CHAPTER

12

使用 Python 爬取 AJAX、互動網頁與 Web API

12-1 AJAX 與 JSON 的基礎

AJAX 是 Asynchronous JavaScript And XML 的縮寫，即非同步 JavaScript 和 XML 技術，AJAX 可以讓 Web 應用程式在瀏覽器建立出更人性化的使用介面。

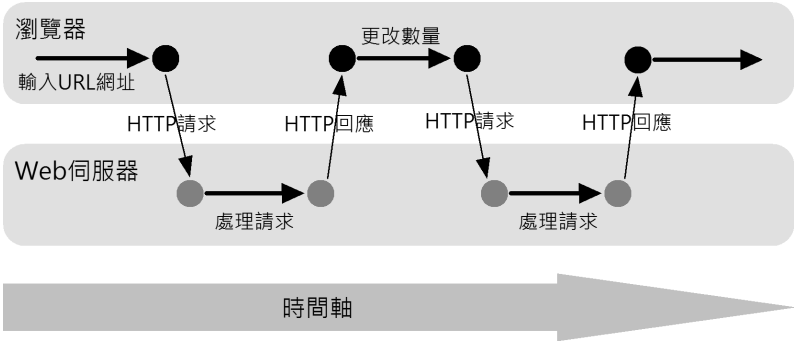
12-1-1 AJAX 的基礎

AJAX 是 Jesse James Garrett 最早提出的名稱，其技術的核心是非同步 HTTP 請求（Asynchronous HTTP Requests），可以讓 HTTP 請求不用等待伺服端的回應，就可以讓使用者執行其他互動操作，例如：更改購物車購買的商品數量後，不需等待重新載入網頁，就可以接著輸入送貨的相關資訊。

簡單的說，非同步 HTTP 請求可以讓網頁使用介面，不會因為 HTTP 請求的等待回應而中斷，因為同步 HTTP 請求需要重新載入整頁網頁內容，如果網路稍慢，可能看見空白頁和網頁逐漸載入的過程，這是和 Windows 應用程式使用者介面之間的最大差異。

同步 HTTP 請求

傳統 HTTP 請求的過程是同步 HTTP 請求(Synchronous HTTP Requests)，當使用者在瀏覽器的網址欄輸入 URL 網址後，按下按鈕，就可以將 HTTP 請求送至 Web 伺服器，在處理後，將請求結果的 HTML 網頁回傳客戶端來顯示，如下圖所示：



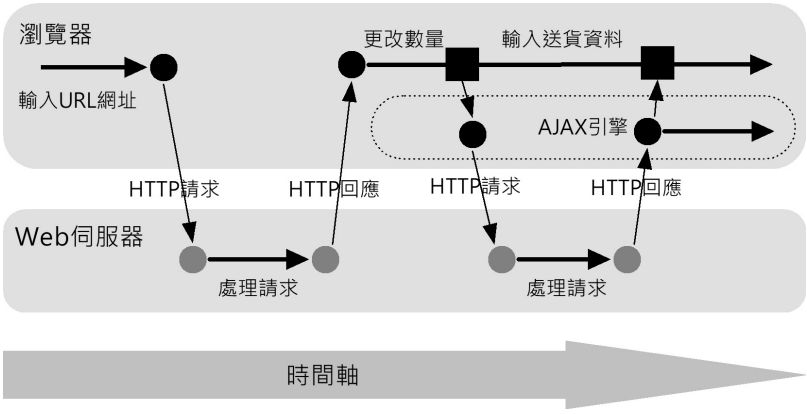
上述圖例在瀏覽器輸入 URL 網址後，將 HTTP 請求送至 Web 伺服器，在處理後，產生購物車網頁傳回瀏覽器顯示，如果數量不對，在更改後，再次送出 HTTP 請求，並且取得回應。

在同步 HTTP 請求的過程中，回應內容都是整頁網頁，所以在等待回應的時間中，使用者唯一能作的就是等待，需要等到回應後，才能執行下一階段的互動，例如：輸入送貨資料。

所以，使用者在網頁輸入資料等互動操作時，是和 HTTP 請求同步的，其過程依序是輸入資料、送出 HTTP 請求、等待、取得 HTTP 回應和顯示結果，完成整個流程後，才能進行下一次互動。

非同步 HTTP 請求

AJAX 是使用非同步 HTTP 請求，除了第 1 次載入網頁外，HTTP 請求是在背景使用 XMLHttpRequest 物件送出 HTTP 請求，在送出後，並不需要等待回應，所以不會影響使用者在瀏覽器進行的互動，如下圖所示：



上述圖例在瀏覽器第 1 次輸入 URL 網址後，將 HTTP 請求送至 Web 伺服器，在處理後，產生購物車網頁傳回瀏覽器顯示，如果數量不對，在更改後，就透過 JavaScript 建立的 AJAX 引擎（AJAX Engine）送出第 2 次的 HTTP 請求，因為是非同步，所以不用等到 HTTP 回應，使用者可以繼續輸入送貨資料。

當送出第 2 次 HTTP 請求在伺服器處理完畢後，AJAX 引擎可以取得回應的 XML 或 JSON 等資料，然後更新指定標籤物件的內容，即更改數量，所以並不用重新載入整頁網頁內容。

AJAX 的 HTTP 請求和使用者輸入資料等互動操作是非同步的，因為 HTTP 請求是在背景執行，執行後也不需等待回應，而是由 AJAX 引擎處理請求、回應和顯示，使用者的操作完全不會因為 HTTP 請求而中斷。

