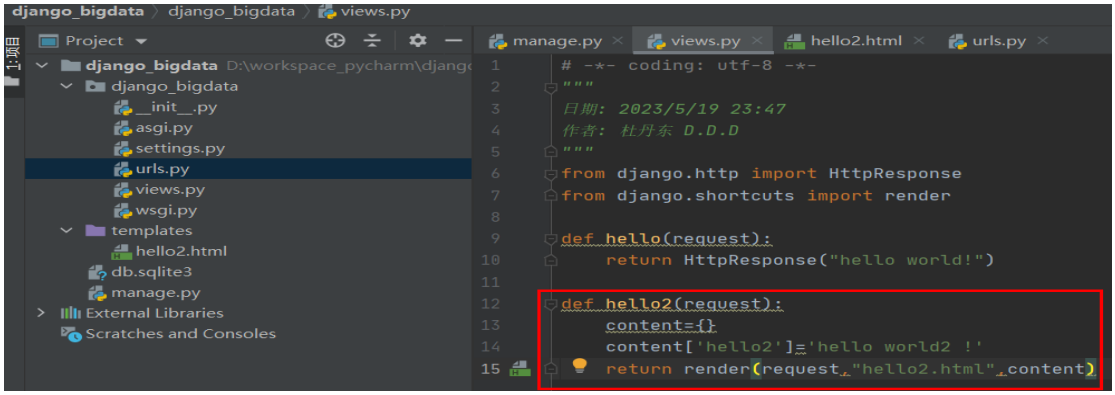
{{ hello2 }}



5.1.2. 编写处理方法

在 views.py 中添加请求方法，添加如下代码：

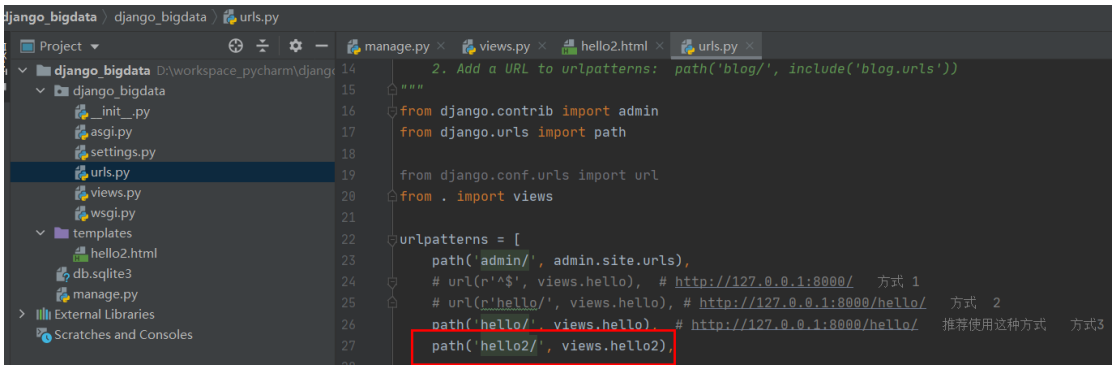
```
def hello2(request):
content={}
content['hello2']='hello world2 !'
    return render(request,"hello2.html",content)
```



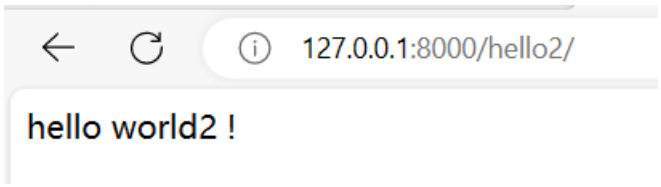
5.1.3. 配置路由

在 urls.py 中添加配置信息，代码如下：

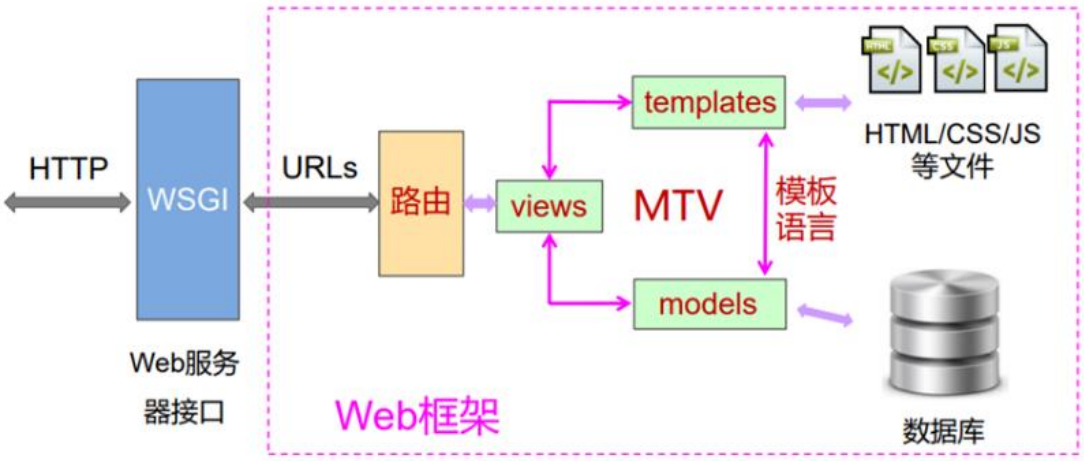
```
path('hello2/', views.hello2),
```



5.1.4. 验证结果



5.1.5. Django 运行原理



1、游览器输入 url。进入服务。

2、wsgi 拿到请求，*封装 socket，*按照 http 协议进行解包

3、解包之后的数据给所有的中间件按照顺序执行一遍。(如果中间件返回 HttpResponse,则直接返回，不走下面的流程)

4、中间件执行完成之后把 url 进行路由分发后映射到对应的 views。

5、然后 view 进行对应的逻辑处理(如果涉及到数据库，则要调用 model 进行操作)。最后返回 json

6、根据 response 进行对应页面渲染或者数据操作。(这里指的是服务端渲染 django 模板数据)

7、把响应数据给中间件执行(中间件请求和响应执行顺序和栈一样)

8、中间件执行完成之后，wsgi 按照 http 协议封装响应数据。返回给游览器，自此结束。

5.2. 模板语法

规则：字典类型

view(python)中：{“变量名”：“值”}

HTML(页面)中：{{ view 中的变量名 }}

5.2.1. 变量类型

例如：views.py 中

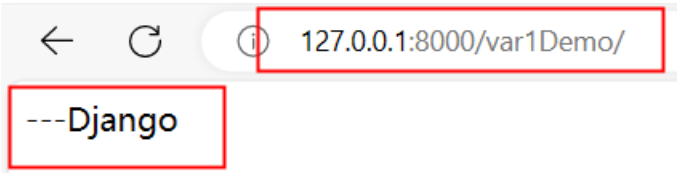
```
def var1Demo(request):
    var1 = 'Django'
    return render(request, 'hello2.html', {'var1': var1})
```

hello2.html 中

```
{{ hello2 }} ---{{ var1 }}
```

urls.py 中:

```
path('var1Demo/', views.var1Demo),
```



5.2.2. 列表类型

views.py:

```
def listDemo(request):
    list1 = ['django0', 'django1', 'django2', 'django3']
    return render(request, 'hello2.html', {'list1': list1})
```

hello2.html:

```
list : {{ list1 }}---{{ list1.1 }}
```

urls.py:

```
path('listDemo/', views.listDemo),
```

5.2.3. 字典类型

views.py：

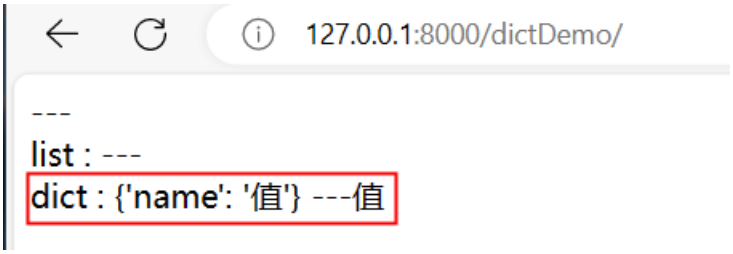
```
def dictDemo(request):
    dict1 = {'name': '值'}
    return render(request, 'hello2.html', {'dict1': dict1})
```

hello.html：

```
dict : {{ dict1 }} ---{{ dict1.name}}
```

urls.py:

```
path('dictDemo/', views.dictDemo),
```

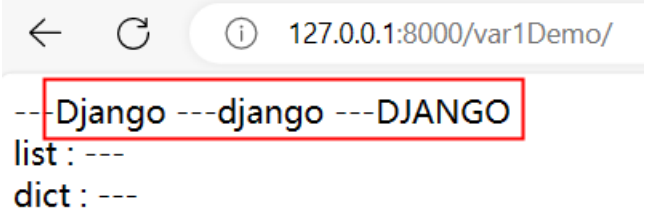


5.2.4. 过滤

1、lower：转小写

2、upper：转大写

```
{{ hello2 }} ---{{ var1 }}  ---{{ var1|lower }} ---{{ var1|upper }}<br/>
```



