北京联合大学

**《Python程序设计》大作业报告**

**天气预报**

爬虫+数据库+Django+网络服务器

姓　 名： 向俊杰 +

班 级： 软件工程2021级

学 号： 2021250360039

学 院： 机器人学院

专 业： 软件工程

2022年12月18日

目录

[1. 总体框架 2](#_Toc300)

[2. Python网络爬虫 5](#_Toc1187)

[2.1相关配置 6](#_Toc1499)

[2.2 Python天气爬虫流程图 8](#_Toc21705)

[2.3重要代码摘要 10](#_Toc20466)

[2.3.1读取城市名 10](#_Toc18316)

[2.3.2模拟鼠标操作 10](#_Toc686)

[2.3.3 XPATH语法规则 11](#_Toc5097)

[2.3.4 dometree 12](#_Toc24324)

[2.4问题与解决方法 16](#_Toc27971)

[2.4.1浏览器渲染未完成导致数据缺失 16](#_Toc28673)

[2.4.2 StaleElementReferenceException 16](#_Toc10948)

[2.4.3 ElementClickInterceptedException 19](#_Toc3894)

[2.4.4提交功能被隐藏 20](#_Toc19098)

[2.4.5网站拒绝连接 21](#_Toc10612)

[2.4.6 driver.get()获取url速度快 22](#_Toc14406)

[3. Django项目+数据库 23](#_Toc11869)

[3.1后端实现 23](#_Toc13923)

[3.1.1创建一个Django项目 23](#_Toc10652)

[3.1.2数据库的相关操作 28](#_Toc3973)

[3.1.3 AdminLTE3美化 44](#_Toc15039)

[3.2前端实现 44](#_Toc3869)

[3.2.1主要功能实现 46](#_Toc2317)

[4. 网络服务器部署+爬虫部署 49](#_Toc2703)

[4.1网络服务器部署 49](#_Toc31432)

[4.2爬虫部署 50](#_Toc17364)

[5. 用户说明 51](#_Toc29624)

[6. 参考文献 52](#_Toc31333)

# 1. 总体框架

成果展示：

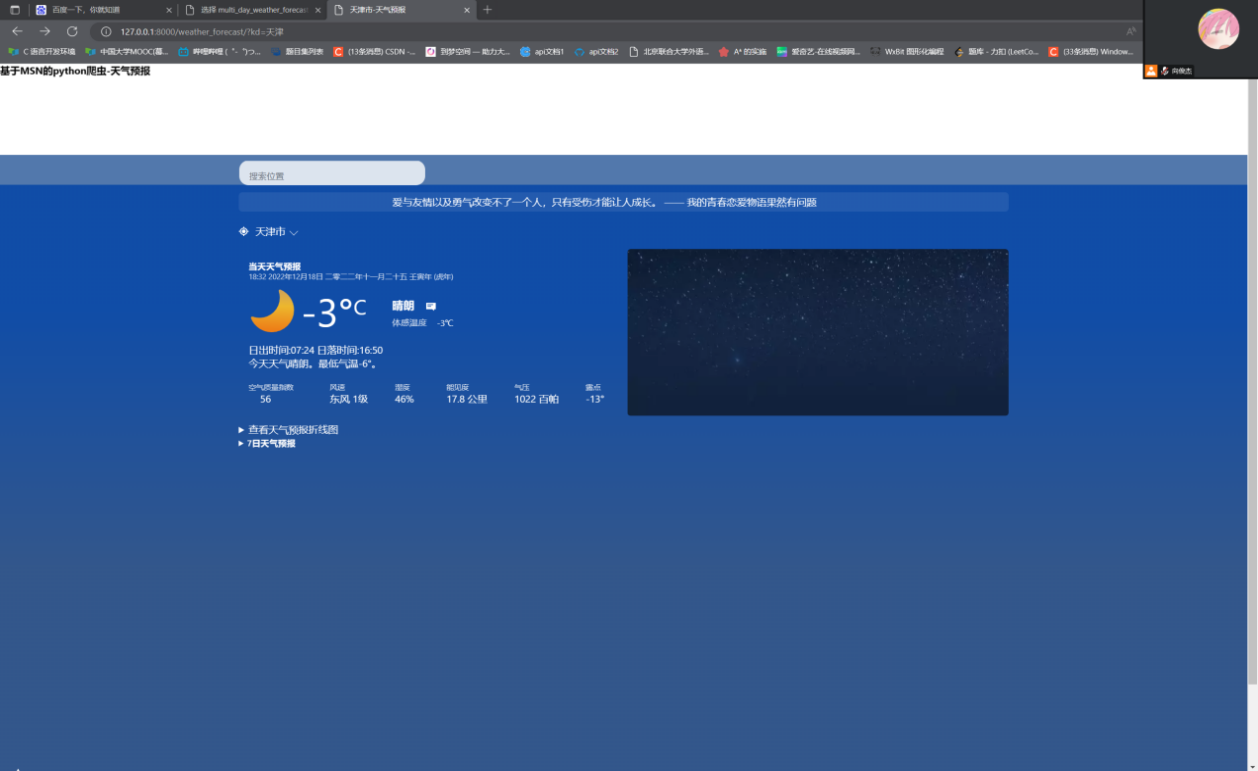


图1 天气预报网页图

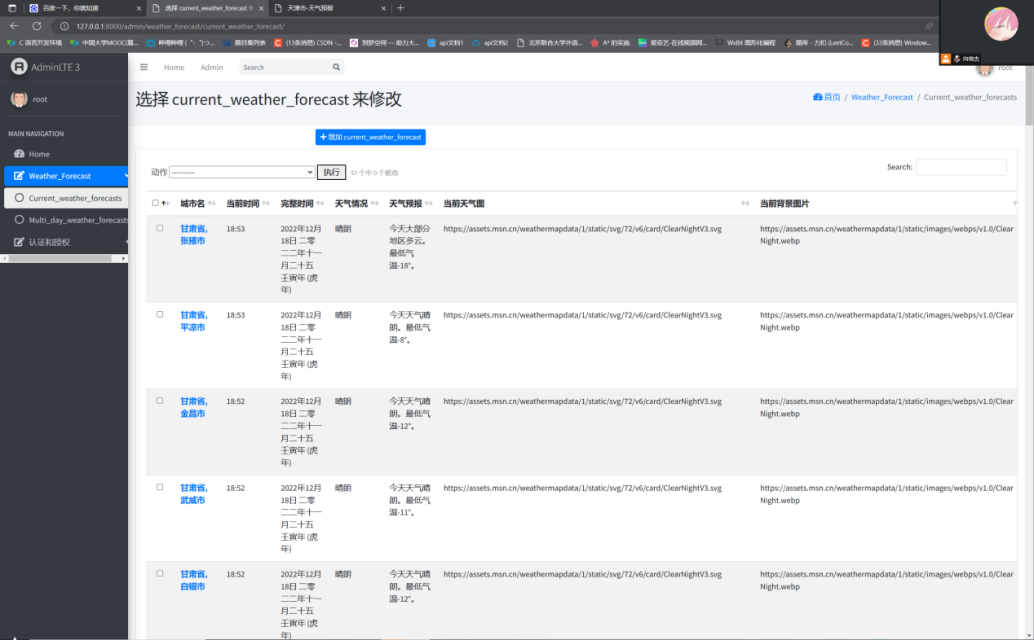


图2 美化后的后台页面

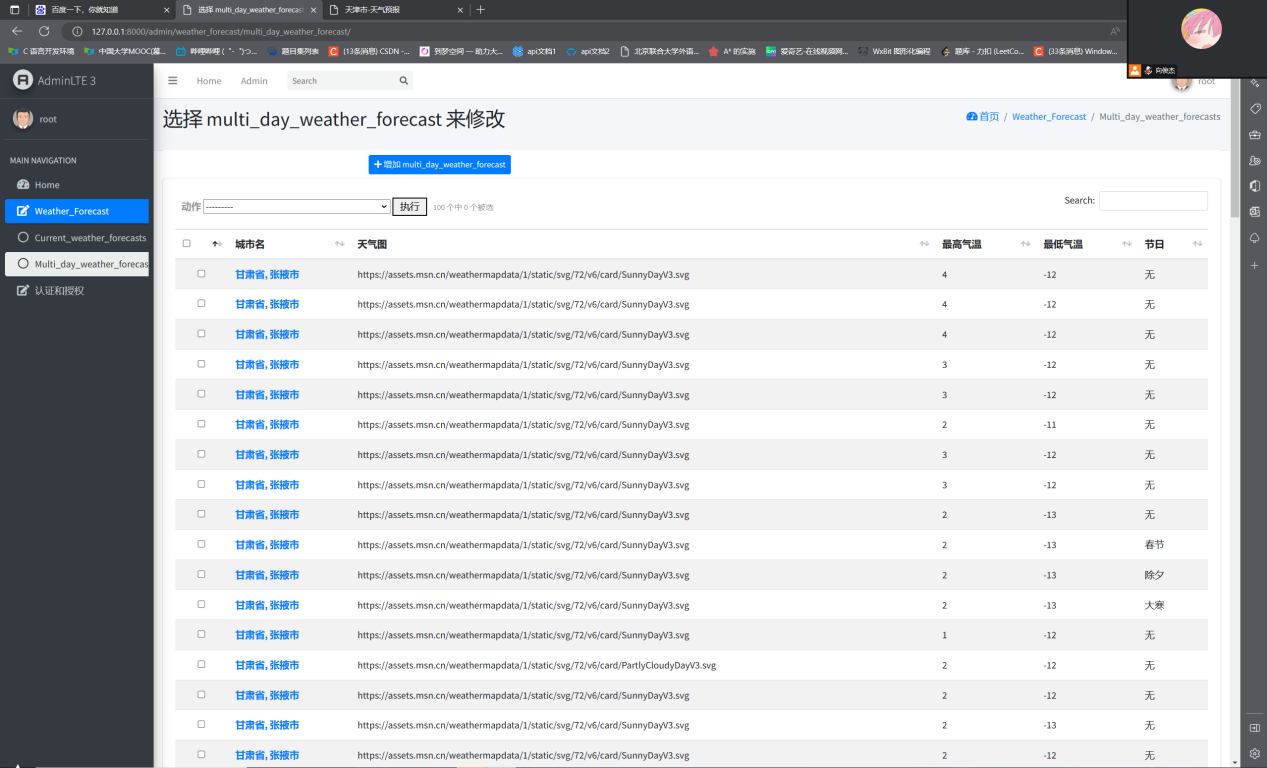


图3 后台数据展示

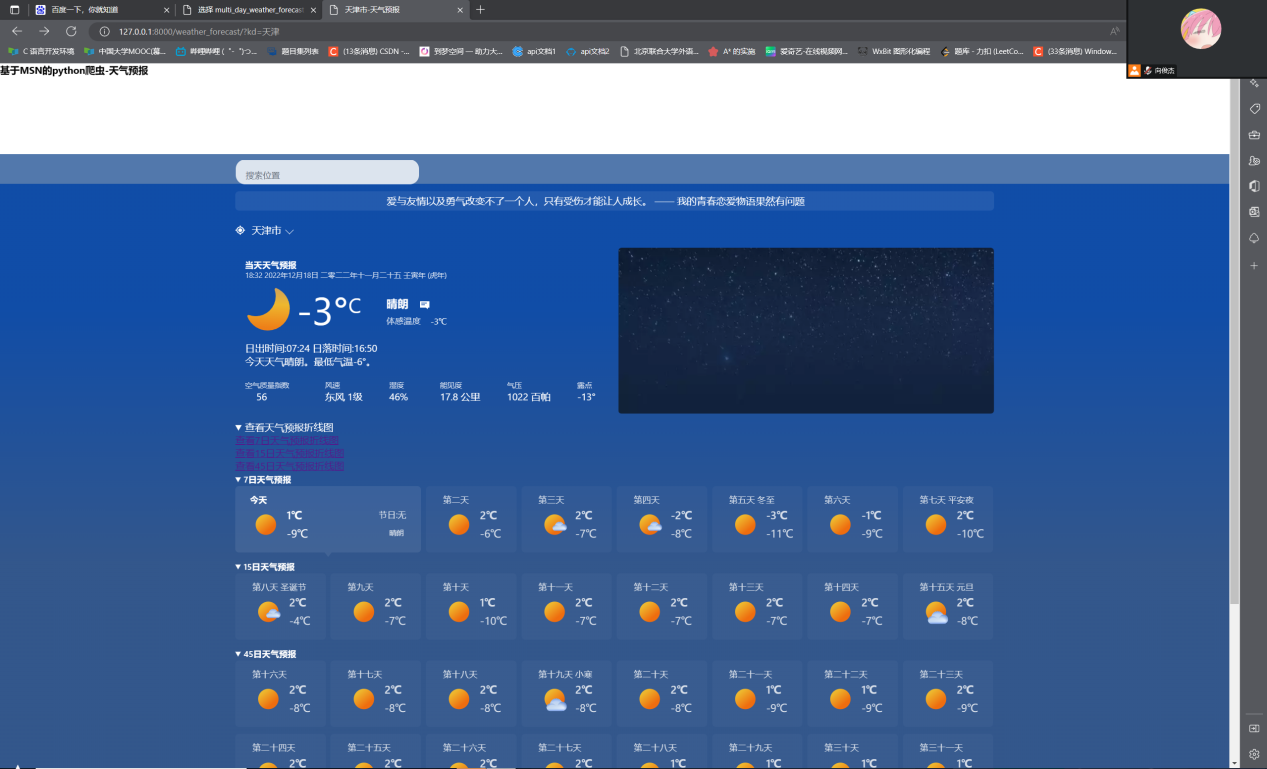


图5 天气预报45天天气数据展示

总体流程图：

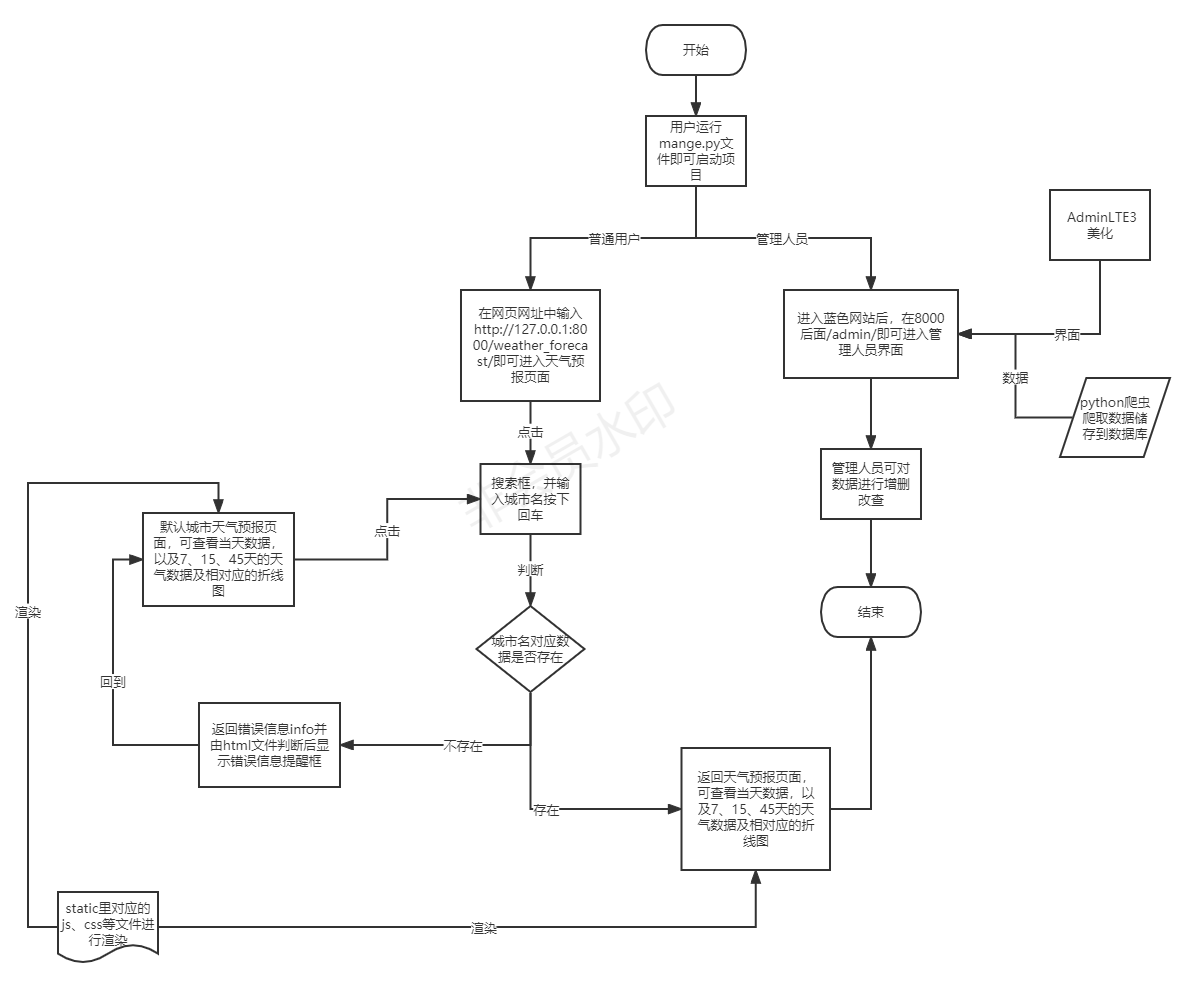


图6 整个项目流程图

# 2. Python网络爬虫

本次作业中的python网络爬虫是基于MSN天气网站进行的动态爬虫，MSN天气是微软公司旗下的天气网站，有一些反爬手段和动态更新技术，故采用Selenium进行本次作业的爬虫工作。（Selenium是一个Web的自动化测试工具，最初是为网站自动化测试而开发的，Selenium可以直接调用浏览器，它支持所有主流的浏览器，可以接收指令，让浏览器自动加载页面，获取需要的数据，甚至页面截屏等）