AJAX 應用程式架構

AJAX 的主要目的是改進 Web 應用程式的使用介面，屬於一種客戶端網頁技術，在實務上，我們可以搭配伺服端網頁技術來建立更佳使用介面的 Web 應用程式，例如：ASP、ASP.NET、PHP 和 JSP 等。

AJAX 應用程式架構的最大差異是在客戶端，客戶端使用 JavaScript 的 AJAX 引擎來處理 HTTP 請求，和取得伺服端回應的文字、HTML、XML 或 JSON 資料（伺服端網頁技術產生），如下圖所示：

上述"author"是欄位名稱的鍵，"陳會安"是值，JSON 的值可以是整數、浮點數、字串（使用「"」括起）、布林值（true 或 false）、陣列（使用方括號括起）和物件（使用大括號括起）。

JSON 物件

JSON 物件是使用大括號包圍的多個 JSON 鍵和值，如下所示：

```
{
  "title": "C 語言程式設計",
  "author": "陳會安",
  "category": "Programming",
  "pubdate": "06/2018",
  "id": "P101"
}
```

JSON 物件陣列

JSON 物件陣列可以擁有多個 JSON 物件，例如："Employees"欄位的值是一個物件陣列，擁有 3 個 JSON 物件，如下所示：

```
{
  "Boss": "陳會安",
  "Employees": [
    { "name" : "陳允傑", "tel" : "02-22222222" },
    { "name" : "江小魚", "tel" : "02-33333333" },
    { "name" : "陳允東", "tel" : "04-44444444" }
  ]
}
```

12-2 使用開發人員工具分析和測試 AJAX 請求

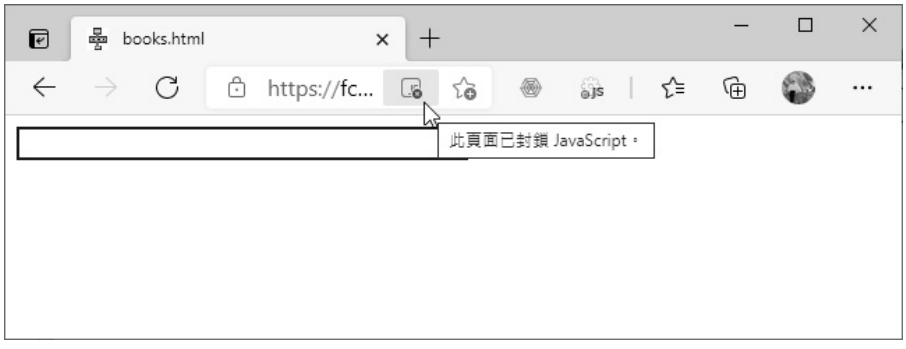
開發人員工具可以擷取網路流量，幫助我們取得客戶端和伺服端之間交換的資料，以便找出目標資料所在的 AJAX 請求。

12-2-1 使用開發人員工具分析 AJAX 請求

我們除了需要找出使用 AJAX 在背後送出的 HTTP 請求，以便找到資料來源，還需檢視 HTTP 標頭資訊找出請求方法，其步驟如下所示：

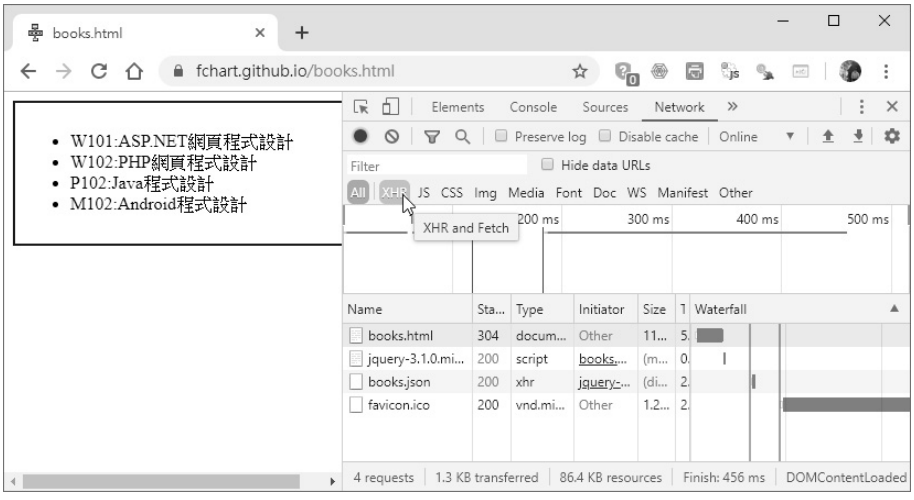
1. 請使用瀏覽器進入 https://fchart.github.io/books.html，可以看到 4 本圖書清單，點選 Quick JavaScript Switcher 擴充功能圖示關閉執行 JavaScript，如下圖所示：