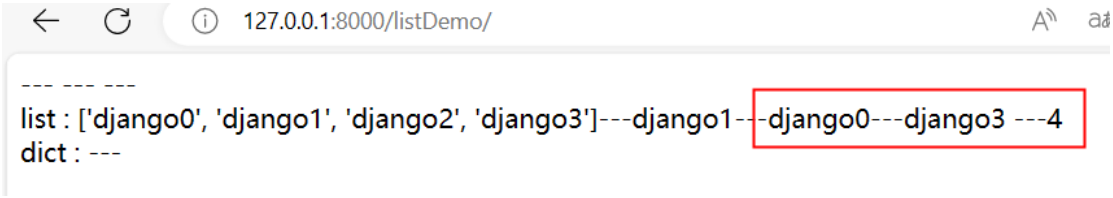
3、first:第一个

4、last：最后一个

5、length：长度

6、truncatewords：前 N 个或字符串的前 N 个字母

```
{{ list1 }}---{{ list1.1 }}---{{ list1|first }}---{{ list1|last }} ---
{{ list1|length }} --{{ list1|truncatewords:'2' }}<br/>
```



7、Date ： 日期格式化

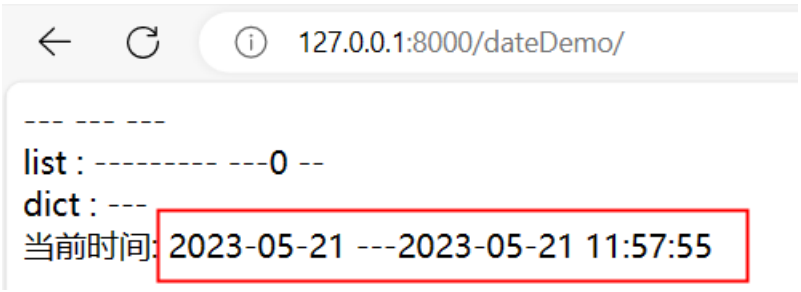
```
def dateDemo(request):
    import datetime
    now = datetime.datetime.now()
    return render(request, 'hello2.html', {'now': now})
```

页面输出时格式化

```
当前时间: {{ now|date:'Y-m-d' }} ---{{ now|date:'Y-m-d h:i:s' }}
```

配置路由

```
path('dateDemo/', views.dateDemo),
```



8、 **filesizeformat ： 文件大小格式化**

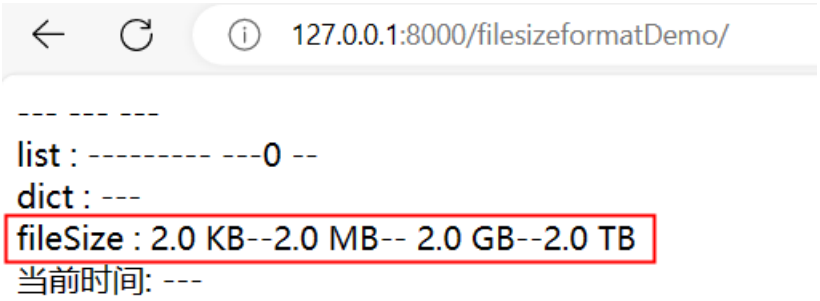
```
def filesizeformatDemo(request):
    fileSize1 = 1024 * 2  # 2K
    fileSize2 = 1024 * 1024 * 2  # 2M
    fileSize3 = 1024 * 1024 * 1024 * 2  # 2G
    fileSize4 = 1024 * 1024 * 1024 * 1024 * 2  # 2T
    fileSizeList = [fileSize1, fileSize2, fileSize3, fileSize4]
    return render(request, 'hello2.html', {'fileSizeList': fileSizeList})
```

页面进行格式化

```
fileSize : {{ fileSizeList.0|filesizeformat }}--{{ fileSizeList.1|filesizeformat }}--{{ fileSizeList.2|filesizeformat }}--{{ fileSizeList.3|filesizeformat }}<br/>
```

配置路由

```
path('filesizeformatDemo/', views.filesizeformatDemo),
```



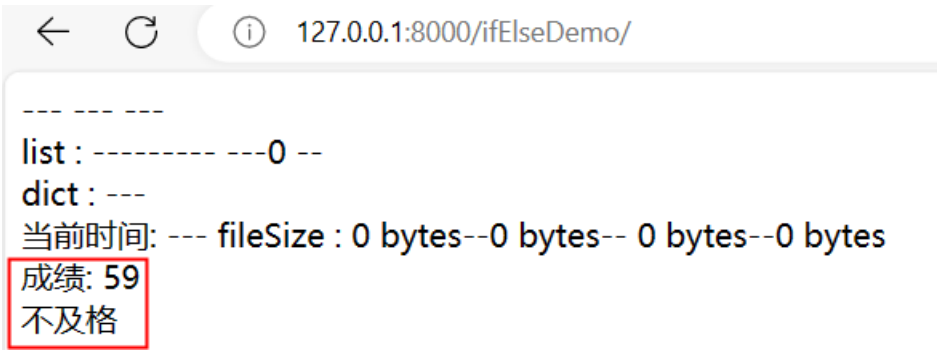
9、if else ： 条件判断

```
def ifElseDemo(request):
    import random
    num = random.randint(1, 100)  # 1-100 的随机数
    return render(request, 'hello2.html', {'num': num})
```

不要忘记 if 的结束标签 endif

```
成绩: {{ num }}<br/>
    {% if num > and num < 90 %} 良好
    {% elif num >= 70 and num < 80 %} 中等
    {% elif num >= 60 and num < 70 %} 及格
    {% else %} 不及格
    {% endif %}
```

```
path('ifElseDemo/', views.ifElseDemo),
```



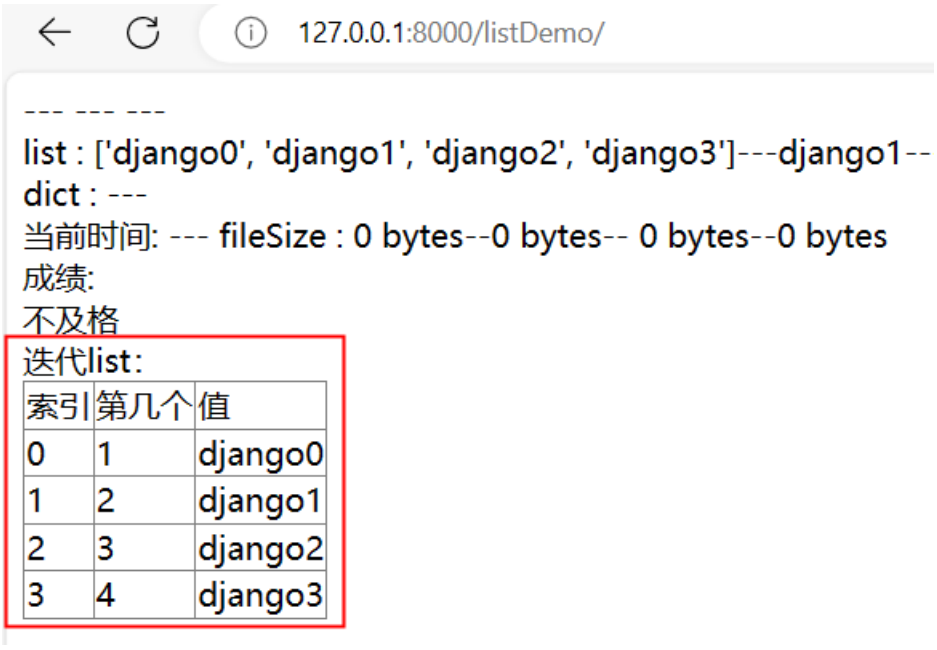
10、for：循环（遍历）

给表格添加样式

```
<style>
    table,tr,td{border: 1px solid gray;border-collapse: collapse;}
</style>
```

1、循环表格和列表，不要忘记 for 的结束标签 endfor

```
<table>
            <td>索引</td><td>第几个</td><td>值</td></tr>
{#            {% for value in list1 %}#}
              {% for value in list1 reversed %}    {# 倒序 #}
                <tr>
                    <td>{{ forloop.counter0 }}</td>
                    <td>{{ forloop.counter }}</td>
                    <td>{{ value }}</td>
                </tr>
            {% endfor %}
        </table>
```



