上海大学

SHANGHAI UNIVERSITY

《Python 计算》期末综合实验

题　　目　基于 Flask 与 Vue.js 的疫情可视化

学　　号　**20121034**

姓　　名　胡才郁

日　　期　**2022 年 5 月 29 日**

一、实验目的与要求

突如其来的疫情扰乱了每一个人的生活节奏，在上海疫情防控的背景下，本实验收集疫情趋势数据与上海居民求助信信息，试对于疫情情况以及民众舆论趋势展开数据分析，并将分析结果使用 Web 技术展示。

本实验属于综合实验选题中的主题(2)Python 网络通信应用设计、(3)Python 舆情分析、(4)Python 数据分析，较为综合的完成了以上主题。

二、实验环境

本项目使用到的编程语言有 Python、JavaScript、SQL，用到的技术栈如下表所示：

表 1. 技术栈说明

技术栈	
技术	说明
前端	
Vue.js	前端框架
Vue router	路由配置
Axios	前端 HTTP 框架
后端	
Flask	后端服务器
MySQL	数据库
PyMySQL	数据库连接池
数据可视化	
Echart	JavaScript 数据可视化
Matplotlib	Python 数据可视化
Seaborn	Python 数据可视化
数据来源	
requests	请求数据接口
Web 部署	
Nginx	反向代理

三、实验内容

本项目希望能利用交互式空间数据分析技术，使用 Flask + Vue.js + Echarts 搭建简单新冠肺炎疫情数据可视化交互分析平台，感知疫情发展趋势与关键节点、分析上海疫情求助信信息的动态演变、对社会舆情进行态势感知。

本实验有效代码 3000 余行，是一个完整的数据可视化项目，从数据的获取，数据清洗，数据分析，数据可视化，再到将结果通过 Web 开发技术完整的展示。

本实验是一个较为综合的实验，涉及到的知识点与技术栈较多。包括以下内容:

1. Python 爬虫与网络编程
2. 数据库操作
3. Flask 框架搭建 Web 后端
4. Vue.js 搭建 Web 前端
5. Echart 进行数据可视化
6. 数据挖掘
7. 可视化结果部署到公网服务器之中

由于此实验已经部署到了个人的阿里云服务器中，因此，可以直接访问网址查看相应的可视化内容，网址如下：

表 2. 数据可视化成果

疫情可视化	
网址	内容
http://47.100.249.168:9999/#/mappage	全国现有确诊
http://47.100.249.168:9999/#/trendpage	全国新增趋势
http://47.100.249.168:9999/#/shanghai	上海新增趋势
http://47.100.249.168:9999/#/helppage	上海求助信息统计

四、实验设计与实现

整个实验中，各种技术协作的架构图如下：

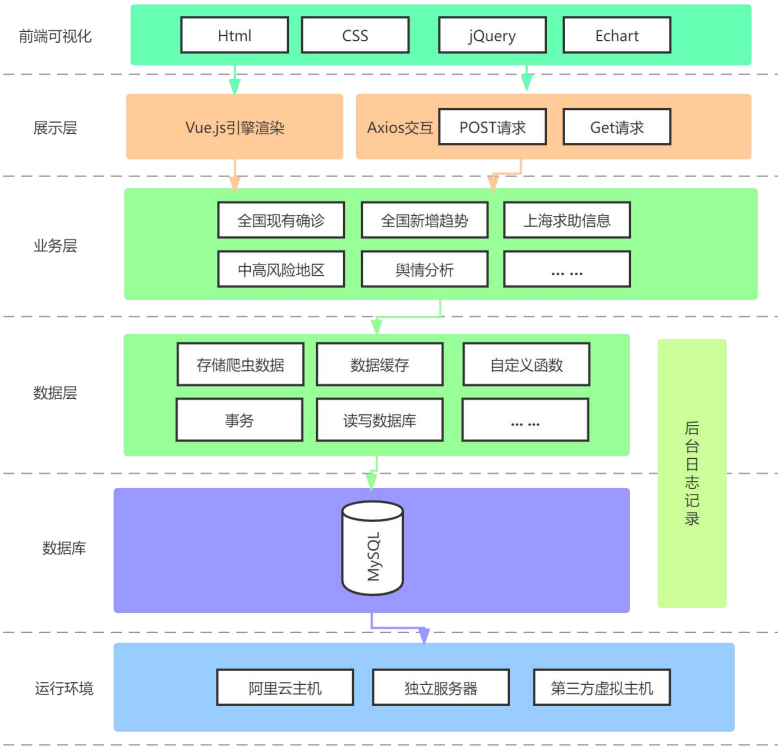


图 1. 系统架构图

接下来通过数据的流通方向，介绍本实验的实现过程。

1. 数据爬取

首先将网站的数据爬取到本地。

由于疫情实时数据更新周期短、更新次数多，且并无直接整理好的数据集共分析使用，因此编写爬虫程序，获取数据。实验中使用 Python 的 requests 库获取数据接口，爬取了从 2 月份到至今的疫情数据、1000 多条疫情谣言与新闻数据、10000 多条上海求助信息。详细数据来源如下表所示：

表 3. 疫情数据来源

网址	内容
腾讯疫情	全国、上海疫情实时数据
丁香园	疫情新闻、谣言
我们来帮你 • 上海抗疫互助	上海居民求助信息

由于腾讯疫情等网站的数据 API 接口并没有详细的文档说明，我只选取了返回数据中，部分自己能力范围内可以处理的数据。此部分但工作量巨大，但重复机械化，因此并不重点介绍。下图是使用接口访问工具 Postman 请求腾讯疫情数据接口所得的数据，展示了部分请求接口得到的数据。由于篇幅有限，无法展示全部数据。