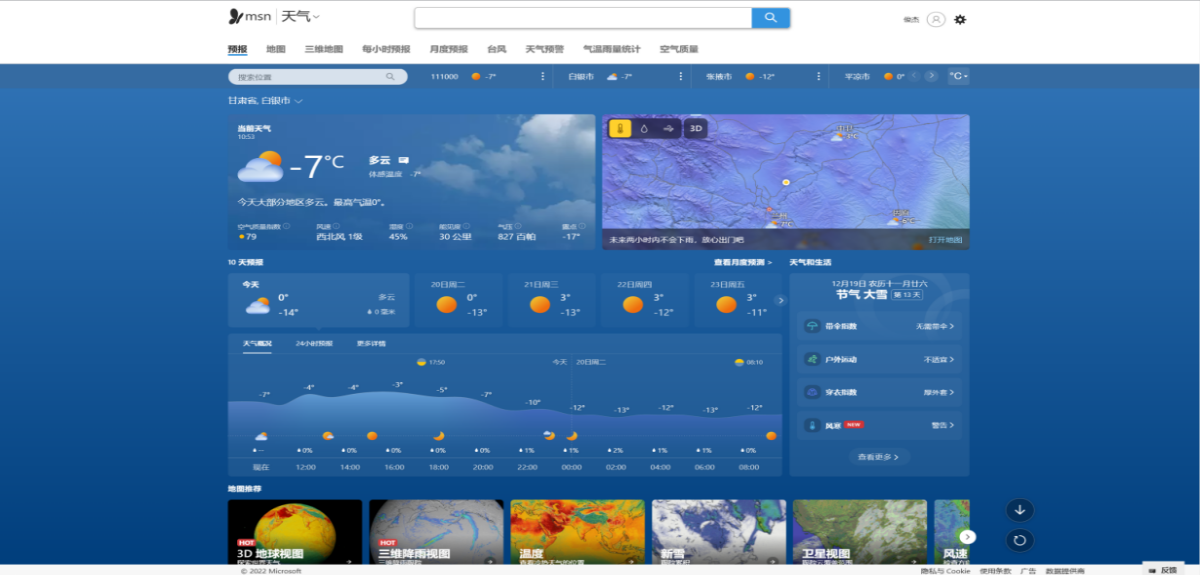


图7 MSN天气网

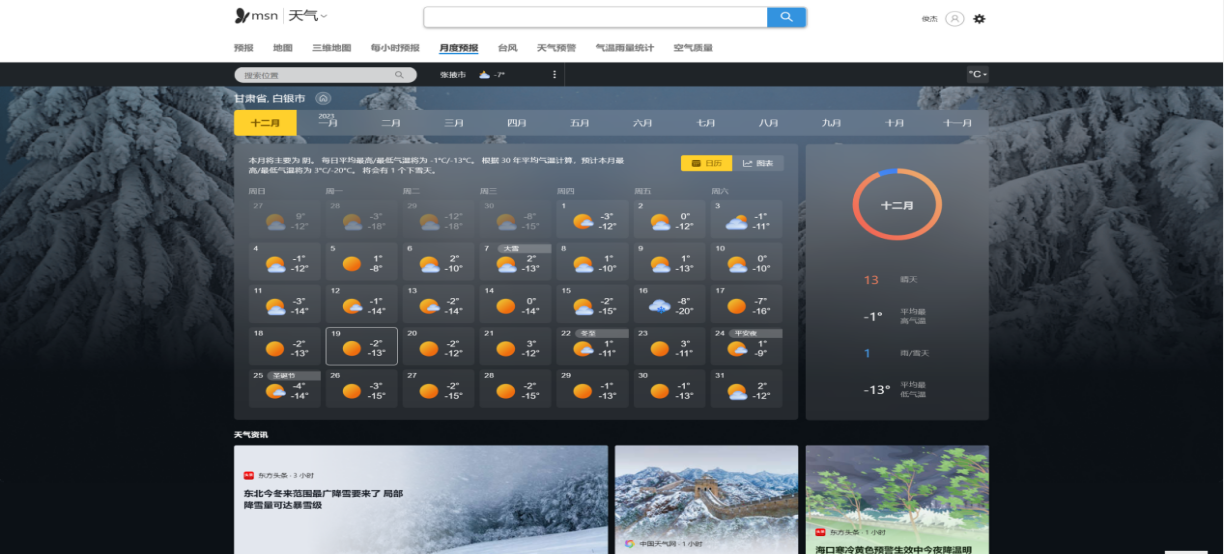


图8 MSN月度预测图

## 2.1相关配置

本次作业采用的开发工具是PyCharm 2021.3.1(专业版)，社区版也行，下载Selenium所需对应浏览器驱动，本次作业选用的是Microsoft edge驱动器，下载好对应版本的edge驱动器，放到你爬虫py文件所在python项目同一文件夹下(这里统一放到Django项目同一文件夹下)，并将edge/applicication中的msedgedriver.exe改名为MicrosoftWebDriver.exe后即可使用。[1]然后在给edge驱动配置环境变量(将MicrosoftWebDriver.exe所在bin目录的路径添加到环境变量里)，[2],如果没有配置环境变量，则需要：

1. *#这是浏览器显示创建的edge实例对象*
2. driver=webdriver.Edge()
3. *#如果driver没有添加到了环境变量，则需要将driver的绝对路径赋值给executable\_path参数*
4. driver=webdriver.Edge(executable\_path='绝对路径')

在PyCharm里点文件(File)点开设置(Settings)，选择python解释器(Python Interpreter)，(或者到第三方包官方网站去下载镜像文件，再用pip安装[3])如图9、10:

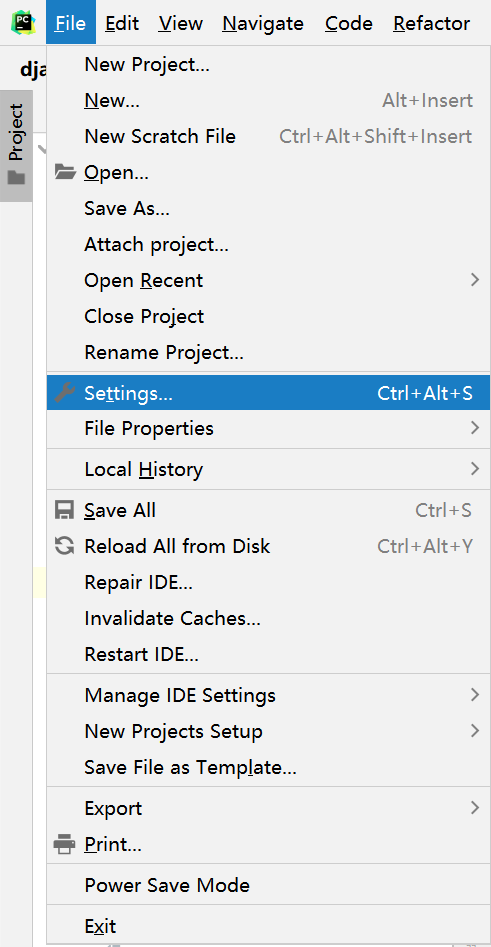


图9 File里的Settings

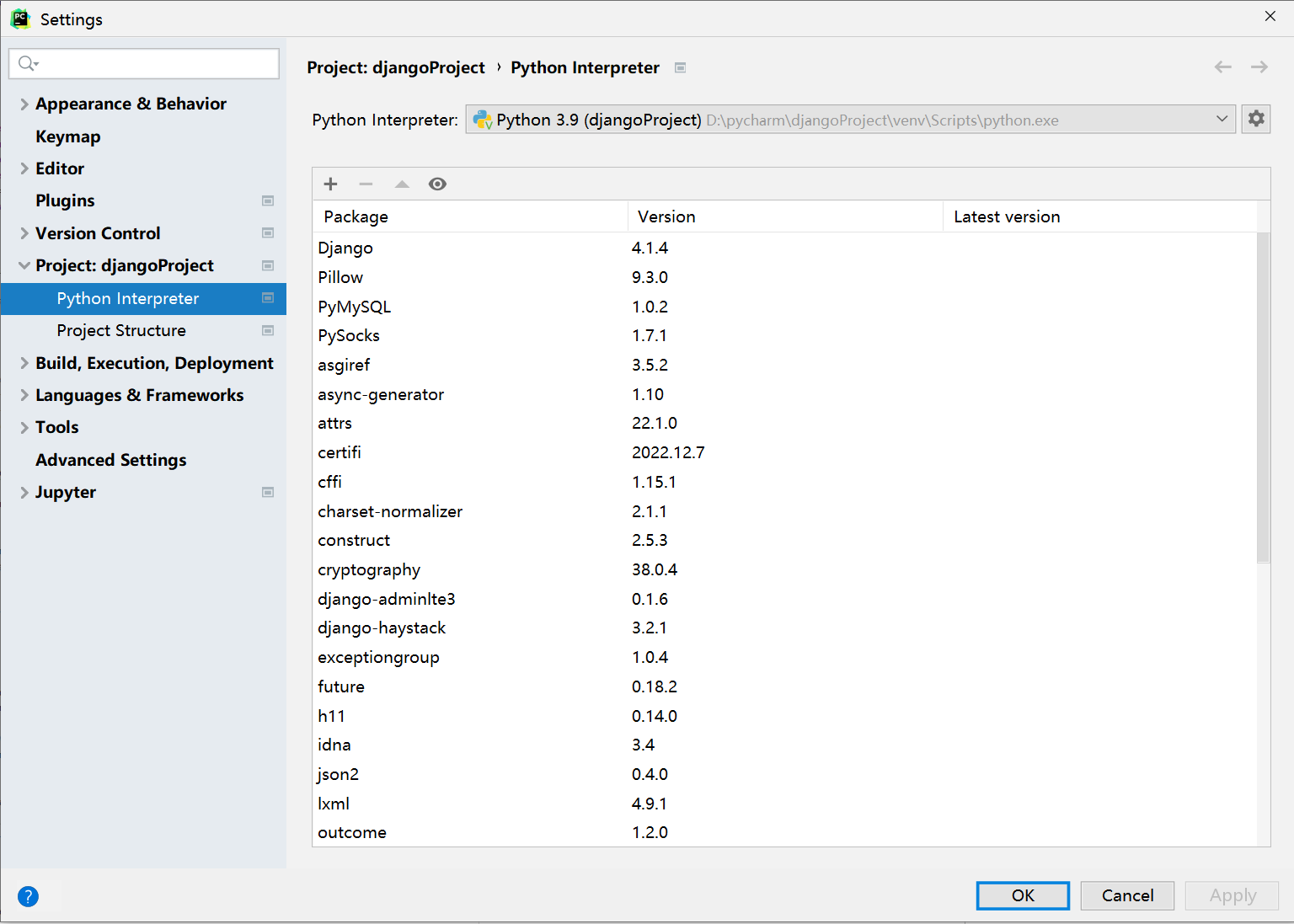


图10 Settings

在这里下载第三方包selenium、lxml,并导入如下包：

1. import time
2. import re
3. from selenium import webdriver
4. from lxml import etree
5. from selenium.webdriver import ActionChains
6. from selenium.webdriver.common.by import By
7. from selenium.webdriver.support.wait import WebDriverWait
8. from selenium.webdriver.support import expected\_conditions as EC
9. from zhdate import\*

这里导入time包是为了暂停进程从而让浏览器对网页渲染完成，强制等待而lxml对应包则是由于lxml模块可以利用XPath规则语法，来快速的定位HTML\XML文档中特定元素以及获取节点信息（文本内容、属性值）XPath(XML Path Language)是一门在HTML\XML文档中查找信息的语言，可用来在HTML\XML文档中对元素和属性进行遍历。[4]而re包则是配合XPATH规则语法通过正则表达式来获取相应所需内容[5],从selenium里导入相关包有webdriver创建一个edge实例对象，ActionChains模拟鼠标操作，By对应查找网页元素的方法，WebDriverWait隐式等待，expected\_conditions as EC限制条件。[5]Zhdate包则是将当前日期转成对应的农历形式。

lxml基本流程：

1. 导入lxml的etree库
2. from lxml import etree
3. 利用etree.HTML，将html字符串（bytes类型或str类型）转化为Element对象，Element对象具有xpath的方法，返回结果的列表html=etree.HTML(body)

xpath方法返回列表的三种情况

* 返回空列表：根据xpath语法规则字符串，没有定位到任何元素。
* 返回由字符串构成的列表：xpath字符串规则匹配的一定是文本内容或某属性的值。
* 返回由Element对象构成的列表：xpath规则字符串匹配的是标签，列表中的Element对象可以继续进行xpath。

## 2.2 Python天气爬虫流程图

如图11，Python天气爬虫流程图，由于是采用模拟点击的方式，故调用驱动不能采用无界面模式。

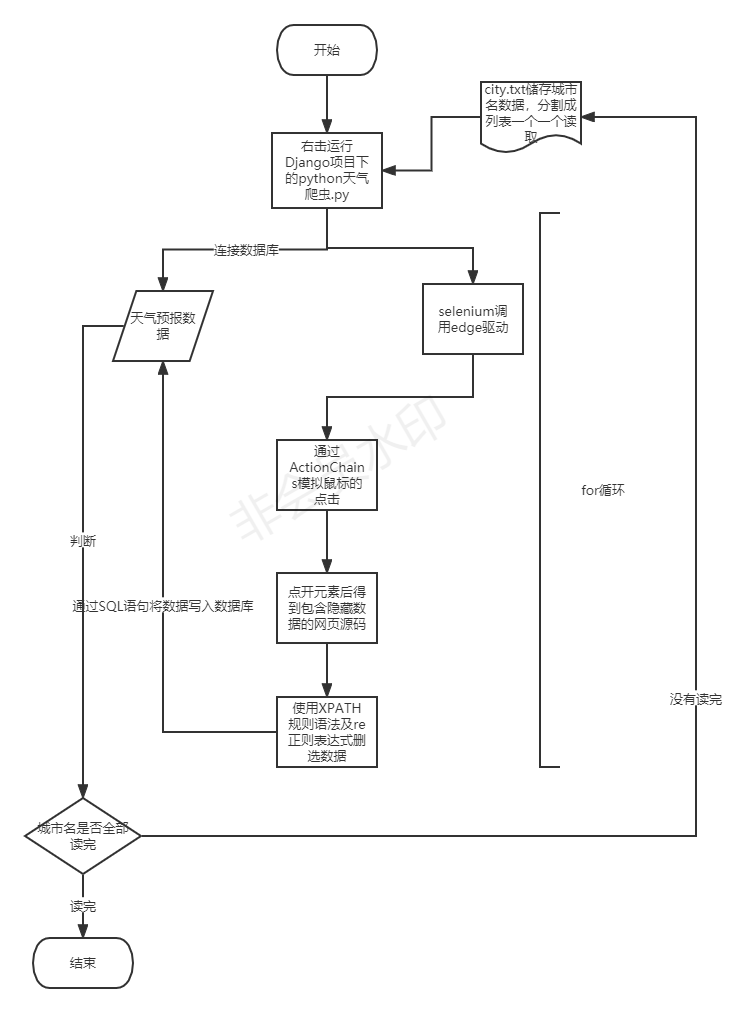


图11 Python天气爬虫流程图

## 2.3重要代码摘要

### 2.3.1读取城市名

由城市名组成的city.txt文件(放在项目里)读入城市名并调用查找函数实现爬取城市名对应天气数据。

1. *#读入城市*
2. with open('D:\pycharm\djangoProject\city.txt','r',encoding='UTF-8')as f:
3. data=f.read()
4. cities=data.split('')
5. for i in range(len(cities)):
6. find\_the\_city(cities[i])

### 2.3.2模拟鼠标操作

由ActionChains模拟鼠标操作，根据网页元素坐标点击来点掉弹窗、点击链接或按钮爬取数据。

1. ActionChains(driver).move\_by\_offset(800,900).click().perform()
2. #由于浏览器渲染时间、浏览器弹窗、网站防爬等原因故采用模拟鼠标点击方法
3. #(800,900)为我接受坐标
4. #(255,220)为选择框坐标
5. #(435,960)更多详情选项
6. #(555,145)为弹窗坐标
7. #(520,100)为月度预测

