东方瑞通
终身学习
Founded in 1998

Python疫情监控实战

讲师 | 朱明虎

目录

CONTENTS

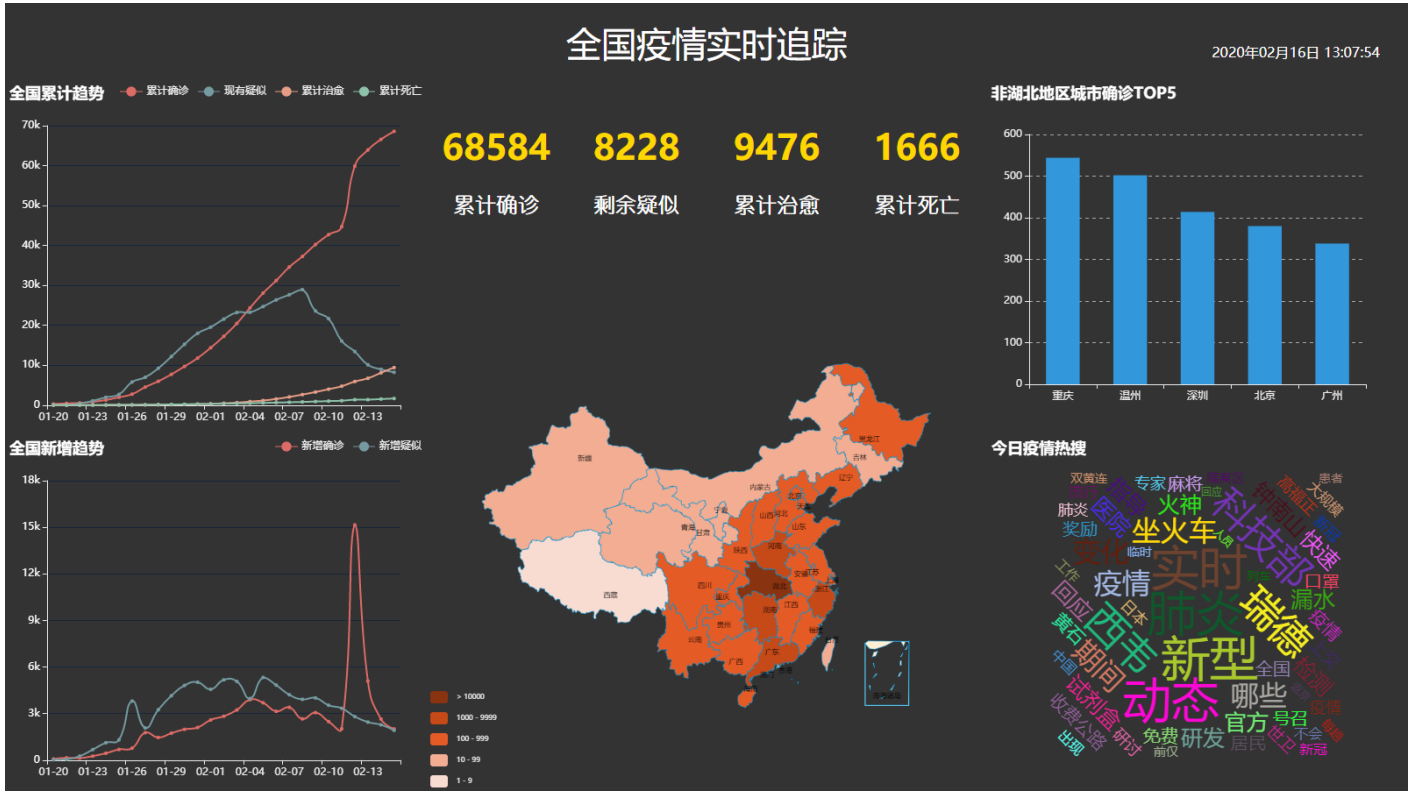
1.1 项目介绍

本项目是一个基于 Python + Flask + Echarts 打造的一个疫情监控系统，涉及到的技术有：

- Python 网络爬虫
- 使用 Python 与 MySQL 数据库交互
- 使用 Flask 构建 web 项目
- 基于 Echarts 数据可视化展示
- 在 Linux 上部署 web 项目及爬虫

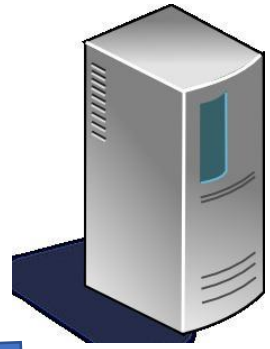
1.1 项目介绍

效果展示：

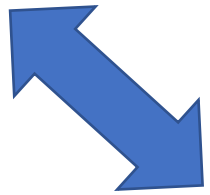

全国疫情实时追踪
2020年02月16日 13:07:54
全国累计趋势
累计确诊
现有疑似
累计治愈
累计死亡
68584
累计确诊
8228
剩余疑似
9476
累计治愈
1666
累计死亡
非湖北地区城市确诊TOP5
重庆
温州
深圳
北京
广州
全国新增趋势
新增确诊
新增疑似
今日疫情热搜
> 10000
1000 - 9999
100 - 999
10 - 99
1 - 9

1.2 项目架构

数据获取



flask 搭建 web 后台



数据持久化

数据可视化 h5 + echarts

1.3 项目环境准备

- Python 3.x
- MySQL
- PyCharm （Python IDE）
- Jupyter notebook (Pyhon IDE)
- Hbuild（前端 IDE，https://www.dcloud.io/hbuilderx.html）
- Linux 主机（后期项目部署）

1.4 notebook

Jupyter Notebook（此前被称为 IPython notebook）是一个基于网页的用于交互计算的应用程序，在数据科学领域很受欢迎。

简言之，notebook 是以网页的形式打开，可以在 code 类型单元格中直接编写代码和运行代码，代码的运行结果也会直接在代码块下显示。如在编程过程中需要编写说明文档，可在 markdown 类型的单元格中直接编写，便于作及时的说明和解释。

1.4 notebook

- 安装

  pip install notebook

- 启动:

  jupyter notebook

- 修改工作目录

  ① jupyter notebook --generate-config

  ② 编辑 jupyter_notebook_config.py 文件

```
260  ## The directory to use for notebooks and kernels.
261  c.NotebookApp.notebook_dir = r"c:\Users\admin\Desktop\notebook"
262
```

1.4 notebook

- notebook 的基本操作

① 新建文件与导入文件

② 单元格分类：code 、markdown

③ 命令模式（蓝色边框）与编辑模式（绿色边框）

④ 常用快键键

单元格类型转换：Y 、M；　　插入单元格：A、B；　　进入命令模式：Esc　　代码补全：Tab

运行单元格：ctrl / shift / alt + enter　　删除单元格：DD

1.4 notebook

- markdown 语法
  - ① 标题：使用1 ~ 6个 # 跟随一个空格来表示1 ~ 6级标题
  - ② 无序列表：使用 *，- 或 + 后跟随一个空格来表示
  - ③ 有序列表：使用数字+点表示
  - ④ 换行：使用两个或以上的空行
  - ⑤ 代码：可以使用 `代码` (反引号)来标记代码部分，使用 ```语言 标记代码块
  - ⑥ 分割线：3个星号 *** 或 3个减号 ---
  - ⑦ 链接与图片： [文字](链接地址) ![图像说明](图片链接地址 "图片说明信息")