通过下图可以观察到请求到的数据嵌套层数多，字段多。

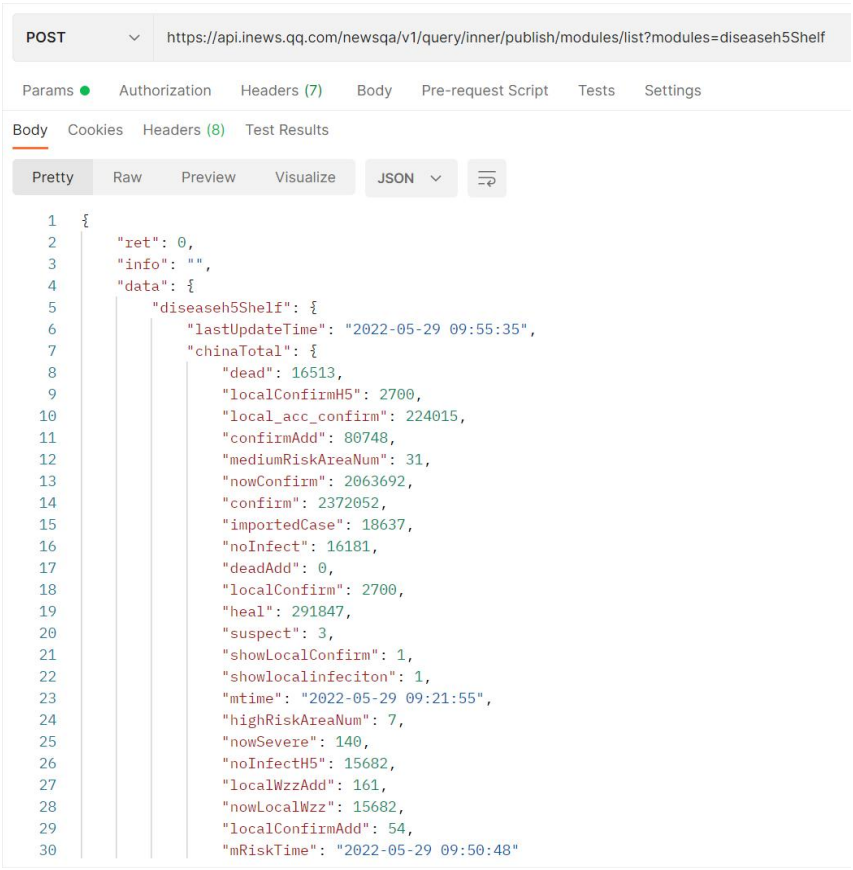


图 2. 爬虫所得部分数据

2. 数据持久化

此处采用的是 pymysql 连接本地 mysql 数据库，并且把爬取到的数据存成 4 张数据表，分别存储全国疫情信息、上海疫情信息、中高风险地区、上海求助信息。

表 4. 数据库设计

数据表名	内容
Details	全国当前疫情情况
History	全国历史疫情趋势
Risk_area	中高风险地区信息
Shanghai_history	上海历史疫情趋势

由于数据表数量较多，篇幅限制，此处不过多介绍，以 Risk_area 表为例介绍数据持久化，存入数据库的过程。

将爬取到的信息封装为 Python 中的字典数据结构，并且使用 PyMySQL，执行预先写好的 SQL 语句，就可以将字典中的 key 映射为数据库之中的字段，而将字典中的 value 映射为数据库之中每一条数据的信息。

	id	end_update_time	province	city	county	address	type
476	837	2022-05-16 21时	北京市	房山区	长阳镇	碧波园小区	中风险
477	838	2022-05-16 21时	北京市	通州区	新华街道	盛业家园社区	中风险
478	839	2022-05-16 21时	北京市	通州区	潞城镇	后北营村	中风险
479	840	2022-05-16 21时	北京市	通州区	马驹桥镇	样本小区	中风险
480	841	2022-05-16 21时	北京市	通州区	中仓街道	新华园社区	中风险
481	842	2022-05-16 21时	河北省	唐山市	路南区	天地福农果品市场	中风险
482	843	2022-05-16 21时	河北省	承德市	滦平县	长山峪镇长山峪村	中风险
483	844	2022-05-16 21时	辽宁省	丹东市	振兴区	知春园小区一期	中风险
484	845	2022-05-16 21时	辽宁省	丹东市	振兴区	丹建锦园93号楼	中风险
485	846	2022-05-16 21时	辽宁省	丹东市	振兴区	丹建锦园95号楼	中风险
486	847	2022-05-16 21时	辽宁省	丹东市	振兴区	福临花园小区	中风险
487	848	2022-05-16 21时	辽宁省	丹东市	振兴区	文安新村小区	中风险
488	849	2022-05-16 21时	辽宁省	丹东市	振兴区	丹建馨园小区	中风险
489	850	2022-05-16 21时	辽宁省	丹东市	振兴区	红房新区小区	中风险
490	851	2022-05-16 21时	辽宁省	丹东市	振兴区	桃源逸景小区	中风险
491	852	2022-05-16 21时	辽宁省	营口市	大石桥市	鑫盛家园	中风险
492	853	2022-05-16 21时	辽宁省	营口市	大石桥市	汤池镇二道河村	中风险
493	854	2022-05-16 21时	辽宁省	营口市	大石桥市	汤池镇三元井村	中风险
494	855	2022-05-16 21时	辽宁省	营口市	大石桥市	百寨街道四季春城小区	中风险
495	856	2022-05-16 21时	黑龙江省	哈尔滨市	宾县	宾州镇绿海家园小区	中风险
496	857	2022-05-16 21时	黑龙江省	哈尔滨市	宾县	宾州镇金穗小区	中风险
497	858	2022-05-16 21时	上海市	上海市	黄浦区	打浦桥街道顺昌路612弄20号	中风险
498	859	2022-05-16 21时	上海市	上海市	闵行区	梅陇镇许泾村八组	中风险
499	860	2022-05-16 21时	上海市	上海市	闵行区	梅陇镇行南村三队	中风险
500	861	2022-05-16 21时	上海市	上海市	闵行区	华漕镇许浦村三队	中风险

图 3. Risk_area 表信息

在 Risk_area 表中，除了设置每一条信息的主键 id 外，存储了省、市、区、地址以及风险情况的有关信息。

数据持久化的操作是必要的，由于要将可视化的结果在 Web 网址上可以访问获得，则后端需要访问数据库返回接口数据。而将数据存入数据库中，也可以避免由于程序中断而导致请求到的数据丢失，完成了数据的持久化操作。

3. 后端接口编写

Flask 是一个可定制的 Web 框架，轻巧、简洁，通过定制第三方扩展来实现具体功能，它可以只用 10 行 Python 代码搭建一个 Web 后端服务器。通过 Flask 作为数据库与后端服务器之间的桥梁。通过设置相对应的路由，使得可以在相应的网址进行访问。此处以获取全国各地目前实时确诊人数信息为例，介绍在此任务之中后端接口的编写。

3.1 数据库查询

编写对应的 sql 查询语句，查询数据库中 details 表中的省份名称与确诊信息，并通过路由配置为/map 路径。当后台 Python 程序启动时，访问对应的路径，即可获取到将对应的数据。如下图所示，http://47.100.249.168:5000/map 网址对应返回为相应的后台数据。

