網路程式設計
網路爬蟲 靜態網頁解析

Instructor: 馬豪尚

Requests向伺服器端GET請求

- 定義請求位置
  - url
- Requests用get方法來請求
  - request.get(url)
- 伺服器端回應屬性
  - text: 編碼的HTML標籤字串
  - contents: 沒有編碼的位元組資料，適用於非文字內容的請求
  - encoding: 取得HTML標籤字串的編碼
  - status_code: 伺服器回應狀態碼

Requests向伺服器端GET請求

- 帶有參數的請求
  - 直接將參數加在url網址之後
  - 使用params參數指定url參數值的字典
    - dic_params = {'name': 'Allen', 'grade': 100}
    - requests.get("url/get", params = dic_params)

Requests - User-agent and Cookie

- 有些網站的HTTP請求需要指定header參數或Cookie資料
- 輸入user-agent在header內
  - header= {'user-agent' : 'Allen'}
  - requests.get(url, headers=headers)
- 輸入Cookie資料
  - Cookies= dict('name'='Allen')
  - requests.get(url, cookies=cookies)

Requests向伺服器端POST請求

- 定義請求位置
  - url
- Requests用POST方法來請求，同時送出要傳送給伺服器的資料，例如表單欄位的輸入
  - post_data={'name':'Allen', 'grade': 100}
  - request.post(url, post_data)

HTML基本架構

- 文件宣告DOCTYPE
- Html
  - 標示網頁的開始與結束，是一個網頁的根元素
- Head
  - 用來標示網頁標頭
  - 網頁編碼方式、標題、關鍵字、連結等
- Body
  - 網頁的主體

