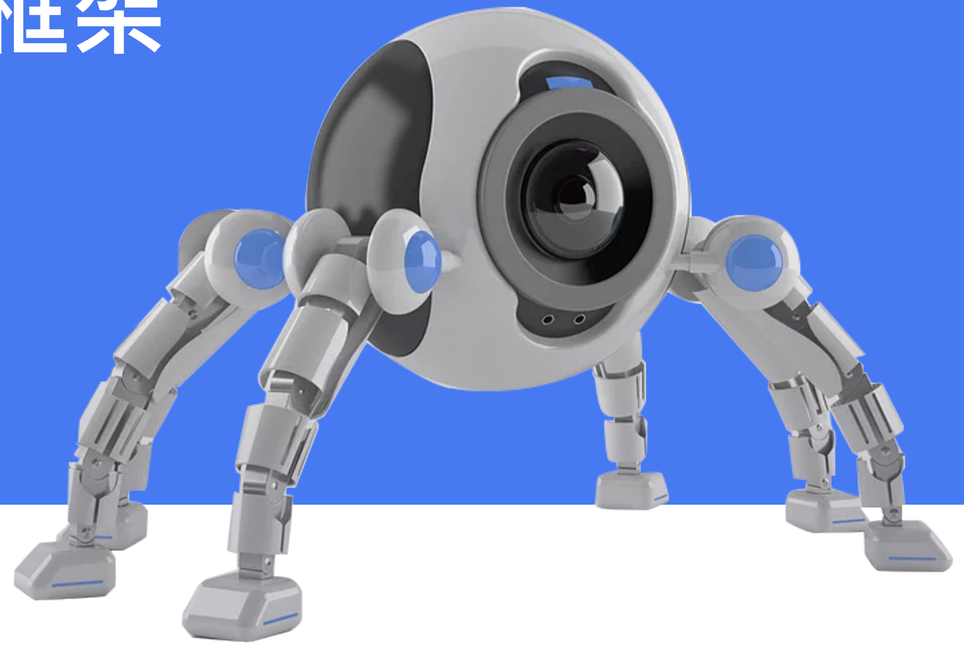
Day 27 SCRAPY 網頁爬蟲框架

Scrapy 元素定位與處理



出題教練：楊鎮銘

python

本日知識點目標

- 了解 Scrapy 內建定位元素的方式
- 了解 Scrapy 針對資料處理邏輯所建立的流程與操作方式

Scrapy 元素定位

爬蟲最重要的兩個部份就是送請求跟定位元素

在介紹 Scrapy 之前我們使用的分別是 requests 與 BeautifulSoup

上個小節我們介紹了 **Scrapy.Request** 來取代 requests，為了符合框架設定所以不適合混用，但定位元素並不限定

這邊為了程式碼的一致性，介紹 Scrapy 定位元素的方式

Scrapy 元素定位

CUPOY

Scrapy 定位元素的方式分別有 XPath 與 CSS selector 兩種

與之前課程中介紹的概念差不多，只是呼叫的 function 不同

- 送出 Scrapy.Request 後取得 Response 物件
- 透過 Response.selector.xpath() 或 Response.selector.css() 定位元素

這邊有建立捷徑可以使用 Response.xpath() 或 Response.css()

範例：parse '<html><body>good</body></html>'

```
response.xpath('//span/text()').get() // good
response.css('span::text').get() // good
```

Scrapy 元素定位比較

CUPOY

Scrapy 框架定位元素方式	非框架元素定位方式
selector.xpah()	lxml.etree.parse().xpath()
selector.css()	BeautifulSoup().select()