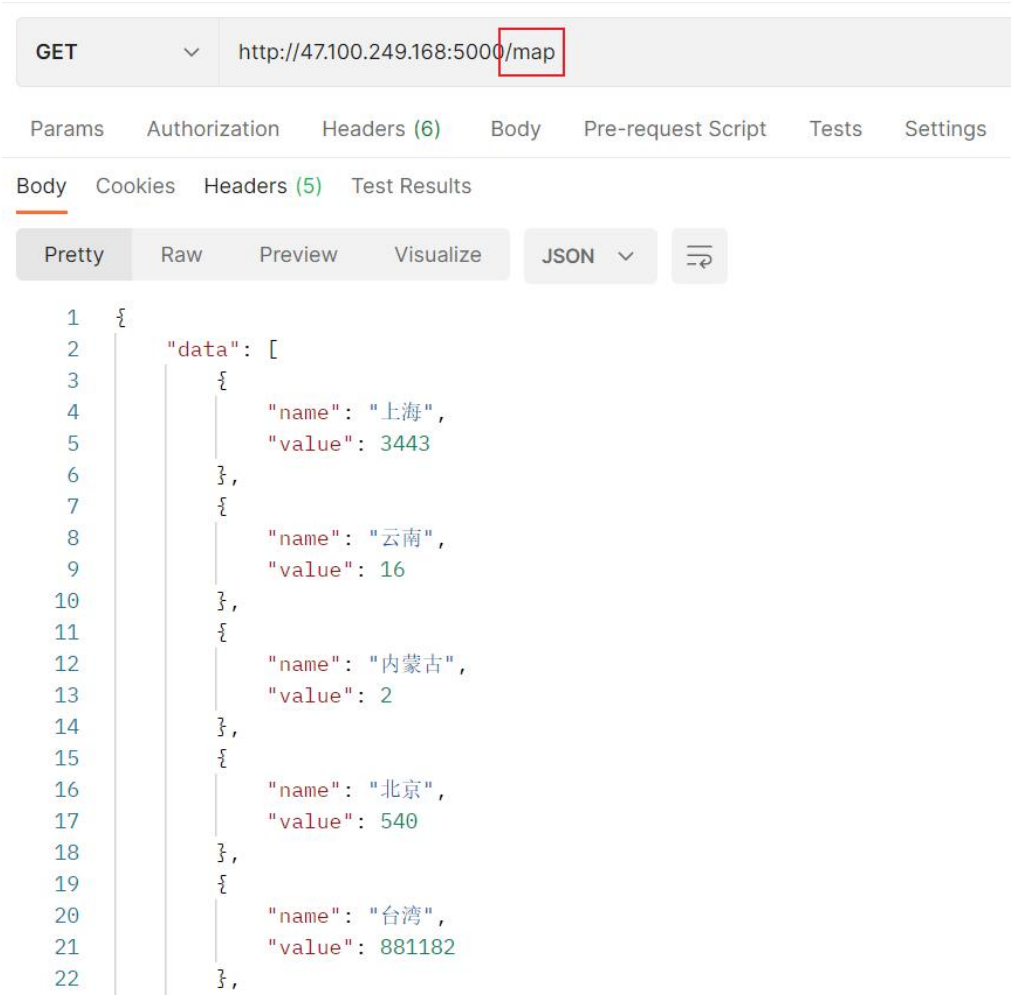


图 4. map 接口返回 json 数据

4. 前端页面

使用 Echart 作为数据可视化组件，并且由 Vue.js 引擎渲染页面。而此页面中的，地图中绑定的数据并非固定，而是通过前端 Axios 发送 HTTP 请求，请求后端 http://47.100.249.168:5000/map 获得，并且将由后端请求到的数据，动态绑定到可视化图形之中。

如下图所示，前端向后端的 Flask 服务器请求数据，在后端服务器的日志文件之中可以找到如下记录：

```
127.0.0.1 - - [29/May/2022 09:59:10] "GET /map HTTP/1.0" 200 -
127.0.0.1 - - [29/May/2022 09:59:10] "GET /map HTTP/1.0" 200 -
127.0.0.1 - - [29/May/2022 09:59:18] "GET /shanghai HTTP/1.0" 200 -
127.0.0.1 - - [29/May/2022 10:01:06] "GET /map HTTP/1.0" 200 -
127.0.0.1 - - [29/May/2022 10:01:57] "GET /trend HTTP/1.0" 200 -
127.0.0.1 - - [29/May/2022 11:17:26] "GET /map HTTP/1.0" 200 -
127.0.0.1 - - [29/May/2022 11:18:06] "GET /shanghai HTTP/1.0" 200 -
```

图 5. 后端日志文件记录

针对上图中日志文件的信息解读，如下表所示：

表 5. 日志文件详情

IP 地址	127.0.0.1
被请求时间	2022.05.09
HTTP 请求方式	GET
请求路由	/map /trend /shanghai
HTTP 状态码	上海历史疫情趋势