2. 可以看到圖書清單不見了，表示圖書資料是在之後才載入，這是一種使用 AJAX 技術產生的網頁內容。

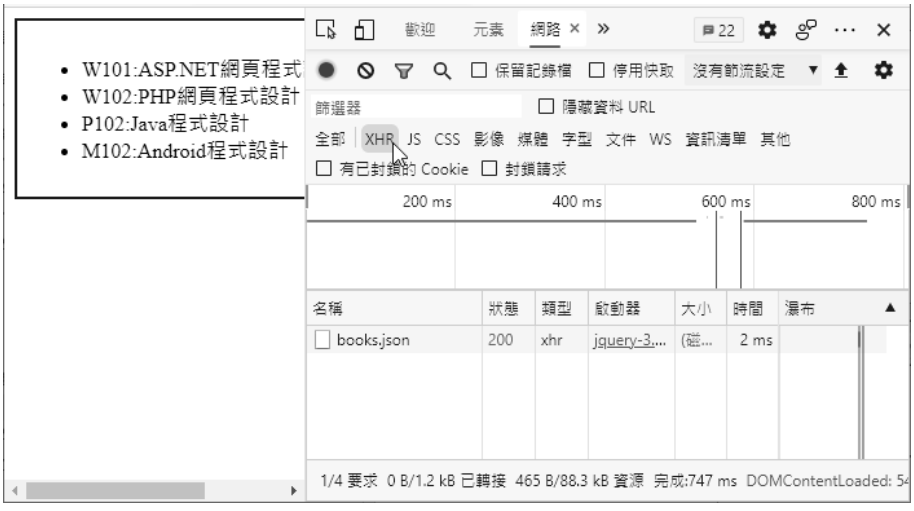


3. 請再次點選 Quick JavaScript Switcher 擴充功能圖示切換執行 JavaScript，可以再度看到 4 本圖書清單後，按 F12 鍵切換至開發人員工具。

4 選【網路】（Network）標籤，按 F5 鍵重新載入網頁，開始擷取網路流量，稍等一下，預設是在【全部】（All）標籤顯示擷取流量的完整項目清單，包含名稱、狀態和類型等資訊，以此例共有 4 個 HTTP 請求，選【XHR】只顯示 AJAX 請求，如下圖所示：

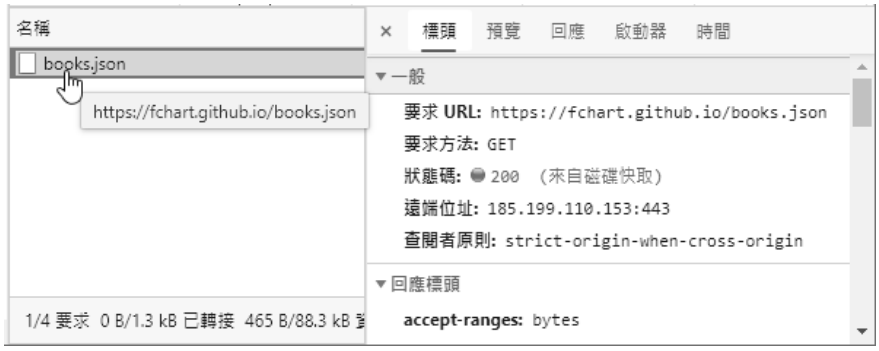


5 可以看到只剩下一個【books.json】項目，如下圖所示：



上述【文件】（Doc）標籤是 document 類型的 HTTP 請求，這些是取得網頁內容的 HTTP 請求。

6 點選欲檢視的流量【books.json】，可以檢視 HTTP 標頭資訊，即【標頭】（Header）標籤，可以看到請求方法是 GET，如下圖所示：

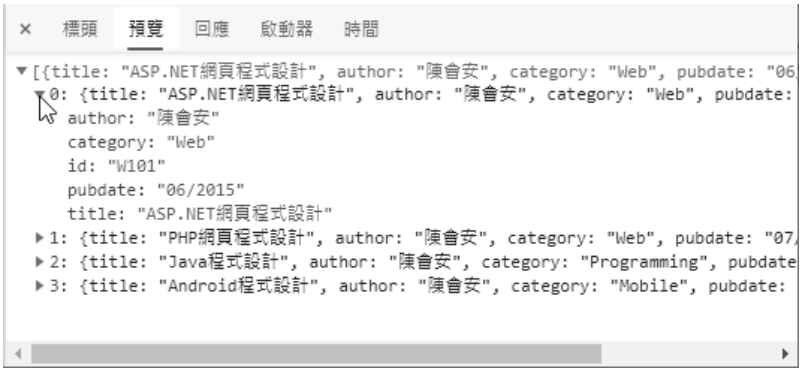


7 選【回應】（Response）標籤，可以看到回傳的 JSON 字串內容，這是 4 本圖書的資料。

從上述回應（Response）標籤的內容可以證明網頁內容是在瀏覽器載入網頁後，才在背景使用 JavaScript 程式碼送出 HTTP 請求來取得 JSON 資料，所以 HTTP 請求是位在【XHR】標籤。

8 如果回應內容很長，我們可以按 Ctrl + F 鍵來搜尋資料，例如：在下方欄位輸入 Java，按 Enter 鍵，可以搜尋到此筆 JSON 物件，如下圖所示：

9 選【預覽】（Preview）標籤，可以看到階層方式顯示的 JSON 資料，點選前方小箭頭展開 JSON 資料，如下圖所示：



10 最後需要取得 AJAX 請求的 URL 網址，請在 XHR 項目【books.json】上，執行【右】鍵快顯功能表的「複製>複製連結位址」命令（Chrome 是「Copy>Copy link address」命令），可以將 URL 網址複製至剪貼簿，如下所示：

https://fchart.github.io/books.json

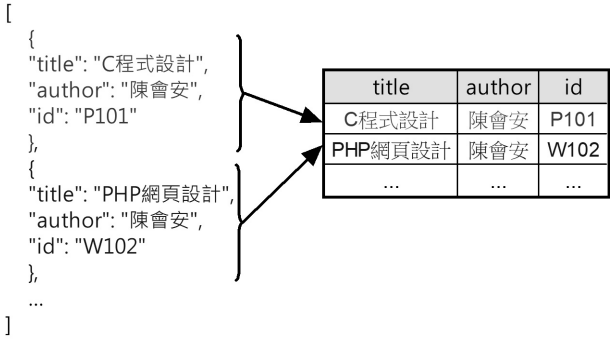
因為 AJAX 請求是 GET 方法，我們可以直接在瀏覽器測試和顯示請求的回應資料，如下圖所示：