HTML基本範例

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>我的網頁</title>
</head>
<body>
        <h1>Hello, HTML5!</h1>
</body>
</html>
```

標籤(tag)與屬性(attribute)

- 標籤(tag)
  - 標示網頁上的內容或描述內容的性質
  - <head>、<body> 、<header> 、<p> 、 、<a> 、<table> 、<form> 、 <img>、 <video>等
- 屬性(attribute)
  - 超連結
    - Google首頁

屬性名稱　　屬性值　　內容

- 元素: 包含開始標籤、內容以及結束標籤

全域屬性 (global attributes)

- 所有的 HTML 元素都有的屬性，我們稱做全域屬性 (global attributes)，可以在所有的元素中使用
- **id元素唯一識別符號**:用來設定 HTML 元素的唯一識別符號 (identifier)，每個 HTML 元素的 id 需要是在整份 HTML 文件中獨一無二 (unique) 不可重複的，且一個元素只能有一個id。
  - 用作 <a> 連結的錨點名稱。例如點擊連結 <a href="#myid"> 會跳到 <tag id="myid"> 元素處
  - 用在 JavaScript 可以透過 id 存取該元素
  - 用在 CSS 可以用 id 當選擇器

Example

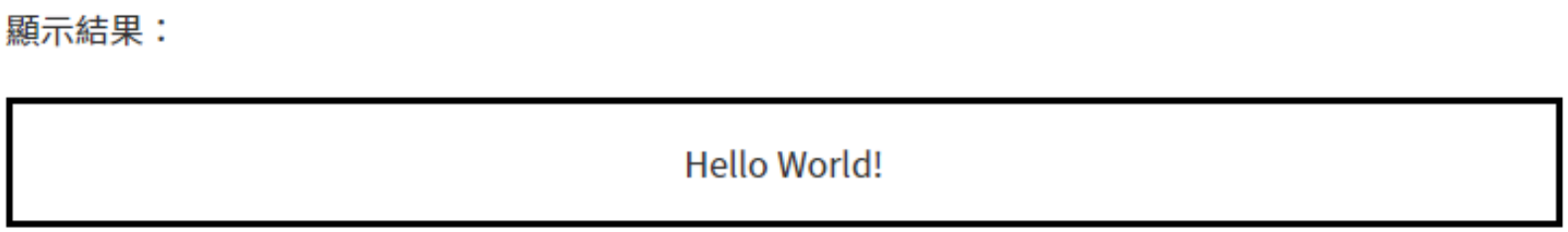
- <p id=“beauty">The most beautiful paragraph on this web. </p>

全域屬性 (global attributes)

- **class 元素類別名稱:**用來設定 HTML 元素的類別名稱 (class names)，每一個 HTML 元素可以有多個類別，你可以用空格分隔 (space-separated) 開不同的類別名稱。
  - 用在 JavaScript 可以透過 class 存取該元素
  - 用在 CSS 可以用 class 當選擇器 (selector)
- Example
  - Above point sounds a bit obvious. Remove/rewrite?

全域屬性 (global attributes)

- **style 樣式:** 用來直接設定該 HTML 元素的 CSS 樣式 (inline style)，而用 style 屬性設定的 CSS 優先權是最高的，會蓋過寫在 <style> 或外部樣式表中的樣式。
- Example
  - Hello World!



靜態網頁分析

- 用requests取回HTML網頁內容
- 搭配BeautifulSoup套件
  - 解析及取得HTML原始碼各個標籤的元素資料
  - pip install bs4
- 載入套件模組
  - from bs4 import BeautifulSoup

BeautifulSoup解析器

- Beautiful Soup支持Python標準庫中的HTML解析器，還支持一些第三方的解析器

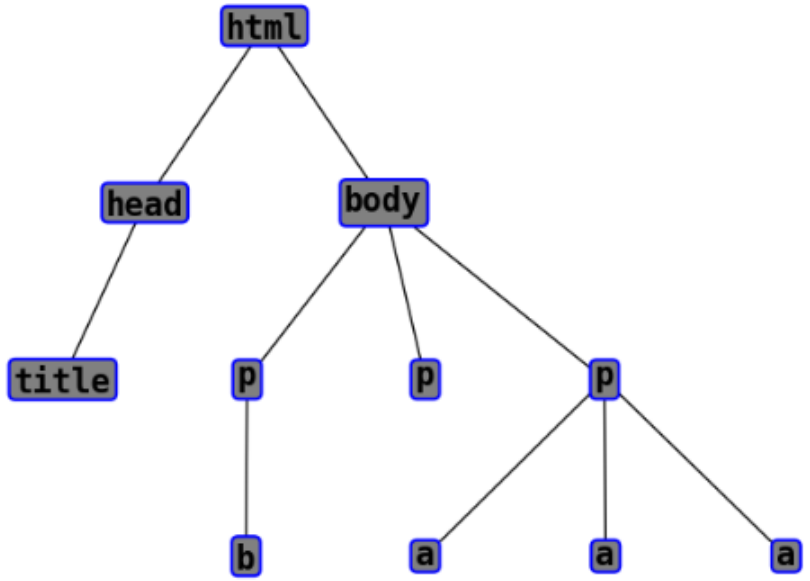
解析器	使用方法	優勢	劣勢
Python標準庫	BeautifulSoup(markup, "html.parser")	執行速度適中 文檔容錯能力強	Python 2.7.3 or 3.2.2前的版本中文檔容錯能力差
lxml HTML 解析器	BeautifulSoup(markup, "lxml")	速度快 文檔容錯能力強	需要安裝C語言庫
lxml XML 解析器	BeautifulSoup(markup, ["lxml-xml"]) BeautifulSoup(markup, "xml")	速度快 唯一支援XML的解析器	需要安裝C語言庫
html5lib解析器	BeautifulSoup(markup, "html5lib")	最好的容錯性 以瀏覽器的方式解析文檔 產生HTML5格式的文檔	速度慢 不依賴外部擴展

BeautifulSoup

- 安裝解析器
  - lxml HTML 解析器
    - pip install lxml
  - html5lib解析器
    - pip install html5lib

BeautifulSoup 物件的種類

- Beautiful Soup將HTML文檔轉換成一個樹形結構，每個節點都是Python物件
  - BeautifulSoup
    - soup = BeautifulSoup(content, "lxml")
    - 第一個參數為含有HTML標籤的文字內容
    - 第二個參數為解析器的名稱
  - Tag
    - 與XML或HTML原始文檔中的tag相同
  - NavigableString
    - 可以遍歷的字符串
  - Comment
    - 註釋及特殊字符串



BeautifulSoup 物件

- Beautiful Soup提供了許多操作和遍歷tag子節點
- 存取某個tag的方法
  - soup.tag名稱
    - tag = soup.a → 獲取解析文檔內的第一個a標籤