目录

CONTENTS

2.1 爬虫概述

爬虫，就是给网站发起请求，并从响应中提取需要的数据的自动化程序

① 发起请求，获取响应

通过 http 库，对目标站点进行请求。等同于自己打开浏览器，输入网址

常用库：urllib、urllib3、requests

服务器会返回请求的内容，一般为：html、二进制文件（视频，音频）、文档，json 字符串等

② 解析内容

寻找自己需要的信息，就是利用正则表达式或者其他库提取目标信息

常用库：re、beautifulsoup4

③ 保存数据

将解析得到的数据持久化到文件或者数据库中

2.1 爬虫概述

- 使用 **urllib** 发送请求
  - request.urlopen()

```
from urllib import request

url = "http://www.baidu.com"

res = request.urlopen(url) #访问url并获得响应

print(res.geturl()) #获取主机地址
print(res.getcode()) #获取请求状态码
print(res.info())  #获取响应头

html = res.read() #取获取的是字节形式的内容
# print(html)
html.decode("utf-8") #解码
print(html)
```

```
from urllib import request

url = "http://www.dianping.com"
header = {
    "User-Agent":"Mozilla/6.0 (iPhone; CPU iPh
}
req = request.Request(url,headers=header)  #添
res = request.urlopen(req) #访问url并获得响应

print(res.geturl()) #获取主机地址
print(res.getcode()) #获取请求状态码
print(res.info())  #获取响应头

html = res.read() #取获取的是字节形式的内容
# print(html)
html.decode("utf-8") #解码
```

2.1 爬虫概述

- 使用 **requests** 发送请求
  - 安装: pip install requests
  - requests.get()

```
import requests

url = 'http://www.baidu.com'

resp = requests.get(url) #发起请求
print(resp.encoding) #查看编码
print(resp.status_code) #获取状态码
html = resp.text
# print(html)
resp.encoding = "utf-8"
html = resp.text
print(html)
```

```
import requests

url = 'http://www.dianping.com'
header = {
    "Host": "www.dianping.com",
    "User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
resp = requests.get(url,headers= header) #发起请求
print(resp.encoding) #查看编码
print(resp.status_code) #获取状态码
# html = resp.text
# print(html)
resp.encoding = "UTF-8"
html = resp.text
# print(html)
```

2.1 爬虫概述

- 使用 **beautifulsoup4** 解析内容

  beautifulsoup4 将复杂的 HTML 文档转换成一个树形结构,每个节点都是 Python 对象

  - 安装：pip install beautifulsoup4
  - BeautifulSoup(html)
    - 获取节点：find()、find_all()/select()、
    - 获取属性：attrs
    - 获取文本：text

```
import requests
from bs4 import BeautifulSoup

url = "http://wsjkw.sc.gov.cn/scwsjkw/gzbd/fyzt.shtml"
resp =requests.get(url)
print(resp.encoding)
resp.encoding = "utf-8"
html = resp.text
soup = BeautifulSoup(html)
for i in soup.find_all("a"):
    print(i)
```

```
ISO-8859-1
<a href="/scwsjkw/gzbd01/2020/2/14/293299f6284b4692aaeabadfd3e94849.
肺炎疫..." src="/scwsjkw/gzbd01/2020/2/14/293299f6284b4692aaeabadfd3
"/></a>
<a href="/scwsjkw/gzbd01/2020/2/14/293299f6284b4692aaeabadfd3e94849.
疫情最新情况"><span>【最新】</span>截至2月13日24时我省新型冠状病毒肺
```

2.1 爬虫概述

- 使用 **re** 解析内容
  - re 是 python 自带的正则表达式模块，使用它需要有一定的 ***正则表达式*** 基础
  - re.search(regex ,str)
    - ① 在 str 中查找满足条件的字符串，匹配不上返回 None
    - ② 对返回结果可以分组，可在字符串内添加小括号分离数据：

      groups()

      group(index)：返回指定分组内容

```
import re

html="2月12日0-24时，我省新型冠状病毒肺炎新增确诊病例15例，治愈出院
confirm_add_patten = "新增确诊病例(\d+)"
confirm_add = re.search(confirm_add_patten,html)
print(confirm_add)
print(confirm_add.groups())
print(confirm_add.group(0))
print(confirm_add.group(1))
```

```
<re.Match object; span=(21, 29), match='新增确诊病例15'>
('15',)
新增确诊病例15
15
```

2.2 爬取腾讯疫情数据

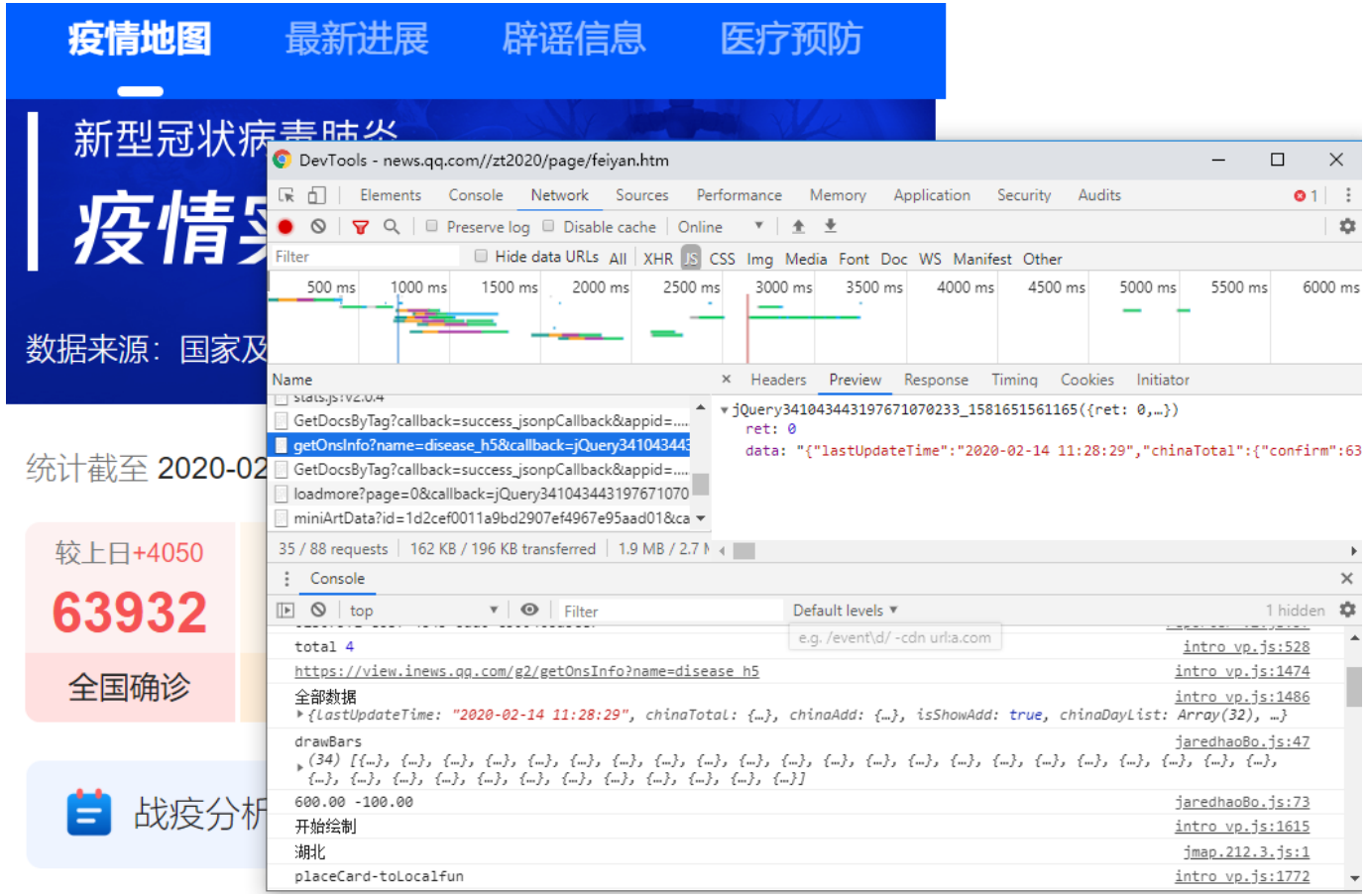
- 有了爬虫基础后，我们可以自行去全国各地的卫健委网站上爬取数据，不过部分网站反爬虫手段很高明，需要专业的反反爬手段
- 我们也可以去各大平台直接爬取最终数据，比如：