請注意！Edge 瀏覽器並不會格式化編排 JSON 資料；Chrome 瀏覽器會格式化編排 JSON 資料。

12-3 Python 處理 JSON 資料

JSON 是一種能夠自我描述和容易了解的資料交換格式，使用大括號定義成對的鍵和值（Key-value Pairs），相當於物件的屬性和值，類似 Python 語言的字典和串列。

JSON 如果是物件陣列，每一個物件是一筆記錄，我們可以使用方括號「[ ]」來定義多筆記錄，如同是一個表格資料，如下圖所示：



Python 的 JSON 處理是使用 json 模組，只需配合檔案處理即可將 JSON 資料寫入檔案，和讀取 JSON 檔案內容。

JSON 和 Python 字典的轉換：ch12-3.py

在 json 模組的 dumps()方法可以將 JSON 字典轉換成 JSON 字串，loads()方法是從 JSON 字串轉換成 JSON 字典，如下所示：

```
import json

data = {
   "name": "Joe Chen",
   "score": 95,
   "tel": "0933123456"
}

json_str = json.dumps(data)
print(json_str)
```

```
data2 = json.loads(json_str)
print(data2)
```

上述程式碼首先呼叫 dumps()方法，將字典轉換成 JSON 資料內容的字串，然後呼叫 loads()方法，再將字串轉換成字典，其執行結果如下所示：

```
{"name": "Joe Chen", "score": 95, "tel": "0933123456"}
{'name': 'Joe Chen', 'score': 95, 'tel': '0933123456'}
```

將 JSON 資料寫入檔案：ch12-3a.py

我們可以使用 json 模組的 dump()方法將 Python 字典寫入 JSON 檔案，如下所示：

```
import json

data = {
   "name": "Joe Chen",
   "score": 95,
   "tel": "0933123456"
}

jsonfile = "Example.json"
with open(jsonfile, 'w') as fp:
    json.dump(data, fp)
```

上述程式碼建立字典 data 後，使用 open()函數開啟寫入檔案，然後呼叫 dump()方法將第 1 個參數的 data 字典寫入第 2 個參數的檔案，可以在 Python 程式的目錄看到建立的 Example.json 檔案。

讀取 JSON 檔案：ch12-3b.py

我們是使用 json 模組的 load()方法將 JSON 檔案內容讀取成 Python 字典，如下所示：

```
import json

jsonfile = "Example.json"
with open(jsonfile, 'r') as fp:
    data = json.load(fp)
```

```
json_str = json.dumps(data)
print(json_str)
```

上述程式碼開啟 JSON 檔案 Example.json 後，呼叫 load()方法讀取 JSON 檔案轉換成字典，接著轉換成 JSON 字串後顯示 JSON 內容，其執行結果如下所示：

```
{"name": "Joe Chen", "score": 95, "tel": "0933123456"}
```

12-4 使用 Web API 取得網路資料

Web API 就是一種 REST API，REST（REpresentational State Transfer）是架構在 WWW 的 Web 應用程式架構，目前政府機構和各大軟體廠商都提供有付費或免費的 Web API，可以讓我們直接撰寫 Python 程式透過 Web API 來取得網路資料。

12-4-1 認識 Web API

Web API（Web Application Programming Interface）是一種標準方法透過 Internet 網際網路來執行其他系統提供的功能，我們就是使用 HTTP 請求來執行其他系統提供的 Web API 方法。

如同在瀏覽器輸入 URL 網址來瀏覽網頁，很多公開 API 可以直接在瀏覽器執行來取得網路資料，回應資料大多是 JSON 格式的資料。

Web API 的種類

基本上，目前 Web API 可以分成兩種，如下所示：

- 公開 API（Public/Open API）：任何人不需註冊帳號就可以使用的 Web API。
- 認證 API（Authenticated API）：需要先註冊帳號後才能使用的 Web API。

上述帳號可能需付費或免費註冊，在註冊後，可以得到 API 金鑰（API Key），執行 Web API 時，需要提供 API 金鑰的認證資料。

Web API 的認證方式

一般來說，當 Web API 是使用 GET 方法的 HTTP 請求時，有些是公開；有些需要認證；POST 方法大部分都需要認證。Web API 的認證方式主要有 2 種，如下所示：

- 使用 API 金鑰認證：當註冊 Web API 帳號取得 API 金鑰後，GET 方法是使用參數來指定認證資料，POST 方法是使用自訂標頭名稱來指定認證資料。
- 使用帳號和密碼認證：直接使用註冊的帳號和密碼進行認證，視 Web API 文件的說明，可能是使用自訂標頭來指定帳號和密碼，或是使用 get()方法的參數來指定帳號和密碼。

12-4-2 直接從網站下載資料

目前很多網站或政府單位的 Open Data 開放資料網站都可以直接下載資料，不用撰寫任何 Python 程式碼就可以取得所需資料。

下載台灣期交所未平倉量

台灣期交所三大法人未平倉量的下載網址，如下所示：

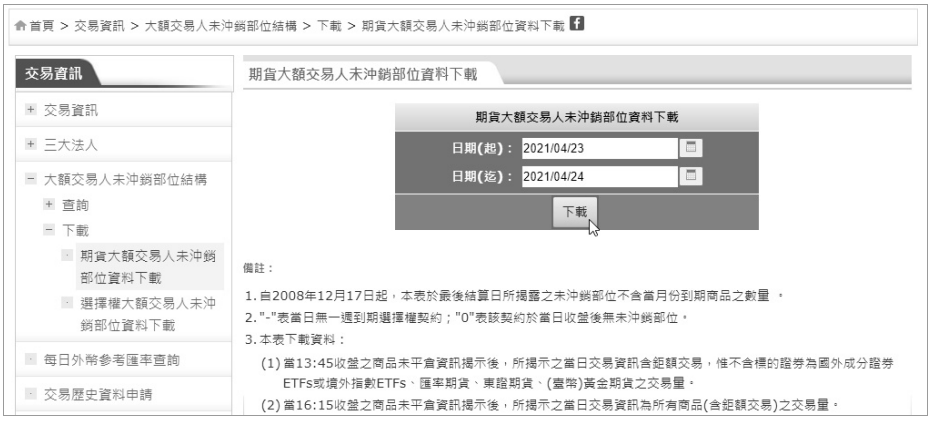
https://www.taifex.com.tw/cht/3/futAndOptDateView

