

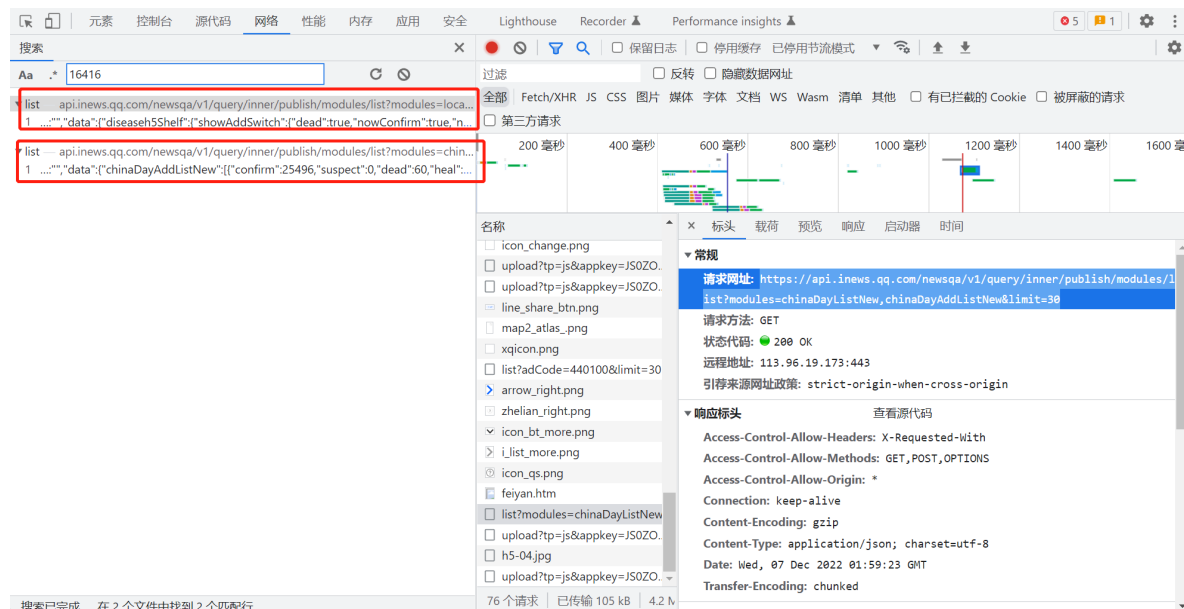
疫情信息获取

01 腾讯疫情

目标页面: <https://news.qq.com/zt2020/page/feiyang.htm#/>

该页面使用json数据来渲染页面

对某个数据进行搜索可以发现:



搜索结果发现, 同一个数据在两个接口中都出现了

1.1 接口分析

为了更加方便地分析接口内容, 推荐使用如下两种工具之一:

- json.cn
- 使用浏览器插件: **json_formatter** (使用群里提供的json_formatter.crx文件安装)

说明: 下面的接口实际上是一样的, 只不过modules携带的参数不同, 导致的结果不同

接口一: <https://api.inews.qq.com/newsqa/v1/query/inner/publish/modules/list?modules=localCityNCOVDataList,diseaseh5Shelf>

- localCityNCOVDataList: 每个城市当天的确诊、风险地区数量的信息
- diseaseh5Shelf:
 - chinaTotal: 中国当天疫情信息汇总
 - chinaAdd: 中国当天新增疫情信息汇总
 - **areaTree**: 地区树, 每个参数的children都包含了当前地区的下级区域 (中国 --- 省份 --- 城市)
 - 每个地区都有的信息: adcode --> 地区代码; today --> 当日新增数据; total --> 汇总数据; name --> 地区名称

这个接口只能获取当天的数据, 它最大的价值是给我们提供了**adcode**

接口二: <https://api.inews.qq.com/newsqa/v1/query/inner/publish/modules/list?modules=chinaDayListNew,chinaDayAddListNew&limit=30>