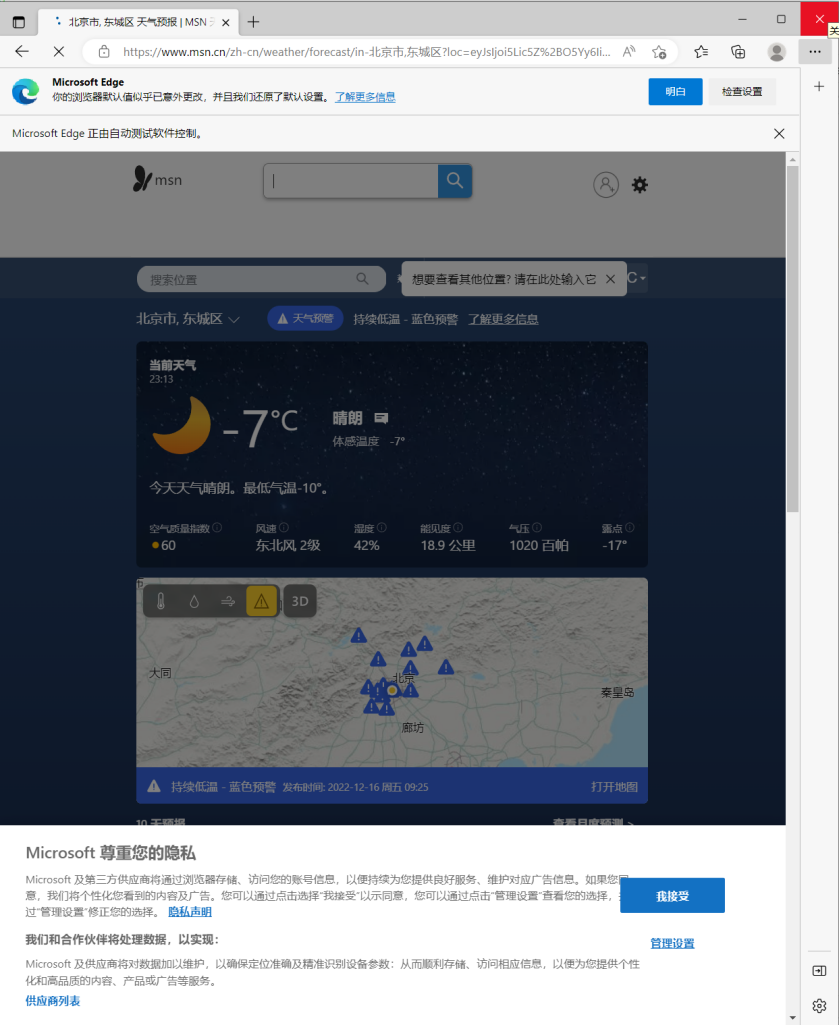


图12 selenium调用edge驱动

### 2.3.3 XPATH语法规则

XPATH规则语法加上正则表达式获取图片url。

1. *#获取当前天气图*
2. sources5=dometree.xpath('//\*[@id="OverviewCurrentTemperature"]/img')
3. for source in sources5:
4. current\_weather\_picture=\
5. re.search(r'https?:[a-zA-Z0-9/.//\_\-]\*\.(webp|png|jpg|svg)\b',''.join(source.xpath("@src")))[0]
6. current[keys[5]].append(current\_weather\_picture)
7. print(f'current\_weather\_picture:{current\_weather\_picture}')

### 2.3.4 dometree

获取天气对应元素并获取其中数据，dometree.xpath获得的是数据列表通过for循环再用jion方法将列表转化成字符串。

1. *#获取正确(有结果的)城市名city\_name*
2. sources2=dometree.xpath('//\*[@id="WeatherOverviewLocationName"]/a')
3. for source in sources2:
4. city\_name=re.search('.+,?',''.join(source.xpath("text()")))[0]
5. print(f'这是{city\_name}当前天气信息')
6. current[keys[0]].append(city\_name)
7. print(f'city\_name:{city\_name}')
8. *#获取当前时间*
9. time1=datetime.now()
10. current\_time=re.search('\d+:\d+',str(time1))[0]
11. current[keys[1]].append(current\_time)
12. print(f'current\_time:{current\_time}')
13. *#获取完整时间*
14. time2=re.search('\d{4}-\d{2}-\d{2}',str(time1))[0].split('-')
15. full\_time=time2[0]+'年'+time2[1]+'月'+time2[2]+'日'+''+ZhDate.from\_datetime(datetime(int(time2[0]),int(time2[1]),int(time2[2]))).chinese()
16. current[keys[2]].append(full\_time)
17. print(f'full\_time:{full\_time}')
18. *#获取天气情况*
19. sources3=dometree.xpath('//\*[@id="OverviewCurrentTemperature"]/div/div[1]')
20. for source in sources3:
21. weather\_condition=''.join(source.xpath("text()"))
22. current[keys[3]].append(weather\_condition)
23. print(f'weather\_condition:{weather\_condition}')
24. *#获取天气预报*
25. sources4=dometree.xpath('//\*[@id="CurrentWeatherSummary"]/p')
26. for source in sources4:
27. weather\_forecast=''.join(source.xpath("text()"))
28. current[keys[4]].append(weather\_forecast)
29. print(f'weather\_forecast:{weather\_forecast}')
30. *#获取当前天气图*
31. sources5=dometree.xpath('//\*[@id="OverviewCurrentTemperature"]/img')
32. for source in sources5:
33. current\_weather\_picture=\
34. re.search(r'https?:[a-zA-Z0-9/.//\_\-]\*\.(webp|png|jpg|svg)\b',''.join(source.xpath("@src")))[0]
35. current[keys[5]].append(current\_weather\_picture)
36. print(f'current\_weather\_picture:{current\_weather\_picture}')
37. *#获取当前背景图*
38. sources6=dometree.xpath('//\*[@id="WeatherOverviewCurrentSection"]/div[2]')
39. for source in sources6:
40. current\_background\_picture=\
41. re.search(r'https?:[a-zA-Z0-9/.//\_\-]\*\.(webp|png|jpg|svg)\b',''.join(source.xpath("@style")))[0]
42. current[keys[6]].append(current\_background\_picture)
43. print(f'current\_background\_picture:{current\_background\_picture}')
44. *#获取当前温度*
45. sources7=dometree.xpath('//\*[@id="OverviewCurrentTemperature"]/a')
46. for source in sources7:
47. current\_temperature=''.join(source.xpath("text()"))+'°'
48. current[keys[7]].append(current\_temperature)
49. print(f'current\_temperature:{current\_temperature}')
50. *#获取体感温度*
51. sources8=dometree.xpath('//\*[@id="OverviewCurrentTemperature"]/div/a/div[2]')
52. for source in sources8:
53. somatosensory\_temperature=''.join(source.xpath("text()"))[:-1]
54. current[keys[8]].append(somatosensory\_temperature)
55. print(f'somatosensory\_temperature:{somatosensory\_temperature}')
56. *#获取日出时间*
57. sources9=dometree.xpath(
58. '//\*[@id="root"]/div/div/div[3]/div[2]/div[2]/div/div[3]/div/div[3]/div[2]/div/div/div/div[1]/div[3]/div[2]/div/div[1]/div[1]')
59. for source in sources9:
60. sunrise\_time=''.join(source.xpath("text()"))
61. current[keys[9]].append(sunrise\_time)
62. print(f'sunrise\_time:{sunrise\_time}')
63. *#获取日落时间*
64. sources10=dometree.xpath(
65. '//\*[@id="root"]/div/div/div[3]/div[2]/div[2]/div/div[3]/div/div[3]/div[2]/div/div/div/div[1]/div[3]/div[2]/div/div[3]/div[1]')
66. for source in sources10:
67. sunset\_time=''.join(source.xpath("text()"))
68. current[keys[10]].append(sunset\_time)
69. print(f'sunset\_time:{sunset\_time}')
70. *#获取空气质量指数*
71. sources11=dometree.xpath('//\*[@id="WeatherOverviewCurrentSection"]/div[2]/div/div[3]/div/div[1]/a/div[2]')
72. for source in sources11:
73. air\_quality\_index=''.join(source.xpath("text()"))
74. current[keys[11]].append(air\_quality\_index)
75. print(f'air\_quality\_index:{air\_quality\_index}')
76. *#获取湿度*
77. *#从span标签里获取内容浏览器提供的xpath后//span*
78. sources12=dometree.xpath('//\*[@id="CurrentDetailLineHumidityValue"]//span')
79. for source in sources12:
80. moisture\_content=''.join(source.xpath("text()"))
81. current[keys[12]].append(moisture\_content)
82. print(f'moisture\_content:{moisture\_content}')
83. *#获取风速*
84. sources13=dometree.xpath('//\*[@id="WeatherOverviewCurrentSection"]/div[2]/div/div[3]/div/div[2]/a/div[2]')
85. for source in sources13:
86. wind\_speed=''.join(source.xpath("text()"))
87. current[keys[13]].append(wind\_speed)
88. print(f'wind\_speed:{wind\_speed}')
89. *#获取可见度*
90. sources14=dometree.xpath('//\*[@id="CurrentDetailLineVisibilityValue"]//span')
91. for source in sources14:
92. visibility=''.join(source.xpath("text()"))
93. current[keys[14]].append(visibility)
94. print(f'visibility:{visibility}')
95. *#获取气压*
96. sources15=dometree.xpath('//\*[@id="CurrentDetailLinePressureValue"]//span')
97. for source in sources15:
98. atmospheric\_pressure=''.join(source.xpath("text()"))
99. current[keys[15]].append(atmospheric\_pressure)
100. print(f'atmospheric\_pressure:{atmospheric\_pressure}')
101. *#获取露点*
102. sources16=dometree.xpath('//\*[@id="CurrentDetailLineDewPointValue"]//span')
103. for source in sources16:
104. dew\_point=''.join(source.xpath("text()"))

## 2.4问题与解决方法

### 2.4.1浏览器渲染未完成导致数据缺失

|  |  |
| --- | --- |
| 浏览器渲染未完成导致数据缺失 | 概述 |
| 问题 | 浏览器由于网络等问题导致调用js、css文件渲染未完成，页面数据未完全显示。 |
| 解决方法 | 采用强制等待和隐式等待相结合的方式。 |
| 强制等待 | 即time.sleep()，最简单的等待方式，缺点：设置时间短了没用，长了又浪费时间。 |
| 隐式等待 | 是全局的是针对所有元素，设置等待时间如10秒，如果10秒内出现，则继续向下，否则抛异常。可以理解为在10秒以内，不停刷新看元素是否加载出来。 |
| 显示等待 | 是单独针对某个元素，设置一个等待时间如5秒，每隔0.5秒检查一次是否出现，如果在5秒之前任何时候出现，则继续向下，超过5秒尚未出现则抛异常。 |

### 2.4.2 StaleElementReferenceException

问题：StaleElementReferenceException:Message:stale element reference:element is not attached to the page document.

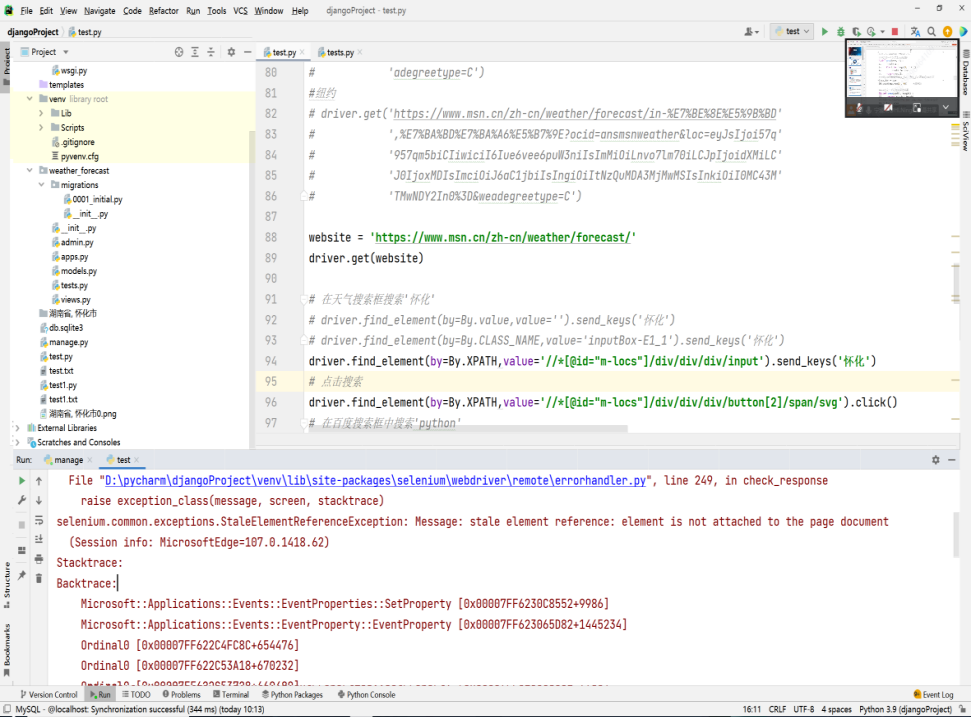


图13 StaleElementReferenceException问题

解决方法：由于selenium版本问题，语法新老版本之间有出入。

1. #在天气搜索框搜索城市名
2. searchbox\_forecast=driver.find\_element(by=By.XPATH,value='//\*[@id="m-locs"]/div/div/div/input')
3. searchbox\_forecast.send\_keys(cityname)
4. time.sleep(2)

find\_element方法发生变化，采用上述查找网页元素的方法。导入By模块，通过By.XPATH来寻找元素(怎样快速获得XPATH表达式？一般浏览器都自带复制XPATH功能。