在表格輸入日期範圍，按【下載】鈕，可以下載三大法人未平倉量。

下載大額交易人未平倉量的 URL 網址，如下所示：

https://www.taifex.com.tw/cht/3/largeTraderFutView

輸入日期範圍，按【下載】鈕，可以下載大額交易人未平倉量。

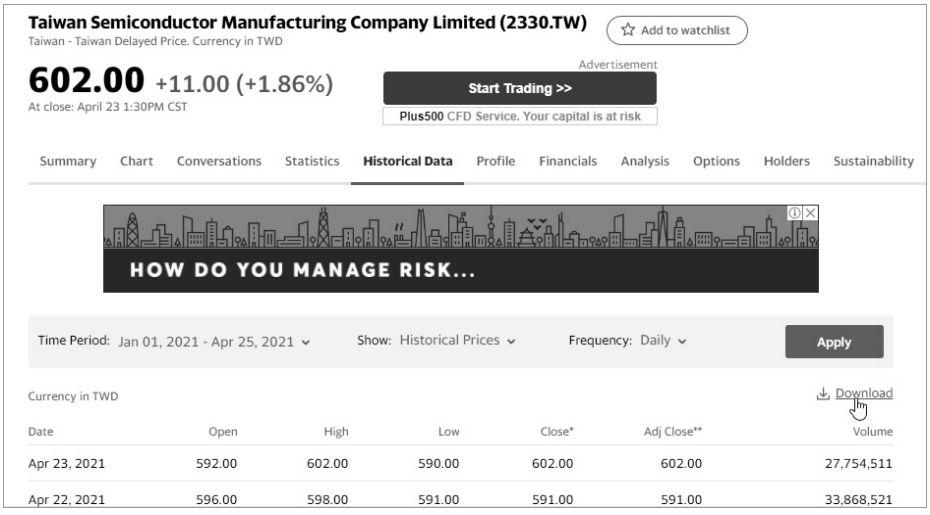


下載美國 Yahoo 的股票歷史資料

在美國 Yahoo 財經網站可以下載股票的歷史資料，例如：台積電，其 URL 網址，如下所示：

https://finance.yahoo.com/quote/2330.TW

上述網址最後的 2330 是台積電的股票代碼，.TW 是台灣股市，如下圖：



請在上述網頁選反白【Historical Data】標籤後，在下方左邊選擇時間範圍，右邊按【Apply】鈕顯示股票的歷史資料後，點選下方【Download Data】超連結，可以下載以股票名稱為名的 CSV 檔案。

12-4-3 Google 圖書查詢的 Web API

Google 圖書查詢是使用 Google Books APIs 查詢圖書資訊，其回應資料是 JSON 格式的資料，在這一節筆者準備建立 Python 程式來查詢 Python 圖書的資訊。

使用 Google Books APIs

Google Book APIs 可以讓我們在線上查詢指定條件的圖書資訊，其格式如下所示：

https://www.googleapis.com/books/v1/volumes?q=<關鍵字>&maxResults=5&projection=lite

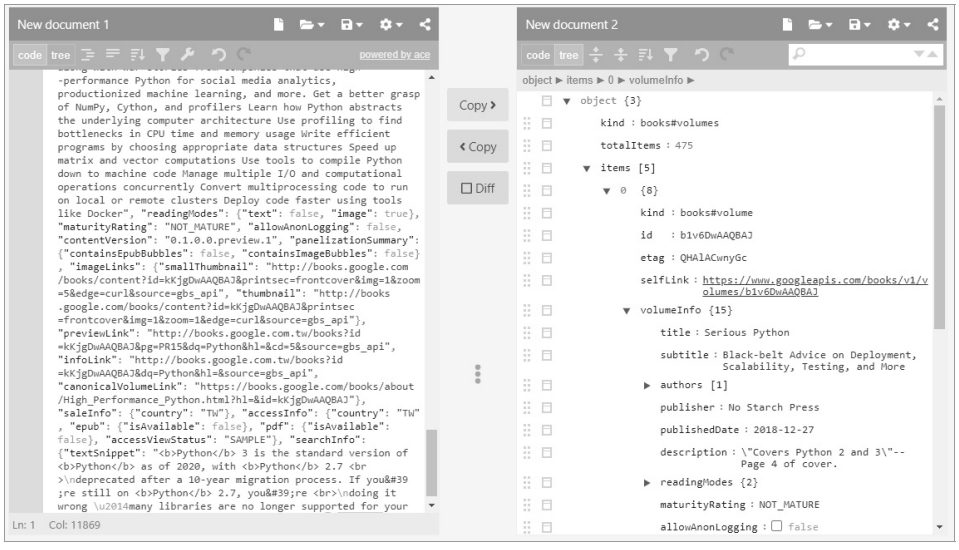
上述網址的 q 參數是關鍵字，maxResults 是最大搜尋筆數，5 是最多 5 筆圖書，最後 1 個參數是取回精簡圖書資料。例如：查詢 Python 圖書，如下：