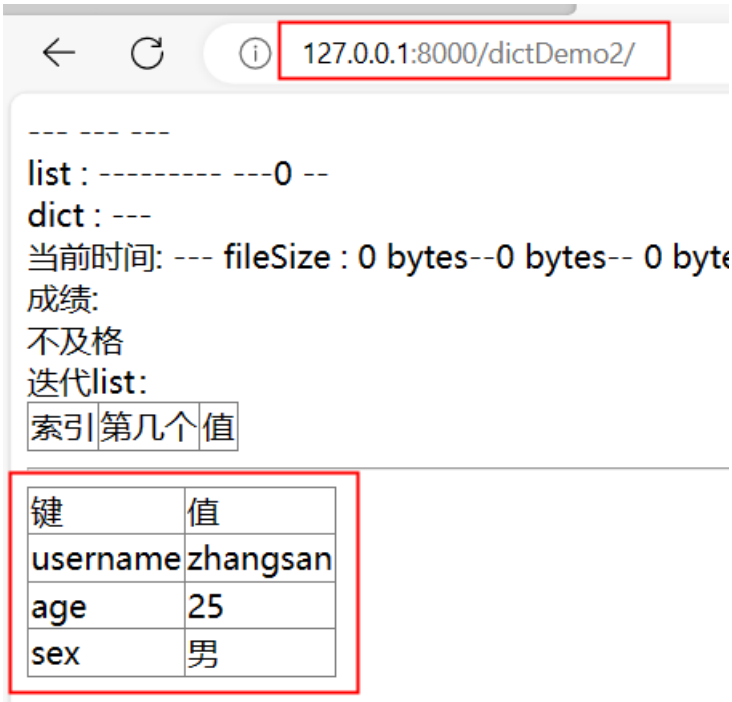
2、迭代字典

```
def dictDemo2(request):
    userInfo = {'username': 'zhangsan', 'age': '25', 'sex': '男'}
    return render(request, 'hello2.html', {'userInfo': userInfo})
```

```
<hr/>
<table>
    <td>键</td><td>值</td></tr>
    {% for key,value in userInfo.items %}
        <tr>
            <td>{{ key }}</td>
            <td>{{ value }}</td>

        </tr>
    {% endfor %}
</table>
```

配置路由

```
path('dictDemo2/', views.dictDemo2),
```

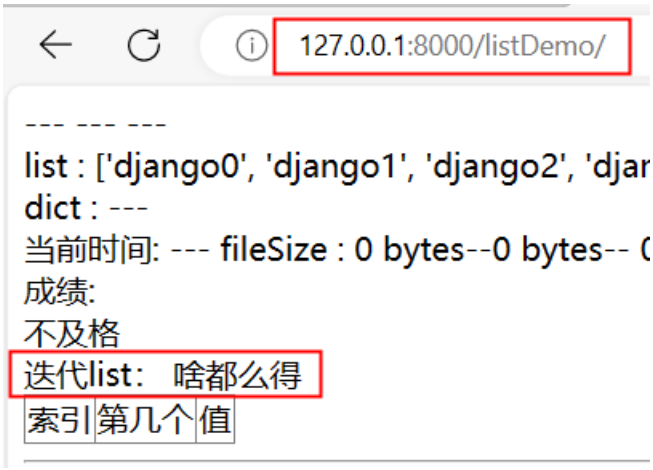


扩展知识：

- forloop.counter：顺序获取循环序号，从 1 开始计算
- forloop.counter0：顺序获取循环序号，从 0 开始计算
- forloop.revcounter：倒序获取循环序号，结尾序号为 1
- forloop.revcounter0：倒序获取循环序号，结尾序号为 0
- forloop.first（一般配合 if 标签使用）：第一条数据返回 True，其他数据返回 False
- forloop.last（一般配合 if 标签使用）：最后一条数据返回 True，其他数据返回 False

11、empty：判空

```
<table>
    <td>索引</td><td>第几个</td><td>值</td></tr>
      {% for value in list1 %}#}
      {% for value in list2 reversed %}    {# 倒序 #}
        <tr>
            <td>{{ forloop.counter0 }}</td>
            <td>{{ forloop.counter }}</td>
            <td>{{ value }}</td>
        </tr>
      {% empty %}  啥都么得
    {% endfor %}
</table>
```



注意：判空分 2 种

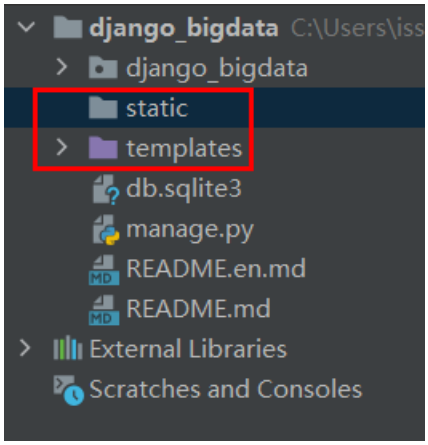
①对象本身为空

②对象本身就不存在（如本例--故意写错）

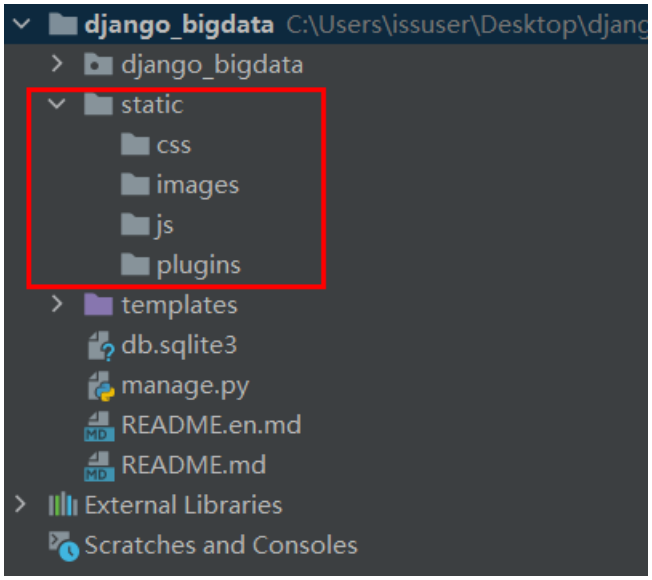
5.3. 静态资源文件访问

5.3.1. 创建 static 文件夹

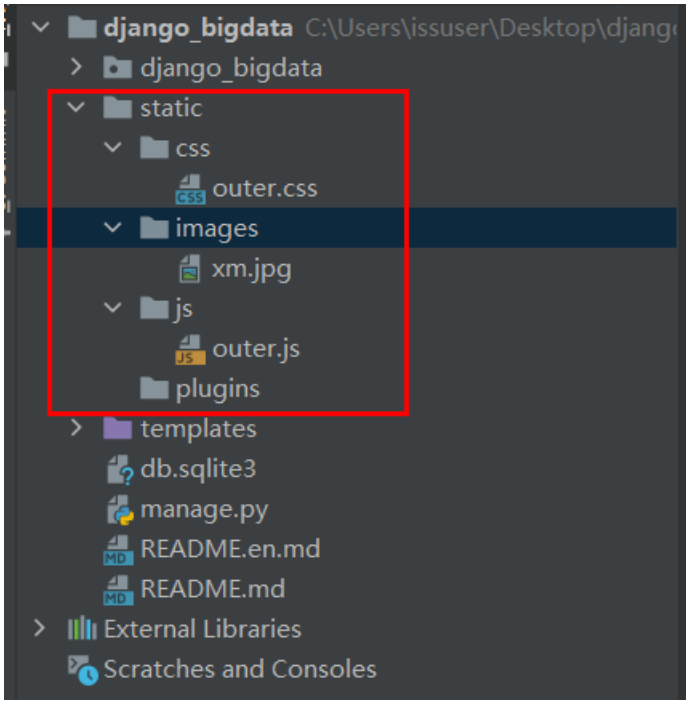
在 tamplates 同级目录创建 static 文件夹



5.3.2. 分别创建静态资源文件存放的文件夹



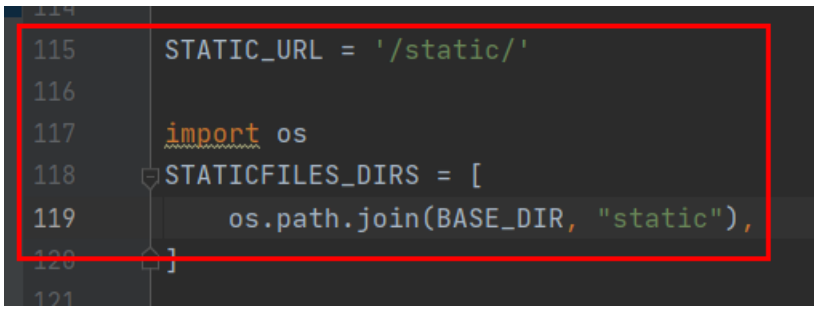
5.3.3. 给对应文件夹放入文件



5.3.4. 引入文件测试

settings.py 中加入如下代码：

```
import os
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
]
```



hello2.html 页面中引入对应文件

```
#引入 css 文件

<link rel="stylesheet" href="/static/css/outer.css">

#引入 js 文件

<script src="/static/js/outer.js"></script>
```

```
#最好在 body 第一行加载静态资源文件
{% load static %}
#引入标签测试
<span>测试</span> <img src=" {% static 'images/xm.jpg' %}" width="100px"
height="100px">
```