  https://voice.baidu.com/act/newpneumonia/newpneumonia/?from=osari_pc_1

  https://news.qq.com//zt2020/page/feiyan.htm

2.2 爬取腾讯疫情数据

- 意外收获



2.2 爬取腾讯疫情数据

- 获取所有病情数据

```
url = 'https://view.inews.qq.com/g2/getOnsInfo?name=disease_h5'
headers = {
    'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebl
}
r = requests.get(url, headers)
res = json.loads(r.text)  # json字符串转字典
data_all = json.loads(res['data'])
```

2.2 爬取腾讯疫情数据

- 分析与处理

```
lastUpdateTime  #最后更新时间
chinaTotal      #总数
chinaDayList    #历史记录
chinaDayAddList #历史新增记录
areaTree:-name        # areaTree[0]，中国数据
         -today
         -total
         -children:-name  #省级数据，列表
                   -today
                   -total
                   -children:-name #市级数据，列表
                             -today
                             -total
```

2.2 爬取腾讯疫情数据

- 分析与处理

```
history = {}  # 历史数据
for i in data_all["chinaDayList"]:
    ds = "2020." + i["date"]
    tup = time.strptime(ds, "%Y.%m.%d")
    ds = time.strftime("%Y-%m-%d", tup)  # 改变时间格式,不然插入数据库会报错，数据库是datetime类型
    confirm = i["confirm"]
    suspect = i["suspect"]
    heal = i["heal"]
    dead = i["dead"]
    history[ds] = {"confirm": confirm, "suspect": suspect, "heal": heal, "dead": dead}
for i in data_all["chinaDayAddList"]:
    ds = "2020." + i["date"]
    tup = time.strptime(ds, "%Y.%m.%d")
    ds = time.strftime("%Y-%m-%d", tup)
    confirm = i["confirm"]
    suspect = i["suspect"]
    heal = i["heal"]
    dead = i["dead"]
    history[ds].update({"confirm_add": confirm, "suspect_add": suspect, "heal_add": heal, "dead_add": dead})
```

```
details = []  # 当日详细数据
update_time = data_all["lastUpdateTime"]
data_country = data_all["areaTree"]  # list 25个国家
data_province = data_country[0]["children"]  # 中国各省
for pro_infos in data_province:
    province = pro_infos["name"]  # 省名
    for city_infos in pro_infos["children"]:
        city = city_infos["name"]
        confirm = city_infos["total"]["confirm"]
        confirm_add = city_infos["today"]["confirm"]
        heal = city_infos["total"]["heal"]
        dead = city_infos["total"]["dead"]
        details.append([update_time, province, city, confirm, confirm_add, heal, dead])
```

2.2 爬取腾讯疫情数据

- 数据存储

history 表存储每日总数据，details 表存储每日详细数据

```
CREATE TABLE `history` (
  `ds` datetime NOT NULL COMMENT '日期',
  `confirm` int(11) DEFAULT NULL COMMENT '累计确诊',
  `confirm_add` int(11) DEFAULT NULL COMMENT '当日新增确诊',
  `suspect` int(11) DEFAULT NULL COMMENT '剩余疑似',
  `suspect_add` int(11) DEFAULT NULL COMMENT '当日新增疑似',
  `heal` int(11) DEFAULT NULL COMMENT '累计治愈',
  `heal_add` int(11) DEFAULT NULL COMMENT '当日新增治愈',
  `dead` int(11) DEFAULT NULL COMMENT '累计死亡',
  `dead_add` int(11) DEFAULT NULL COMMENT '当日新增死亡',
  PRIMARY KEY (`ds`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
CREATE TABLE `details` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `update_time` datetime DEFAULT NULL COMMENT '数据最后更新时间',
  `province` varchar(50) DEFAULT NULL COMMENT '省',
  `city` varchar(50) DEFAULT NULL COMMENT '市',
  `confirm` int(11) DEFAULT NULL COMMENT '累计确诊',
  `confirm_add` int(11) DEFAULT NULL COMMENT '新增确诊',
  `heal` int(11) DEFAULT NULL COMMENT '累计治愈',
  `dead` int(11) DEFAULT NULL COMMENT '累计死亡',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

2.2 爬取腾讯疫情数据

- 数据存储
  - 使用 pymysql 模块与数据库交互
  - 安装： pip install pymysql
    1. 建立连接
    2. 创建游标
    3. 执行操作
    4. 关闭连接

```python
import pymysql

# 创建连接
conn = pymysql.connect(host="122.51.251.91",
                       user="user1",
                       password="123456",
                       db="cov2019",
                       charset="utf8")
# 创建游标
cursor = conn.cursor()# 执行完毕返回的结果集默认以元组显示

# 执行操作
sql = "select * from details limit 3"
cursor.execute(sql)
res = cursor.fetchall()   #获取执行结果
# conn.commit()    #如果是增删改操作 需要提交事务
print(res)

#关闭连接
cursor.close()
conn.close()
```

2.2 爬取腾讯疫情数据

- 数据存储

```
def update_details():
    """
    更新 details 表
    :return:
    """
    cursor = None
    conn = None
    try:
        li = get_tencent_data()[1]  #  0 是历史数据字典,1 最新详细数据列表
        conn, cursor = get_conn()
        sql = "insert into details(update_time,province,city,confirm,confirm_add,heal,dead) values(%s,%s,%s,%s,%s,%s,%s)"
        sql_query = 'select %s=(select update_time from details order by id desc limit 1)' #对比当前最大时间戳
        cursor.execute(sql_query,li[0][0])
        if not cursor.fetchone()[0]:
            print(f"{time.asctime()}开始更新最新数据")
            for item in li:
                cursor.execute(sql, item)
            conn.commit()  # 提交事务 update delete insert操作
            print(f"{time.asctime()}更新最新数据完毕")
        else:
            print(f"{time.asctime()}已是最新数据！")
    except:
        traceback.print_exc()
    finally:
        close_conn(conn, cursor)
```

2.2 爬取腾讯疫情数据

- 数据存储