https://www.googleapis.com/books/v1/volumes?maxResults=5&q=Python&projection=lite


```
{
    "kind": "books#volumes",
    "totalItems": 475,
    "items": [
        {
            "kind": "books#volume",
            "id": "b1v6DwAAQBAJ",
            "etag": "isspJhOPFWY",
            "selfLink": "https://www.googleapis.com/books/v1/volumes/b1v6DwAAQBAJ",
            "volumeInfo": {
                "title": "Serious Python",
                "subtitle": "Black-belt Advice on Deployment, Scalability, Testing, and More",
                "authors": [
                    "Julien Danjou"
                ],
                "publisher": "No Starch Press",
                "publishedDate": "2018-12-27",
                "description": "\"Covers Python 2 and 3\"--Page 4 of cover.",
                "readingModes": {
                    "text": false,
                    "image": false
                },
                "maturityRating": "NOT_MATURE",
                "allowAnonLogging": false,
                "contentVersion": "preview-1.0.0",
                "panelizationSummary": {
                    "containsEpubBubbles": false,
                    "containsImageBubbles": false
                },
                "imageLinks": {
                    "smallThumbnail": "http://books.google.com/books/content?id=b1v6DwAAQBAJ&p
                    "thumbnail": "http://books.google.com/books/content?id=b1v6DwAAQBAJ&prints
                },
```


上述圖例是 RestMan 格式化顯示的 JSON 資料，可以看出 JSON 資料的結構是一個 JSON 物件。我們可以使用線上 JSON 編輯器來顯示 JSON 資料的階層結構，如下所示：

https://jsoneditoronline.org/



請將 JSON 資料複製至左邊編輯區域，按【Copy >】鈕，就可以在右邊看到 JSON 資料的階層結構。請展開階層結構，totalItems 鍵的值是圖書總數 475，items 鍵的值是 JSON 物件陣列，共有 5 本書，這是每一本圖書的 JSON 物件，在展開後，在 volumeInfo 鍵的值是圖書資訊，title 鍵值是書名"Serious Python"；authors 鍵的值是 JSON 陣列的作者清單。

將 Google 圖書查詢的 JSON 資料寫入檔案：ch12-4-3.py

Python 程式可以將 Google 圖書查詢的 JSON 資料寫入 Books.json 檔案，如下所示：

```
import json
import requests

url =
"https://www.googleapis.com/books/v1/volumes?maxResults=5&q=Python&project
ion=lite"
jsonfile = "Books.json"
r = requests.get(url)
r.encoding = "utf8"
json_data = json.loads(r.text)
with open(jsonfile, 'w') as fp:
    json.dump(json_data, fp)
```

上述程式碼使用 requests.get()方法送出 HTTP 請求後，呼叫 json.loads()方法將讀取資料轉換成字典，然後開啟寫入檔案 Books.json，呼叫 josn.dump()方法寫入 JSON 檔案，可以在 Python 程式的相同目錄看到建立的 Books.json 檔案。