在爬取页面F12打开开发者工具

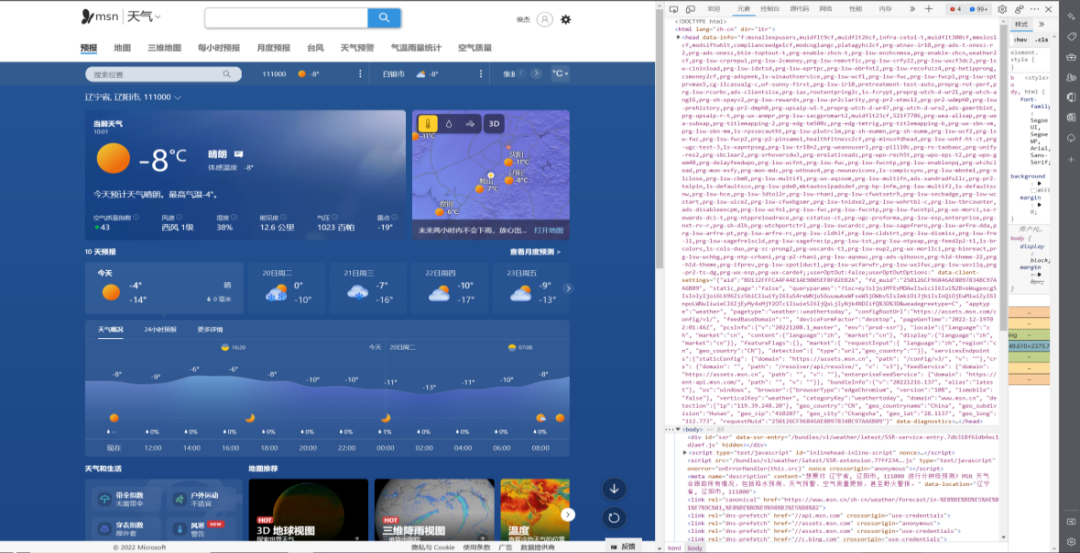


图14 F12打开开发者工具

接着点击左上角，再到网页上选择页面元素，会自动给你跳到相应元素位置，

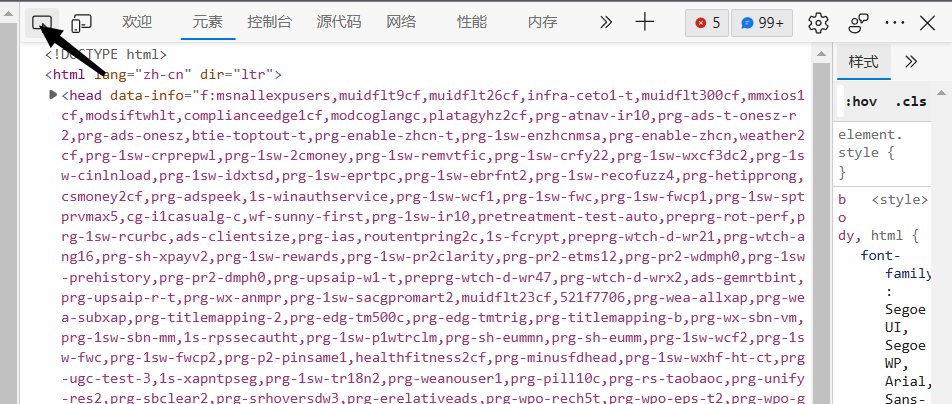


图15选择页面元素

接着在所选元素的位置，右击选择复制，复制xPath。

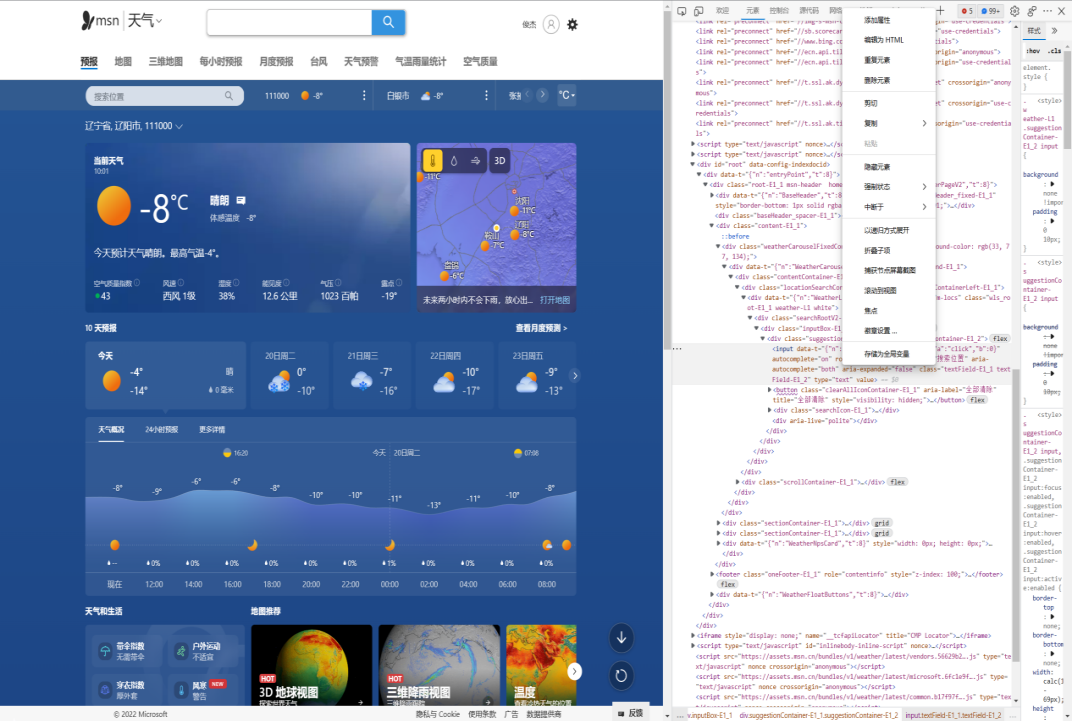


图16右击元素选择复制

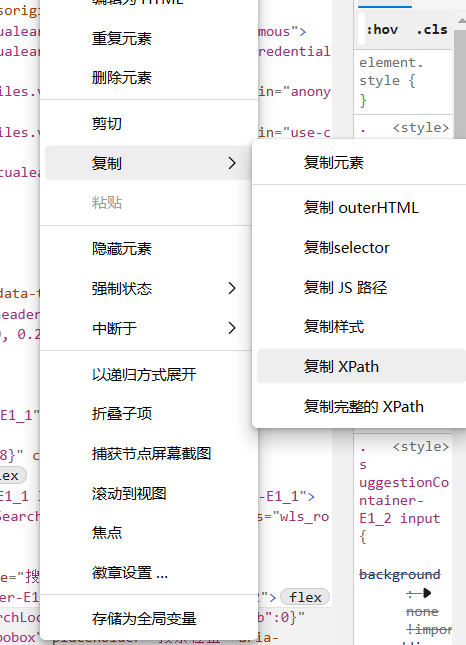


图17 复制xPath

### 2.4.3 ElementClickInterceptedException

问题：selenium-解决ElementClickInterceptedException元素不可点击问题。

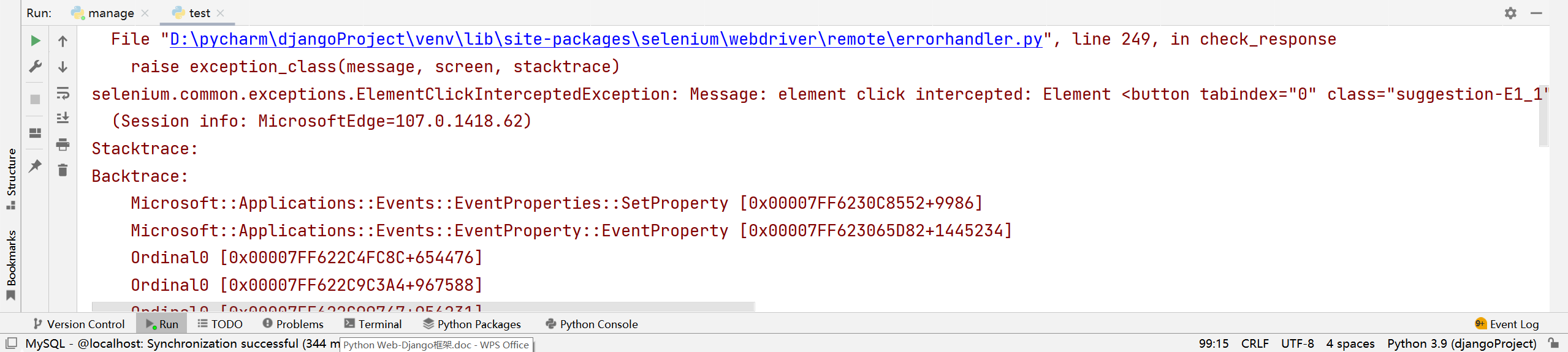


图18 ElementClickInterceptedException问题

解决办法：由于网站自身问题，对按钮、链接进行了隐藏，所以通过find\_element方法找到网页元素后，无法直接调用.click()方法去直接点击，故我们采用ActionChains模拟鼠标操作进行点击。

1. #在天气搜索框搜索城市名
2. searchbox\_forecast=driver.find\_element(by=By.XPATH,value='//\*[@id="m-locs"]/div/div/div/input')
3. searchbox\_forecast.send\_keys(cityname)
4. time.sleep(2)
5. #上一步鼠标坐标(800,900)
6. ActionChains(driver).move\_by\_offset(-545,-680).click().perform()
7. time.sleep(2)

ActionChains(driver).move\_by\_offset(-545,-680).click()是准备一个鼠标点击动作，后面必须跟上perform()来执行该动作。(怎样来确定网页元素的坐标？通过context\_click()方法右击网页，由此出现选择项，由此确定不同坐标元素所在位置，当确定好后，将context\_click()方法改成click()方法实现点击)

### 2.4.4提交功能被隐藏

问题：网页提交功能被隐藏，无法通过.submit()方法提交信息

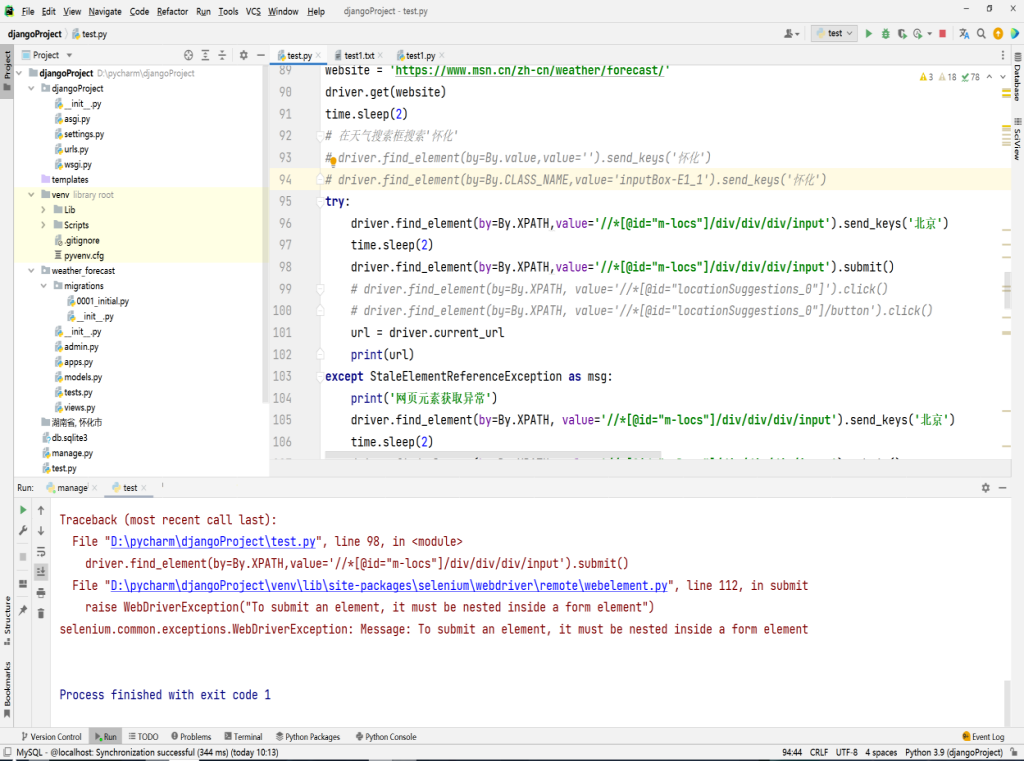


图19 无法通过.submit()方法提交信息问题

解决方法：采用导入Keys模块来模拟键盘操作(Keys.ENTER)无法解决问题，有由于将城市名输入后网页会自动加载复选框，故同样采用模拟鼠标点击选中项的方法，这里点击点一个选择。

1. #在天气搜索框搜索城市名
2. searchbox\_forecast=driver.find\_element(by=By.XPATH,value='//\*[@id="m-locs"]/div/div/div/input')
3. searchbox\_forecast.send\_keys(cityname)
4. time.sleep(2)
5. #上一步鼠标坐标(800,900)
6. ActionChains(driver).move\_by\_offset(-545,-680).click().perform()
7. time.sleep(2)



图20 输入信息加载出来了复选框

### 2.4.5网站拒绝连接

问题：同时创建两个edge实例对象，导致网站拒绝连接。



图21 网站拒绝连接问题

解决方法：采用内嵌函数的方法。

1. url=driver.current\_url
2. find\_current\_weather\_forecast(cityname,url)

### 2.4.6 driver.get()获取url速度快

问题：原是想通过查找月度预测(以获取45天天气数据)链接后获取url，然后通过driver.get()来重新加载月度预测页面，但由于driver.get()获取url速度比较快，时常出现url还为空的时候就已经被driver.get()调用了。