5. 数据可视化

实验的这一部分使用 Matplotlib 与 Seaborn 数据可视化库对于上海目前的疫情状态进行分析。

本部分上海疫情趋势数据爬取自腾讯疫情实时追踪，https://news.qq.com/zt2020/。

上海市民求助信息爬取自我们来帮你·上海抗疫互助，https://www.helpothers.cn/help/。

5.1 上海疫情趋势分析

绘制并分析以下两张图表：

- 上海新增趋势图
- 上海市民求助信息趋势图

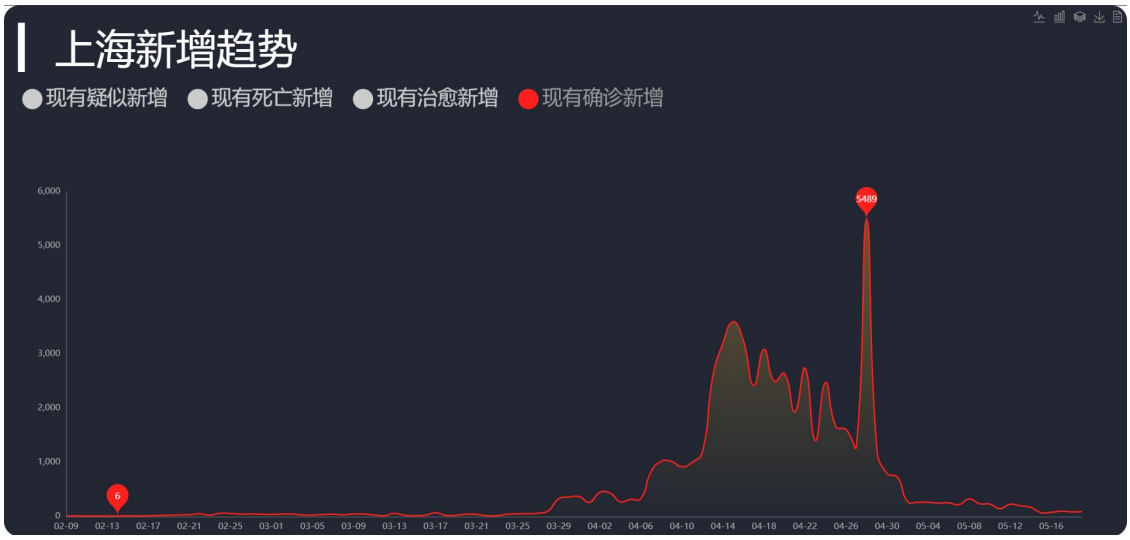


图 6. 上海新增趋势图

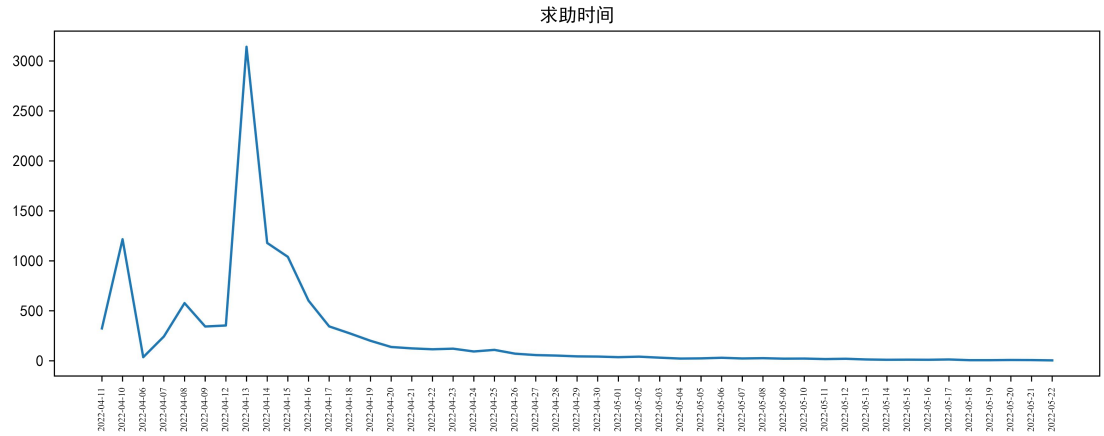


图 7. 上海市民求助趋势图

数据分析后，可以得出以下结论：

- 确诊新增从 4 月 6 日起，开始显著增加。与此同时，上海求助信息也在显著增多。并且在 4 月 14 日左右达到一个小高峰，与此同时上海求助信息也达到了峰值。
- 4 月 14 日之后新增情况逐渐下降，而在 4 月 27 日至 4 月 29 日间爆发时增长与回落。
- 从 4 月底开始，上海新增逐渐趋于 0，而市民求助信息量在每一天只有个位数，逐渐趋于 0。

5.2 上海市民求助信息分析

下面对于上海疫情趋势变化进行分析，自 3 月 31 日起，上海的防疫政策由精准防控调整为全员核酸，同时进行严格的区分防控，共计分为封控区、管控区和防范区实施网格化管理，这三个区域实施动态调整，管控区、防范区内一旦发现感染者，对所在区域实施升级管控并落实相应防控措施。

而在政策初步实行阶段，出现了新增确诊病例的明显增长，居民求助信息也开始增多。而随着政策的一步步调整与优化，在 4 月 10 日时，上海市场监管局发布关于规范疫情防控期间“社区团购”价格行为的提示函，针对市民担心的“买不到”“难买到”，尤其是主副食品和生活必需品问题，上海市聚焦民生物资，重点加大食品、母婴、药品等物资的供应力度。

因此在政策的支持之下，市民向社会发出求助的数量明显减少，疫情得到了有效控制。

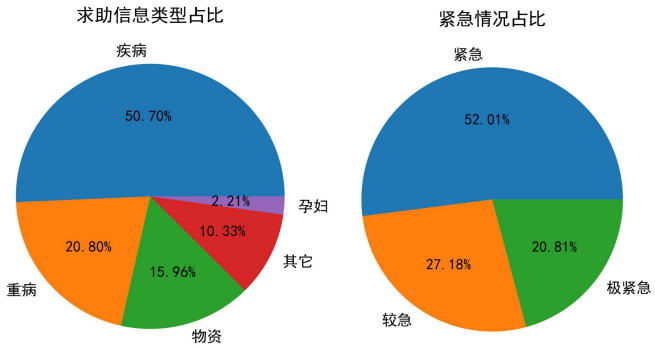


图 8. 上海市民求助信息情况占比

针对于各类求助信息进行数据分析，绘制词云图与求助关键词柱状图如下：

图 9. 上海市民求助信息词云

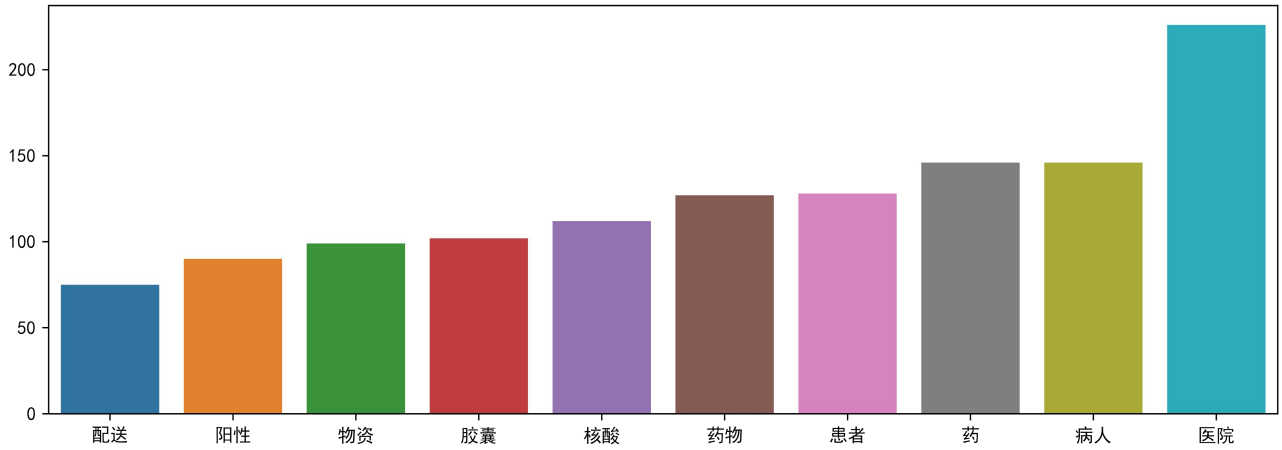


图 10. 上海市民求助信息

分析数据可知，总计 11457 条上海市民求助信息之中，因为 1847 种不同的具体原因求助，其中上海求助关键词 60%以上与疾病有关，且疾病类型多样，情况复杂。

结合紧急情况占比，紧急与极紧急之和占 75%，也应证了疾病无法治疗等确实属于紧急情况，也正由于遭遇疾病无法治疗此类紧急事件，求助者才选择将信息发在网上求助。

5.3 上海疫情地图分析

结合上海疫情地图，分析求助者来自的区域。可以得出不同地区疫情严重程度与求助人数具有强相关性，疫情较为严重的地区，例如浦东、徐汇、杨浦等，来自这些地区的求助者也相应较多。

