



網路程式設計

網路爬蟲 靜態網頁解析

Instructor: 馬豪尚

常見的防爬機制

- › 檢查http標頭資訊的user-agent
 - 偽裝成各個瀏覽器送出請求
- › 檢查是否短時間同一個ip位置送出大量請求
 - 搭配使用time.sleep()延遲送出請求

Python fake-useragent套件

› 模組安裝

- `pip install fake_useragent`
- `From fake_useragent import UserAgent`

› 宣告物件

- `ua = UserAgent()`

Python fake-useragent套件

- › 產生各瀏覽器的假useragent
 - ua.ie
 - ua.google
 - ua.firefox
 - ua.safari
- › 將useragent加入header一併送出
 - headers={'user-agent': ua.google}
 - request.get(url, headers=headers)

Selenium

- › Selenium是一個web應用程式的軟體測試框架
- › Selenium可以模擬出使用者在瀏覽器的所有操作行為
- › 常作為「自動化測試」使用的工具，在網站開發完成後，透過自動化的腳本測試所有功能是否正常
- › Selenium還可以擷取動態網頁的內容和HTML表單自動化
 - 能夠輕鬆與JavaScript的事件合作
 - 可以處理網頁的AJAX請求
 - 自動化操作網頁上的元素

Selenium 下載和安裝

- › Python Selenium 套件
 - pip install selenium
 - from selenium import webdriver
- › 瀏覽器驅動程式
 - Chrome:
 - › <https://sites.google.com/chromium.org/driver/downloads>
 - Edge:
 - › <https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/>

Chrome瀏覽器webdriver物件

- › 宣告webdriver物件
 - `webdriver.Chrome("chromedriver", options=optionsobj)`
- › 宣告webdriver的選項參數
 - `optionsobj = webdriver.ChromeOptions()`

加入選項參數到webdriver

- › 不讓瀏覽器執行在前景，而是在背景執行
 - options.add_argument('--headless')
- › 以最高權限來執行
 - options.add_argument('--no-sandbox')
- › 使用 /tmp 而非 /dev/shm 作為暫存區，避免chrome崩潰
 - options.add_argument('--disable-dev-shm-usage')
- › 設定連線用的跳板
 - options.add_argument('--proxy-server')
- › 不使用gpu，避免不必要的bug
 - options.add_argument('--disable-gpu')

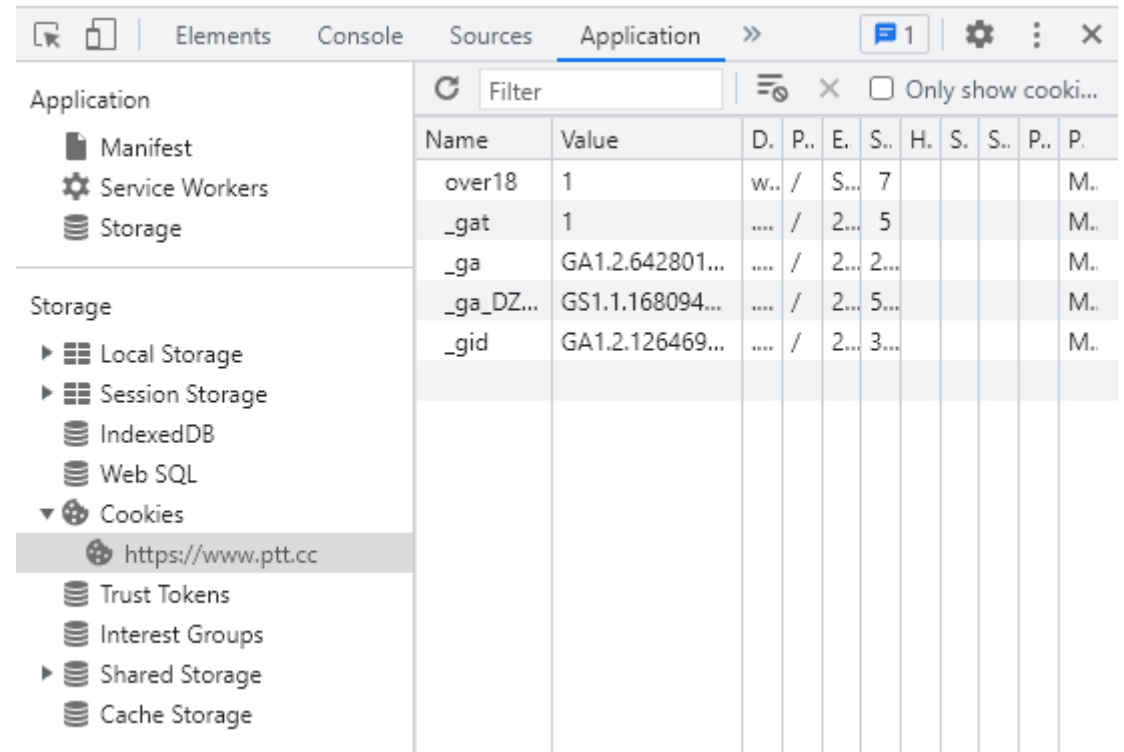
Selenium取得網頁資料

- › 使用webdriver物件中的get函數來取得網頁資料
 - driver.get("url")
- › get取得網頁資料之後
 - 使用webdriver物件中的title屬性取得網頁的標題
 - › driver.title
 - 使用webdriver物件中的page_source屬性取得網頁的html原始碼
 - › driver.page_source

Selenium取得網頁資料-cookie

- › 定義網頁需要的cookie
 - Cookie={key: value}
- › 加入cookies到driver
 - driver.add_cookie(Cookie)

檢查->Application



The screenshot shows the Chrome DevTools Application tab. The left sidebar lists storage areas: Manifest, Service Workers, Storage, Local Storage, Session Storage, IndexedDB, Web SQL, Cookies (selected), Trust Tokens, Interest Groups, Shared Storage, and Cache Storage. The main pane shows a table of cookies for the URL https://www.ptt.cc. The table has columns for Name, Value, and various attributes (D., P., E., S., H., S., S., P., P.).