```
def insert_history():
    """
        插入历史数据
    :return:
    """
    cursor = None
    conn = None
    try:
        dic = get_tencent_data()[0]  # 0 是历史数据字典,1 最新详细数据列表
        print(f"{time.asctime()}开始插入历史数据")
        conn, cursor = get_conn()
        sql = "insert into history values(%s,%s,%s,%s,%s,%s,%s,%s,%s)"
        for k, v in dic.items():
            # item 格式 {'2020-01-13': {'confirm': 41, 'suspect': 0, 'heal': 0, 'dead': 1}
            cursor.execute(sql, [k, v.get("confirm"), v.get("confirm_add"), v.get("suspect"),
                                 v.get("suspect_add"), v.get("heal"), v.get("heal_add"),
                                 v.get("dead"), v.get("dead_add")])

        conn.commit()  # 提交事务 update delete insert操作
        print(f"{time.asctime()}插入历史数据完毕")
    except:
        traceback.print_exc()
    finally:
        close_conn(conn, cursor)
```

2.2 爬取腾讯疫情数据

- 数据存储

```
def update_history():
    """
    更新历史数据
    :return:
    """
    cursor = None
    conn = None
    try:
        dic = get_tencent_data()[0]  #  0 是历史数据字典,1 最新详细数据列表
        print(f"{time.asctime()}开始更新历史数据")
        conn, cursor = get_conn()
        sql = "insert into history values(%s,%s,%s,%s,%s,%s,%s,%s,%s)"
        sql_query = "select confirm from history where ds=%s"
        for k, v in dic.items():
            # item 格式 {'2020-01-13': {'confirm': 41, 'suspect': 0, 'heal': 0, 'dead': 1}
            if not cursor.execute(sql_query, k):
                cursor.execute(sql, [k, v.get("confirm"), v.get("confirm_add"), v.get("suspect"),
                                     v.get("suspect_add"), v.get("heal"), v.get("heal_add"),
                                     v.get("dead"), v.get("dead_add")])
        conn.commit()  # 提交事务 update delete insert操作
        print(f"{time.asctime()}历史数据更新完毕")
    except:
        traceback.print_exc()
    finally:
        close_conn(conn, cursor)
```

2.3 爬取百度热搜数据

百度的数据页面使用了动态渲染技术，我们可以用 **selenium** 来爬取

- selenium 是一个用于 web 应用程序测试的工具,直接运行在浏览器中，就像真正的用户在操作一样
- 安装： pip install selenium
- 安装浏览器（谷歌、火狐等）
- 下载对应版本浏览器驱动：http://npm.taobao.org/mirrors/chromedriver/
  1. 创建浏览器对象
  2. 浏览器.get()
  3. 浏览器.find()

2.3 爬取百度热搜数据

- 数据爬取

```
def get_baidu_hot():
    """
    :return: 返回百度疫情热搜
    """
    option = ChromeOptions()  # 创建谷歌浏览器实例
    option.add_argument("--headless")  # 隐藏浏览器
    option.add_argument('--no-sandbox')

    url = "https://voice.baidu.com/act/virussearch/virussearch?from=osari_map&tab=0&infomore=1"
    browser = Chrome(options=option, executable_path="./chromedriver.exe")
    browser.get(url)
    # 找到展开按钮
    dl = browser.find_element_by_xpath('//*[@id="main"]/div/div/section/div[2]/div/div[2]/section/div')
    dl.click()
    time.sleep(1)
    # 找到热搜标签
    c = browser.find_elements_by_xpath('//*[@id="main"]/div/div/section/div[2]/div/div[2]/section/a/div/span[2]')
    context = [i.text for i in c]  # 获取标签内容
    print(context)
    return context
```

2.3 爬取百度热搜数据

- 数据存储

  同样，我们也需要把数据存储到 mysql 数据库

```
CREATE TABLE `hotsearch` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `dt` datetime DEFAULT NULL ON UPDATE
CURRENT_TIMESTAMP,
  `content` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4
```