5.4. ORM

5.4.1. 优点

1、提高开发效率。

2、不同数据库可以平滑切换。

5.4.2. 缺点

1、ORM 代码转换为 SQL 语句时，需要花费一定的时间，执行效率会有所降低。

2、长期写 ORM 代码，会降低编写 SQL 语句的能力。

5.4.3. ORM 解析过程

1、ORM 会将 Python 代码转成为 SQL 语句。

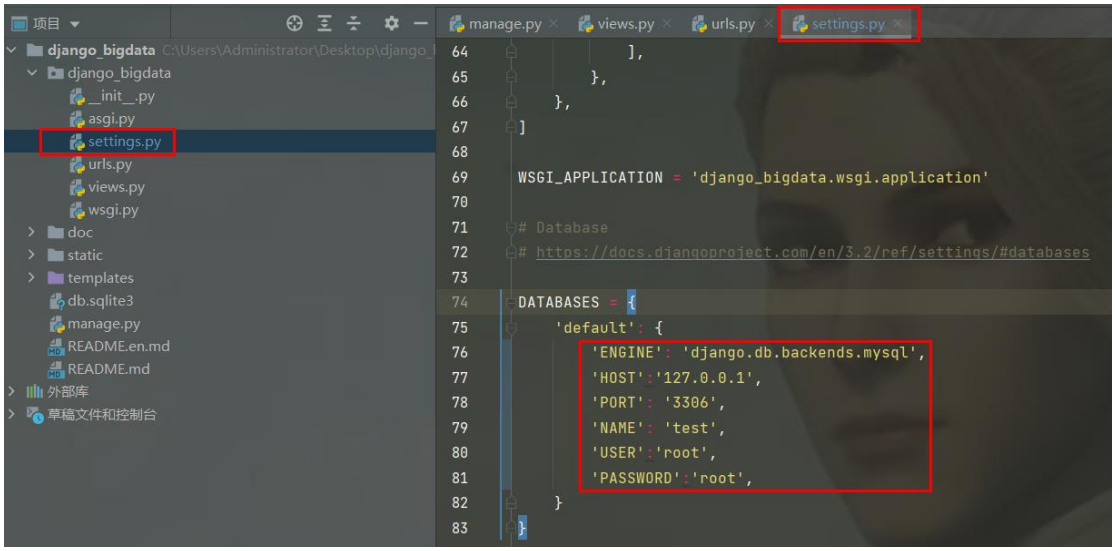
2、SQL 语句通过 pymysql 传送到数据库服务端。

3、在数据库中执行 SQL 语句并将结果返回。

5.4.4. 数据库配置

5.4.4.1. settings.py 中修改 DATABASE 的配置项

```
'ENGINE': 'django.db.backends.mysql',
'HOST':'127.0.0.1',
'PORT': '3306',
'NAME': 'test',
'USER':'root',
'PASSWORD':'root',
```



5.4.4.2. 初始化数据库

与 settings.py 同级目录的__init__.py 中添加如下代码：

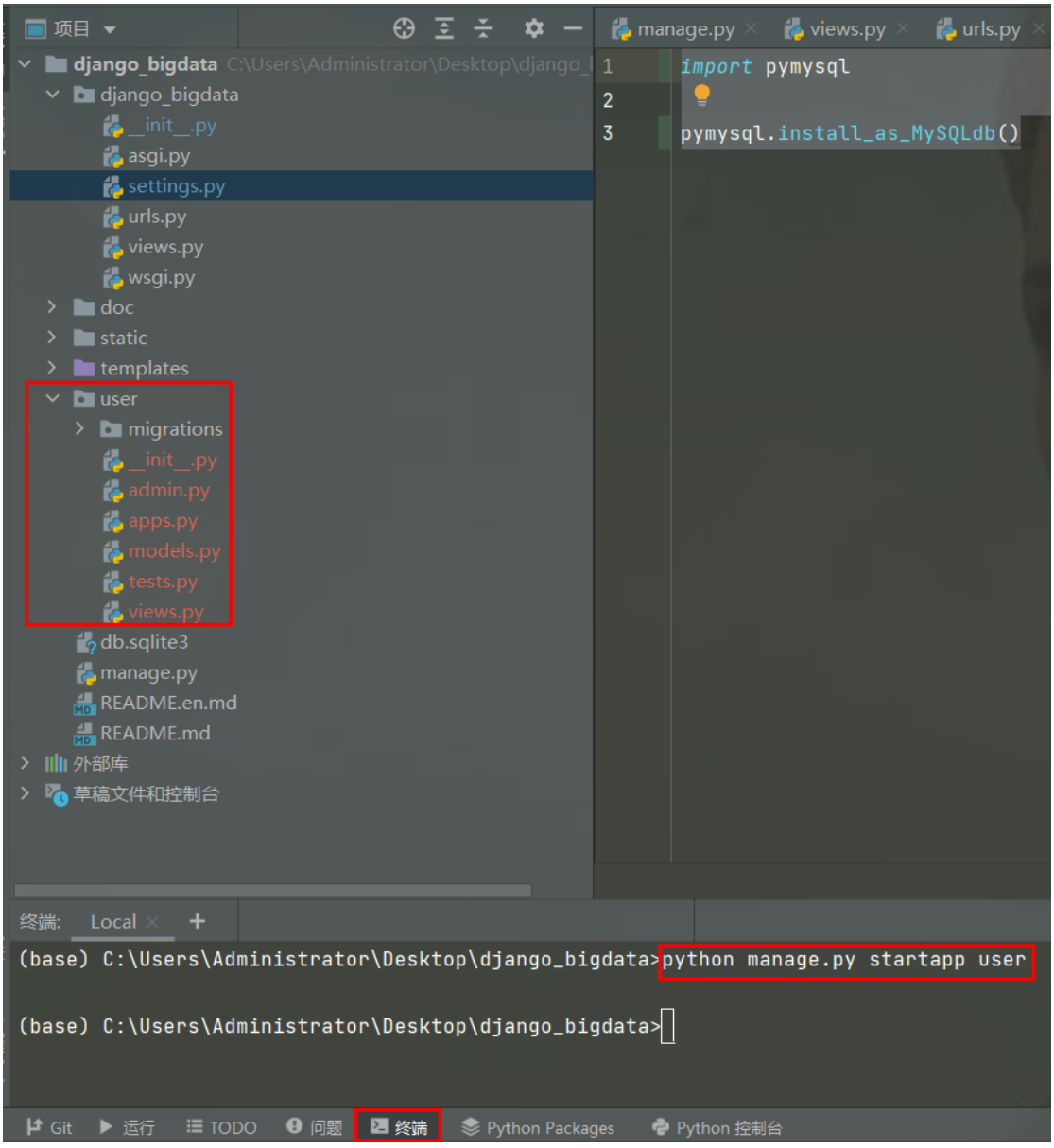
```
import pymysql

pymysql.install_as_MySQLdb()
```

5.5. 定义模型

5.5.1. 创建 app

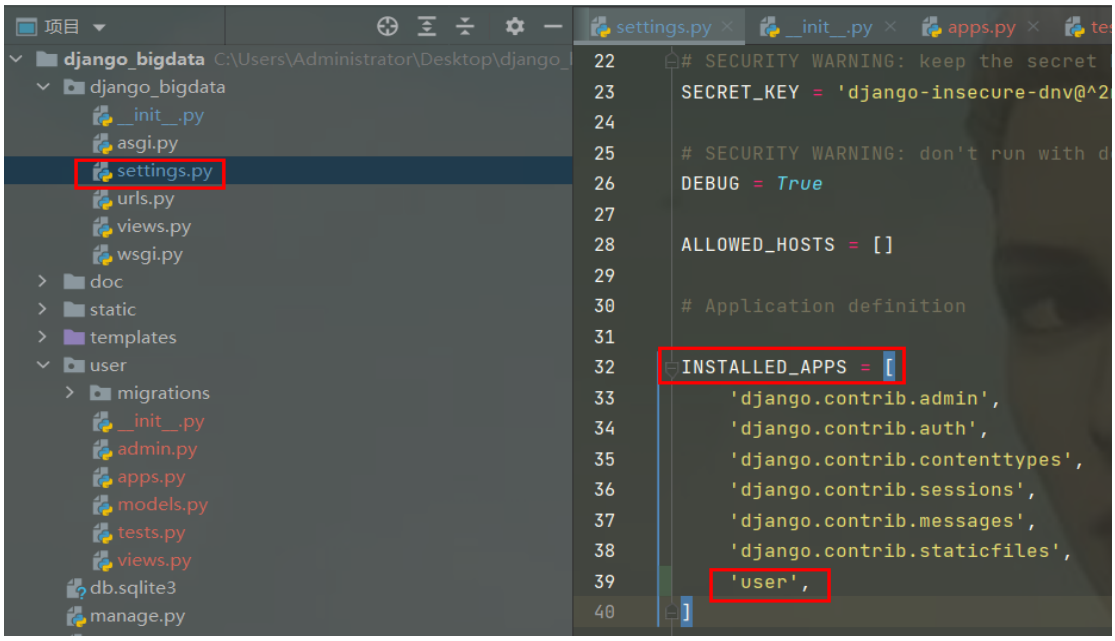
Django 规定，如果要使用模型，必须要创建一个 app。我们使用以下命令创建一个 user 的 app

```
python manage.py startapp user
```



