Cathay interview's report Take Home Exam

Content

- 第一題
  - 實作架構
  - 處理過程
  - 處理結果
- 第二題
  - 實作架構
  - 處理過程
  - 實作營運化方法&機制
- 成長
  - 專案進度規劃
  - 遇到的問題
  - 解決方法
  - 展望

Github repo: littlefish0331/cathay_intervie_591

Folder Structure

簡單說明一下資料夾的結構，這部分也可以觀看 Github 的 READMD

```
Interview_cathay
├── README.md: 專案說明
├── codePy/: Python 程式碼
├── result/: 處理好的資料
├── data/: raw data
├── image/
├── key/: MongoDB 的 connection string
├── report_youjun.pptx
└── Take_Home_Exam.md : 擔心放 PDF 有智慧財產權，所以額外寫一份
```

Quiz 01

第一題 - 實作架構

見 quiz01_report.ipynb

```
Interview_cathay
├── codePy/: Python 程式碼
├── result/: 處理好的資料
└── data/: raw data
```

處理好的資料這裡

自動下載的資料放這裡

第一題 – 處理過程

- 觀察內政部不動產時價登陸網站，取得 csv 資料下載的 API endpoint
- 自動下載
  - 檢查資料夾位置是否存在
  - 下載資料
- 資料讀取
  - 放入 list
  - 合併為 pd.Dataframe
- 資料處理
  - 「樓層數」的中文數字轉一般數字
  - 擷取「交易筆棟數」的車位數量
  - 計算平均注意實質意義(除去車位數=0)
- 資料匯出

第一題 – 處理結果

filter_a結果

```
In [6]: pd.read_csv(os.path.join(path_result, "./filter_a.csv"), nrows=5)
```

Out[6]:

鄉鎮市區	交易標的	土地區段位置建物區段門牌	土地移轉總面積平方公尺	都市土地使用分區	非都市土地使用分區	非都市土地使用編定	交易年月日	交易筆棟數	移轉層次	...	車位移轉總面積(平方公尺)	車位總價元	備註	編號	主建物面積	附屬建物面積	陽台面積	電梯	移轉編號	total_floor
大同區	房地(土地+建物)+車位	民權西路121~150號	4.25	都市：其他:第三種商業區。	NaN	NaN	1100319	土地1建物1車位1	六層	...	0.00	0	陽台外推；夾層；其他增建；	RPOOMLQJNHGGFBA17DA	23.58	1.23	3.58	有	5.0	14
萬華區	房地(土地+建物)+車位	環河南路二段1~30號	8.62	都市：其他:第肆種商業區(依都市計畫說明書圖規定辦理，始得	NaN	NaN	1100414	土地2建物1車	四層	...	19.94	0	NaN	R						

```
In [4]: pd.read_csv(os.path.join(path_result, "./filter_b.csv"))
```

Out[4]:

	attr	value
0	總件數	8348.00
1	總車位數	4143.00
2	平均總價元	15301387.96
3	平均車位總價元	995157.89

filter_b結果

Quiz 02

第二題 - 實作架構

見 quiz02 系列，以及資料夾 flask_api/

- quiz02_report01_crawl_rent591.ipynb
- quiz02_report02_ETL_json.ipynb
- quiz02_report03_crawl_contact.ipynb
- quiz02_report04_Flask.ipynb

```
Interview_cathay
├── codePy
│   ├── flask_api
│   ├── quiz
│   ├── quiz02*. ipynb
├── key
└── result/: 爬取資料
```

API 主程式

function

程式碼: 爬蟲、資料前處理、塞入mongoDB

爬取的資料、清理好的資料

第二題 – 實作架構 – 爬蟲

依照 quiz02 系列說明步驟

- 利用 session 機制先取得網頁 token
  - 使用 591 的 API 抓取租屋內容
  - 中間更改縣市時，重新建立連線取得新的 token (避免被擋)
  - 紀錄每篇貼文的['post_id', 'refreshtime']，後續更新會用到
- 資料初步清理，並上傳到 mongoDB
  - 使用 mongodb atlas，並設定使用者與防火牆，以達到安全機制
  - 啟用 Replica Set – 3 nodes(atlas 預設)
  - 另將資料備援到自己的 mongodb container
- 補爬po者的連絡資訊，並 update 到 mongodb
- 建立 API