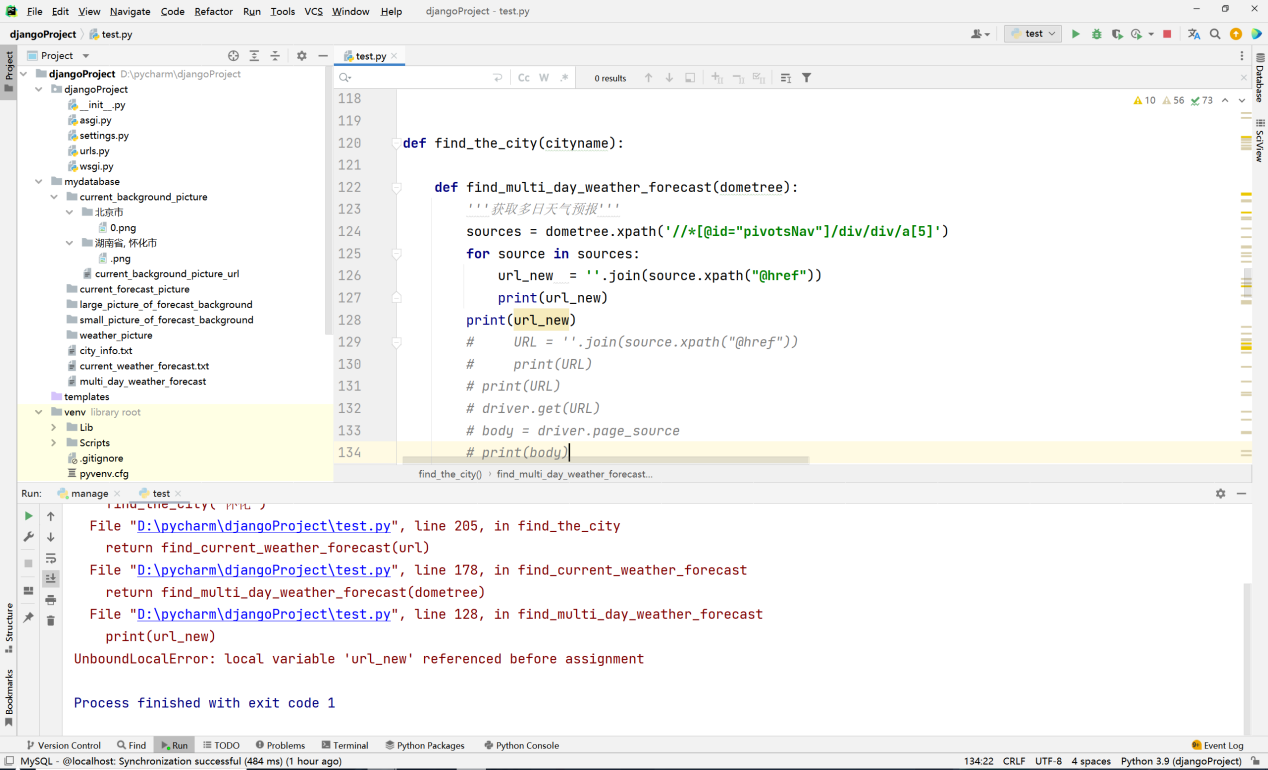


图22 driver.get()获取url速度快问题

解决方法：直接弃用该思路，采用模拟鼠标点击链接。



图23 预报为主页面，而月度预报为链接所在

# 3. Django项目+数据库

## 3.1后端实现

### 3.1.1创建一个Django项目

#### 3.1.1.1创建虚拟环境

﻿﻿

图24 创建虚拟环境

#### 3.1.1.2配置启动命令

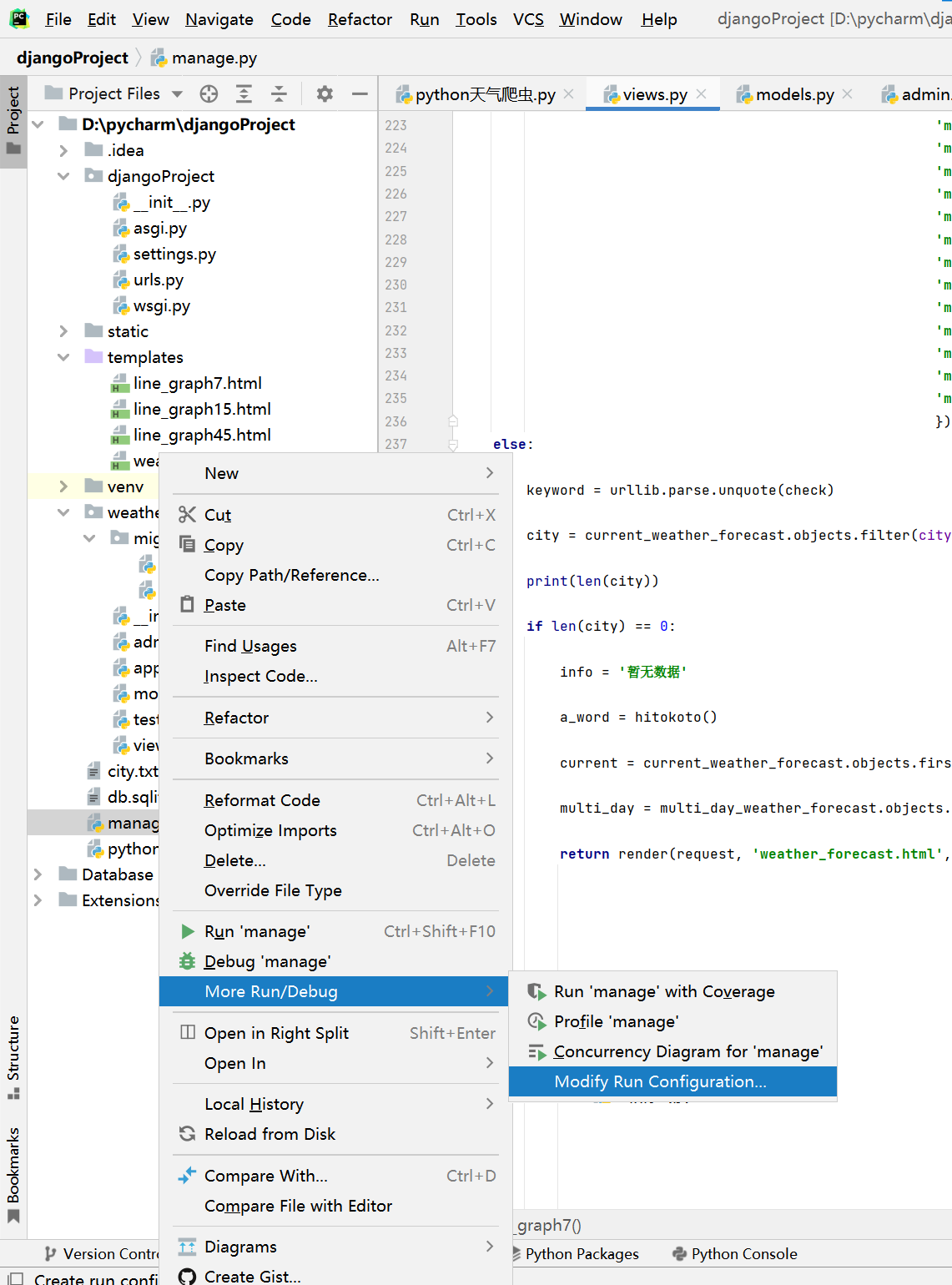
﻿

图25 配置启动环境﻿

在Parameters中添加启动参数runserver。

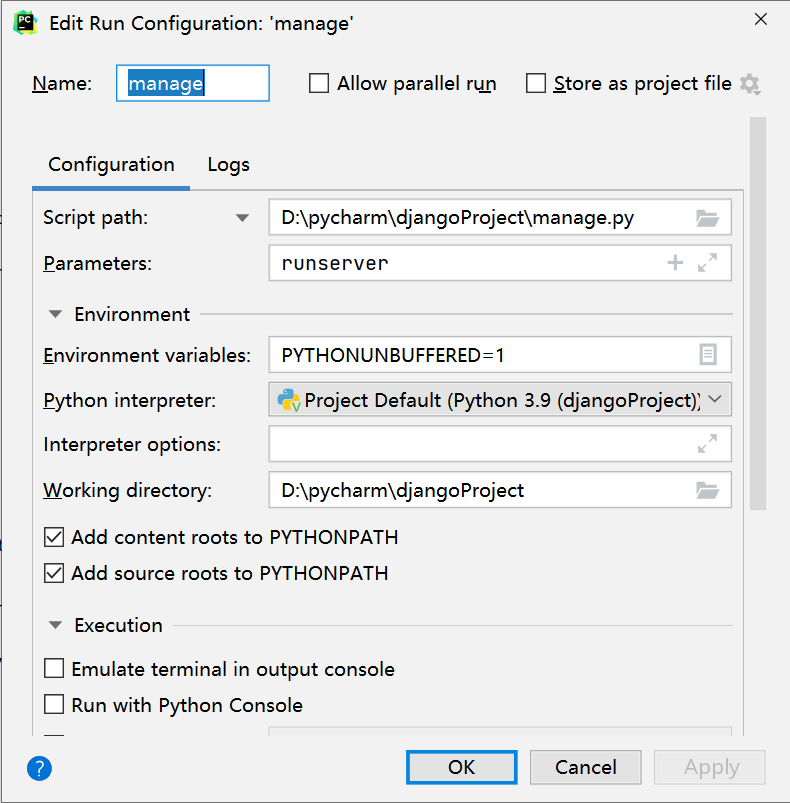


图26 配置启动参数﻿﻿

#### 3.1.1.3启动项目

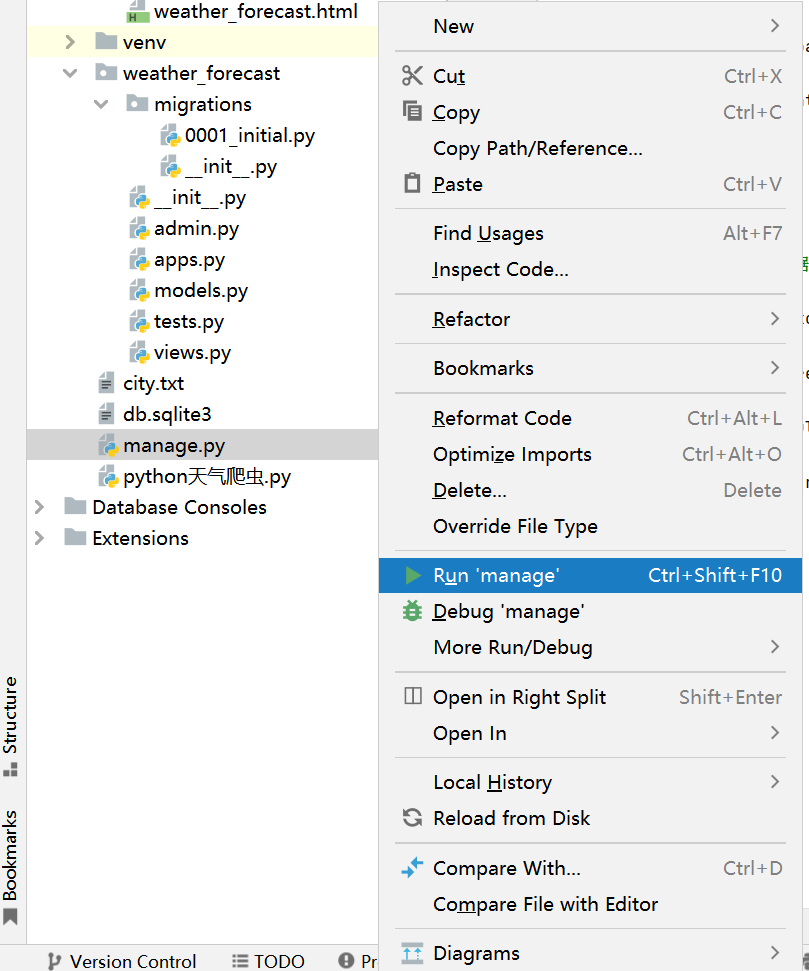
﻿

图27 启动项目

启动成功。

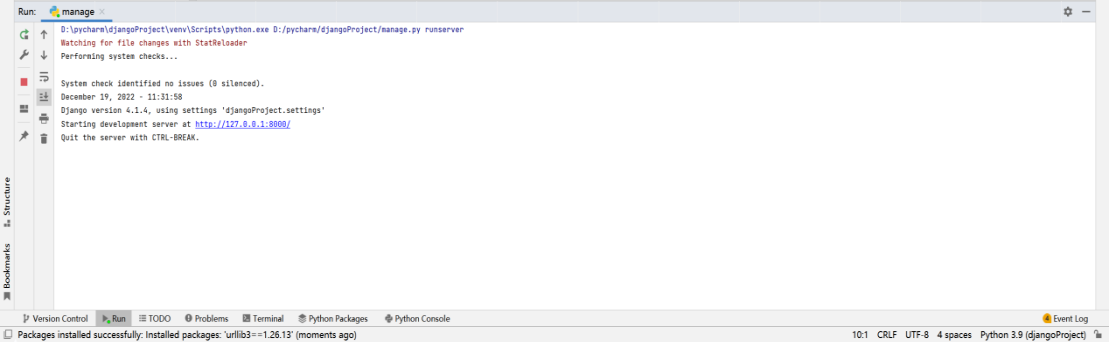
﻿

图28 启动成功

点击蓝色网址http://127.0.0.1:8000进入欢迎页面。

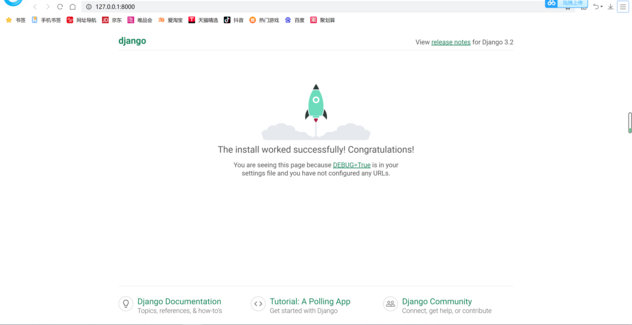


图29 欢迎页面﻿

#### 3.1.1.4切换语言

考虑到当前语言环境为英文，我们修改一下语言环境

﻿﻿

图30 settings文件

1. *#在settings.py文件中找到106行*
2. *#将*
3. LANGUAGE\_CODE='en-us'
4. TIME\_ZONE='UTC'
5. *#修改为*
6. *#语言环境*
7. LANGUAGE\_CODE='zh-hans'
8. *#亚洲上海时区*
9. TIME\_ZONE='Asia/Shanghai'
10. *#修改保存后会自动重启,我们刷新页面,会发现页面已经变成中文的了*﻿﻿

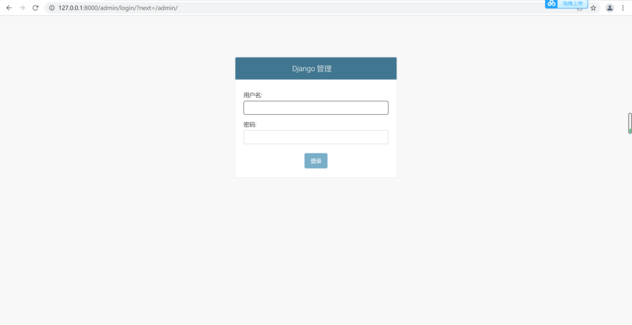


图31 中文欢迎页面

#### 3.1.1.5进入后台

Django自带后台,我们直接在浏览器地址中输入

http://127.0.0.1:8000/admin就可以进入后台了。

﻿

1.2

图32 后台登录页面

### 3.1.2数据库的相关操作

#### 3.1.2.1同步数据库

Django后台功能需要使用到数据库功能,所以我们需要先同步数据库。所谓的同步数据库,就是创建数据库,并且在数据库中创建项目启动所需要的数据表这里需要数据库，Django支持很多的数据库,本次作业采用MySQL数据库。

MySQL数据库的配置如下:

需要提前安装pip install PyMySQL(可以到Python解释器去下载第三方包)

在Django的工程同名子目录的\_\_init\_\_.py文件中添加如下语句：

1. import pymysql
2. pymysql.install\_as\_MySQLdb()

还要改数据库配置，找到在settings.py文件中找到76行

1. DATABASES={
2. 'default':{
3. 'ENGINE':'django.db.backends.mysql',
4. 'NAME':'databasename',#（自己建的数据库名，自己先在cmd窗口建一个数据库，不然会报错找不到该数据库）
5. 'USER':'databaseuser',#（MySQL用户名）
6. 'PASSWORD':'password',#（密码）
7. 'HOST':'127.0.0.1',
8. 'PORT':'3306',
9. }
10. }

[7]

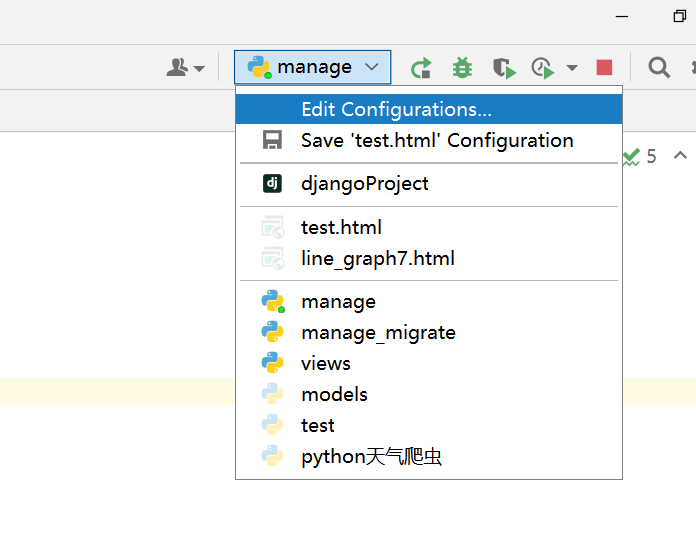


图33 编辑

1. 点击+下拉选择Python。
2. 给当前的命令取一个名字为manage\_migrate。
3. 找到Script path栏目,选择当前项目中的manage.py。
4. 找到Parameters栏目,默认是空白的,在这里填写命令migrate。

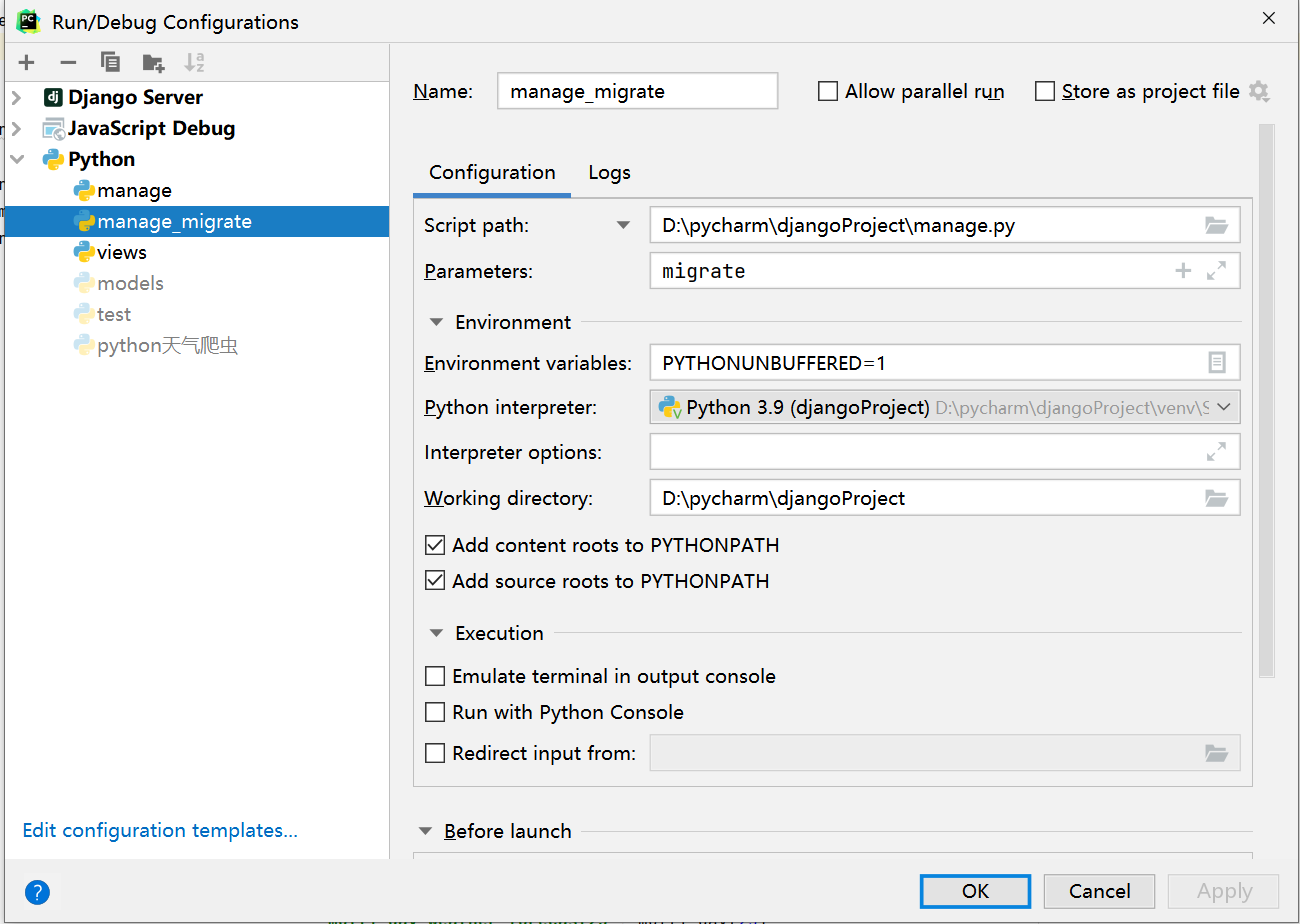


图34 配置完成

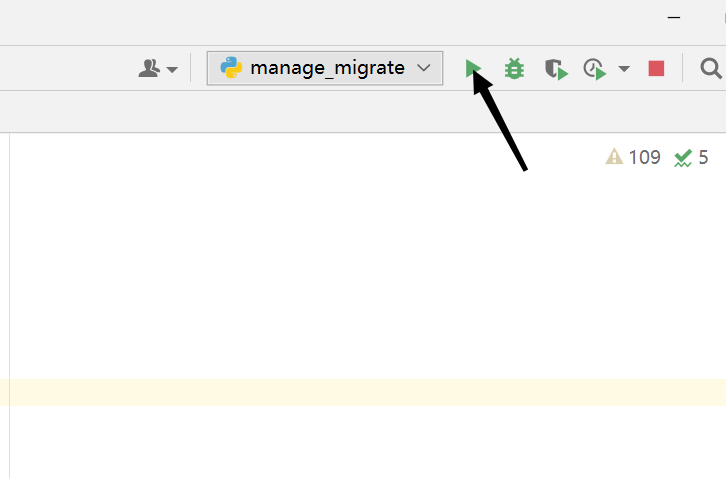


图35 启动命令

点击运行后，即可实现数据库同步。

#### 3.1.2.2开发模块

##### 3.1.2.2.1.创建子应用

在PyCharm终端运行一下两条指令创建子应用。

1. *#在PyCharm终端运行一下两条指令创建子应用*
2. python manage.py startapp子应用名称
3. python manage.py startapp book