    - tag = soup.body.p → 獲取解析文檔內的body標籤底下的第一個p標籤
  - 使用find()
    - soup.find('tag名稱')
    - 只能存取第一個名為'tag名稱'的節點
- 要存取/查詢所有指定名稱的子節點(tag)
  - soup("tag名稱")
    - 返回一個列表為符合tag名稱的所有tag和內容

BeautifulSoup 物件

- find_all(name, attrs, string, recursive, **kwargs)
  - name 參數可以查找所有名字為 name 的tag
    - soup.find_all("title")
  - attrs 參數搜尋時可以把該參數當作指定名字tag的”屬性值”來搜尋
    - 可以使用的屬性值支援字符串、正規表達式、列表、True
    - soup.find_all(id='link2')
    - soup.find_all(href=re.compile("elsie"), id='link1')
    - 有些tag屬性在搜索不能使用,比如HTML5中的 data-* 屬性
  - string 參數可以搜尋文檔中的符合字符串的內容
    - soup.find_all(string="Elsie")
    - string 參數能夠支援字符串、正規表達式、列表、True

BeautifulSoup 物件

- find_all(name, attrs, string, recursive, **kwargs)
  - recursive=True/False
    - 只想搜索tag的直接子節點，使用參數 recursive=False
    - 預設搜尋tag的所有子孫節點，recursive=True
  - limit=數字
    - 搜尋到的結果數量達到 limit 的限制時，就停止搜尋返回結果
- find_all可以使用css的類別方式來查詢
  - soup.find_all("a", class_="sister")
  - 支援字符串、正規表達式、方法或 True

BeautifulSoup 物件

- 使用參數CSS選擇器的搜尋方法
  - select(“選擇器”)
    - soup.select(“選擇器”) → 搜尋所有符合該選擇器的節點(tag)
    - soup.select(“#id”) → 若使用id為選擇器則選擇器名稱為#id
    - tag.select(“選擇器”) → 搜尋tag內符合該選擇器的子節點
    - 以上都會返回一個列表，包含所有符合的節點內容
  - select_one(“選擇器”)
    - soup.select_one(“選擇器”) → 搜尋第一個符合該選擇器的節點(tag)
    - soup.select_one(“#id”) → 若使用id為選擇器則選擇器名稱為#id
    - tag.select_one(“選擇器”) → 搜尋tag內符合該選擇器的第一個子節點

BeautifulSoup 物件搜尋方法小結

搜尋方法	說明
select_one()	使用參數CSS選擇器字串搜尋HTML標籤，返回第一個符合的HTML標籤物件
select()	使用參數CSS選擇器字串搜尋HTML標籤，返回所有符合的HTML標籤物件的串列
find()	使用參數的標籤名稱或屬性值來搜尋HTML標籤，返回第一個符合的標籤物件
find_all()	使用參數的標籤名稱或屬性值來搜尋HTML標籤，返回所有符合的HTML標籤物件的串列
soup.tag名稱	返回一個符合該名稱的標籤物件
soup(“tag名稱”)	返回一個串列包含所有符合該名稱的標籤物件

BeautifulSoup Tag 物件

- Tag與XML或HTML原始文檔中的tag相同，
- tag.name
  - 每個tag都有自己的名稱，透過這個name屬性來獲取
  - 可以改變tag的名稱但會影響整個透過當前Beautiful Soup物件解析的HTML文檔樹
    - tag.name = "新的名稱"
- tag.text
  - 取得當前tag的內容

BeautifulSoup Tag 物件

- tag.attrs : 一個tag可能有很多個屬性，tag的屬性操作方法與字典相同
  - 一個屬性的情況
    - <b class="boldest"> 有一個 "class" 的屬性，其值為 "boldest"
    - tag[屬性名稱] → ex. tag['class']
  - 多個屬性的情況
    - 有一個 "class" 的屬性，其值為 "box"，和另一個"id"屬性，值為"one"
    - tag.attrs → 返回所有屬性名稱和值的字典

BeautifulSoup Tag 物件

- tag.get(“屬性名稱”, None)
  - 第一個參數為指定屬性，意思是取得當前tag內指定屬性的值
  - 第二個參數為，若沒有符合的屬性而返回的值

BeautifulSoup Tag 物件

- 文檔樹中tag的屬性可以被增加，刪除或修改，操作方式採用python字典的操作方式
- 存取屬性
  - tag[‘屬性名稱’]
- 增加或修改屬性
  - tag[‘屬性名稱’] = 屬性值 → ex. tag['id'] = 1