```
def update_hotsearch():
    """
    将疫情热搜插入数据库
    :return:
    """
    cursor = None
    conn = None
    try:
        context = get_baidu_hot()
        print(f"{time.asctime()}开始更新热搜数据")
        conn, cursor = get_conn()
        sql = "insert into hotsearch(dt,content) values(%s,%s)"
        ts = time.strftime("%Y-%m-%d %X")
        for i in context:
            cursor.execute(sql, (ts, i))  # 插入数据
        conn.commit()  # 提交事务保存数据
        print(f"{time.asctime()}数据更新完毕")
    except:
        traceback.print_exc()
    finally:
        close_conn(conn, cursor)
```

目录

CONTENTS

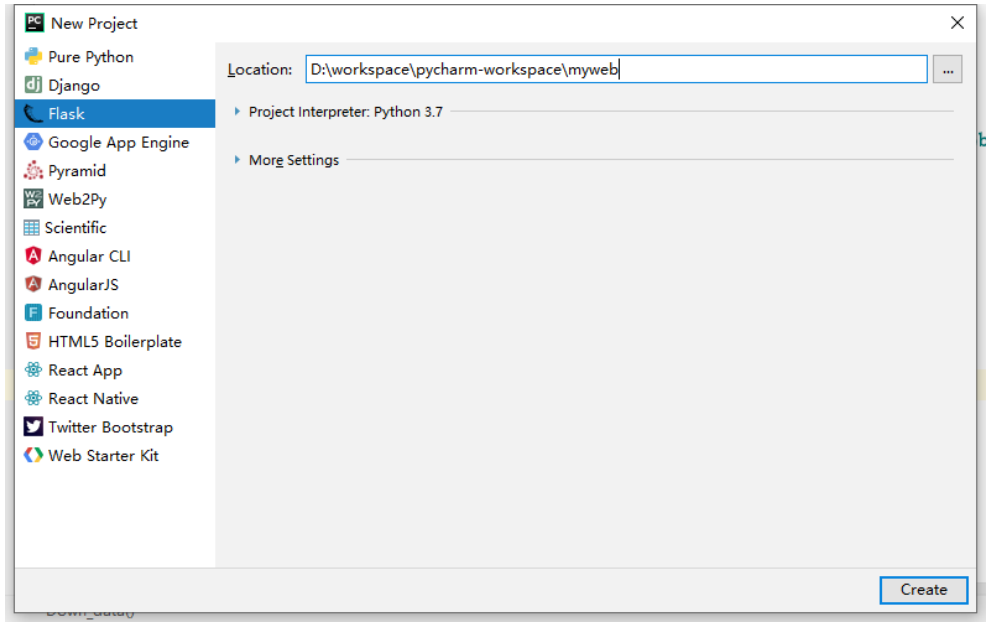
3.1 Flask 快速入门

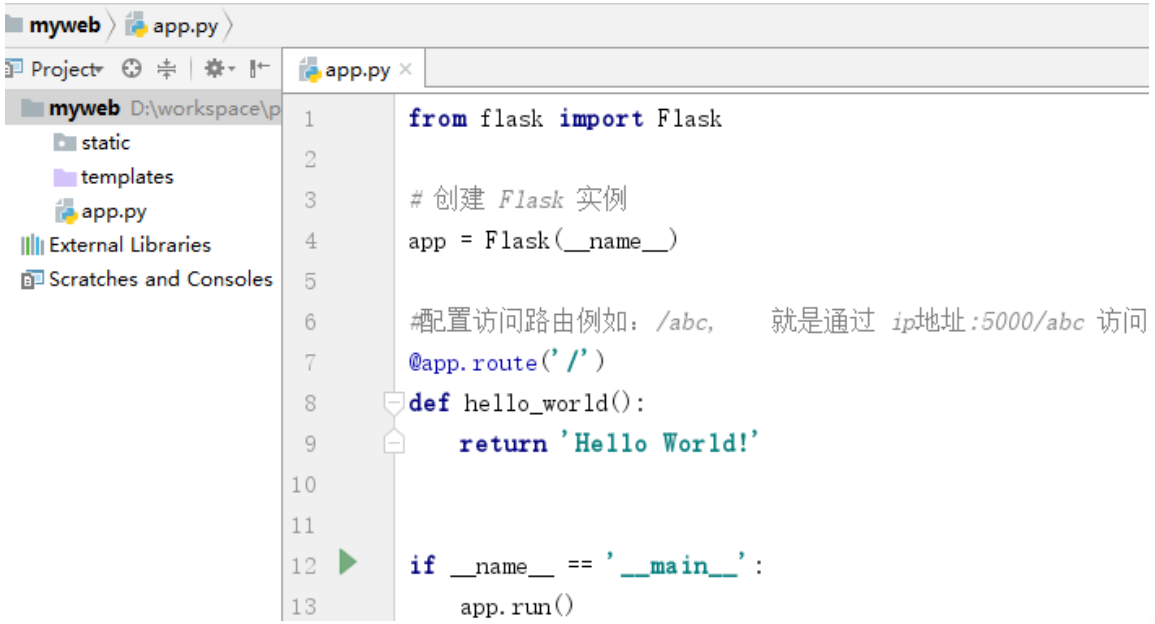
Flask 是一个使用 Python 编写的轻量级 Web 应用框架。其 WSGI（Python Web Server Gateway Interface） 工具包采用 Werkzeug， 模板引擎则使用 Jinja2， 是目前十分流行的 web 框架。

- 安装：pip install flask

3.1 Flask 快速入门

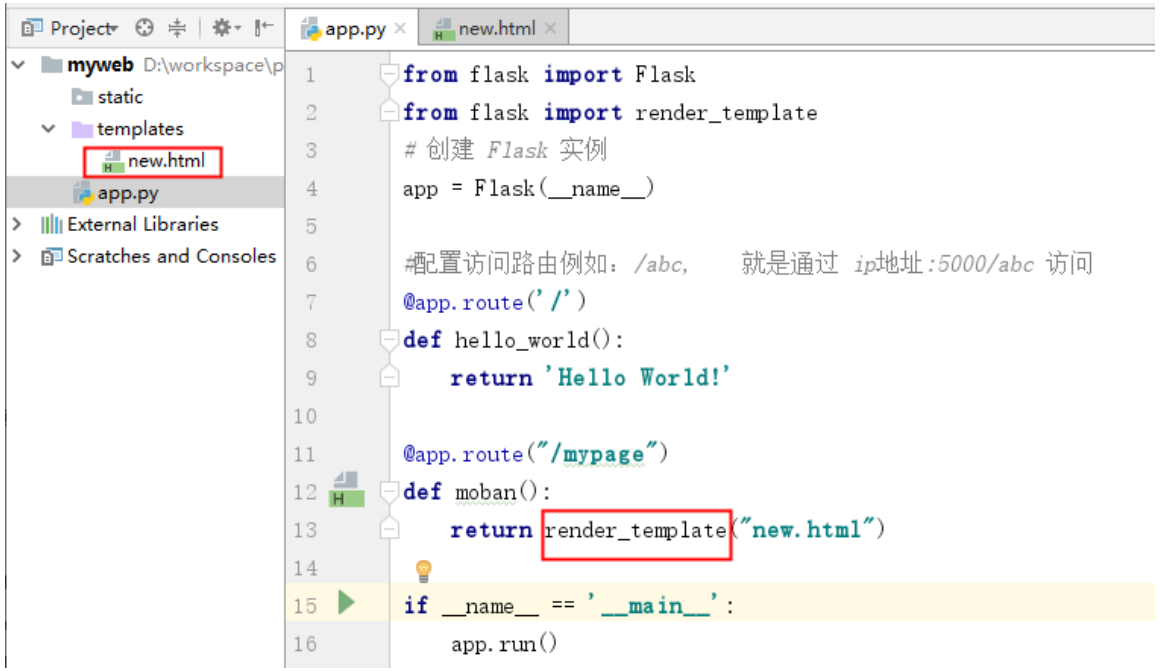
- 创建 Flask 项目





3.1 Flask 快速入门

- 模板的使用
  - 模板就是预先写好的页面，里面可以使用特殊语法引入变量
  - 使用 render_template 返回模板页面



3.1 Flask 快速入门

- Flask 获取请求参数
  - 使用 request 对象获取参数
  - ① request.values　　获取参数字典
  - ② request.values.get("参数名")

3.1 Flask 快速入门

- 使用 Ajax 局部刷新页面
  - Ajax 是 Asynchronous JavaScript and XML 的简称，通过 Ajax 向服务器发送请求，接收服务器返回的 json 数据，然后使用 JavaScript 修改网页的来实现页面局部数据更新
  - 使用 jquery 框架可方便的编写 ajax 代码，需要 jquery.js 文件

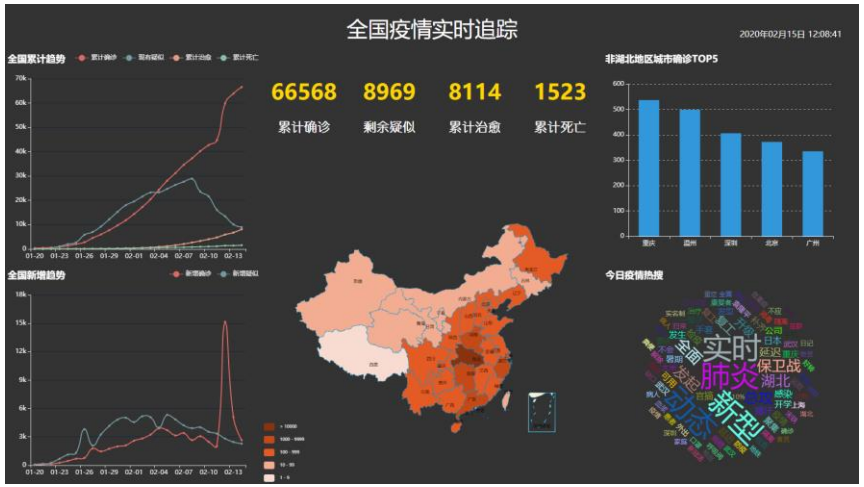
基本格式：

```
$.ajax({
    type:"post",  //请求类型
    url:"/目标路由",  //请求地址
    data:{"k1":"v1","k2":"v2"}, //数据
    datatype:"json",   //返回的数据类型
    success:function (datas) {
        //请求成功的回调函数，datas 是返回的数据
    },
    error:function () {
        //请求失败时执行
    }
})
```

3.2 可视化大屏模板制作

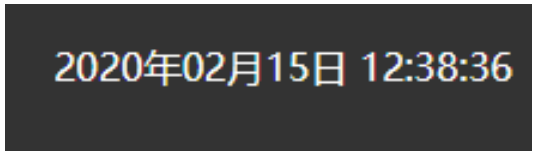
- 使用绝对定位划分版块