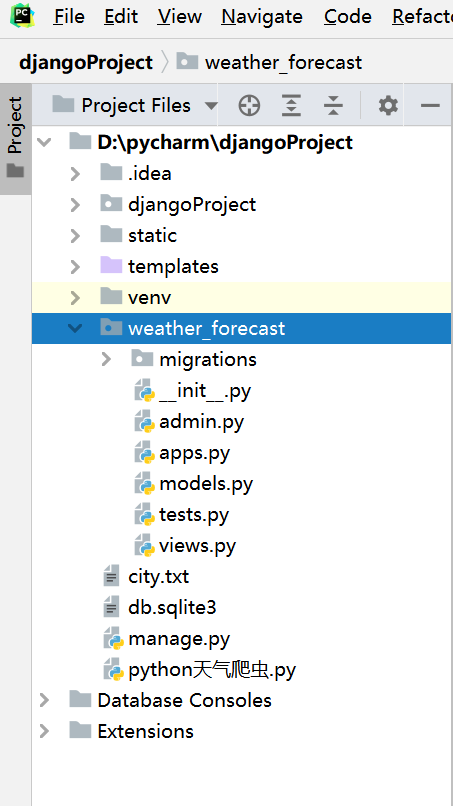


图36生成子应用如weather\_forecast

* admin.py文件跟网站的后台管理站点配置相关。
* apps.py文件用于配置当前子应用的相关信息。
* migrations目录用于存放数据库迁移历史文件。
* models.py文件用户保存数据库模型类。
* tests.py文件用于开发测试用例，编写单元测试。
* views.py文件用于编写Web应用视图。

注册子应用，在工程配置文件settings.py中，INSTALLED\_APPS项保存了工程中已经注册安装的子应用，初始工程中的INSTALLED\_APPS如下：

1. INSTALLED\_APPS=[
2. *#adminlte3的相关配置*
3. 'adminlte3',
4. 'adminlte3\_theme',
5. 'django.contrib.admin',
6. 'django.contrib.auth',
7. 'django.contrib.contenttypes',
8. 'django.contrib.sessions',
9. 'django.contrib.messages',
10. 'django.contrib.staticfiles',
11. *#注册子应用*
12. 'weather\_forecast.apps.WeatherForecastConfig',
13. ]

注册安装一个子应用，即是将子应用的配置信息文件apps.py中的Config类添加到INSTALLED\_APPS列表中。

1. from django.apps import AppConfig
2. class WeatherForecastConfig(AppConfig):
3. default\_auto\_field='django.db.models.BigAutoFiel'
4. name='weather\_forecast'

例如，将刚创建的weather\_forecast子应用添加到工程中，可在INSTALLED\_APPS列表中添加：'weather\_forecast.apps.WeatherForecastConfig’

##### 3.1.2.2.2.定义模型类

当前天气预报信息表

表1：当前天气预报信息表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 字段说明 |
| id | BigAutoField | 主键 |
| city\_name | CharField | 城市名 |
| current\_time | CharField | 当前时间 |
| full\_time | CharField | 完整时间 |
| weather\_condition | CharField | 天气情况 |
| weather\_forecast | CharField | 天气预报 |
| current\_weather\_picture | CharField | 当前天气图 |
| current\_background\_picture | CharField | 当前背景图片 |
| current\_temperature | CharField | 当前温度 |
| somatosensory\_temperature | CharField | 体感温度 |
| sunrise\_time | CharField | 日出时间 |
| sunset\_time | CharField | 日落时间 |
| air\_quality\_index | CharField | 空气质量指数 |
| moisture\_content | CharField | 湿度 |
| wind\_speed | CharField | 风速 |
| visibility | CharField | 能见度 |
| atmospheric\_pressure | CharField | 气压 |
| dew\_point | CharField | 露点 |

多日天气预报信息表

表2：多日天气预报信息表

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 字段说明 |
| id | BigAutoField | 主键 |
| city\_name | CharField | 城市名 |
| weather\_picture | CharField | 天气图 |
| highest\_temperature | CharField | 最高气温 |
| minimum\_temperature | CharField | 最低气温 |
| holiday | CharField | 节日 |

在models.py定义模型类，代码如下：

1. from django.db import models
2. *#Create your models here.*
3. *#准备当前天气预报的模型类*
4. class current\_weather\_forecast(models.Model):
5. *#current\_id=models.BigAutoField('当前天气预报id')*
6. city\_name=models.CharField(max\_length=200,verbose\_name='城市名')
7. current\_time=models.CharField(max\_length=200,verbose\_name='当前时间')
8. full\_time=models.CharField(max\_length=200,verbose\_name='完整时间')
9. weather\_condition=models.CharField(max\_length=20,verbose\_name='天气情况')
10. weather\_forecast=models.CharField(max\_length=200,verbose\_name='天气预报')
11. current\_weather\_picture=models.CharField(max\_length=200,verbose\_name='当前天气图')
12. current\_background\_picture=models.CharField(max\_length=200,verbose\_name='当前背景图片')
13. current\_temperature=models.CharField(max\_length=20,verbose\_name='当前温度')
14. somatosensory\_temperature=models.CharField(max\_length=20,verbose\_name='体感温度')
15. sunrise\_time=models.CharField(max\_length=100,verbose\_name='日出时间')
16. sunset\_time=models.CharField(max\_length=100,verbose\_name='日落时间')
17. air\_quality\_index=models.CharField(max\_length=20,verbose\_name='空气质量指数')
18. moisture\_content=models.CharField(max\_length=20,verbose\_name='湿度')
19. wind\_speed=models.CharField(max\_length=20,verbose\_name='风速')
20. visibility=models.CharField(max\_length=20,verbose\_name='能见度')
21. atmospheric\_pressure=models.CharField(max\_length=20,verbose\_name='气压')
22. dew\_point=models.CharField(max\_length=20,verbose\_name='露点')
23. class My:
24. db\_table='current\_weather\_forecast'
25. verbose\_name\_plural='当前天气预报'
26. def\_\_str\_\_(self):
27. """将模型类以字符串的方式输出"""
28. return self.city\_name
29. *#准备多日天气预报的模型类*
30. class multi\_day\_weather\_forecast(models.Model):
31. *#multi\_day\_id=models.BigAutoField('多天天气预报id')*
32. city\_name=models.CharField(max\_length=200,verbose\_name='城市名')
33. weather\_picture=models.CharField(max\_length=200,verbose\_name='天气图')
34. highest\_temperature=models.CharField(max\_length=20,verbose\_name='最高气温')
35. minimum\_temperature=models.CharField(max\_length=20,verbose\_name='最低气温')
36. holiday=models.CharField(max\_length=20,verbose\_name='节日')
37. class My:
38. db\_table='multi\_day\_weather\_forecast'
39. verbose\_name\_plural='多日天气预报'
40. def\_\_str\_\_(self):
41. """将模型类以字符串的方式输出"""
42. return self.city\_name

[8]

##### 3.1.2.2.3.注册模型类

在应用的admin.py文件中注册模型类：

1. from django.contrib import admin
2. *#Register your models here.*
3. from weather\_forecast.models import\*
4. @admin.register(current\_weather\_forecast)
5. class current\_weather\_forecastAdmin(admin.ModelAdmin):
6. list\_display=('city\_name',
7. 'current\_time',
8. 'full\_time',
9. 'weather\_condition',
10. 'weather\_forecast',
11. 'current\_weather\_picture',
12. 'current\_background\_picture',
13. 'current\_temperature',
14. 'somatosensory\_temperature',
15. 'sunrise\_time',
16. 'sunset\_time',
17. 'air\_quality\_index',
18. 'moisture\_content',
19. 'wind\_speed',
20. 'visibility',
21. 'atmospheric\_pressure',
22. 'dew\_point')*#列表显示字段*
23. @admin.register(multi\_day\_weather\_forecast)
24. class multi\_day\_weather\_forecastAdmin(admin.ModelAdmin):
25. list\_display=('city\_name',
26. 'weather\_picture','highest\_temperature',
27. 'minimum\_temperature',
28. 'holiday')*#列表显示字段*

##### 3.1.2.2.4.模型迁移

在PyCharm终端运行：

1. *#生成迁移文件：根据模型类生成创建表的语句*
2. python manage.py makemigrations
3. *#执行迁移：根据第一步生成的语句在数据库中创建表*
4. python manage.py migrate

迁移成功后，在PyCharm终端创建后台管理用户及密码:

1. *#输入命令*
2. python manage.py createsuperuser*#按照提示输入用户名密码信息*

再次进入后台登录界面，输入设置好的用户名密码就可以进入后台了。

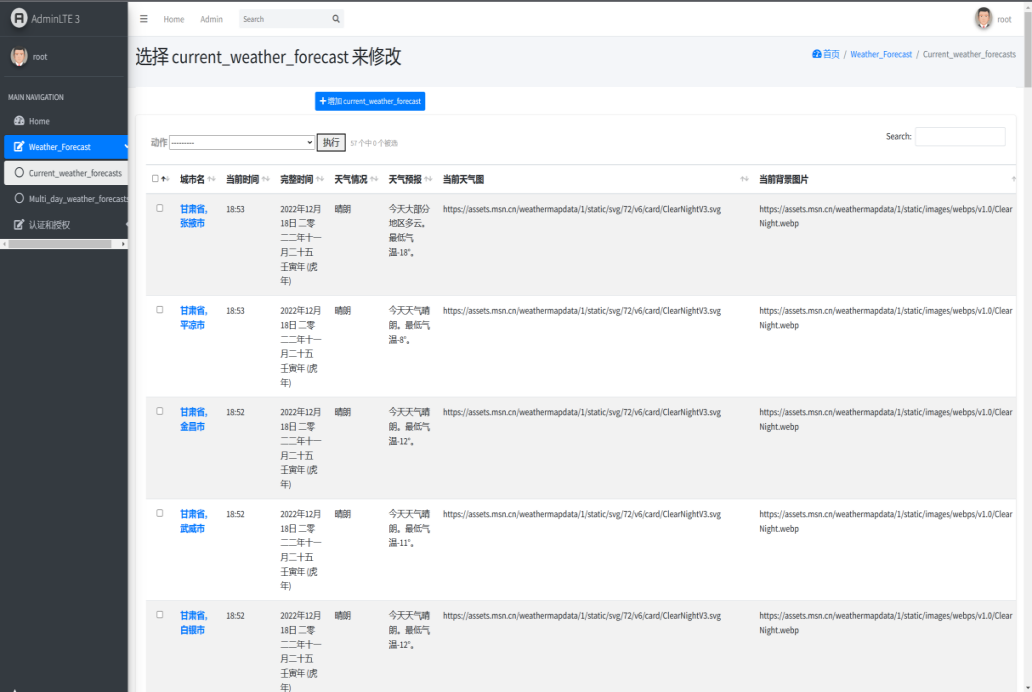


图37 数据在后台显示，这里的后台是经过AdminLTE3美化后的

#### 3.1.2.3操作数据库

##### 3.1.2.3.1.爬虫数据存入数据库

1. *#将数据写入数据库*
2. import pymysql
3. from zhdate import\*
4. conn=pymysql.Connect(host='127.0.0.1',
5. port=3306,
6. user='root',
7. passwd='123456',
8. db='mydatabase',
9. charset='utf8')
10. print('连接数据库')
11. cur=conn.cursor()
12. *#将当前天气信息写入weather\_forecast\_current\_weather\_forecast*
13. cur.execute(
14. "INSERT INTO`weather\_forecast\_current\_weather\_forecast`(`city\_name`,`current\_time`,`full\_time`,`weather\_condition`,`weather\_forecast`,`current\_weather\_picture`,`current\_background\_picture`,`current\_temperature`,`somatosensory\_temperature`,`sunrise\_time`,`sunset\_time`,`air\_quality\_index`,`moisture\_content`,`wind\_speed`,`visibility`,`atmospheric\_pressure`,`dew\_point`)VALUES('%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s');"%(current[keys[0]][0],current[keys[1]][0],current[keys[2]][0],current[keys[3]][0],current[keys[4]][0],current[keys[5]][0],current[keys[6]][0],current[keys[7]][0],current[keys[8]][0],current[keys[9]][0],current[keys[10]][0],current[keys[11]][0],current[keys[12]][0],current[keys[13]][0],current[keys[14]][0],current[keys[15]][0],current[keys[16]][0],))
15. conn.commit()
16. *#将数据写入weather\_forecast\_multi\_day\_weather\_forecast*
17. cur.execute(
18. "INSERT INTO`weather\_forecast\_multi\_day\_weather\_forecast`(`city\_name`,`weather\_picture`,`highest\_temperature`,`minimum\_temperature`,`holiday`)VALUES('%s','%s','%s','%s','%s');"%(
19. multi\_day[keys[0]],multi\_day[keys[1]][i],multi\_day[keys[2]][i],multi\_day[keys[3]][i],
20. multi\_day[keys[4]][i]))
21. conn.commit()
22. *#断开数据库连接*
23. conn.close()
24. print('连接关闭')

[9]

##### 3.1.2.3.2.在PyCharm里操作数据库

找到PyCharm右边的Database插件，然后点击进去，第一次用要下载一个驱动(它会爆红提示)