第二題 – 處理結果(API test by Postman)

```
{
    "shape": 1,
    "kind": 4,
    "renter": {
        "renter_role": "屋主",
        "renter_fname": "余" ,
        "renter_title": "先生"
    },
    "sex": 2,
    "contact": "0921173418",
    "sectionid": 1
}
```

json template

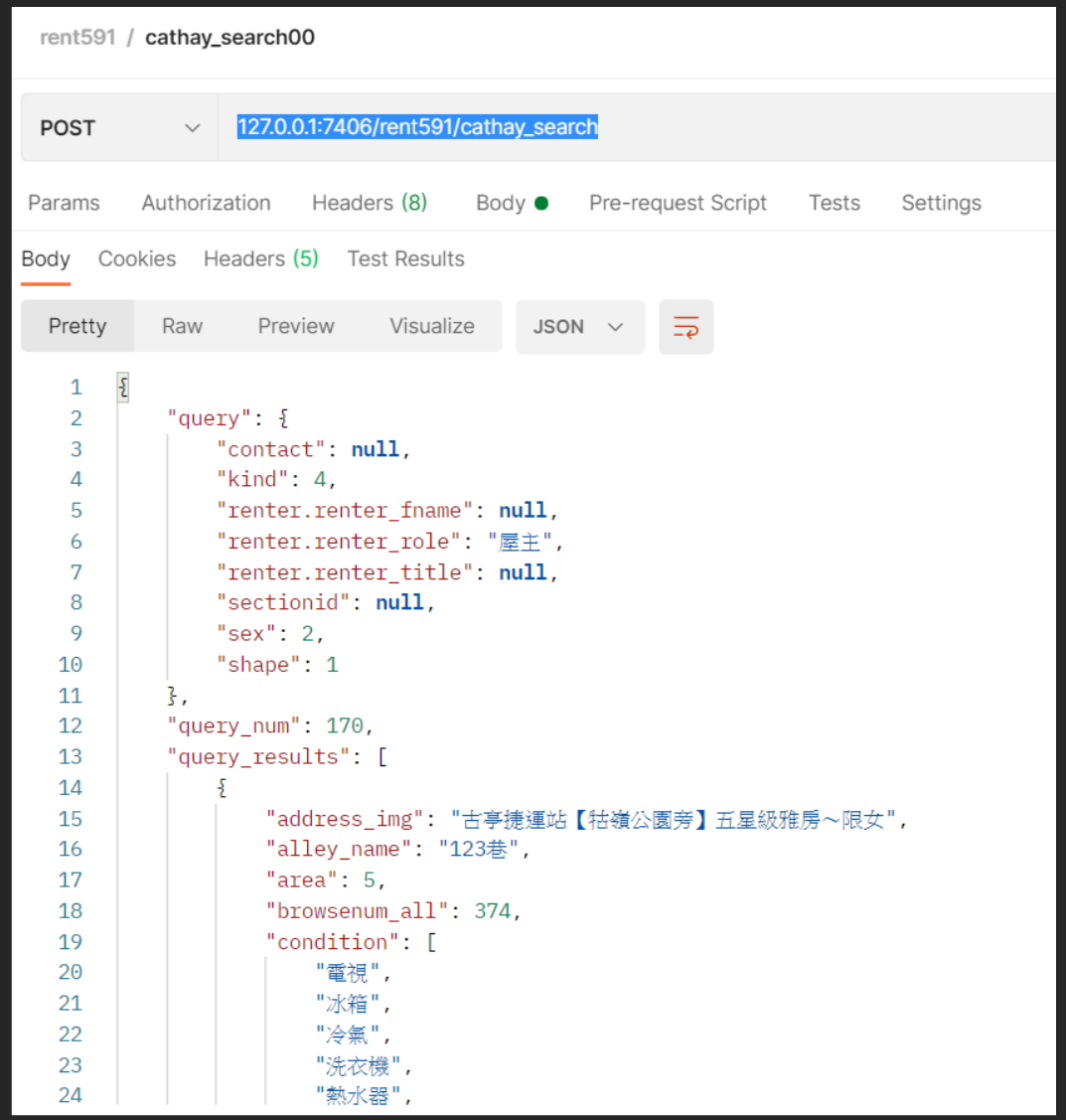
- Host: 127.0.0.1 or 203.145.218.12:7406*
- API endpoint:
  - POST: /rent591/cathay_search

json 格式如右，設計時依照面試考題要求
所以只有這些選項，另有做一個比較多參數，並提供 **query string** 做彈性呼叫的。

回傳結果為符合條件貼文的 **meta data**，見下頁。

*: 目前對外防火牆未開

第二題 – 處理結果(API test by Postman)



回傳下列資訊

- `query`: 使用者的 `json` 篩選條件
- `query_num`: 符合條件的po文數
- `query_results`: 查詢結果

第二題 – 營運化方法&機制 – 更新

利用 `/result/meta_refresh.csv`

- 爬取網頁時發現，591會依照PO文時間做排序，所以先預設抓取第一頁
- 觀察 post_id, refreshtime(po文更新時間)，對照上述的 csv，便可知道po文是否更新
  - 若有更新(不管事新po文或是舊po文修改)，將資料依照 post_id，update 資料庫
  - 不斷爬取，直到發現 post_id, refreshtime 一致，就跳出