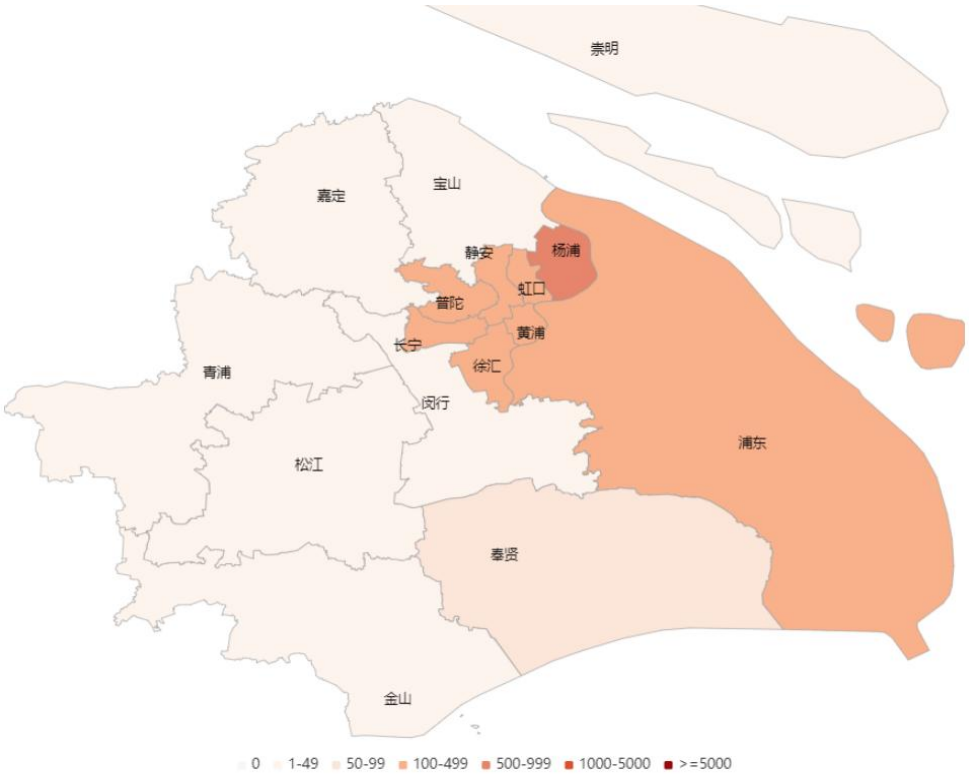


图 11. 上海疫情地图

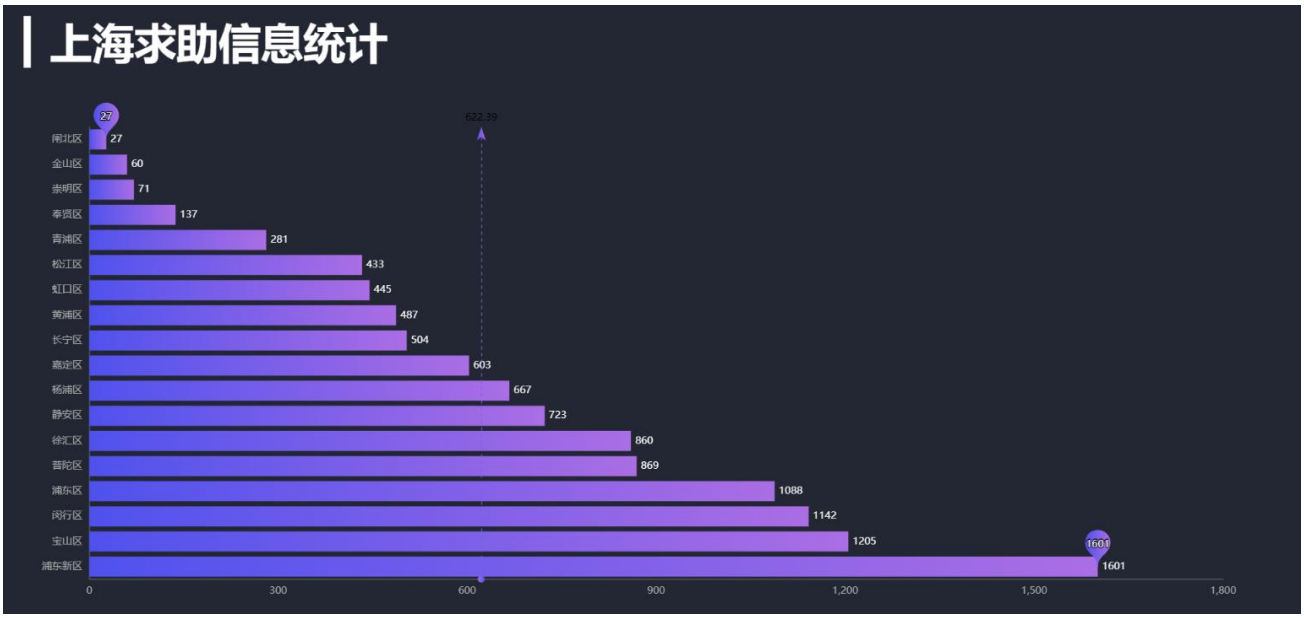


图 12. 上海各区市民求助信息统计

5.4 疫情与谣言分析

疫情谣言信息爬取自丁香园，https://portal.dxy.cn/。

TF-IDF 是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF 加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。

本实验中共获取到 400 余条谣言信息，对这些谣言信息构建 TF-IDF 模型，寻找最容易被造谣的关键信息。

分析结果可知，造谣者最喜欢以病毒口罩等作为造谣的关键词，此外“钟南山”、“卫健委”等关键词重要性也较高，与预期相符，造谣者借“钟南山”、“卫健委”的口吻散播谣言，恶意骗取点击量与关注度。

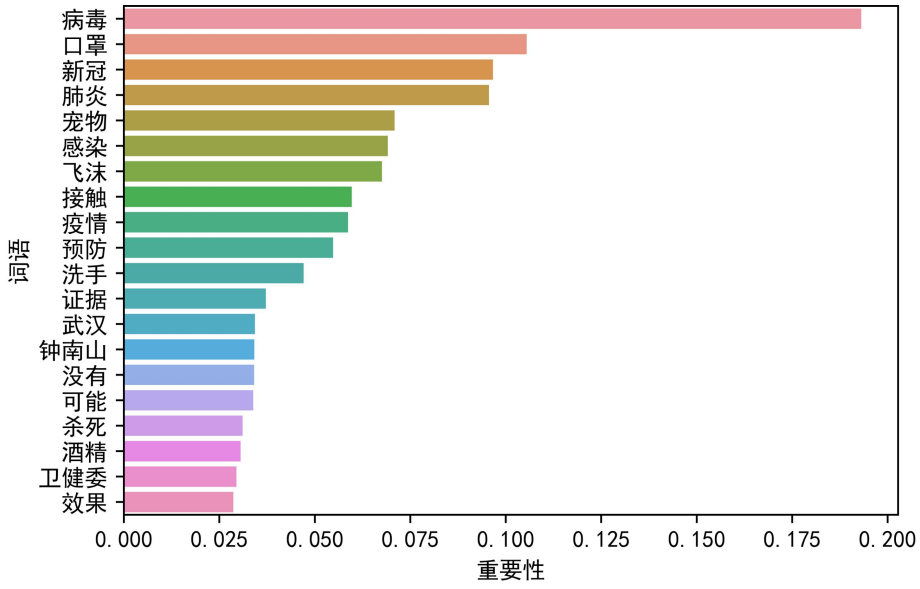


图 13. 疫情谣言 TF-IDF 分析

五、测试用例

1. 全国新增趋势

此处对于全国新增趋势堆叠图以及全国疫情地图进行功能测试。

本网页展示了全国现有死亡新增、现有治愈新增、现有确诊新增的堆叠图。此页面可以随时间变化详细展示每一天的具体信息。并且将这三种信息的最大值，最小值，以“水滴”的样式展现在图中，而每一项指标的平均值以折线的形式展示。