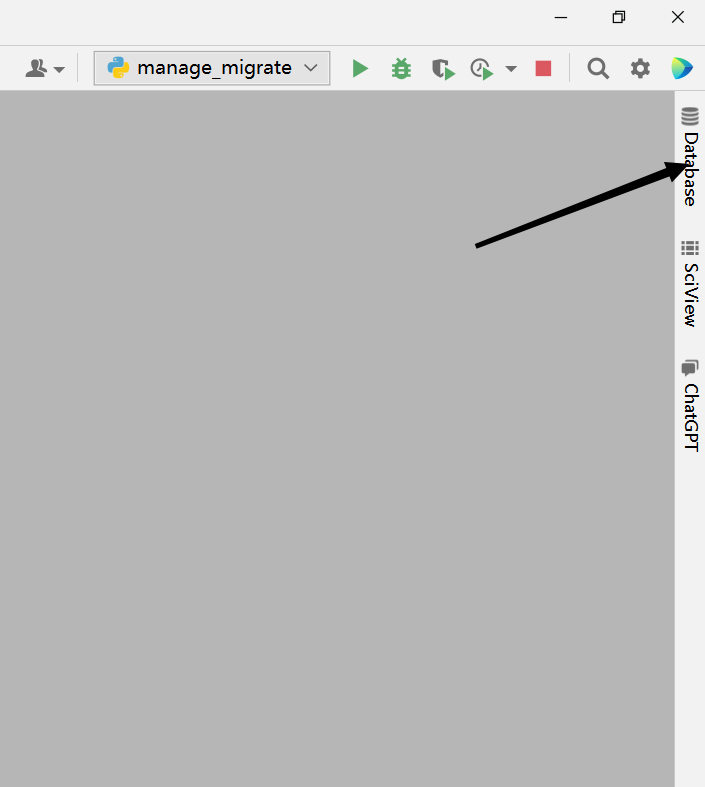


图38 Database插件

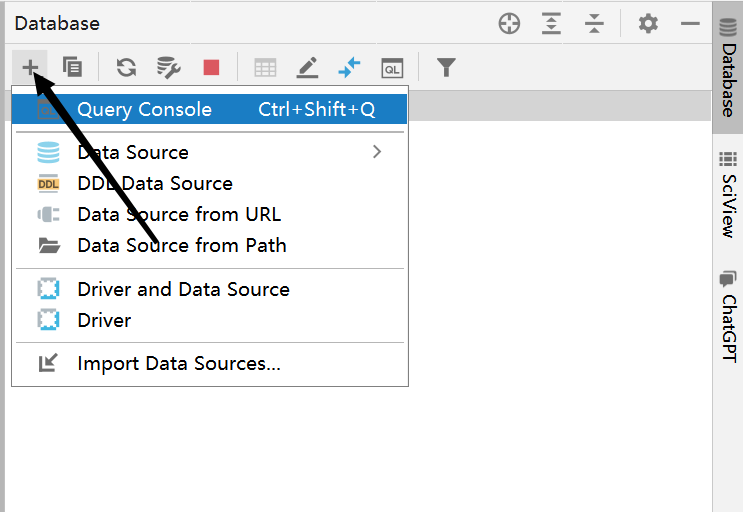


图39 选择Data Source

点击Data Source后选择MySQL

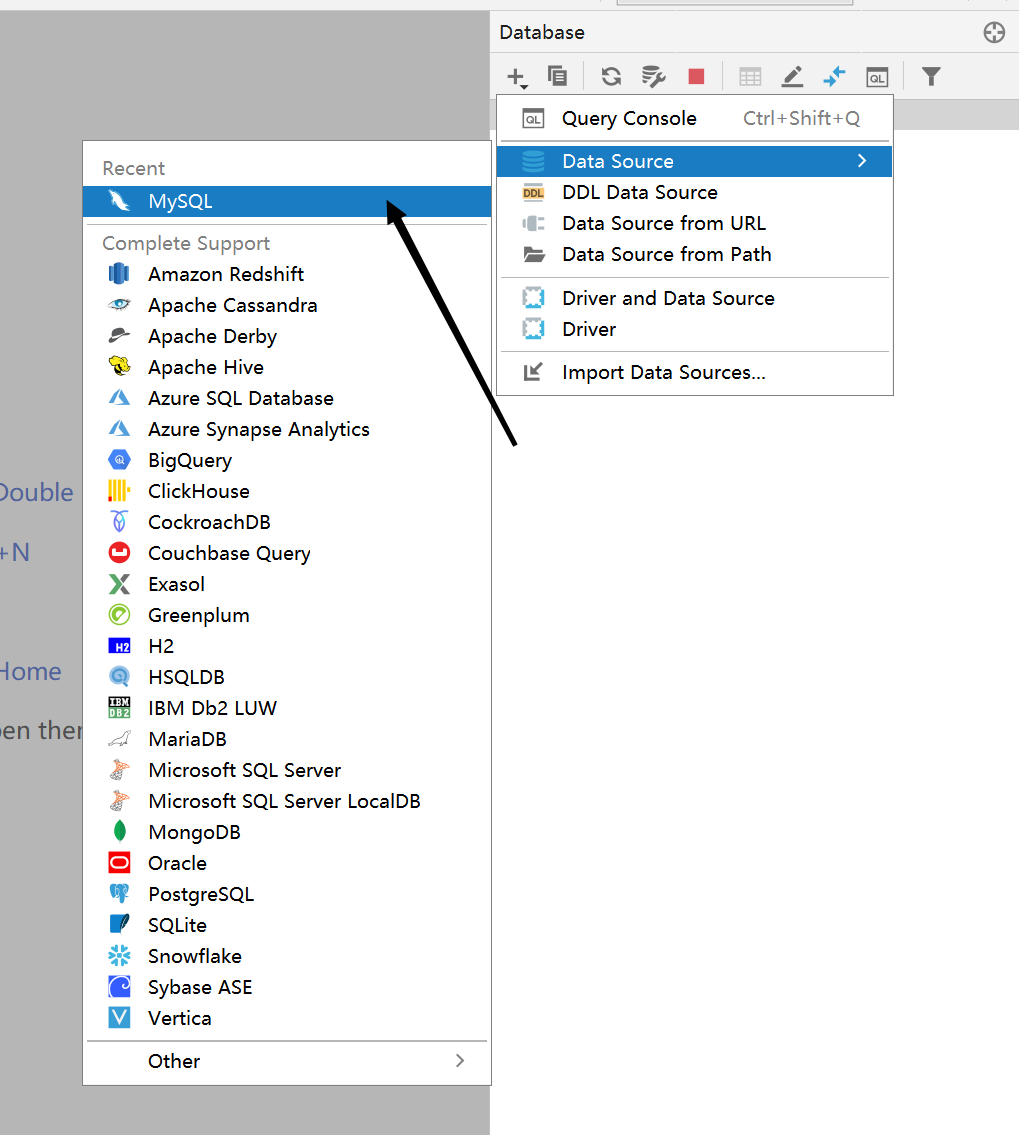
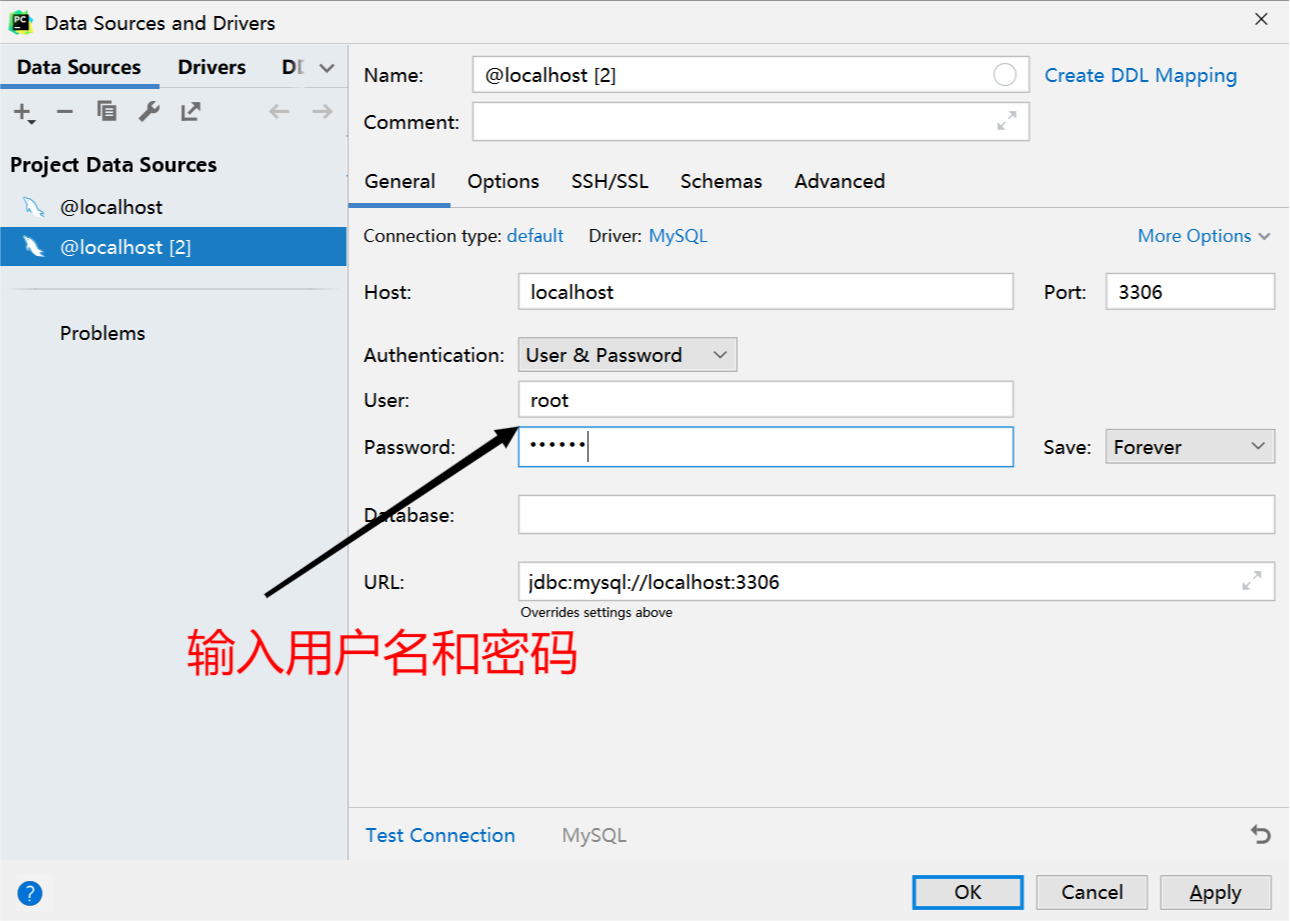
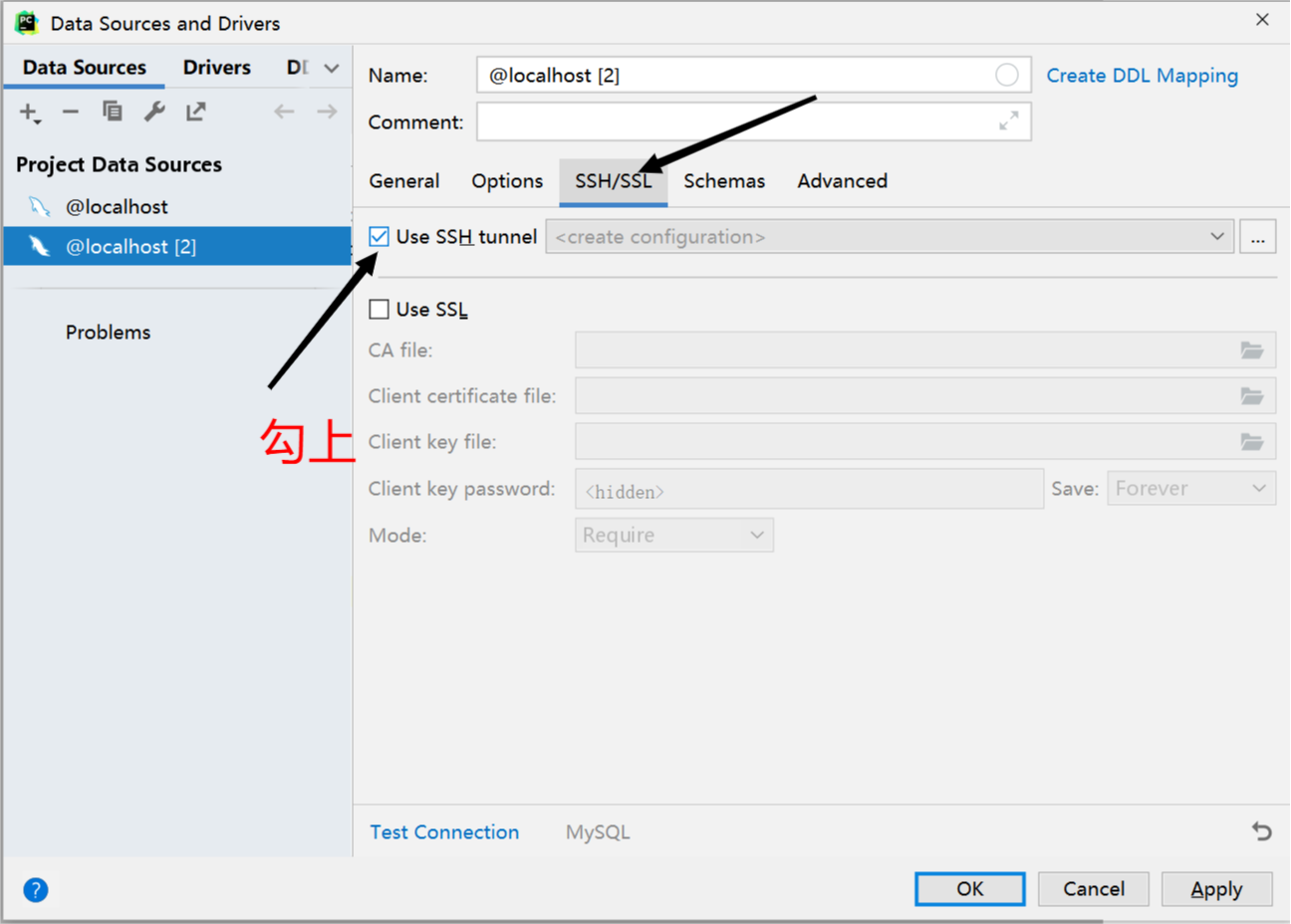