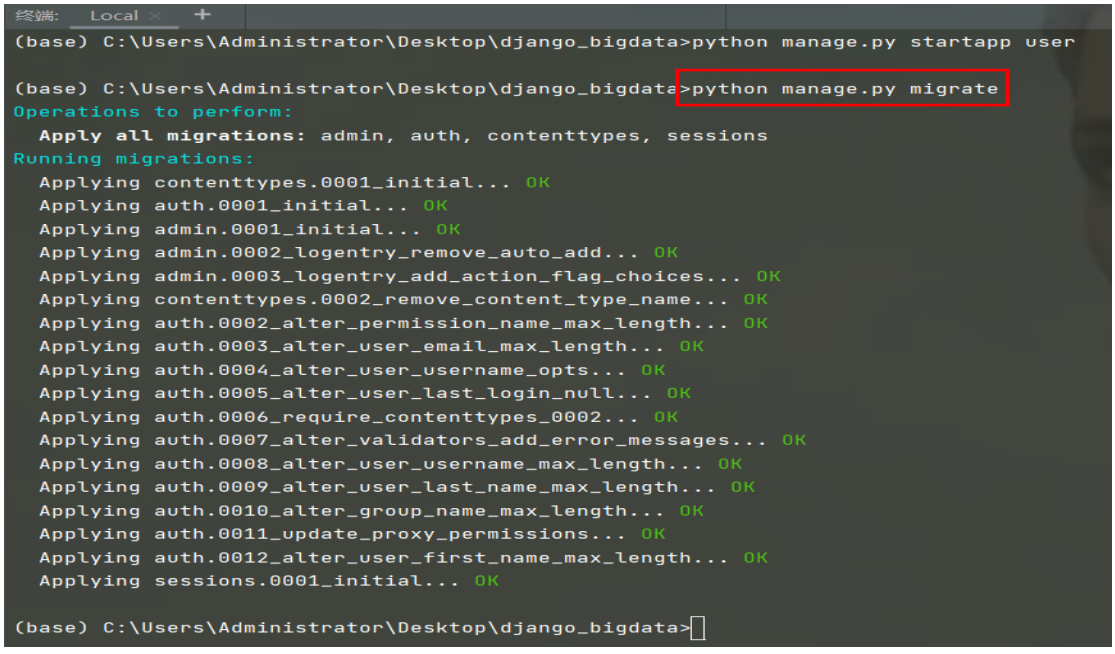
5.5.2. 配置 app

settings.py 的 INSTALLED_APPS 中配置 app



5.5.3. 创建表结构

创建表结构：python manage.py migrate （所有 app（模块）---包括系统自带的）



注意 ： 如果执行即便还是报错，就在后面加—fake(忽略已存在的表)：如下

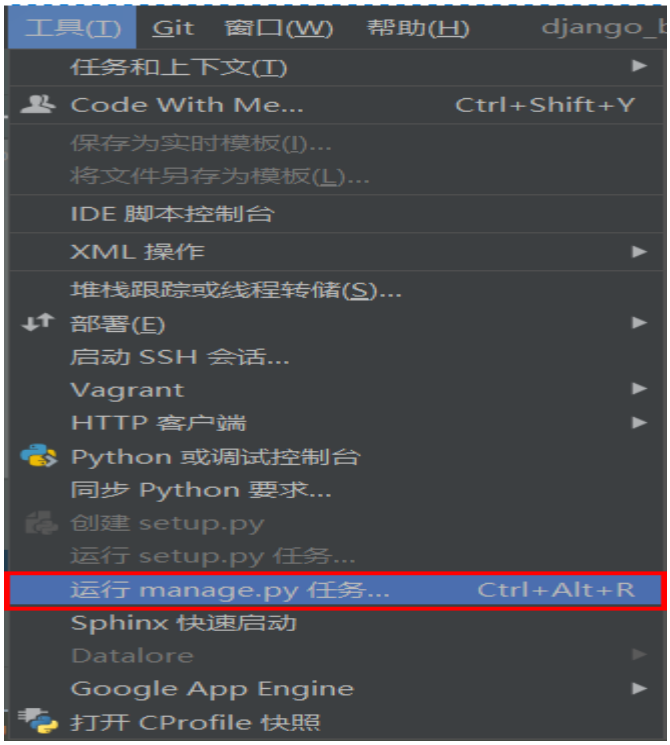
python manage.py migrate --fake

注意： 1、表名组成结构为：应用名_类名（如：user_users）。

2、无论是生成表还是自己创建表都要遵循这个规则。

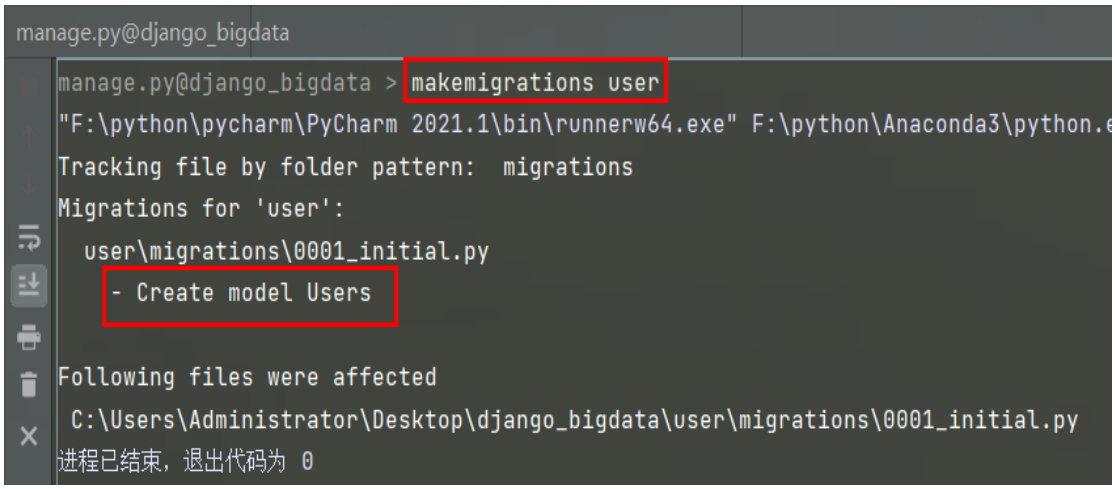
3、只要修改了 models.py 就必须重新执行此命令。

5.5.4. 只添加某一个模块



python manage.py migrate user　告诉框架模型有变

makemigrations user　　创建表结构



5.5.5. 测试

1、在 user 模块下的 models.py 中添加添加用户的代码：

```
from django.db import models

class Users(models.Model):
    userid = models.AutoField(primary_key=True)
    username = models.CharField(max_length=32, unique=True)
    password = models.CharField(max_length=32)
    sex = models.CharField(max_length=4, default='男')
    nc = models.CharField(max_length=32)
    roleId = models.IntegerField(default='3')
    state = models.CharField(max_length=16)  # 数据库里已经设置，这里测试不设置可以不
    img = models.CharField(max_length=128)
```

注意:类名代表表名，userid 等代表字段名称

2、在 user 模块下的 views.py 中添用户添加的代码：

```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse
from user.models import Users

def testAdd(request):
    print('添加...')
    user = Users(username='令狐冲 4', password='dgjj')
    user.save()
    return HttpResponse("添加成功!")
```

3、urls.py 中配置路由

```
# 地址栏拼参
path('user/testAdd/', user.views.testAdd),
```

4、结果

自此，说明 Django 的 ORM 框架已经配置完成

6. 案例

用户管理系统

Post 请求时报错（跨域请求保护），页面中加入

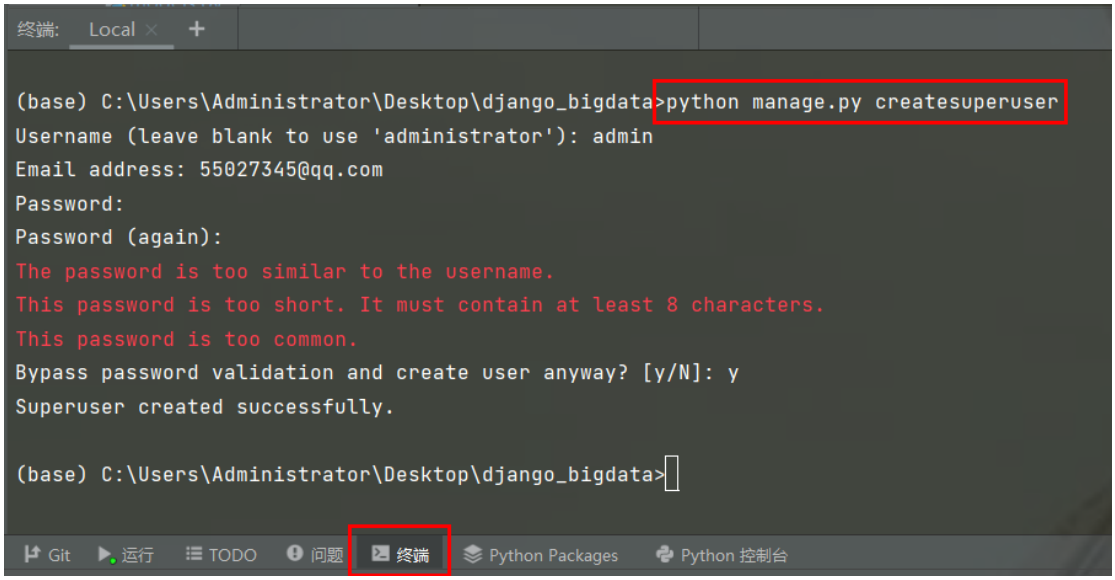
```
{% csrf_token %}    <!--加入这行 -->
```