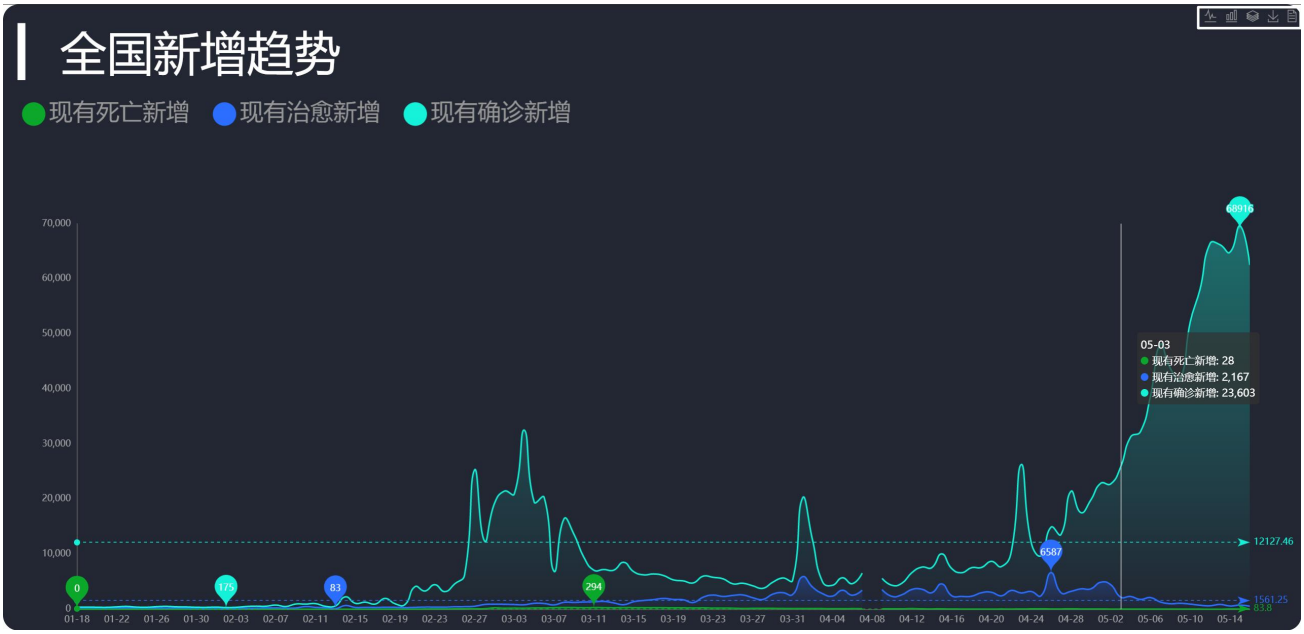


图 14. 全国新增趋势折线堆叠图

并且在页面右上角处(上图中白框区域)增加了由折线图一键转变为条状堆叠图的功能，样例如下图所示：

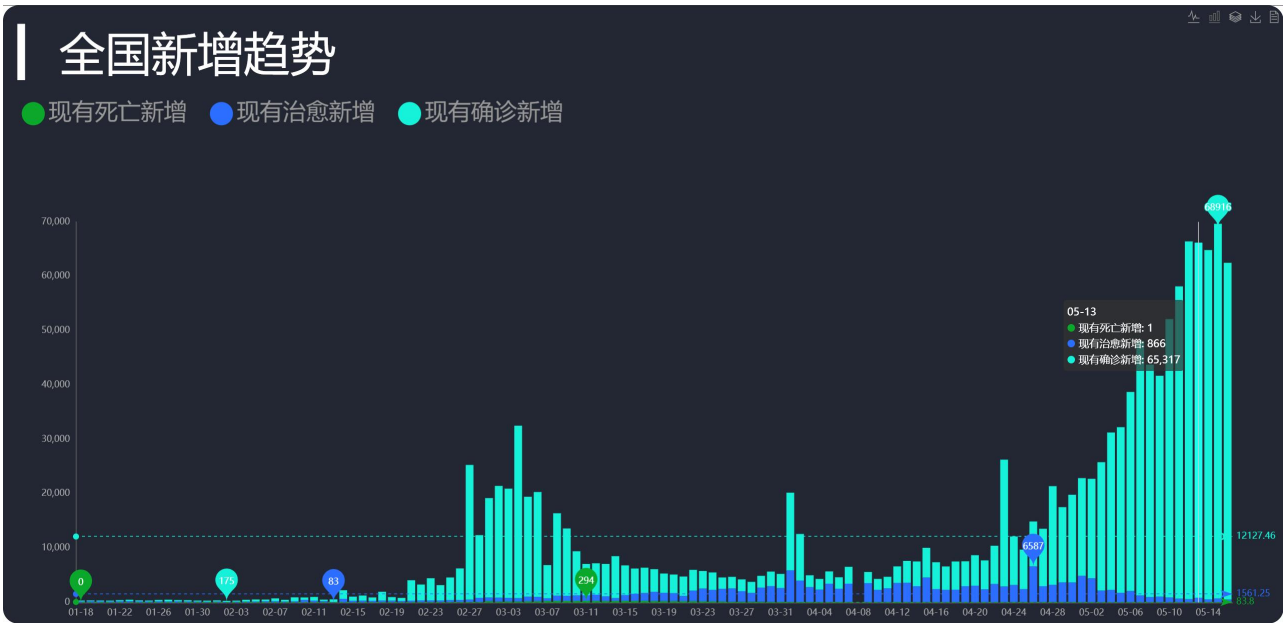


图 15. 全国新增趋势条形堆叠图

2. 全国现有确诊

下面以全国现有确诊疫情地图为例，介绍前端的编写。

下方疫情地图可以根据现有确诊人数，选择不同确诊人数区间内的省份。并且根据不同省份间确诊人数的多少，对于不同地区的疫情严重程度进行颜色区分。例如对于下方图片而言，当只选择确诊人数在[1-9]区间内的地区时，疫情地图中只存在粉色区域与无确诊人数的白色区域，如图 17 所示。