如同 BeautifulSoup 一樣，搜尋符合條件的元素可以選擇要回傳一個或是多個

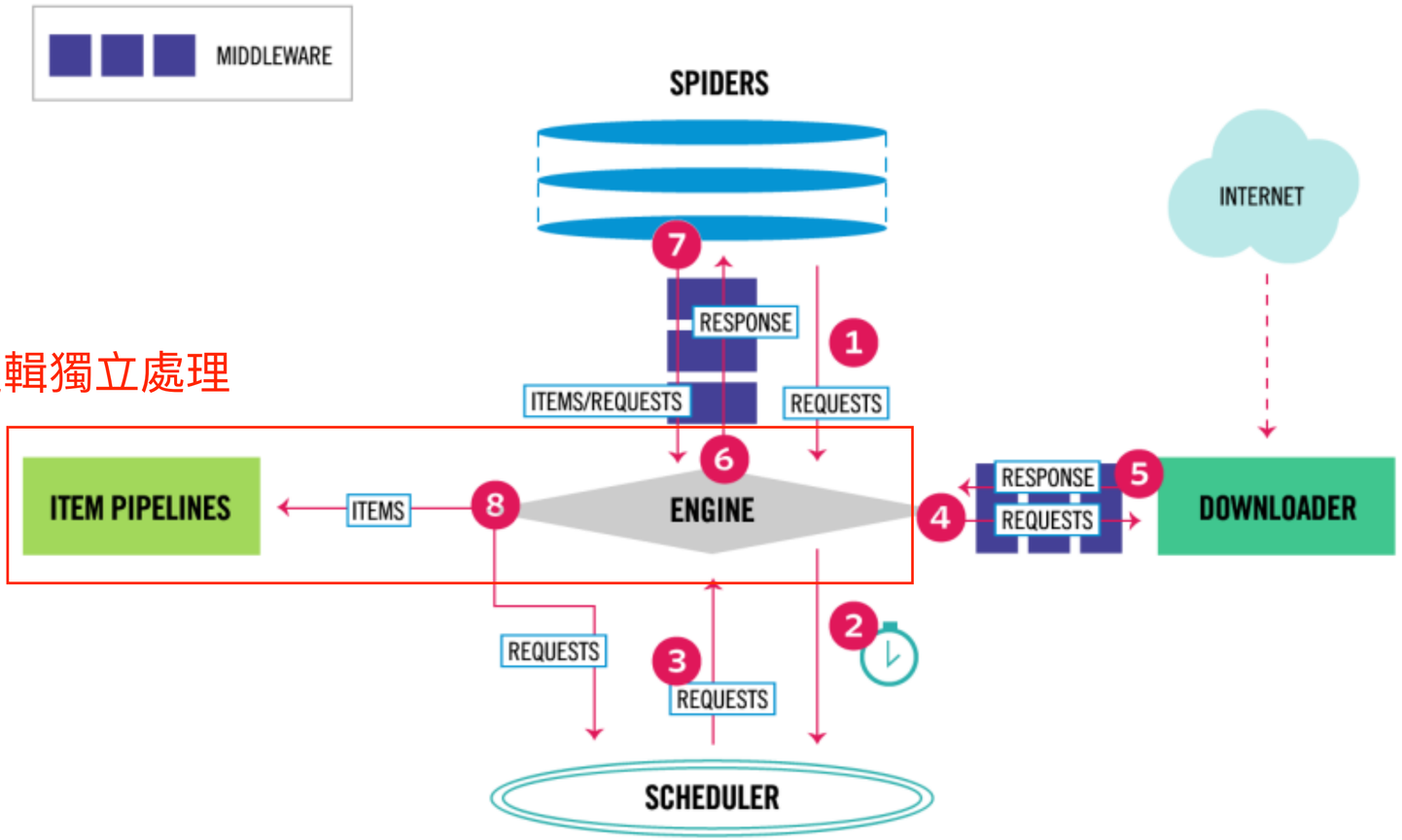
- BeautifulSoup

```
.find() / .find_all()
```

- Scrapy.Selector

```
.get() / .getall()
```

Scrapy 獨立資訊處理的邏輯



框架把爬完的資訊處理邏輯獨立處理

參考來源： *Architecture overview*

定義資料格式

CUPOY

通常我們爬完資料都會以字典型式傳遞與儲存

當爬蟲愈來愈多，資料的格式也愈來愈多種，將會變得難以管理

由於字典格式 (dict) 本身容易新增欄位與覆蓋

容易在沒有注意到的地方改變資料格式與型態

```
data = {'name': '', 'year': 2019} // 定義資料欄位只包含名字與年份
data['month'] = 1 // 容易新增欄位，改變原本定義的資料格式
data['year'] = [2019, 2020] // 容易覆蓋也容易更改資料型態
print(data) // {'name': '', 'year': [2019, 2020], 'month': 1
```

定義資料格式(範例)

CUPOY

Scrapy 透過明確定義資料格式來解決這個問題

```
class Product(scrapy.Item): // 定義 Product 格式
    name = scrapy.Field()
    price = scrapy.Field()
    last_updated = scrapy.Field()
```

scrapy.Field() 定義了每個資料屬性的型態

Product() 定義了資料格式

定義資料格式

CUPOY

當爬完資料要儲存時，需要決定儲存成哪一種格式 (e.g. Product)