或者注释掉配置文件中的东西

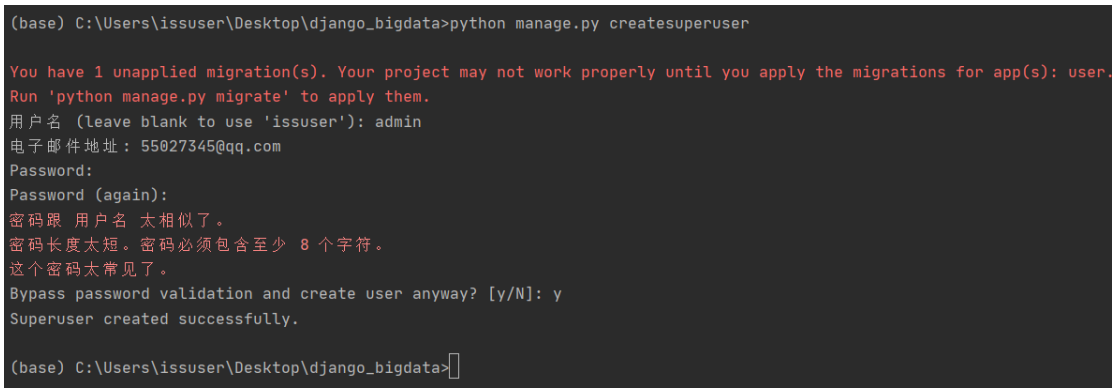
7. 后台管理系统

7.1. 初始化超管账号

python manage.py createsuperuser

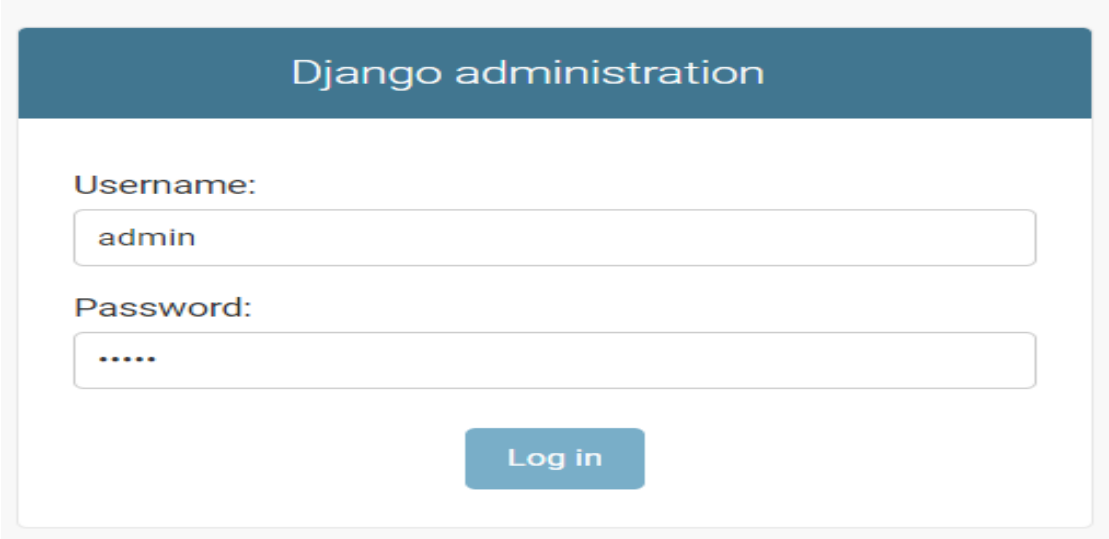


汉化后再初始化，提示信息变中文了

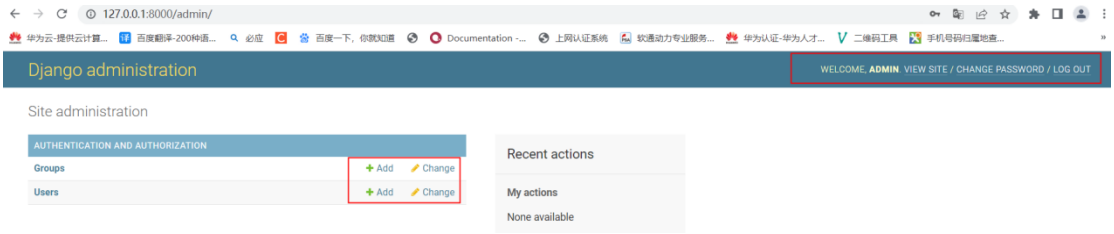


7.2. 进入后台

地址栏输入：http://127.0.0.1:8000/admin，输入用户名和密码，点击登录



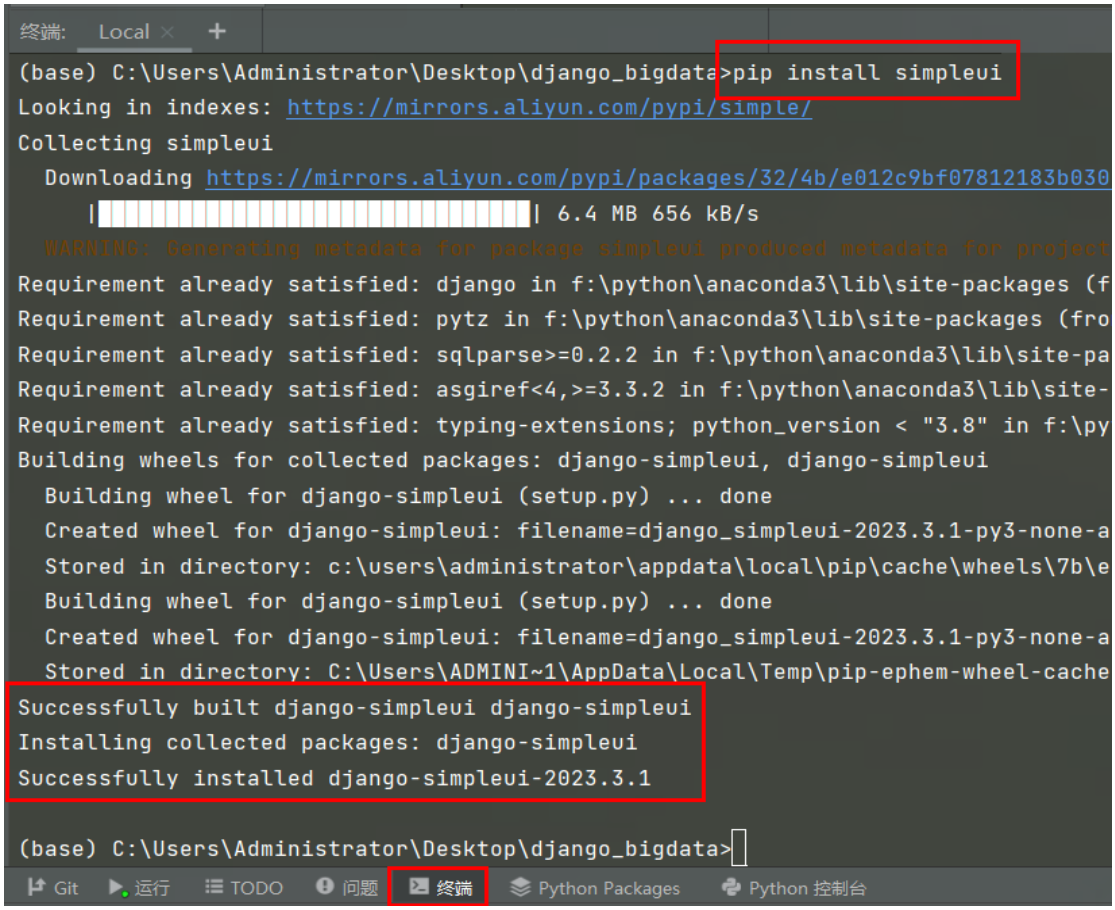
7.3. 后台界面预览



7.4. 美化后台

7.4.1. 安装 UI 包

pip install simpleui



pip install django-simpleui　这个也可以，不过网络好像不太行（实在不行用下面的）

如果下载超时

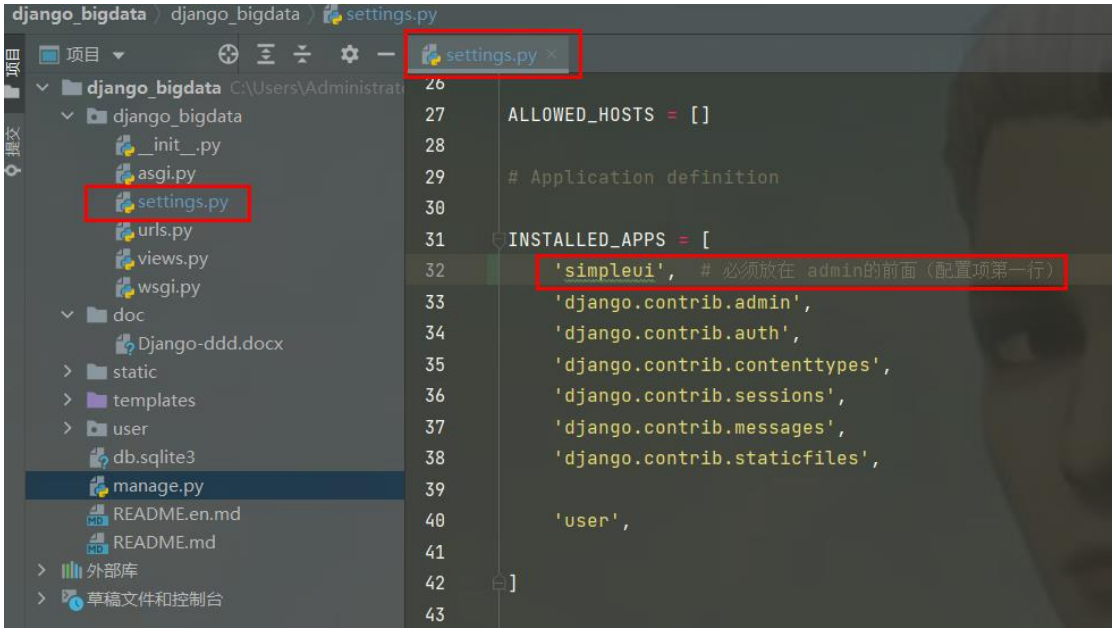
使用如下命令：

pip --default-timeout=1688 install simpleui -i http://pypi.douban.com/simple/ --trusted-host pypi.douban.com