- chinaDayAddListNew: 中国历史疫情**新增**数据表
- chinaDayListNew: 中国历史疫情数据汇总表

其中, limit参数可以修改查询的天数

接口三: <https://api.inews.qq.com/newsqa/v1/query/pubished/daily/list?adCode=440000&limit=30>

- adCode: 地区代码 (可以是省份, 也可以是城市)
- limit: 修改查询的天数

所以接口一用于获取adCode; 接口二用于获取全国历史疫情信息; 接口三用于获取各个省份/城市的历史疫情信息

1.2 全国疫情信息获取

```
import requests
from pprint import pprint
from datetime import datetime
from dateutil.relativedelta import relativedelta

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36'
}

def cal_limit_days(month=3):
    '''获取完整月份数据, 默认获取三个完整的月份'''
    now = datetime.now() # 今天的日期
    # 计算month个月前的日期, 与now相差多少天
    pre_date = now - relativedelta(months=month)
    # print(pre_date) # 2022-09-07 14:26:30.839619
    res_date = datetime(pre_date.year, pre_date.month, 1) # 相当于获取2022-09-01
    # 计算两个日期之间相差多少天
    diff_day = (now - res_date).days # days参数用于查看相差多少天
    return diff_day + 1, res_date

def turn_to_sql_date(year, date, min_date):
    '''判断日期界限并转为mysql支持的日期格式'''
    ds = year + '.' + date
    # 为了与min_date进行对比, 需要转化为datetime对象
    ds_tmp = datetime.strptime(ds, '%Y.%m.%d')
    if ds_tmp < min_date:
        print(f'{ds_tmp.date()} 日期超过最小边界, 跳过! ')
        return None
    return ds.replace('.', '-') # # ds_tmp.strftime('%Y-%m-%d')

def get_tencent_data():
    '''获取国内疫情信息'''
    limit, min_date = cal_limit_days()
    print('-----正在获取全国疫情信息-----')
    print(f'当前limit值为: {limit} | 最小的日期界限为: {min_date.date()}')

    contry_url = 'https://api.inews.qq.com/newsqa/v1/' \
```

```

        'query/inner/publish/modules/list?modules' \
        '=chinaDayListNew,chinaDayAddListNew&limit=' + str(limit)
    con_resp = requests.get(contry_url, headers=headers)
    china_data = con_resp.json()['data']
    # 按照日期来整理全国疫情信息表
    history = {} # 保存数据的字典，是以日期作为主键
    for i in china_data['chinaDayAddListNew']:
        ds = turn_to_sql_date(i['y'], i['date'], min_date) # 日期，后续需要转为
2022-11-06或2022/11/06
        if not ds: # 如果为None，则跳过当前的日期
            continue
        # 添加疫情信息数据
        history[ds] = {
            'confirm_add': i['confirm'], 'heal_add': i['heal'],
            'dead_add': i['dead'], 'importedCase_add': i['importedCase']
        }

    for i in china_data['chinaDayListNew']:
        ds = turn_to_sql_date(i['y'], i['date'], min_date)
        if not ds: # 如果为None，则跳过当前的日期
            continue
        if ds not in history: # 检查日期是否存在于上述的字典中，仅当日期存在才执行写入
            continue
        # 更新已经存在的日期的记录
        history[ds].update({
            'confirm': i['confirm'], 'confirm_now': i['nowConfirm'],
            'heal': i['heal'], 'dead': i['dead'], 'importedCase':
i['importedCase']
        })

    pprint(history)

def get_province_data():
    '''获取省份的信息'''
    pass

def main():
    '''启动的主程序'''
    get_tencent_data()

if __name__ == '__main__':
    main()

```

1.3 数据持久化

- 打开mysql终端: `mysql -uroot -p密码`
- 创建数据库: `create database covid19 charset=utf8mb4`
- 启用数据库: `use covid19`
- 创建如下3个信息表:

```

CREATE TABLE `history` (
  `ds` datetime NOT NULL COMMENT '日期',
  `confirm` int(255) DEFAULT NULL COMMENT '累计确诊',
  `confirm_add` int(255) DEFAULT NULL COMMENT '当日新增确诊',
  `confirm_now` int(255) DEFAULT NULL COMMENT '剩余确诊',
  `heal` int(255) DEFAULT NULL COMMENT '累计治愈',
  `heal_add` int(255) DEFAULT NULL COMMENT '当日新增治愈',