```
#tit {
    color: #FFFFFF;  /* 设置字体 */
    position: absolute;   /* 绝对定位 */
    height: 10%;
    width: 40%;
    left: 30%;
    top: 0;
    /* 居中分布 */
    display: flex;
    align-items: center;
    justify-content: center;
}
```



3.2 可视化大屏模板制作

- 统计数字及时间
  - 调整 css 样式
  - 获取数据




```
def get_c1_data():
    """
    :return: 返回大屏div id=c1 的数据
    """
    # 因为会更新多次数据，取时间戳最新的那组数据
    sql = "select sum(confirm)," \
          "(select suspect from history order by ds desc limit 1)," \
          "sum(heal)," \
          "sum(dead) " \
          "from details " \
          "where update_time=(select update_time from details order by update_time desc limit 1) "
    res = query(sql)
    return res[0]
```

3.2 可视化大屏模板制作

- echarts 快速入门

  ECharts，缩写来自 Enterprise Charts，商业级数据图表，是百度的一个开源的数据可视化工具，提供了丰富的图表库，能够在 PC 端和移动设备上流畅运行

官网网站：https://echarts.apache.org/zh/index.html

3.2 可视化大屏模板制作

- echarts 快速入门

第一步，在<script>标签中引入 Echarts 文件及 jQuery

```
<script src="../static/js/echarts.min.js"></script>
<script src="../static/js/jquery-1.11.1.min.js"></script>
```

第二步，准备一个放图表的容器

```
<!-- 为ECharts准备一个具备大小（宽高）的DOM容器1-->
<div id='pane' style="height: 500px;"></div>
```

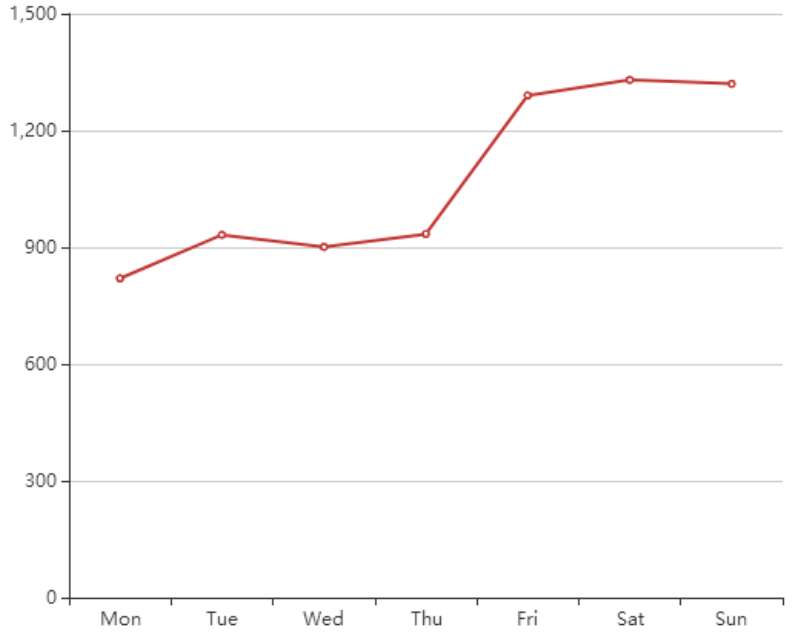
3.2 可视化大屏模板制作

- echarts 快速入门

  第三步，设置参数，初始化图表

```
var option = {
    xAxis: {
        type: 'category',
        data: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
    },
    yAxis: {
        type: 'value'
    },
    series: [{
        data: [820, 932, 901, 934, 1290, 1330, 1320],
        type: 'line'
    }]
};
// 基于准备好的dom，初始化echarts图表
var my_chart = echarts.init(document.getElementById('pane'));

//为echarts对象加载数据
my_chart.setOption(option);
```



3.2 可视化大屏模板制作

- echarts 快速入门

  第四步，更改对应 option 里面 data 的值，让图表变化

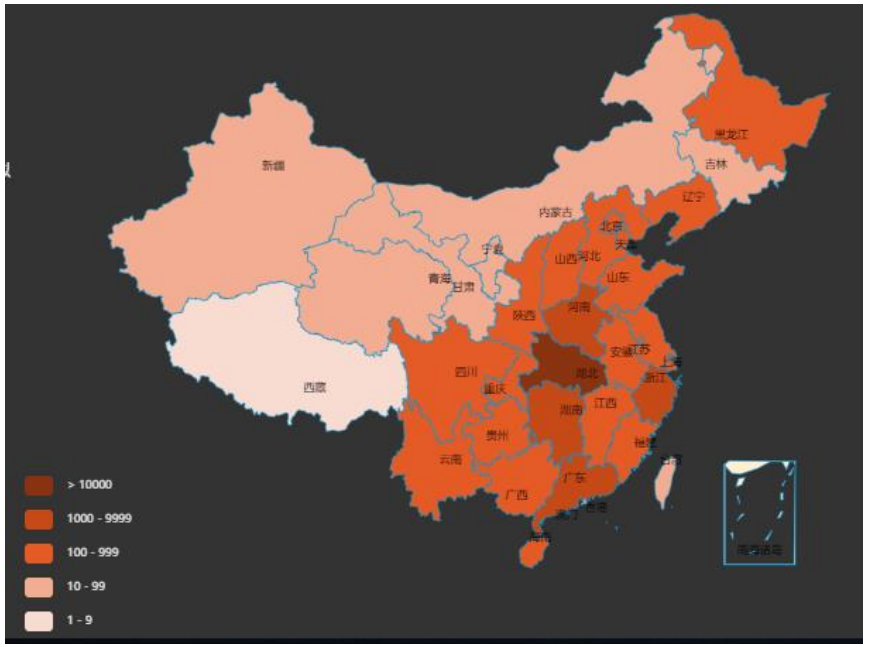
```
$("#change").click(function () {
    // $.ajax 从后台获取数据