或

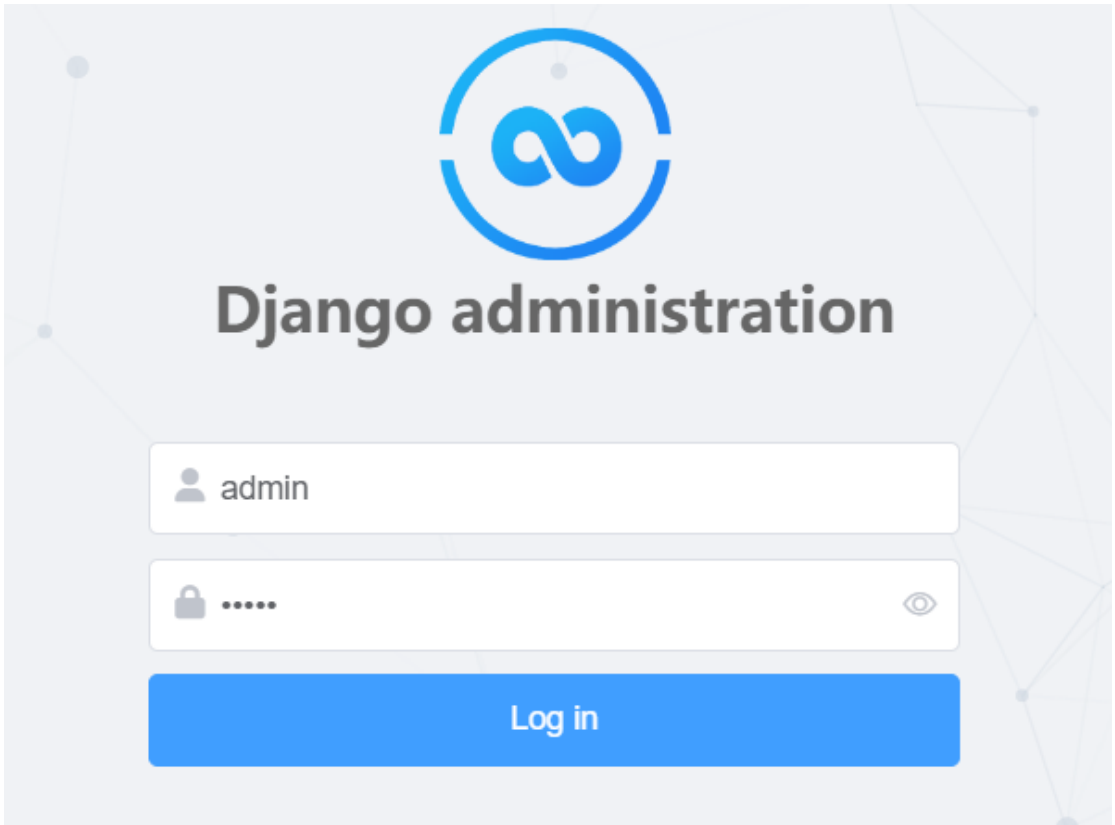
pip install -i http://pypi.doubanio.com/simple/ --trusted-host pypi.doubanio.com simpleui

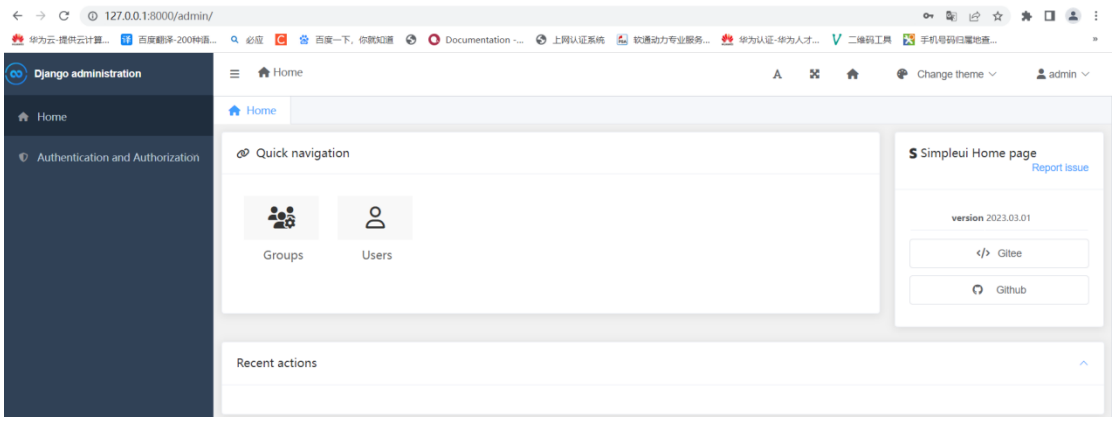
7.4.2. 将 simpleui 加入 settings.py 中



如果样式没有正常显示：python manage.py collectstatic

7.5. 重新进入页面





7.6. 汉化并修改图标及标题

7.6.1. 在 settings.py 中添加如下配置

```
# 更改默认语言为中文
LANGUAGE_CODE = 'zh-hans'
TIME_ZONE = 'Asia/Shanghai'
USE_TZ = False # 这里务必调整为 False，否则时区设置无效

# 去掉默认 Logo 或换成自己 Logo 链接
SIMPLEUI_LOGO = '/static/images/xm.jpg'
#下载时将上一个配置注释掉
STATIC_ROOT = os.path.join(BASE_DIR, "static")
SIMPLEUI_STATIC_OFFLINE = True # 离线模式
```

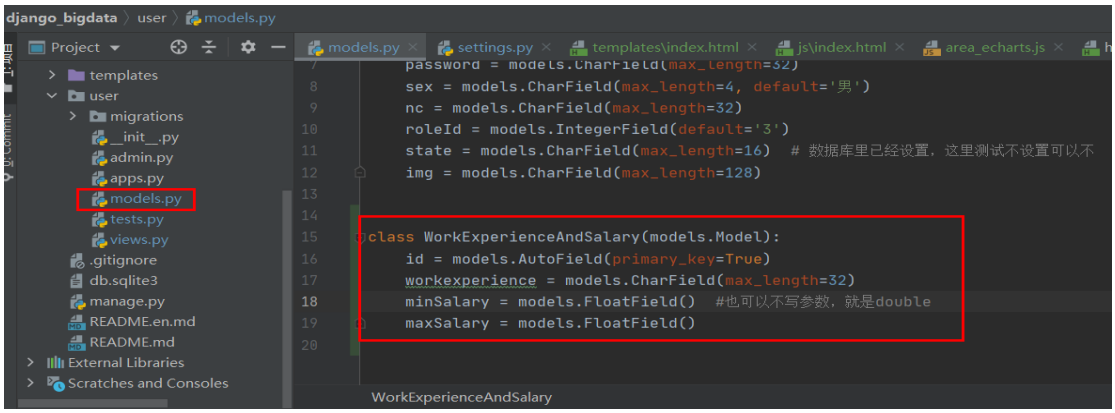
7.6.2. 下载时需要注释，完成后再把追后两行注释掉

```
import os

# STATICFILES_DIRS = [
#     os.path.join(BASE_DIR, "static"),
# ]

# 更改默认语言为中文
LANGUAGE_CODE = 'zh-hans'
TIME_ZONE = 'Asia/Shanghai'
USE_TZ = False # 这里务必调整为False，否则时区设置无效

# 去掉默认Logo或换成自己Logo链接
SIMPLEUI_LOGO = '/static/images/xm.jpg'
#下载时将上一个配置注释掉
STATIC_ROOT = os.path.join(BASE_DIR, "static")
SIMPLEUI_STATIC_OFFLINE = True # 离线模式
```