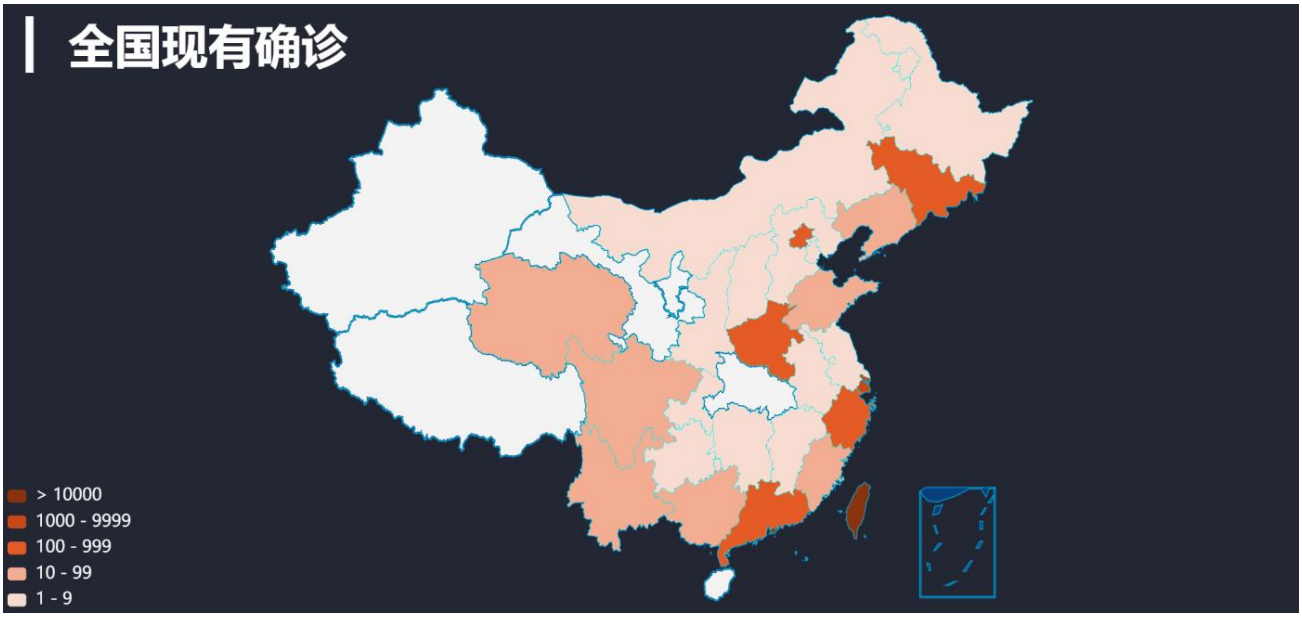


图 16. 全国现有确诊疫情地图

图 17. 全国现有确诊疫情地图

六、收获与体会

春季学期注定对我们来说是个难以忘记的学期，它有着两个学期的考试都堆在一起的紧张刺激，也有着疫情突如其来的措手不及。不知不觉，这也是我足不出校的整整第 90 天。查询了学校的健康之路我才发现，自 3 月 16 日以来，我已经做过了 40 次核酸，以及数不清楚多少次抗原。因此对于 Python 计算的期末实验，我想做贴近生活的主题，于是从疫情数据获取开始，一步步做到疫情数据可视化，分析疫情的进展。

本实验有效代码 3000 余行，可以说，在本次实验过程中，我 80%的代码编写时间用于爬取数据、后端代码编写、数据库操作前端数据绑定的编写，而数据可视化部分只使用了 20%的时间。然而，大家了解我的工作大多来自于数据可视化的 20%。与数据可视化的美观与华丽相比，数据的提供部分更像默默奉献的角色，为可视化的展示提供了夯实的条件与基础。颇有“台上一分钟，台下十年功”的感受。

我的主力编程语言是 Java，我更多使用 Java 的 SpringBoot 框架做后端开发。在此次实验中，我接触了 Python 的 Flask 后端开发框架，我明显的感受到 Python 作为后端开发的语言使用并不如 Java 方便，Flask 的 ORM 操作并不流畅，并且它在返回多层嵌套的 json 数据时很容易逻辑不清晰，导致代码写的很乱。在遇到相关问题时，很难找到对应的解决方式。与 Python 相关的后端开发的学习资料与 Java 相比特别少，主流的学习网站之中，用 Java 后端教学视频点击量几乎是用 Python 后端教学视频点击量的近 10 倍。但是在使用 Python 编写数据爬虫与机器学习时，强大的第三方库的确能减轻我的工作量，Python 语言还是更适合机器学习任务与数据分析任务。

选修 Python 计算这门课，除了提高了自学能力，最大的感受锻炼了自己的辨别能力。我深刻的明白了不同编程语言各有所长，C++速度快，适合硬件处理，Java 生态好，适合做后端开发，Python 第三方库强大，适合机器学习、数据分析等等。因此在自己大二时间充沛的情况下，更要广泛的接触知识，取其所长，使用才能够得心应手。

由于学校三学期制度，使得我们的节奏进度较快，导致自己没有充足的时间去细细探究每一个 Python 第三方库的延申和拓展，让自己可以熟练掌握其应用，并且举一反三，亲自动手实现，属实有些遗憾，在春季学期考试结束之后，我也希望自己能够再次学习相关知识。

最后，希望疫情早日结束，大家都能回到正常的生活轨迹之中。

七、核心代码

因篇幅有限，此处仅展示 Python 后端的部分代码，爬虫、前端、可视化部分不再展示。

后端业务代码：

```
from flask import Flask, jsonify
from flask_cors import CORS
from jieba.analyse import extract_tags
import string
import db


app = Flask(__name__)
CORS(app, supports_credentials=True)

@app.route("/map")
def get_map_data():
    res = []
    for tup in db.get_c2_data():
        # [{'name': '上海', 'value': 318}, {'name': '云南', 'value': 162}]
        res.append({"name": tup[0], "value": int(tup[1])})
    return jsonify({"data": res})


@app.route("/trend")
def get_add_trend():
    data = db.get_add_data()
    day_ls, confirm_add_ls, suspect_add_ls, heal_add_ls, dead_add_ls = [], [], [], [], []
    for date, confirm_add, suspect_add, heal_add, dead_add in data:
        day_ls.append(date.strftime("%m-%d"))  # a 是 datatime 类型
        confirm_add_ls.append(confirm_add)
        suspect_add_ls.append(suspect_add)
        heal_add_ls.append(heal_add)
        dead_add_ls.append(dead_add)

    return jsonify(
        {"day": day_ls, "confirm_add_ls": confirm_add_ls, "suspect_add_ls": suspect_add_ls, "heal_add_ls": heal_add_ls,
         "dead_add_ls": dead_add_ls})


if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

数据库操作代码：