    var d = new Array(7)
    for (var i = 0; i <7 ; i++) {
        d[i] = Math.floor(Math.random()*1000+1);
    }
    // console.log(d)
    option.series[0].data=d
    my_chart.setOption(option)
});
```

3.2 可视化大屏模板制作

- 中国地图
  - 复制中国地图 option，导入 **china.js**
  - 获取数据

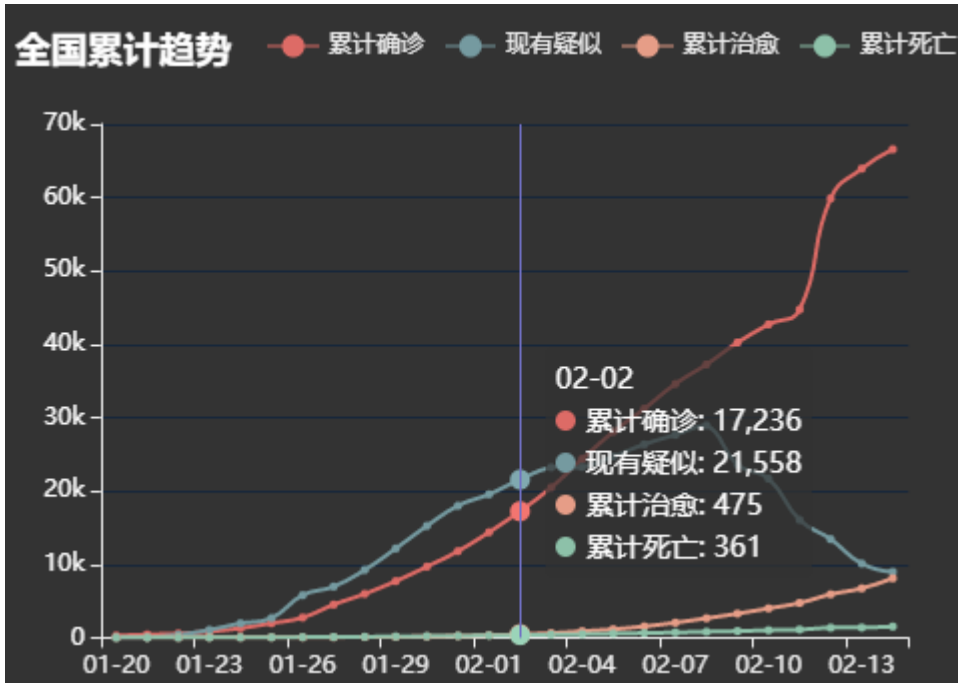
```
def get_c2_data():
    """
    :return: 返回各省数据
    """
    # 因为会更新多次数据，取时间戳最新的那组数据
    sql = "select province,sum(confirm) from details " \
          "where update_time=(select update_time from details " \
          "order by update_time desc limit 1) " \
          "group by province"
    res = query(sql)
    return res
```



3.2 可视化大屏模板制作

- 全国累计趋势
  - 复制折线图 option
  - 获取数据

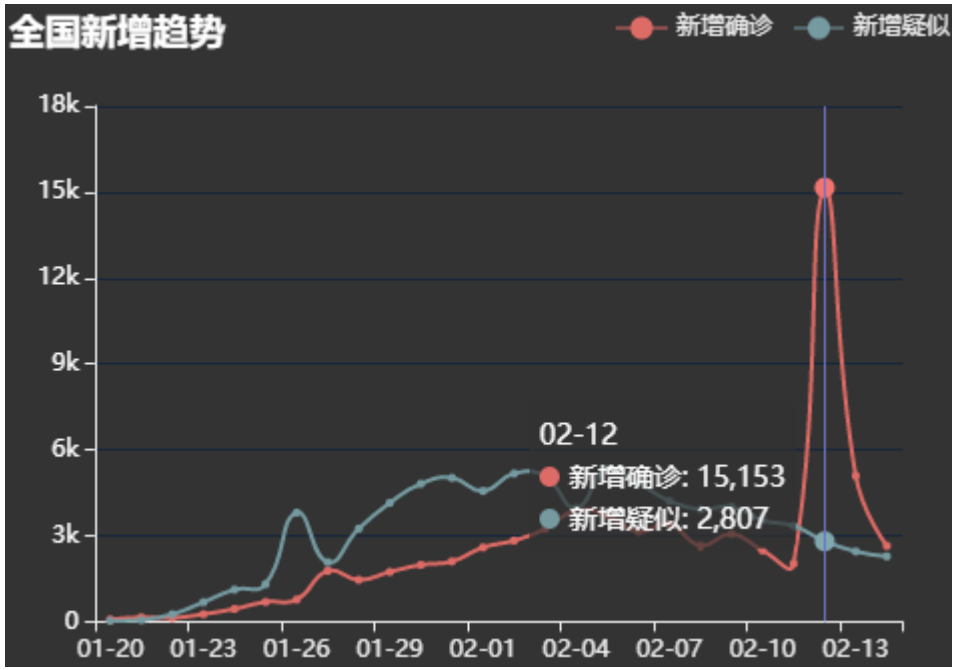
```
def get_l1_data():
    """
    :return:  返回每天历史累计数据
    """
    sql = "select ds, confirm, suspect, heal, dead from history "
    res = query(sql)
    return res
```



3.2 可视化大屏模板制作

- 全国新增趋势
  - 复制折线图 option
  - 获取数据

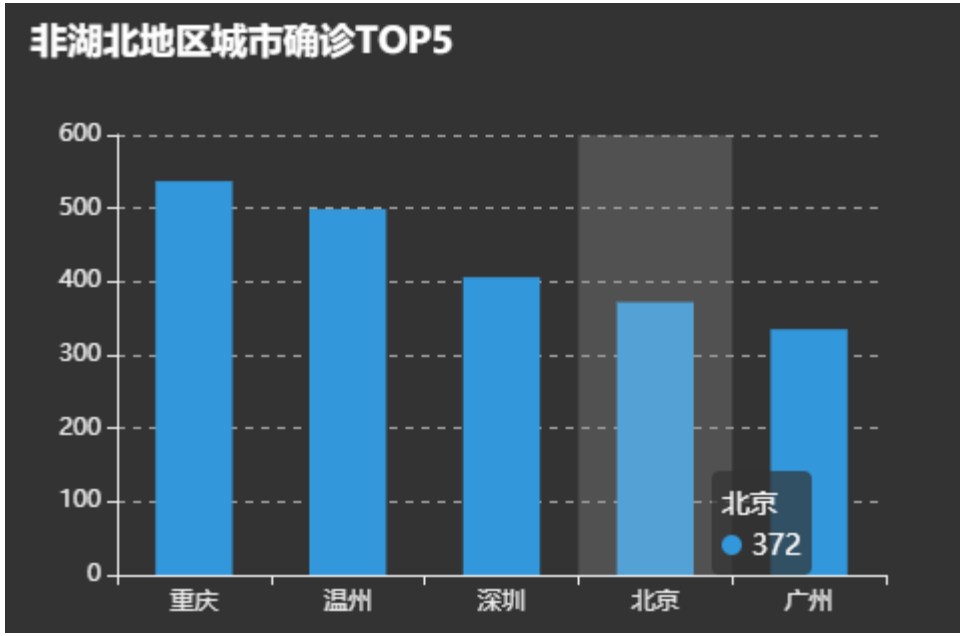
```
def get_12_data():
    """
    :return:  返回每天新增确诊和疑似数据
    """
    sql = "select ds, confirm_add, suspect_add from history "
    res = query(sql)
    return res
```



3.2 可视化大屏模板制作

- 非湖北地区TOP5
  - 复制柱状图 option
  - 获取数据

```
def get_rl_data():
    """
    :return:  返回非湖北地区城市确诊人数前5名
    """
    sql = 'SELECT city,confirm FROM ' \
          '(select city,confirm from details  ' \
          'where update_time=(select update_time from details order by update_time desc limit 1) ' \
          'and province not in ("湖北","北京","上海","天津","重庆") ' \
          'union all ' \
          'select province as city,sum(confirm) as confirm from details  ' \
          'where update_time=(select update_time from details order by update_time desc limit 1) ' \
          'and province in ("北京","上海","天津","重庆") group by province) as a ' \
          'ORDER BY confirm DESC LIMIT 5'
    res = query(sql)
    return res
```



3.2 可视化大屏模板制作

- 疫情热搜
  - 复制词云图 option，导入**wordcloud.js**
  - 获取数据，使用 jieba 获取关键字