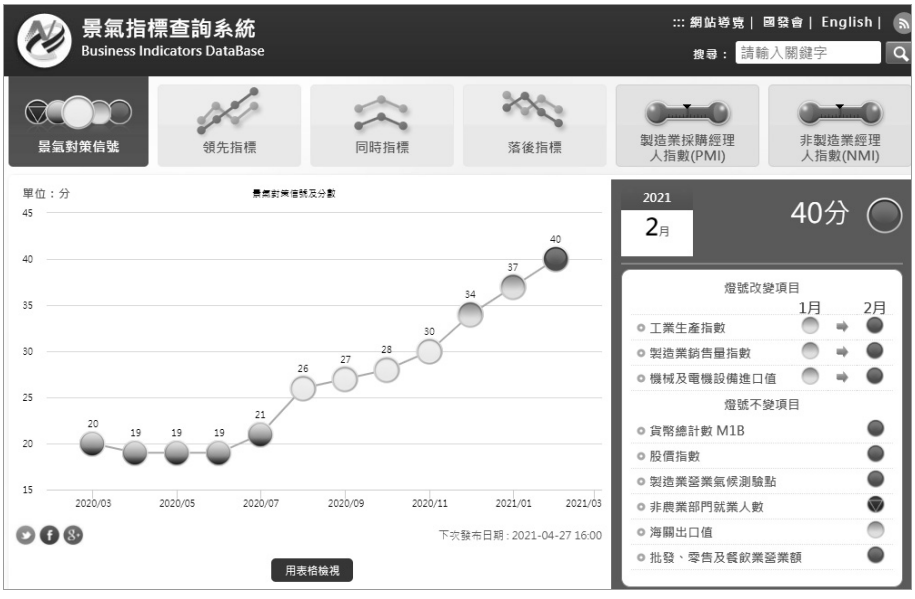
12-5 Python 爬蟲實戰：爬取景氣對策信號分數

我們準備建立 Python 爬蟲程式來爬取國家發展委員會的景氣對策信號分數，其 URL 網址如下所示：

https://www.ndc.gov.tw/

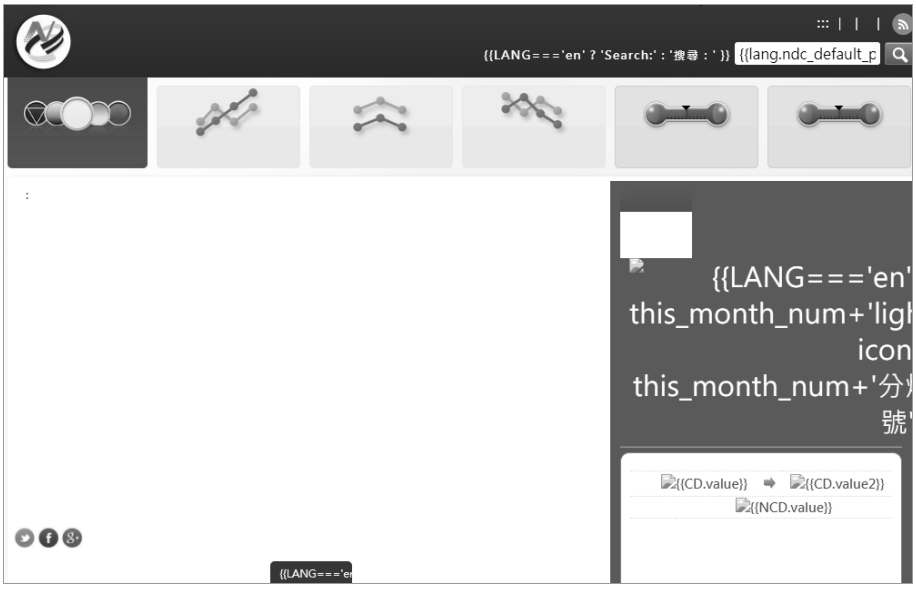
請捲動視窗至下方圖表，可以看到圖表下方的三個按鈕，按中間的【查詢系統】鈕，如下圖所示：

可以看到景氣對策信號及分數的圖表，在圖表繪出的是每一個月份的分數，如下圖所示：



步驟一：判斷網頁內容是否是 JavaScript 動態產生

請使用 Quick JavaScript Switcher 擴充功能關閉執行 JavaScript，可以看到圖表不見了，因為這是使用 JavaScript 程式碼繪出的圖表，如下圖所示：

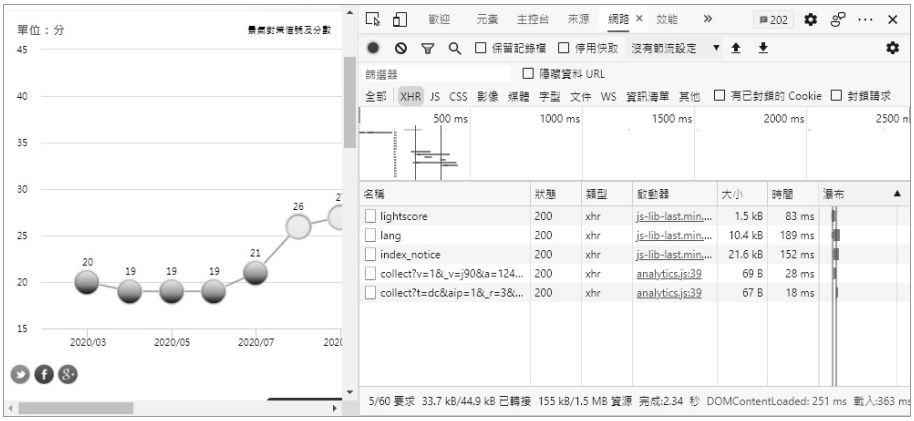


雖然，圖表的景氣對策信號分數的資料可能是直接寫在 JavaScript 程式碼之中，不過，大部分情況是在背景使用 AJAX 請求來取得景氣對策信號分數的資料。

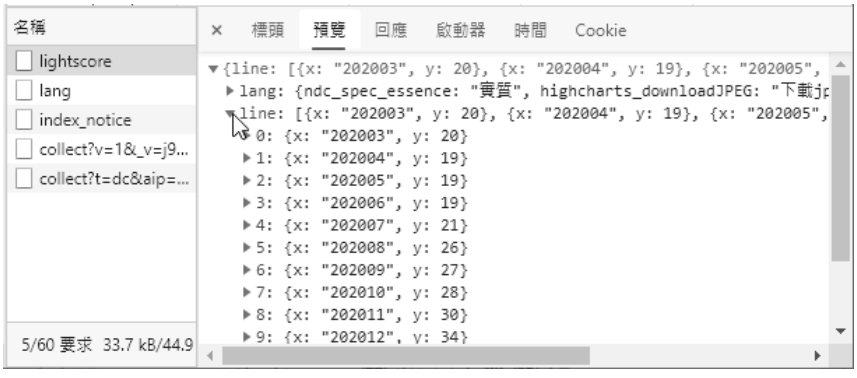
步驟二：使用開發人員工具分析 AJAX 請求

現在，請點選 Quick JavaScript Switcher 擴充功能圖示切換執行 JavaScript 後，使用開發人員工具分析 AJAX 請求，其步驟如下所示：

1. 請開啟開發人員工具，選【網路】（Network）標籤，按 F5 鍵重新載入網頁，即可開始擷取網路流量，選【XHR】只顯示 AJAX 請求，如下圖：

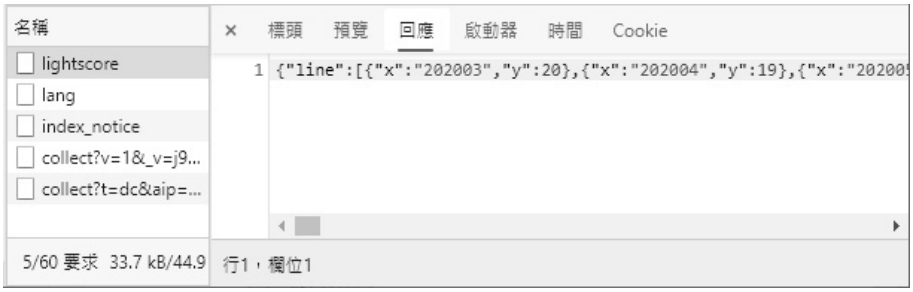


2 點選第 1 個【lightscore】項目，再選【預覽】（Preview）標籤，可以看到回傳的資料，展開 line 可以看到每月的景氣對策信號分數，如下圖所示：



