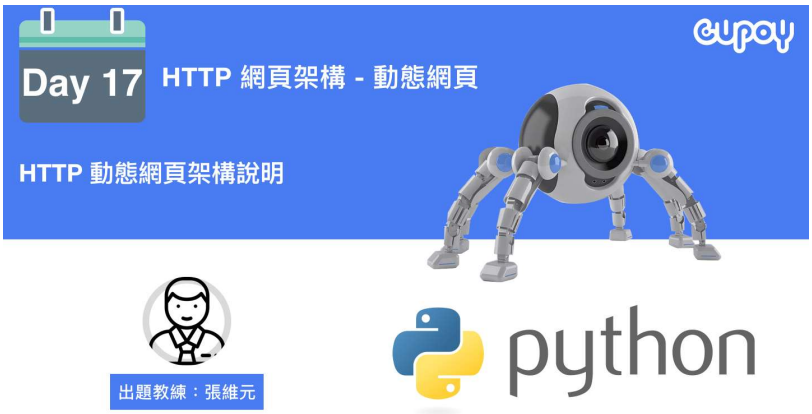
D17：HTTP 動態網頁架構說明與非同步取得資料

簡報閱讀 範例與作業 問題討論

HTTP 動態網頁架構說明



本日知識點目標

- 了解動態網頁的資料爬蟲策略

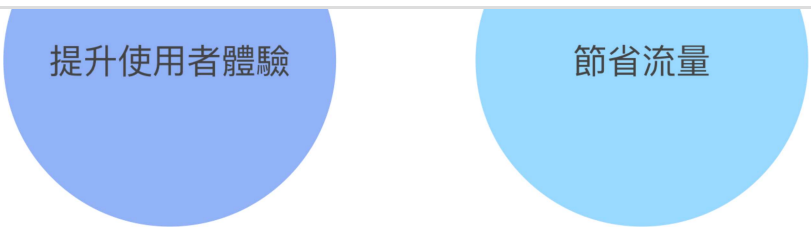
動態網頁架構

動態網頁有別於靜態網頁產生資料的方式。靜態網頁是透過每一次使用者請求，後端會產生一次網頁回傳，所以請求與回傳是一對一的，有些人把他們稱為同步。在動態網頁的話，是透過 Ajax 的技術，來完成非同步的資料傳輸。換句話說，就是在網頁上，任何時間點都可以發送請求給後端，後端只回傳資料，而不是回傳整個網頁。

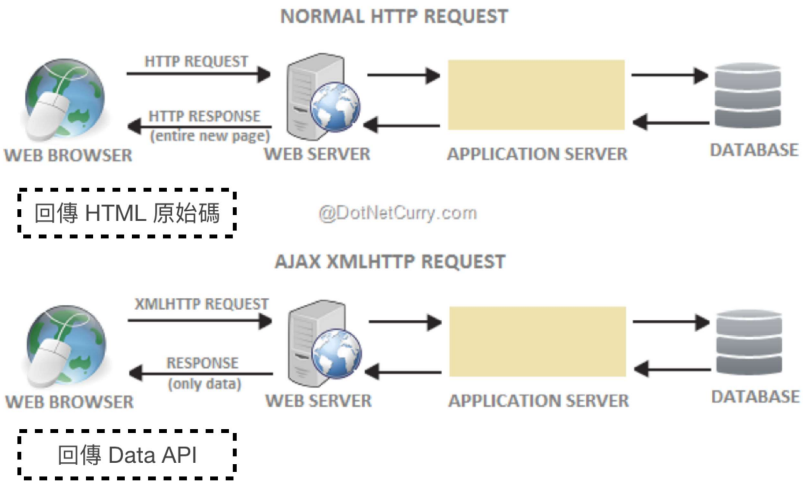
AJAX（Asynchronous JavaScript and XML）

AJAX（Asynchronous JavaScript and XML）是一種在瀏覽器中讓頁面不會整個重載的情況下發送 HTTP 請求的技術。使用 AJAX 來與伺服器溝通的情況下，不會重新載入整個頁面，而只是傳遞最小的必要資料。原生的老舊 AJAX 實現標準為 XHR，設計得十分粗糙不易使用，而 jQuery 其中的 AJAX 功能是前端早期相當普及的 AJAX 封裝，使得 AJAX 使用起來容易許多。