- 刪除屬性
  - del tag[‘屬性名稱’] → ex. del tag['class']

BeautifulSoup Tag 物件

- HTML多值屬性，一個屬性內含有多個值(用空白隔開)
  - 最常見的多值的屬性是 class
  - 有一個 "class" 的屬性，其值為 "one" 和"box"兩個
  - 操作和單一屬性值的時候一樣，但是會用列表返回
    - tag[屬性名稱] → ex. tag['class']
    - 返回 ["one ", " box"]的列表
  - 如果該屬性為不支援包含多值的屬性，返回會和單一屬性值一樣為字串

BeautifulSoup Tag 物件子節點

- tag.contents
  - tag的 .contents 屬性可以將tag的子節點以列表的方式返回
- tag.children
  - tag的 .children 生成器，可以對tag的子節點進行循環
  - For child in tag.children:
- tag.descendants
  - .contents 和 .children 屬性僅包含tag的直接子節點
  - .descendants 屬性會返回一個可迭代的物件
  - 得到該節點的所有子孫節點

BeautifulSoup Tag 物件子節點

- tag.string
  - tag僅有一個子節點可以用tag.string來存取
  - tag只有一個 NavigableString 類型子節點，也可以用tag.string來存取
  - tag包含了多個子節點，tag就無法確定 .string 方法應該存取哪個子節點的內容，就會返回None

BeautifulSoup Tag 物件子節點

- tag.strings
  - tag中包含多個字符串，可以使用 .strings 來循環獲取
    - for string in soup.strings:
- tag.stripped_strings
  - 輸出的字符串中可能包含了很多空格或空行，使用.stripped_strings這個方法來去除
    - for string in soup.stripped_strings:

BeautifulSoup Tag 物件父節點

- tag.parent
  - 獲取某個tag的父節點
  - 最頂層節點的父節點就是BeautifulSoup物件
  - BeautifulSoup 物件的父節點會返回None
- tag.parents
  - .parents 屬性可以返回一個可迭代的物件
  - 得到該節點的所有父輩節點

BeautifulSoup Tag 物件兄弟節點

- tag.next_sibling
  - 查詢指定一個tag的下一個兄弟節點
    - tag = soup.b
    - tag.next_sibling
- tag.previous_sibling
  - 查詢指定一個tag的上一個兄弟節點
    - tag = soup.b
    - tag.previous_sibling
- tag.next_siblings 和 tag.previous_siblings
  - 透過 .next_siblings 和 .previous_siblings 屬性可以對當前節點的兄弟節點返回一個可迭代的物件

BeautifulSoup NavigableString

- NavigableString
  - 被定義為可以迭代遍歷的字符串，字符串常被包含在tag內
  - NavigableString沒有子節點的方法可以使用，因為沒有子節點
  - tag.string → 存取這個標籤內的字符串

BeautifulSoup修改文檔樹

- 修改tag的名稱和屬性
  - tag.name = “新名稱"
  - tag[‘屬性’] = ‘新屬性值’
- 刪除屬性
  - del tag[‘屬性']
- 修改.string
  - tag.string = “新的String值”
- 創建一個新tag
  - soup.new_tag(“tag name”, 屬性=“屬性值”)
- 創建一段新文字內容
  - soup.new_string(“文字內容")

BeautifulSoup修改文檔樹

- 增加tag內的文字內容
  - tag.append(“增加的內容”)
- 插入tag或文字內容
  - insert() : 插入到指定位置
    - tag.insert(指定位置, 文字內容)
  - insert_before() : 插入到當前tag之前
    - 當前tag.insert_before(soup.new_tag(“tag name”, 屬性=“屬性值”))
  - insert_after() : 插入到當前tag之後
    - 當前tag.insert_after(soup.new_string(文字內容))

練習

- 用requests爬取bbc news金融財經新聞網站
  - https://www.bbc.com/zhongwen/trad/topics/cq8nqywy37yt
- 用BeautifulSoup套件解析網站內容
  - 將新聞清單解析出來包含以下欄位的內容
    - title、headline 、url網址、時間
  - 存到csv成為一個表格