图40 选择MySQL





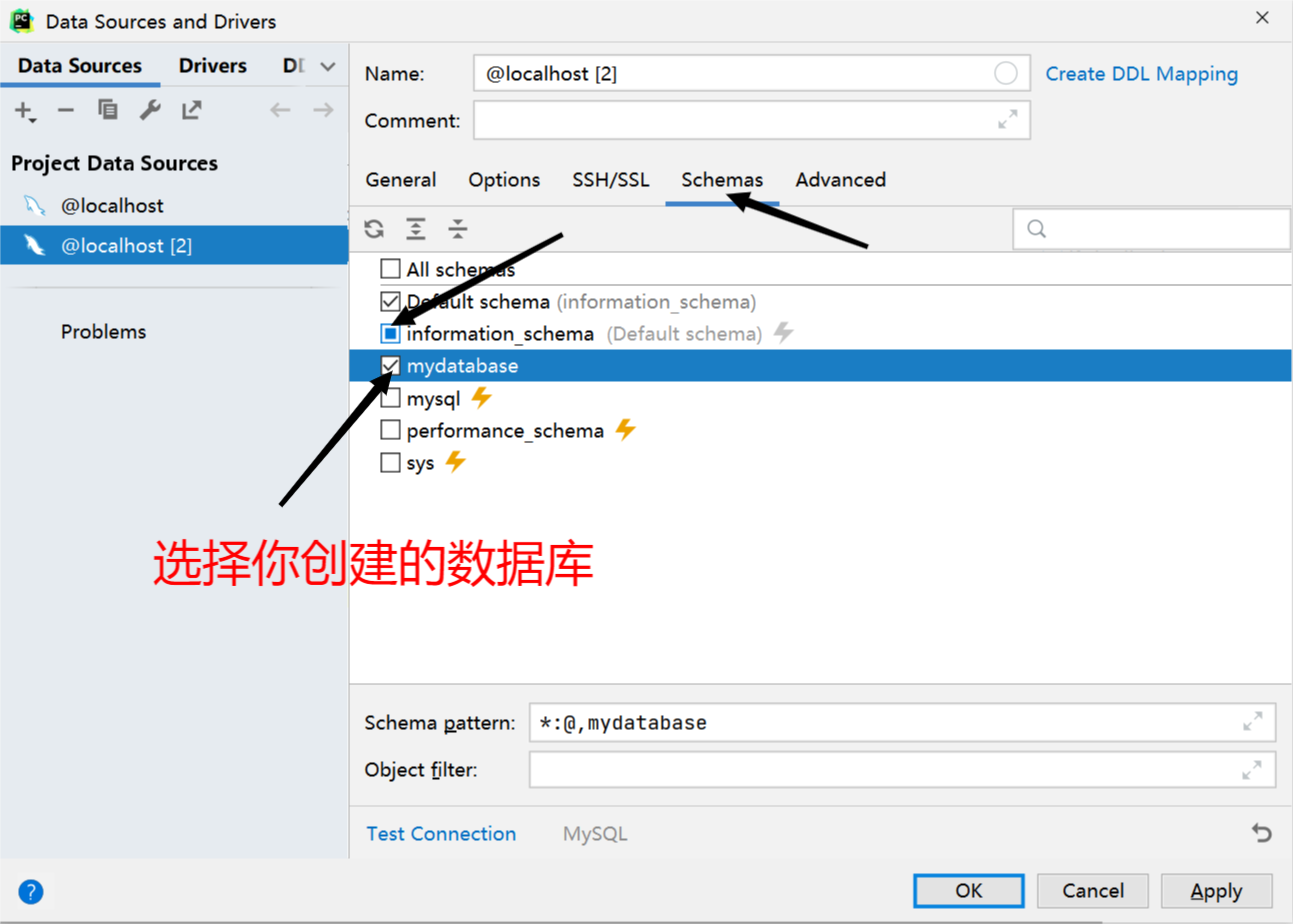


图41-43相关配置

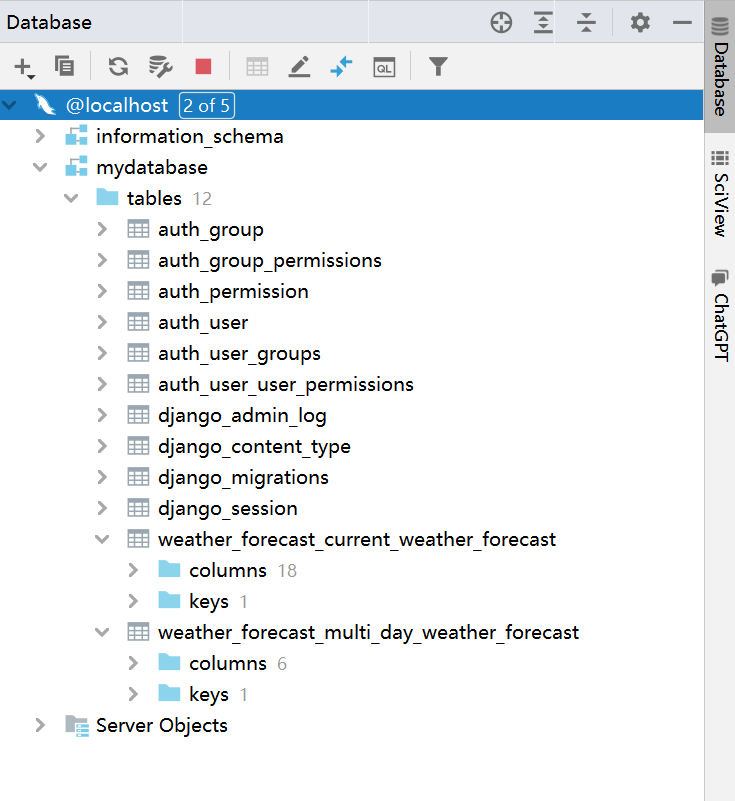


图44 成功建好

这里需要注意到的一点就是，你在这个插件上修改（表）增加（表）并不会影响Django后台的，这个插件的主要功能就是方便你查询数据，修改数据。

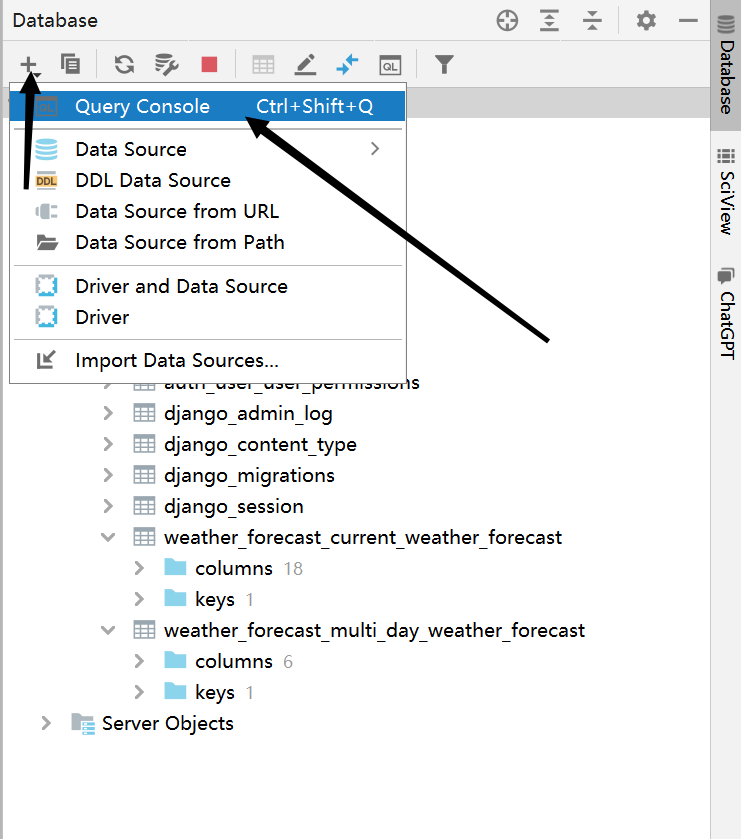


图45 打开命令窗口

在命令窗口即可输入SQL语句和cmd打开运行的效果是一样的，如果不清楚SQL语句可以右击你想要修改的地方：

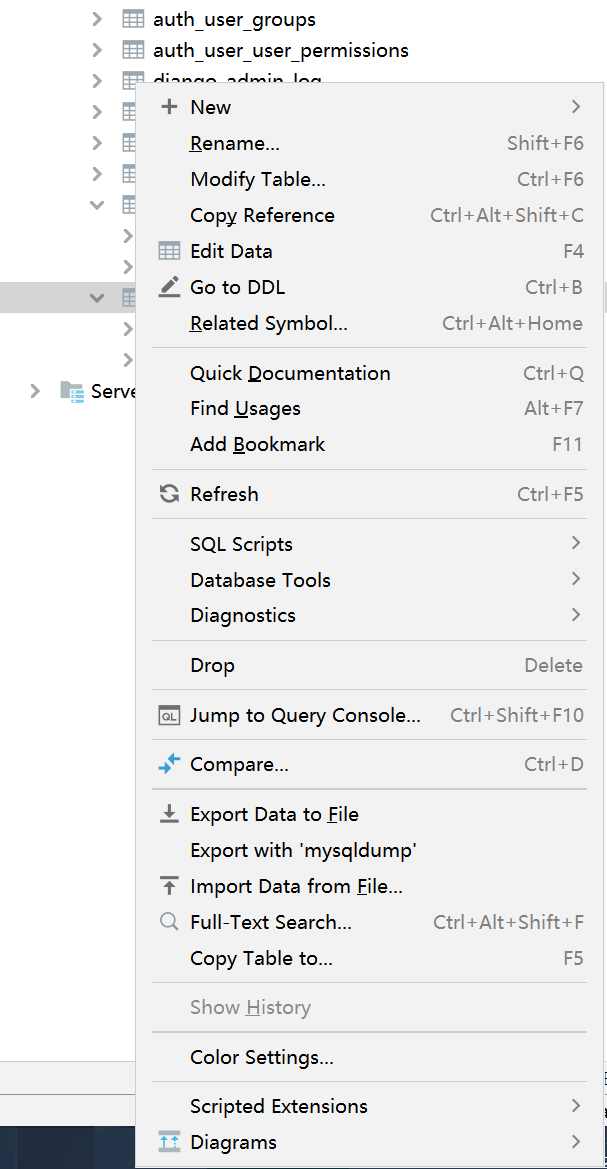


图46操作数据库

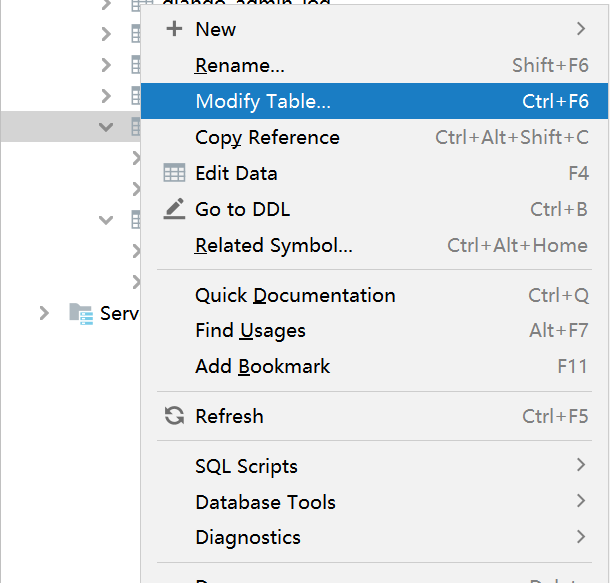


图47 选择你所要进行的操作，点击

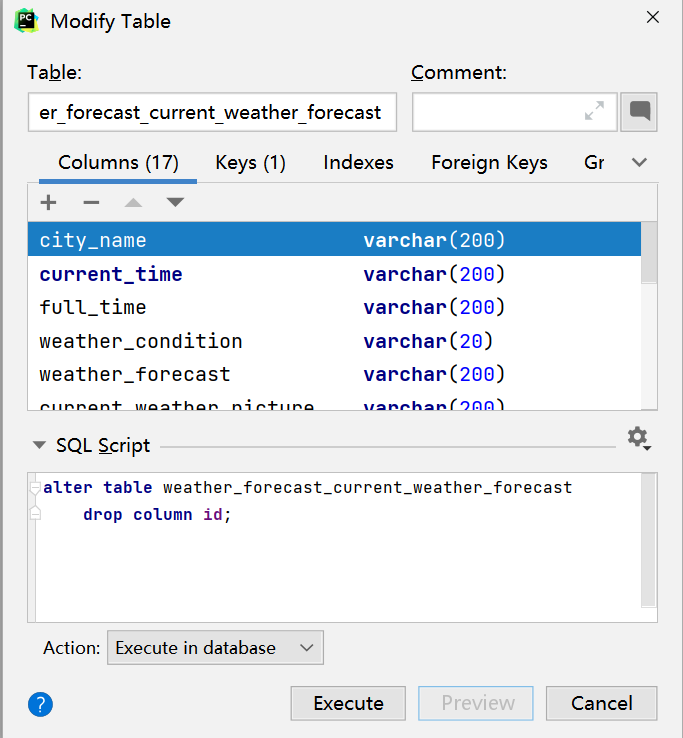


图48 SQL Script会自动给你生成相应的SQL语句

### 3.1.3 AdminLTE3美化

修改一下后台的模板把后台的模板修改为AdminLTE3

下载一个第三方的模块Django-adminlte

(1)在第三方包下载处下载，或在Pycharm终端运行以下命令

1. pip install django-adminlte3

(2)settings.py里面开发插件

1. INSTALLED\_APPS=[
2. *#General use templates&template tags(should appear first)*
3. 'adminlte3',
4. *#Optional:Django admin theme(must be before django.contrib.admin)你的插件必须放在django.contrib.admin的前面*
5. 'adminlte3\_theme',]

(3)这个插件也有一些表生成这个模板的一些模板表使用迁移命令操作数据库

1. python manage.py migrate

(4)同步静态资源

现在settings.py里面添加两个配置

1. STATIC\_URL='/static/'
2. STATICFILES\_DIRS=[os.path.join(BASE\_DIR,'static')]

再执行

1. python manage.py collectstatic

## 3.2前端实现

本次作业前端实现主要采用模板技术，下载一个仿站小工具[10]爬取所需网页获取相应的html、js、css等文件，然后在配置urls路由，在views视图里创建函数，调用html。html文件放在文件夹template存放模板的，而文件夹static存放静态资源css js images

在项目文件夹下的urls文件配置路由：

1. from django.contrib import admin
2. from django.urls import path
3. *#配置路由*
4. import weather\_forecast.views
5. urlpatterns=[
6. path('admin/',admin.site.urls),
7. path('weather\_forecast/line\_graph7/',weather\_forecast.views.line\_graph7),
8. path('weather\_forecast/line\_graph15/',weather\_forecast.views.line\_graph15),
9. path('weather\_forecast/line\_graph45/',weather\_forecast.views.line\_graph45),
10. path('weather\_forecast/',weather\_forecast.views.weather\_forecast,name='kd'),
11. ]

然后到子应用文件夹下的views文件下新建函数即可。

使用仿站小工具的exe可执行文件下载对应网页的页面。



图49 仿站小工具使用

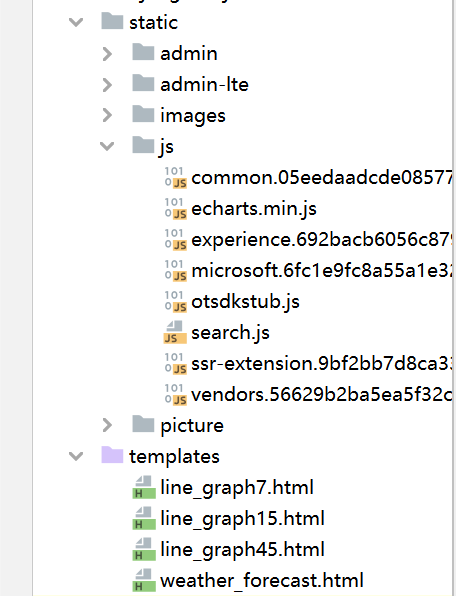


图50 网页文件放置

### 3.2.1主要功能实现

#### 3.2.1.1折线图

折线图主要采用第三方的js文件来渲染(echarts)，下载好js文件将其放置于项目的static文件夹下的js文件夹[11]。

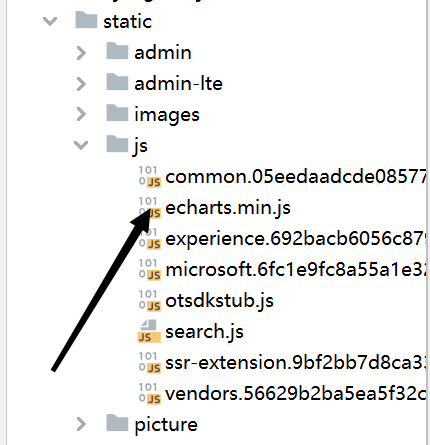


图51 echarts.min.js放置

然后在html文件里创建实例。

如line\_graph7.html中的相关代码。

#### 3.2.1.2发送错误信息

在html文件中采用表单的形式发送搜索的城市名，网页请求对应的urls调用views里定义的函数：

1. *#实现搜索*
2. check=request.GET.get('kd')
3. *#调用urllib.parse对get获得的信息解码*
4. keyword=urllib.parse.unquote(check)
5. city=current\_weather\_forecast.objects.filter(city\_name\_\_contains=keyword)
6. print(len(city))
7. if len(city)==0:
8. info='暂无数据'

然后通过return返回info信息，相应html文件中有判断语句：

1. {%if info%}
2. <script>
3. window.alert('{{info}}');
4. </script>
5. {%endif%}

跳出提示窗口：

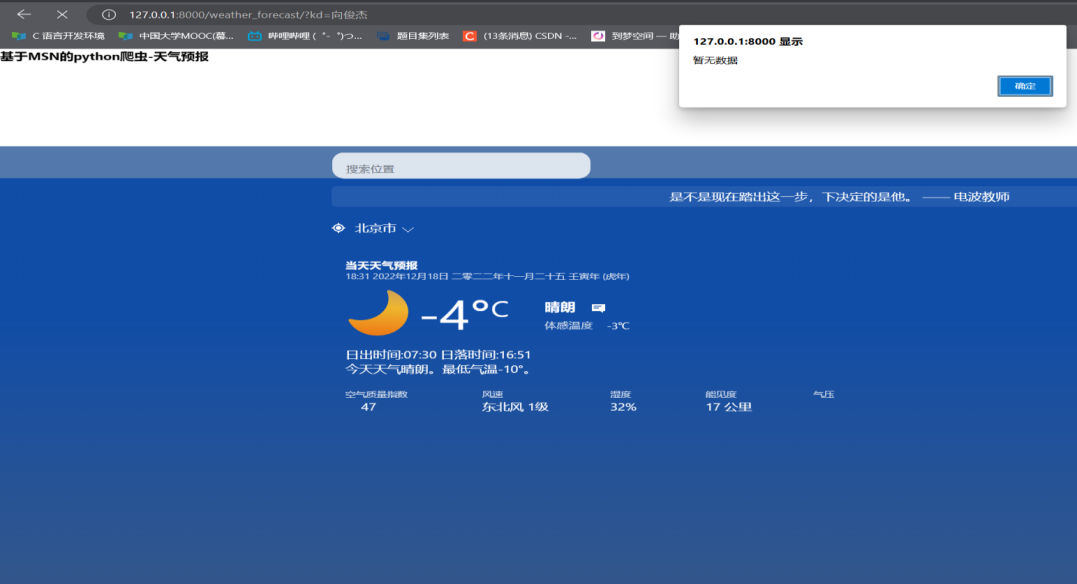


图52 提示窗口

# 4. 网络服务器部署+爬虫部署

## 4.1网络服务器部署

阿里云服务器的购买,学生认证可免费领一台(http://www.aliyun.com)使用远程工具去连接它mobaxtream(https://mobaxterm.mobatek.net/download.html),宝塔运维面板的安装，在mobaxtream中打开命令指示符输入：yum install-y wget&&wget-O install.sh http://download.bt.cn/install/install\_6.0.sh&&sh install.sh，然后：

1. 安装成功之后会自动生成访问地址+用户名+密码。
2. 安装服务器环境LNMP=NGINX+MYSQL+PHP+FTP工具。
3. 安装python项目管理器软件商店->宝塔插件->python项目管理器->设置->版本管理-安装Python。

再部署我们Django的项目：

1. 准备工作:将项目打包上传到宝塔上去。
2. 将模块打包pip freeze>requirements.txt。
3. 项目资源打包将项目中.idea还有env虚拟环境全部删除在上传到宝塔。
4. 数据库导出脚本到数据库的脚本并且上传到宝塔的数据库。
5. 修改配置：
6. #在settings.py中
7. DEBUG=True调试模式修改为False
8. ALLOWED\_HOSTS=[]在['\*']添加'\*'表示任何人都可以访问
9. STATIC\_ROOT='static'静态访问放过
10. 在urls.py
11. 放过的配置
12. 导入相关的包Django 4.0的版本就是这么配置的
13. from django.urls import path,include,re\_path
14. from django.views import static
15. from django.conf import settings
16. 动态匹配
17. urlpatterns=[
18. 添加这个配置就可以
19. re\_path(r'^static/(?P<path>.\*)$',static.serve,{'document\_root':settings.STATIC\_ROOT},name='static')
20. ]

最后绑定一个二级域名：

1. 绑定主机的IP。
2. 使用二级域名来完成绑定。

## 4.2爬虫部署

布置脚本[12]。

# 5. 用户说明

爬虫文件在Django项目下的python天气爬虫.py，直接右击运行即可。



Django项目直接运行mange.py这个文件即可启动，进入蓝色网站后，在8000后面/admin/即可进入管理人员界面查看数据。

在网页网址中输入http://127.0.0.1:8000/weather\_forecast/即可进入天气预报页面，点击搜索框输入你想要查询的城市名后点击回车，即可查询，搜索框下是调用的一言网接口获得的名言名句，刷新即可重新调用接口。在所查询页面中，点击查看天气预报折线图即可查看7、15、45的天气折线图。点击7日天气预报并接着点下去即可查看全部45天的天气预报展示。



# 6. 参考文献

1. Microsoft edge驱动器下载以及使用https://blog.csdn.net/ weixin\_50233989/article/details/113242332
2. 配置环境变量https://jingyan.baidu.com/article/00a07f387 6cd0582d128dc55.html
3. 镜像文件下载地址pypi.org(由于国内访问该网站速度慢，可以 使用华为、阿里、清华的镜像网站)
4. lxml官方文档http://www.w3school.com.cn/xpath/index.asp
5. 【正则表达式1-《python零基础到全栈系列》-哔哩哔哩】htt ps://b23.tv/BJsE88b
6. selenium官方文档The Selenium Browser Automation Proje ct|Selenium
7. 同步数据库 https://www.cnblogs.com/wangdianchao/p/12561 931.html#:~: Django框架基础-定义模型类（model） https:// blog.csdn.net/qq\_38582906/article/details/106772125
8. Python写入MySQL数据库 https://blog.csdn.net/m0\_4344566 8/article/details/84111157
9. 仿站小工具下载https://smalltool.github.io/files/%E4%BB% BF%E7%AB%99%E5%B0%8F%E5%B7%A5%E5%85%B7V10.6.zip
10. echarts教程 https://www.runoob.com/echarts/echarts-in stall.html

Echarts利用多X轴实现七天天气预报效果 https://blog.csdn. net/FengChSvip/article/details/1209507 31

1. 宝塔上布置爬虫脚本http://t.csdn.cn/veM5g、http://t.csd n.cn/0CXjl、http://t.csdn.cn/hgtuz