```

```

`dead` int(255) DEFAULT NULL COMMENT '累计死亡',
`dead_add` int(255) DEFAULT NULL COMMENT '当日新增死亡',
`importedCase` int(255) DEFAULT NULL COMMENT '境外输入案例',
`importedCase_add` int(255) DEFAULT NULL COMMENT '新增境外输入',
PRIMARY KEY (`ds`) USING BTREE
) DEFAULT CHARSET=utf8mb4;

CREATE TABLE `details` (
`id` int(255) NOT NULL AUTO_INCREMENT,
`update_time` datetime DEFAULT NULL COMMENT '数据最后更新时间',
`province` varchar(255) DEFAULT NULL COMMENT '省',
`confirm` int(255) DEFAULT NULL COMMENT '累计确诊',
`confirm_add` int(255) DEFAULT NULL COMMENT '新增确诊',
`confirm_now` int(255) DEFAULT NULL COMMENT '现有确诊',
`heal_add` int(255) DEFAULT NULL COMMENT '新增治愈',
`heal` int(255) DEFAULT NULL COMMENT '累计治愈',
`dead` int(255) DEFAULT NULL COMMENT '累计死亡',
`dead_add` int(255) DEFAULT NULL COMMENT '新增死亡',
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `risk_area` (
`id` int(10) PRIMARY KEY AUTO_INCREMENT,
`end_update_time` varchar(255) COMMENT '数据最后更新时间',
`province` varchar(255) COMMENT '省',
`city` varchar(255) COMMENT '市',
`county` varchar(255) COMMENT '县',
`address` varchar(10000) COMMENT '详细地址',
`type` varchar(10) COMMENT '风险类型'
)ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```

1.4 更新全国疫情信息表

安装pymysql模块: `pip install pymysql`

```

import traceback
from pymysql import connect # 数据库链接对象

def get_conn():
    '''用于获取mysql的连接对象'''
    conn = connect(
        user='root',
        password='123456',
        database='covid19_1',
        host='127.0.0.1', # 如果是远程服务器就需要改成服务器的ip
        charset='utf8'
    )
    # 得到游标对象, 只有游标对象可以执行sql语句
    cursor = conn.cursor()
    return conn, cursor

def get_tencent_data():
    ...
    for ...
        ....
        insert_into_history(history) # 将history的内容写入到数据库

```

```

def insert_into_history(data):
    '''写入数据到history表中'''
    conn, cursor = get_conn()
    print(f'开始更新全国疫情信息...')
    try:
        # history表中需要输入的字段有10个
        sql = "insert into history values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
        # 查询某个日期的数据是否存在
        sql_query = "select confirm from history where ds=%s"
        for k, v in data.items(): # items将键值对进行拆分
            # k = 时间; v = 9个字段的数据
            if not cursor.execute(sql_query, k):
                cursor.execute(sql, [k, v.get('confirm'), v.get('confirm_add'),
                                     v.get('confirm_now'), v.get('heal'),
                                     v.get('heal_add'),
                                     v.get('dead'),
                                     v.get('dead_add'), v.get('importedCase'),
                                     v.get('importedCase_add')])
                print(f'[history] | [{k}] 记录写入成功! ')
            # 提交事务, 注意, 如果不写这句, 那么sql中更新数据的内容不会被执行
            conn.commit()
        except: # 异常处理, 数据库一般会执行回滚的操作
            conn.rollback() # 回滚数据库
            traceback.print_exc() # 打印详细的错误信息
        finally: # 关闭链接
            cursor.close()
            conn.close()

```

注意: 11月4号的 confirm_add 数据异常, 后续数据分析时去掉即可

1.5 各省份数据获取

02 风险地区信息