Name	Value	D.	P.	E.	S.	H.	S.	S.	P.	P.
over18	1	w..	/	S...	7					M..
_gat	1	/	2...	5					M..
_ga	GA1.2.642801...	/	2...	2...					M..
_ga_DZ...	GS1.1.168094...	/	2...	5...					M..
_gid	GA1.2.126469...	/	2...	3...					M..

Selenium資料尋找方法

- › **find_element()**: 用來取得網頁中的第一個定位到的HTML元素
 - driver.find_element()
- › **find_elements()**: (名稱中多了一個s)，用來取得所有網頁中定位到的元素，會以串列方式回傳找到的元素
 - driver.find_elements()
- › find_element和find_elements皆可搭配by模組來尋找資料位置

Selenium資料尋找方法-by

› 載入selenium by模組

– from selenium.webdriver.common.by import By

By.ID, id	透過 id，尋找第一個相符的網頁元素。
By.CLASS_NAME, class	透過 class，尋找第一個相符的網頁元素。
By.CSS_SELECTOR, css selector	透過 css 選擇器，尋找第一個相符的網頁元素。
By.NAME, name	透過 name 屬性，尋找第一個相符的網頁元素。
By.TAG_NAME, tag	透過 HTML tag，尋找第一個相符的網頁元素。
By.LINK_TEXT, text	透過超連結的文字，尋找第一個相符的網頁元素。
By.PARTIAL_LINK_TEXT, text	透過超連結的部分文字，尋找第一個相符的網頁元素。
By.XPATH, xpath	透過 xpath 的方式，尋找第一個相符的網頁元素。

Selenium資料定位方法-by

- › 透過id
 - driver.find_element(By.ID, 'id_name')
- › 透過class
 - driver.find_element(By.CLASS_NAME, 'class_name')
- › 透過css selector
 - driver.find_element(By.CSS_SELECTOR, 'css_selector')
- › 透過xpath
 - 取得 html > body > select 這個網頁元素
 - driver.find_element(By.XPATH, '/html/body/select')

Selenium取得網頁元素的內容

內容	說明
text	元素的內容文字。
get_attribute	元素的某個 HTML 屬性值。
id	元素的 id。
tag_name	元素的 tag 名稱。
size	元素的長寬尺寸。
screenshot	將某個元素截圖並儲存為 png。
is_displayed()	元素是否顯示在網頁上。
is_enabled()	元素是否可用。
is_selected()	元素是否被選取。
parent	元素的父元素。

Selenium取得網頁元素的內容

- › `a = driver.find_element(By.ID, 'a')`
- › 取得元素的內容文字
 - `a.text`
- › 取得元素的id (不是html裡面的id)
 - `a.id`
- › 取得元素的標籤名稱
 - `a.tag_name`
- › 取得元素中的某個屬性值
 - `a.get_attribute('屬性名')`

Selenium操作網頁元素

方法	說明
click()	按下滑鼠左鍵。
click_and_hold()	滑鼠左鍵按著不放。
double_click()	連續按兩下滑鼠左鍵。
context_click()	按下滑鼠右鍵 (需搭配指定元素定位) 。
drag_and_drop()	點擊 source 元素後，移動到 target 元素放開。
drag_and_drop_by_offset()	點擊 source 元素後，移動到指定的座標位置放開。
move_by_offset()	移動滑鼠座標到指定位置。
move_to_element()	移動滑鼠到某個元素上。
move_to_element_with_offset()	移動滑鼠到某個元素的相對座標位置。
release()	放開滑鼠。
send_keys()	送出某個鍵盤按鍵值。
send_keys_to_element()	向某個元素發送鍵盤按鍵值。
key_down()	按著鍵盤某個鍵。
key_up()	放開鍵盤某個鍵。
reset_actions()	清除儲存的動作 (實測沒有作用，查訊後是 Bug) 。
pause()	暫停動作。
perform()	執行儲存的動作。

Selenium操作網頁元素

- › 針對指定元素呼叫方法
 - `button = driver.find_element(By.ID, 'id_name')`
 - `button.click()`
- › 使用「ActionChains」的方式
 - 宣告ActionChains object
 - › `actions = ActionChains(driver)`
 - 將想要的操作串起來
 - › `actions.double_click(add).pause(1).click(add).pause(1).click(add)`
 - 執行ActionChains
 - › `actions.perform()`

爬取網站

- › 分析該網站的HTML
- › 選擇取得資源的方式
 - Request
 - Selenium
- › 撰寫爬蟲和網頁分析語法
 - BeautifulSoup
 - Selenium

練習

- › 分析PTT網站
 - <https://www.ptt.cc/bbs/index.html>
- › 爬取熱門看板中前三名的看板，每個板爬取5頁的文章
- › 將各個看板爬到的文章標題和連結存到CSV檔內

看板名稱	文章標題	連結
Gossiping	xxxxxx	yyyyyy
NBA	XXXXXX	YYYYYY