```
def get_r2_data():
    """
    :return:  返回最近的20条热搜
    """
    sql = 'select content from hotsearch order by id desc limit 20'
    res = query(sql) #格式 (('民警抗疫一线奋战16天牺牲1037364',), ('四川
    return res
```

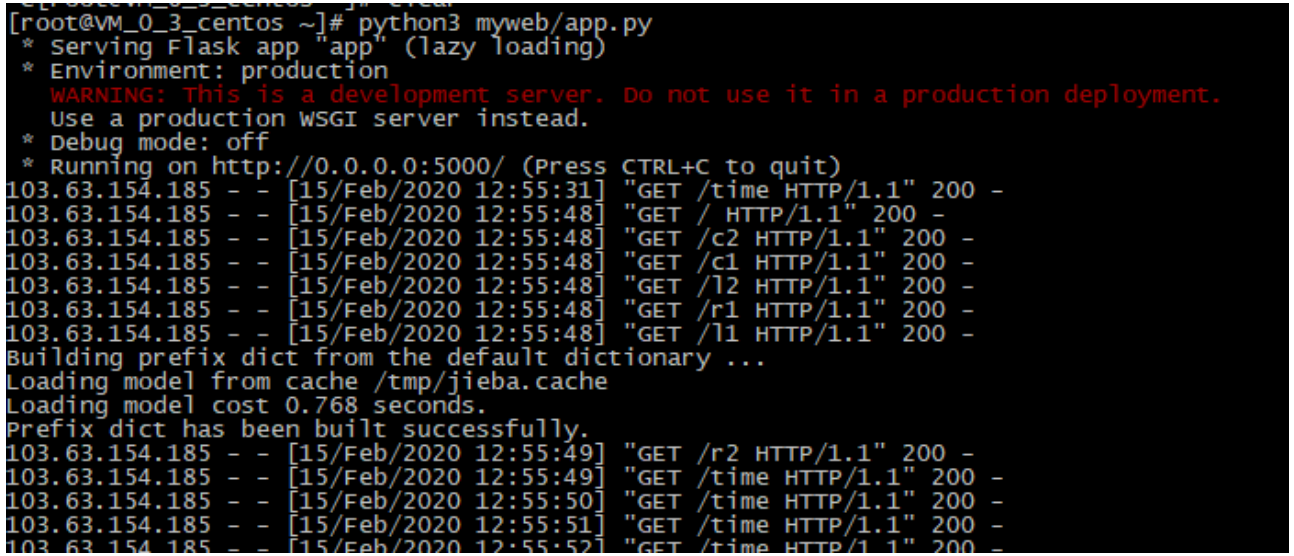
目录

CONTENTS

4.1 部署 Flask 项目

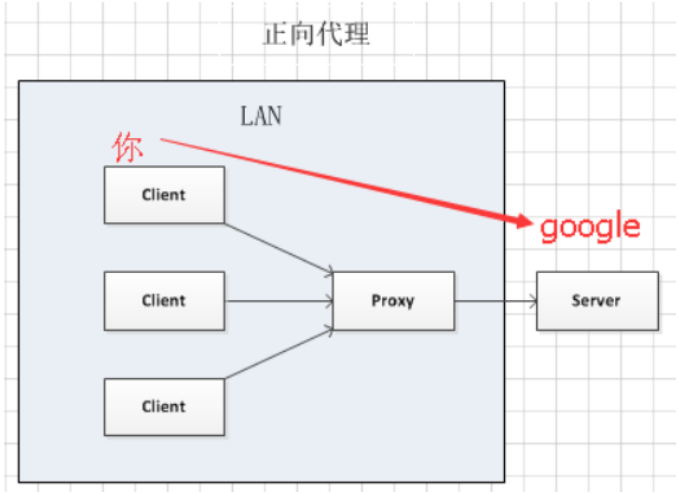
- 开发模式部署

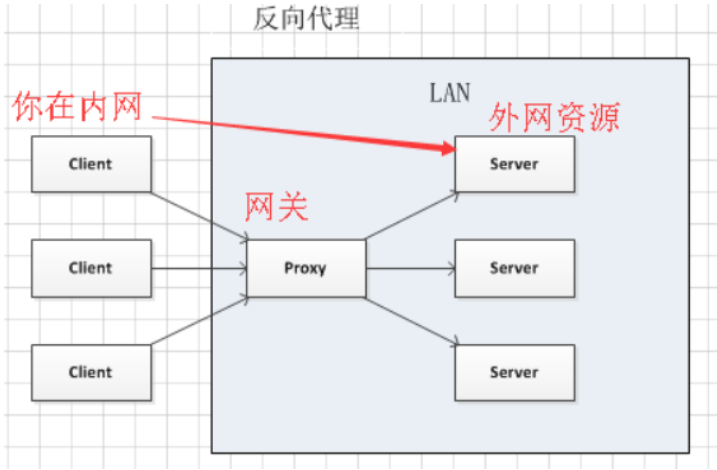
  修改 app.run() 的 host 为 0.0.0.0



4.1 部署 Flask 项目

- 生产模式部署
  - 部署 Flask 应用时，通常都是使用一种 WSGI 应用服务器搭配 Nginx 作为反向代理
  - 常用的 WSGI 服务器： gunicorn、uwsgi
  - 反向代理和正向代理：





4.1 部署 Flask 项目

- 生产模式部署
  - 安装 Nginx：yum install nginx　　(红帽系列Linux发行版上的指令)
  - 安装 Gunicorn: pip install gunicorn
  - 启动 Gunicorn： gunicorn -b 127.0.0.1:8080 -D my_app:app
  - 编辑 Nginx 配置文件　vim /etc/nginx/nginx.conf
  - 启动 Nginx ： /usr/sbin/nginx

```
upstream mycluster {
        server 127.0.0.1:8080 weight=1;
}
server {
    listen       80 ;
    server_name  127.0.0.1;

    location / {
    proxy_pass http://mycluster;
    proxy_redirect      off;
    proxy_set_header    Host                $http_host;
    proxy_set_header    X-Real-IP           $remote_addr;
    proxy_set_header    X-Forwarded-For     $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Proto   $scheme;
    }
```

4.2 部署定时爬虫

- 获取脚本参数
  - sys.argv
  - sys.argv[0] 是脚本所在绝对路径
  - 根据不同参数调用不用方法
- Linux 安装 chrome
  - yum install https://dl.google.com/linux/direct/google-chrome-stable_current_x86_64.rpm
- 下载 chromedriver
  - http://npm.taobao.org/mirrors/chromedriver/ 项目中使用的 79.0.3945.36/

4.2 部署定时爬虫

- 获取crontab 定时调度
  - crontab –l　列出当前任务
  - crontab –e　编辑任务

```
crontab: no changes made to crontab
[root@VM_0_3_centos ~]# crontab -l
*/1 * * * * /usr/local/qcloud/stargate/admin/start.sh > /dev/null 2>&1 &
0 0 * * * /usr/local/qcloud/YunJing/YDCrontab.sh > /dev/null 2>&1 &
30 * * * * python3 /root/spider.py up_his >> /root/log_his 2>&1 &
3 */2 * * * python3 /root/spider.py up_hot >> /root/log_hot 2>&1 &
*/5 * * * * python3 /root/spider.py up_det >> /root/log_det 2>&1 &
[root@VM_0_3_centos ~]#
```

  - 格式：* * * * * 指令

    五个星号分别代表 分 、时 、日 、月、周

东方瑞通
终身学习
Founded in 1998

THANKS !