非同步載入的優點



示意圖



動態網頁爬蟲如何進行？

動態網頁與靜態網頁最大的不同是資料是在什麼時間點取得的。動態網頁是在瀏覽器已經取得 HTML 後，才透過 JavaScript 在需要時動態地取得資料。因此，爬蟲程式也必須要考慮動態取得資料這件事情，才有辦法正確地找到想要的資料。

先思考一下動態網頁的運作流程

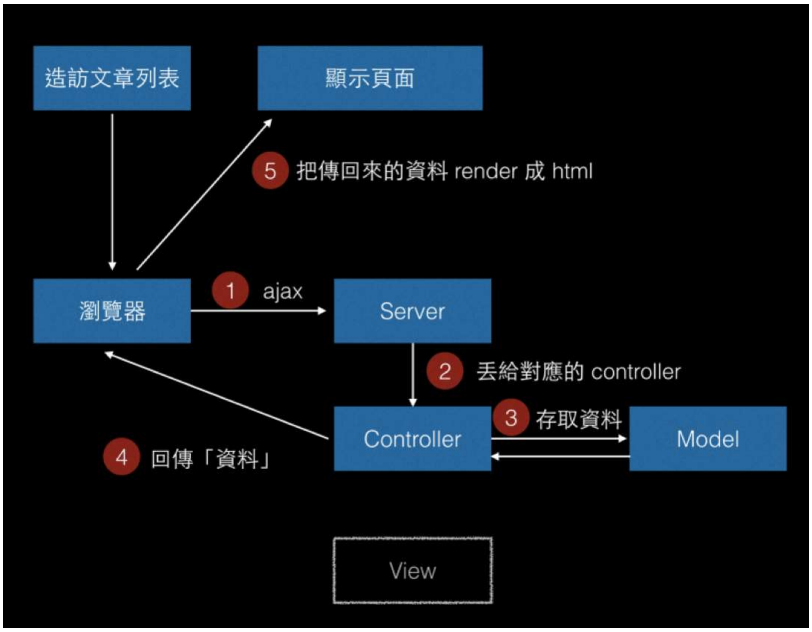
1. 使用者（Client-side）發出請求，稱為是 Request。

3. 產生的回應如果是純資料的話，屬於 API 的一種；如果是網頁的話，就會回傳一個包含 HTML 標籤的網頁格式。
4. 瀏覽器接收包含 HTML 標籤的網頁格式，呈現網頁給使用者。

=> 此時是還沒有資料的！

1. 當瀏覽器解析 HTML 後，開始運行 JavaScript 時會動態的呼叫 API Request 取得資料。
2. 瀏覽器中的 JavaScript 會將資料更新到現有的 HTML 上，呈現網頁給使用者。

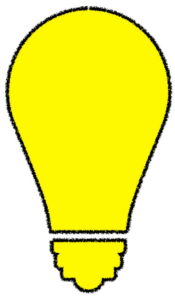
流程圖



動態網頁的爬蟲問題是什麼？

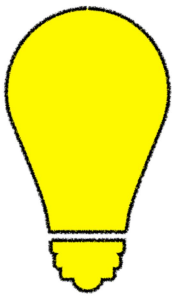
的 HTTP 一來一回的機制，會找不到時機點執行動態的 Request 更新。另外單純靠 Python 程式，也無法執行 JavaScript。

兩種策略



法一

模擬使用者打開瀏覽器



法二

模擬 JavaScript 取得新資料

法一：模擬使用者打開瀏覽器

原本靜態爬蟲的策略是模擬 Request，那我們現在可以模擬更多一點，改為模擬使用者從「發出 Request」到「JavaScript 動態載入資料」的過程。也就是說，這邊的做法是從模擬使用者打開瀏覽器的行為，到模擬器執行JavaScript 動態載入 之後。

法二：模擬 JavaScript 取得新資料

另外一種方法是我們知道 Python 無法直接執行 JavaScript 的。但本質上 JavaScript 也是透過呼叫 API 的方式拉資料，因此我們只要模仿 JavaScript 呼叫 API 這個動作，改由 Python 執行即可。

重要知識點複習

- 了解動態網頁的資料爬蟲策略
- 知道非同步網頁載入機制（Ajax）
- 學習兩種對應動態網頁爬蟲的的策略

解題時間

下一步：閱讀範例與完成作業