3 選【標頭】（Header）標籤，可以看到請求方法是 POST 請求，如下圖：

4 選【回應】（Response）標籤，可以看到回應的原始 JSON 字串內容，如下圖所示：



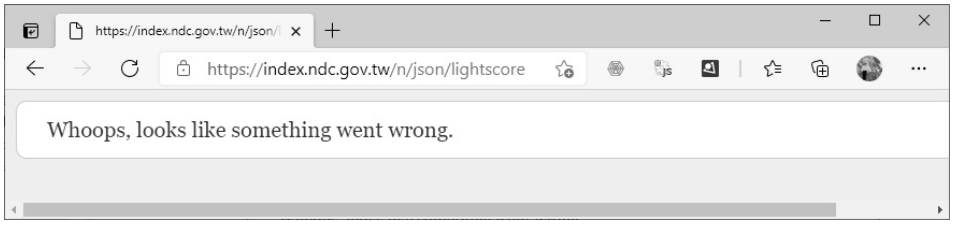
5 請在 XHR 項目上，執行【右】鍵快顯功能表的「複製>複製連結位址」命令（Chrome 是執行「Copy>Copy link address」命令），將 AJAX 請求的 URL 網址複製至剪貼簿。

步驟三：測試 AJAX 請求的 URL 網址

在步驟二已經找出目標資料的 AJAX 請求和其 URL 網址，如下所示：

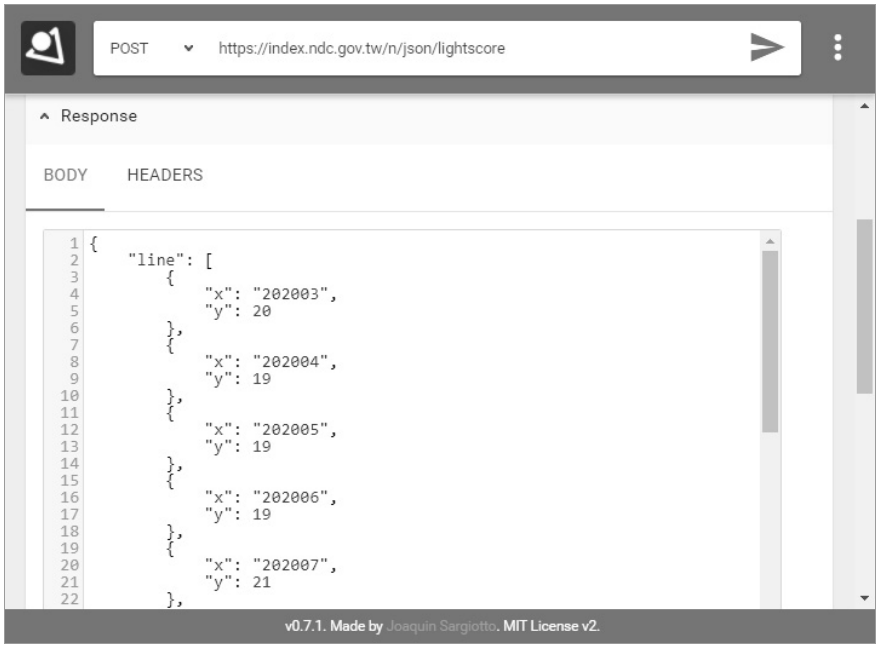
https://index.ndc.gov.tw/n/json/lightscore

首先使用瀏覽器測試上述 AJAX 請求的 URL 網址，並無法取得回應資料，因為是 POST 方法，如下圖所示：



接著使用 RestMan 擴充功能測試 AJAX 請求，選【POST】方法和輸入 URL 網址 https://index.ndc.gov.tw/n/json/lightscore，按游標按鈕送出請求，如下圖所示：

稍等一下，請捲動視窗，可以看到回應的 JSON 資料，如下圖所示：



上述 line 鍵的值是每月的景氣對策信號分數的 JSON 陣列，每一個月是一個 JSON 物件，x 鍵是年/月；y 是分數，如下所示：

```
{
    "x": "202003",
    "y": 20
}
```

步驟四：建立 Python 爬蟲程式爬取 AJAX 請求的資料

現在，我們可以建立 Python 爬蟲程式 ch12-5.py，爬取 AJAX 請求的景氣對策信號分數，在取得回應的 JSON 資料後，儲存成 JSON 檔案 line.json。

Python 程式碼是在第 1~2 行匯入 json 和 requests 模組，第 4 行是 AJAX 請求的 URL 網址，如下所示：

```
01: import json
02: import requests
03:
04: url = "https://index.ndc.gov.tw/n/json/lightscore"
05: jsonfile = "line.json"
06: r = requests.post(url)
07: r.encoding = "utf8"
08: json_data = json.loads(r.text)
09: with open(jsonfile, 'w') as fp:
10:     json.dump(json_data, fp)
```

上述第 6 行呼叫 requests.post()方法送出 HTTP 請求後，在第 8 行呼叫 json.loads()方法將讀取資料轉換成字典後，第 9~10 行開啟寫入檔案 line.json，呼叫 josn.dump()方法寫入 JSON 檔案，可以在 Python 程式的相同目錄看到建立的 line.json 檔案。