Scrapy 框架會幫你檢查儲存格式的正確性

```
product = Product(name='Sam', 'price': 1000)
product['status'] = 'sell' // ERROR:不允許新增未定義欄位
product['last_updated'] = 'today' // 只能改有定義的欄位
```

處理資料流程

前面我們定義了資料格式 (Item)

框架會接著會進入資料處理的流程 (Itme Pipeline)

主要處理資料的目的包含

- 檢查爬到的數據是否正確
- 檢查是否重複，是否需要丟棄資料
- 將爬完的資料存到資料庫或是文檔

處理資料流程範例

CUPOY

Scrapy 框架主要處理資料的幾個時機點

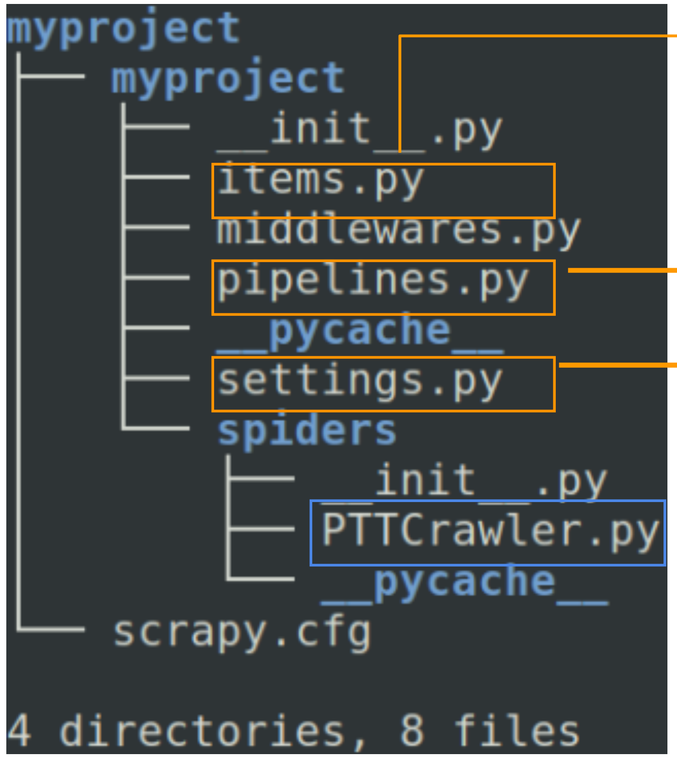
- process_item
  - 每個 Item Pipeline 都需要實作，用來檢查資料數據與是否丟棄等決定
- open_spider
  - 當爬蟲開啟時需要處理的流程 (e.g. 檢查資料庫是否可用)
- close_spider
  - 當爬蟲關閉時需要處理的流程 (e.g. 關閉資料庫連線)

處理資料流程 (範例)

CUPOY

```
class JSONPipeline(object):
    def open_spider(self, spider):
        // 檢查存檔位置
        …
    def close_spider(self, spider):
        // 將爬文的資訊存檔
        …
    def process_item(self, item, spider):
        // 檢查 item 資料的正確性並回傳(或是選擇丟棄)
        …
        return item
```

資料處理的檔案位置



定義資料格式的位置 e.g. ProductItem(scrapy.Item)

定義資料處理流程的位置 e.g. JSONPipeline(object)

根據設定 **ITEM_PIPELINES** 中的設定決定是否資料流程是否執行與執行順序

爬完資料的時候以 item.py 裡定義的資料格式儲存

- Item官方文件
- Item Pipeline 官方文件

重要知識點複習

CUPOY

- 了解資料定位在 Scrapy 框架中對應的寫法為何
  - 資料定位使用 selector，有 xpath 與 css selector 的寫法
- 考慮到爬蟲專案會愈來愈複雜，獨立處理資料邏輯
  - 透過 Item 定義資料格式
  - 透過 Item Pipeline 定義資料處理流程

延伸閱讀

- Srcapy Selector 官方文件
- 進階功能：Item Loader
  - 在 Scrapy 中定義好如何定位元素，以及該元素應該如何存到 Item 格式中，爬蟲過程框架可以幫你自動爬玩送到 pipeline 處理的功能
- 進階功能：Scrapy Feed exports
  - 我們在 pipline 中寫入 JSON 只是熟悉 pipeline 的操作，實際上如果要把資料存成某種格式應該參考 feed exports 的方式

解題時間
LET'S CRACK IT

請跳出 PDF 至官網 Sample Code & 作業

開始解題

START