8. 项目

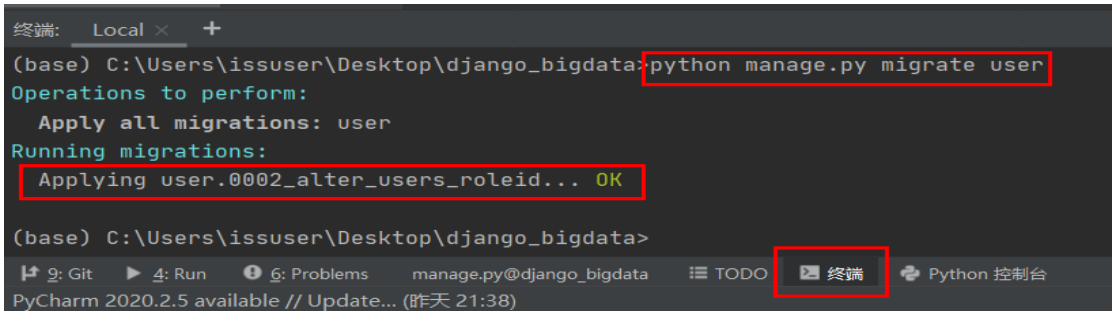
8.1. 创建 models

创建数据库中表对应的类



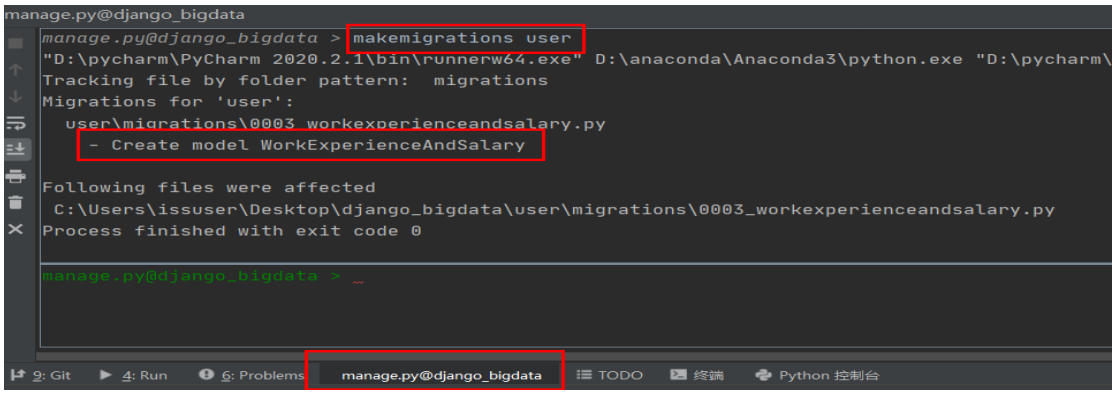
8.2. 迁移模块

python manage.py migrate user



8.3. 创建每张表

makemigrations user



8.4. 创建请求方法

在 user 文件夹下的 views.py 中创建请求方法如下：

```
def getWorkExperienceAndSalary(request):
    print('getWorkExperienceAndSalary...\n')
    weasList = WorkExperienceAndSalary.objects.all()  # ORM 调用，返回 QuerySet--list
    print(weasList.values())  #QuerySet --json 要双引号，结果里面是单引号
    res=querySetToJson(weasList.values()) # 封装 querySet 转 json 的方法
    print(res)
    return HttpResponse(res, content_type='application/json')
```

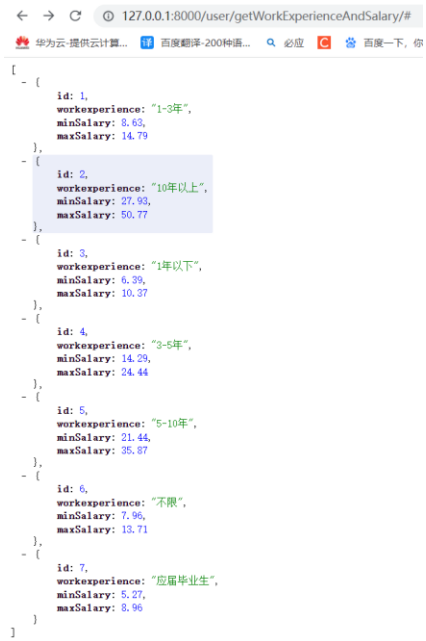
注意：Django 框架查询的结果是 QuerySet(相当于是 list)，python 中的列表内容是单引号“’”，json 中需要的是双引号，所以必须进行类型转换，这里采用的是自己封装方法进行转换。

8.5. 配置路由

在 urls.py 中创建请求路径

```
# 项目
path('user/getWorkExperienceAndSalary/', user.views.getWorkExperienceAndSalary),
```

8.6. 测试



8.7. 大屏

8.7.1. 编写 models.py

```
from django.db import models


class Users(models.Model):
    userid = models.AutoField(primary_key=True)
    username = models.CharField(max_length=32, unique=True)
    password = models.CharField(max_length=32)
    sex = models.CharField(max_length=4, default='男')
    nc = models.CharField(max_length=32)
    roleId = models.IntegerField(default='3')
    state = models.CharField(max_length=16)  # 数据库里已经设置，这里测试不
设置可以不
    img = models.CharField(max_length=128)


class WorkExperienceAndSalary(models.Model):  # 工作经验和工资
    id = models.AutoField(primary_key=True)
    workexperience = models.CharField(max_length=32)
    minSalary = models.FloatField()  # 也可以不写参数，就是 double
    maxSalary = models.FloatField()


class EducationAndSalary(models.Model):  # 学历和工资
    id = models.AutoField(primary_key=True)
    education = models.CharField(max_length=32)
    minSalary = models.FloatField()
    maxSalary = models.FloatField()


class PostnameAndPostnum(models.Model):  # 岗位名称和岗位总数
    # 没有 id 列时，Django 无法工作，解决办法：随便选择一列设置成主键即可
    name = models.CharField(primary_key=True, max_length=32)
    value = models.IntegerField()


class WelfareAndNum(models.Model):  # 福利与数量
    name = models.CharField(primary_key=True, max_length=32)
    value = models.IntegerField()
```

```
class TotalPostAnd51post(models.Model):  # 岗位总数和包含五险一金的岗位总数
    id = models.AutoField(primary_key=True)
    totalPost = models.IntegerField()
    post51 = models.IntegerField()


class CityAndPostTotal(models.Model):  # 每个省份岗位总数
    name = models.CharField(primary_key=True, max_length=128)
    value = models.IntegerField()
```

8.7.2. 编写 views.py

```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse
from user.models import Users, WorkExperienceAndSalary,
EducationAndSalary, PostnameAndPostnum, WelfareAndNum, \
    TotalPostAnd51post, CityAndPostTotal
from utils import querySetToJson

def testAdd(request):
    print('添加...\n')
    user = Users(username='令狐冲 5', password='dgjj')
    user.save()  # 不是 ORM
    return HttpResponse("添加成功!")


def getWorkExperienceAndSalary(request):
    print('getWorkExperienceAndSalary...\n')
    weasList = WorkExperienceAndSalary.objects.all()  # ORM 调用，返回
QuerySet--list
    print(weasList.values())  #QuerySet --json 要双引号，结果里面是单引号
    res=querySetToJson(weasList.values()) # 封装 querySet 转 json 的方法
    print(res)
    return HttpResponse(res, content_type='application/json')

def getEducationAndSalary(request):
    print('getEducationAndSalary...\n')
    easList = EducationAndSalary.objects.all()
    res = querySetToJson(easList.values())
    return HttpResponse(res, content_type='application/json')
```

```
def getWelfareAndNumTop10(request):
    print('getWelfareAndNumTop10...\n')
    wanList=WelfareAndNum.objects.all()
    res = querySetToJson(wanList.values())
    return HttpResponse(res, content_type='application/json')


def getPostnameAndPostNumTop10(request):
    print('getPostnameAndPostNumTop10...\n')
    papList=PostnameAndPostnum.objects.all()
    res = querySetToJson(papList.values())
    return HttpResponse(res, content_type='application/json')


def getTotalOfPostAnd51OfPost(request):
    print('getTotalOfPostAnd51OfPost...\n')
    tpa5List=TotalPostAnd51post.objects.all()
    res = querySetToJson(tpa5List.values())
    return HttpResponse(res, content_type='application/json')


def getPostTotalByCity(request):
    print('getPostTotalByCity...\n')
    captList = CityAndPostTotal.objects.all()
    res = querySetToJson(captList.values())
    return HttpResponse(res, content_type='application/json')
```

8.7.3. 配置所有页面路由

```
# 项目
path('user/getWorkExperienceAndSalary/', user.views.getWorkExperienceAndSalary),
path('user/getEducationAndSalary/', user.views.getEducationAndSalary),
path('user/getWelfareAndNumTop10/', user.views.getWelfareAndNumTop10),
path('user/getPostnameAndPostNumTop10/', user.views.getPostnameAndPostNumTop10),
path('user/getTotalOfPostAnd51OfPost/', user.views.getTotalOfPostAnd51OfPost),
path('user/getPostTotalByCity/', user.views.getPostTotalByCity),
```

8.7.4. 最终效果


127.0.0.1:8000
全国招聘信息可视化大屏
2023年5月28日 1时49分1秒
工作年限与薪资
1-3年
10年以上
1年以下
3-5年
5-10年
不限
应届毕业生
119698
总岗位数
19688
五险一金岗位数
全国岗位分布情况
学历与薪资
61.66
37.31
30.82
18.42
不限
博士
大专
硕士
本科
岗位名称和数量Top10
ANDROID开发工程师
IOS开发工程师
销售经理
运营助理
测试工程师
新媒体运营
前端开发工程师
产品经理
WEB前端开发工程师
UI设计师
福利Top10