```
import pymysql

def get_conn():
    """
    :return: 连接，游标
    """
    # 创建连接
    conn = pymysql.connect(host="127.0.0.1",
                           user="root",
                            password="12345",
                            db="cov",
                            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
    return conn, cursor

def close_conn(conn, cursor):
    cursor.close()
    conn.close()


def query(sql, *args):
    conn, cursor = get_conn()
    cursor.execute(sql, args)
    res = cursor.fetchall()
    close_conn(conn, cursor)
    return res


def get_shanghai_data():
    """
    :return:  返回上海疫情数据
    """
    sql = "select date_, con_num, sus_num, cure_num, death_num, con_add, sus_add, cure_add, dea
th_add from shanghai_history order by date_ desc limit 100"
    res = query(sql)
    return res[::-1]
```

八、课堂研讨

Flask

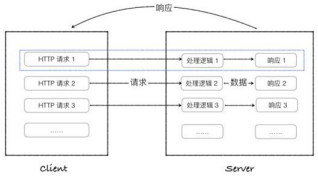
第八小组

20121034 胡才郁
20121036 黄逸弘
20121054 徐鼎力
20121059 朱若时
20121706 张俊雄

开场介绍

Web框架

•所有Flask程序都必须创建一个程序实例。
•当客户端想要获取资源时，一般会通过浏览器发起HTTP请求。
•Web服务器把来自客户端的请求都交给Flask程序实例。
•Flask使用来做路由分发在Flask程序中，路由一般是通过程序实例Python的装饰器实现。
•通过调用视图函数，获取到数据后，然后由Flask返回响应数据给浏览器，最后浏览器显示返回的结果。



Web 框架介绍

简单启动

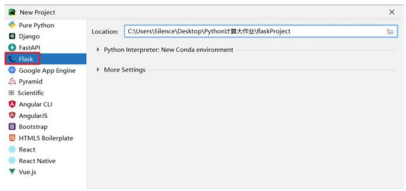
代码实现

```python
# 导入Flask类
from flask import Flask

#Flask类接收一个参数__name__
app = Flask(__name__)

# 装饰器的作用是将路由映射到视图函数index
@app.route('/')
def index():
    return 'Hello World'

# Flask应用程序实例的run方法启动WEB服务器
if __name__ == '__main__':
    app.run()
```

浏览器访问

Flask 启动方式

PyCharm 对于主流框架支持



PyCharm 支持

路由传参

代码实现

```python
# 路由传递的参数默认当做string处理，这里指定int，尖括号中冒号后面的内容是动态的
@app.route('/user/<int:id>')
def router_test(id):
    return '%d' % id
```

浏览器访问

服务器输出

```
 * Serving Flask app 'app.py' (lazy loading)
 * Environment: development
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [23/May/2022 18:17:34] "GET /user/20121034 HTTP/1.1" 200 -
```

路由传参

数据库配置

```python
from flask import Flask
from flask_sqlalchemy import SQLAlchemy


app = Flask(__name__)

#设置连接数据库的URL
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://root:root@127.0.0.1:3306/Flask_test'

#设置每次请求结束后会自动提交数据库中的改动
app.config['SQLALCHEMY_COMMIT_ON_TEARDOWN'] = True

app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True
#查询时会显示原始SQL语句
app.config['SQLALCHEMY_ECHO'] = True
db = SQLAlchemy(app)
```

数据库配置

ORM

ORM是通过使用描述对象和数据库之间映射的元数据，将程序中的对象自动持久化到关系数据库中。

说人话：**不写sql语句，直接操作对象**

例如:查找用户表中id为2的学生

不使用ORM的Sql写法

```
sql = "SELECT ... FROM persons WHERE id = 2"
```

使用ORM的写法

```
User.query.filter(User.id == 2).all()
```

User表

	id	name	email	pswd	role_id
1	1	wang	wang@163.com	123456	1
2	2	zhang	zhang@189.com	201512	2
3	3	chen	chen@126.com	987654	2
4	4	zhou	zhou@163.com	456789	1

ORM 介绍

数据库支持 Flask-SQLalchemy

常用的SQLAlchemy字段类型

类型名	python中类型	说明
Integer	int	普通整数，一般是32位
SmallInteger	int	取值范围小的整数，一般是16位
BigInteger	int或long	不限制精度的整数
Float	float	浮点数
Numeric	decimal.Decimal	普通整数，一般是32位
String	str	变长字符串
Text	str	变长字符串，对较长或不限长度的字符串做了优化
Unicode	unicode	变长Unicode字符串
UnicodeText	unicode	变长Unicode字符串，对较长或不限长度的字符串做了优化
Boolean	bool	布尔值
Date	datetime.date	时间
Time	datetime.datetime	日期和时间
LargeBinary	str	二进制文件

数据库支持

模型类配置

```
class Role(db.Model):
    # 定义表名
    __tablename__ = 'roles'
    # 定义列对象
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(64), unique=True)
    us = db.relationship('User', backref='role')

    #repr()方法显示一个可读字符串
    def __repr__(self):
        return 'Role:%s'% self.name

class User(db.Model):
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(64), unique=True, index=True)
    email = db.Column(db.String(64),unique=True)
    pswd = db.Column(db.String(64))
    role_id = db.Column(db.Integer, db.ForeignKey('roles.id'))

    def __repr__(self):
        return 'User:%s'%self.name
```

Role表

	id	name
1	1	admin
2	2	user

User表

	id	name	email	pswd	role_id
1	1	wang	wang@163.com	123456	1
2	2	zhang	zhang@189.com	201512	2
3	3	chen	chen@126.com	987654	2
4	4	zhou	zhou@163.com	456789	1

模型类配置

增

```
# 创建表
db.create_all()
# 删除表
db.drop_all()

# 插入数据
ro1 = Role(name='admin')
db.session.add(ro1)
db.session.commit()
# 再次插入一条数据
ro2 = Role(name='user')
db.session.add(ro2)
db.session.commit()
# 一次插入多条数据
us1 = User(name='wang',email='wang@163.com',psmd='123456',role_id=ro1.id)
us2 = User(name='zhang',email='zhang@189.com',psmd='201512',role_id=ro2.id)
us3 = User(name='chen',email='chen@126.com',psmd='987654',role_id=ro2.id)
us4 = User(name='zhou',email='zhou@163.com',psmd='456789',role_id=ro1.id)
db.session.add_all([us1,us2,us3,us4])
db.session.commit()
```

删

```
# 查询数据后删除
user = User.query.first()
db.session.delete(user)
db.session.commit()
User.query.all()
```

数据库增删操作

夹带一些私货(tù cáo)...

Flask作为Web框架

- 社区小、资料难查
- 工作岗位少、需求小
- 生态不好、工具少、重复造轮子
- 复杂的数据接口难以编写

个人认为的适用范围?

- Python作为第一语言的同学
- Python计算大作业....数据库大作业.... 各种大作业....
- 编写不需要定制的数据接口
- 机器学习项目WEB端展示

个人使用心得

谢谢观看

FLASK

20121034 胡才郁

结束致谢