優點:

- 如果有更新(未刪除po文)，此機制一定可以update

缺點:

- 如果貼文刪除，無法得知是哪一篇刪除，或許可以比對總筆數，但仍無法知曉確切post_id，此部分建議為使用時，若 query 發現問題，就回傳失敗，並更新資料庫

第二題 – 營運化方法&機制 – 資料庫設計

第一次接觸 NoSQL，依照官方建議，將資料分成

- meta data 以及 search 的 coolection
- One-to-many 的機制，減少單一 collection query 負擔。

API的效能可能要考慮用 AWS CloudWatch + auto scaling 去協助增減，或是考慮去偶，增加SQS 機制

API的規格文件: ~~從缺(自首)~~

summary

成長 – 專案規劃進度

過去公司的專案皆使用 R 進行操作，包過資料分析、資料建模等等。
所以這次面試最大的難題就是**『靈活使用 python』**

老實說，自己的 python 只有跑過簡單的 pandas, numpy 以及 matplotlib，
而且都是最基礎的範例 code，工作上最多曾使用 python 進行經緯度轉換。

因此，針對這次會用到的基本功(包含 python, pandas, mongodb, flask)，
我稍微為自己做了學習與實作進度。~~(不過很多 DELAY)~~

成長 – 專案規劃進度

*: 表示進度拖延

成長 – 遇到的問題

[第一題]

- 很多 R 與 Python 操作的思維不同，所以比預期花了更多的時間。
- 想將整個 script 都全部自動化(包含一開始的下載動作)

[第二題]

- 不會爬蟲
- 不會 mongoDB
- 不會 Flask

成長 – 解決辦法

[第一題]

- 很快速地看完線上的 pandas – groupby 操作的線上課程，順便作為 python 的複習。
  - 課程名稱: Pandas Basics and GroupBy: Intro to Python Data Science
- 大量閱讀文章

[第二題 1/2]

- 爬蟲*
  - 閱讀書籍(《Python 網路爬蟲 – 王者歸來) 完整看完)
  - 參考書籍(《Python 網路爬蟲實戰 by 清華大學出版社》)
  - 大量閱讀文章
- mongoDB
  - Youtube 影片教學、官方文件
  - 實作 (local, atlas, docker)

*: 針對爬蟲遇到的問題，後續補充

成長 – 解決辦法(續)

[第二題 2/2]

- mongoDB 資料結構與collection設計(續)
  - 官方文件、blog
- Flask
  - Youtube 影片
  - 大量文章

[爬蟲問題與解決]

- 網路教學不可行，因為網站改版，有基本防爬，API 需要 token
- 第一版使用 selenium，爬取實在是太緩慢了
- **改用 session 機制，取得 token 後，便可用 API 取得大量資料，問題解決。**
- 發現 591 的 API 內容沒有聯絡電話，只好再重新爬取網頁。

展望 – 可以做得更好

- 不熟悉 python，後來才發現 Django 可以像 R:plumber自動生成 openapi.json 規格文件 然後接 swagger 觀看，有機會可以把這塊完成。(flask也有，但好像推薦用 Django ?)
- 有設計自己的 query 邏輯，但時間上來不及完成(假日也許會做完)，並介接 Line Chatbot 做測試。(順便找台北房子)
- 爬蟲的程式碼可以有更好、有更彈性的寫法
- tryCatch 的機制不夠完善，認真測試一定會出問題
- 沒有考慮負載問題。
- 更新機制尚未啟動
- 尚未搬移到 VM 上，讓API endpoint公開。*

*: 這個遷移部署動作有經驗，但老實說自己到最後一刻才完成作業要求，所以還來不及轉移。連測試API都只能用 Postman 截圖展示。

展望 – 未來

[Python 能力需再精進]

- 可以體悟到，其實想做的事情每個程式語言都可以完成，但是主流的 python 是資料工程師的必要技能。如果有時間會再精進下列技術
  - **不同方法的爬蟲技術、網路架構**
  - API 的撰寫(Flask, Django)的模組化

[資料工程方面]

- **API 負載能力、高可用性的架構考量，以及如何測試?**
- Docker network 原理實作，增加資料庫安全性

[結果互動呈現]

- LINE chatbot (有稍微花1/3天研究，但是時間不太夠，後續有機會會完成)
- 網頁介面，考慮 RWD 響應式模組，可快速建立(Django? Softr 2.0?)
- 商業價值與分析: 資料有了，可以再結合時價登錄，做進一步的研究XD

Thank You
Any question?