C#: 使用 AngleSharp 爬蟲工具來抓取網頁內容吧 - 伊果的沒人看筆記本

igouist.github.io (https://igouist.github.io/post/2022/06/angle-sharp/)

前一次用到 AngleSharp 已經是去年抓網路小說的時候，想不到最近又用上了，乾脆就來筆記一下。

AngleSharp (https://anglesharp.github.io/) 是一款簡單方便的 C# 爬蟲套件，撈網頁時支援 QuerySelector (https://developer.mozilla.org/zh-TW/docs/Web/API/Document/querySelector) 的語法來篩選網頁元素，並且撈回來的資料集合也都能用 Linq 操作，讓我們能對爬取的網頁內容快速進行篩選和處理，只需要短短的語法就可以開心抓想要的內容。

說到要示範爬蟲，果然還是要用爬蟲界默認的經典範例 PTT 表特版 (https://www.ptt.cc/bbs/Beauty/index.html) 來操作（？），接著就讓我們來寫一個簡單的腳本來抓取文章吧！

安裝套件

首先要先從 Nuget (https://www.nuget.org/packages/AngleSharp) 安裝 AngleSharp

確認安裝完畢後就可以開始撰寫囉 ~

> 本篇接下來會使用 Linqpad 來進行簡單的範例，使用 VisualStudio 的朋友遇到 `Dump()` 之類的語法就請再自己調整一下呦

建立 Browser 抓取網頁內容

首先我們需要先建立一個 Browser 來代替我們做事：

```
void Main()
{

        var config = AngleSharp.Configuration.Default.WithDefaultLoade
r();


        var browser = BrowsingContext.New(config);
}
```

建立之後就可以**用這個 browser 的 `OpenAsync()` 來抓取網頁內容囉**：

```
async Task Main()
{
        var config = AngleSharp.Configuration.Default.WithDefaultLoader();
        var browser = BrowsingContext.New(config);


        var url = new Url("https://www.ptt.cc/bbs/Beauty/index.html");


        var document = await browser.OpenAsync(url);
        document.Dump();
}
```

可以看到抓了一堆東西回來，包含網頁的 Uri、Body 等等：

調整配置、準備 Cookie

在前一個步驟我們雖然把網頁抓回來了，但讓我們看一下 `context.Body.InnerHtml` 的內容：

看起來跟我們要的文章列表有點差距啊！這是因為 PTT 會先跳出一個視窗詢問是否滿十八歲，選擇「是」之後會記錄一筆 `over18=1` 到 Cookie 中，只有持有這個 Cookie 才能進入到文章列表。

因此我們現在要先把答案準備到 Cookie 裡，這時候我們就需要調整一下前面的配置內容。

首先先在配置裡加上預設的 Cookies：

```
var config = AngleSharp.Configuration.Default
    .WithDefaultLoader()
    .WithDefaultCookies();

var browser = BrowsingContext.New(config);
```

接著在我們開啟網頁之前，**使用 `SetCookie` 對目標指定要用的 Cookie**：

```
var url = new Url("https://www.ptt.cc/bbs/Beauty/index.html");


browser.SetCookie(url, "over18=1'");

var document = await browser.OpenAsync(url);
```

現在會長得像這樣：

```
async Task Main()
{
        var config = AngleSharp.Configuration.Default
                .WithDefaultLoader()
                .WithDefaultCookies();

        var browser = BrowsingContext.New(config);

        var url = new Url("https://www.ptt.cc/bbs/Beauty/index.html");
        browser.SetCookie(url, "over18=1'");

        var document = await browser.OpenAsync(url);
        document.Body.InnerHtml.Dump();
}
```

接著讓我們來確認爬回來的 HTML：

可以看到我們成功進到看板囉！

> 註：另一個很常在配置處理的是使用 LoaderOptions 加上 AngleSharp.Css 提供的 `WithCss()` 來抓取 CSS 處理後的結果，例如：
>
> ```
> Configuration.Default
>     .WithDefaultLoader(new LoaderOptions
>     {
>         IsResourceLoadingEnabled = true
>     })
>     .WithCss();
> ```
>
> 不過這些場景（需要掛 Cookie 啦、要先等 CSS 啦）通常都是遇到之後才去 Google 的，這邊就不再贅述。

> 註：`browser.OpenAsync` 除了提供網址讓他直接抓回來以外，有時候我們也會遇到本機已經有 Html 檔案要處理，或是已經用 HttpClient 等方法把網頁內容拿回來了的狀況
>
> 這種時候也可以用 `OpenAsync` 提供的委派方法來把 HTML 字串讀取成 AngleSharp 的物件。例如：`OpenAsync(res => res.Content(html))`，同時委派中的 `VirtualResponse` 也提供了對這物件設定 Cookie 等資訊的方法，有這個需求的朋友可以再動手試試。

使用篩選器來抓指定的內容

現在我們已經成功把網頁內容抓回來了，接著就是要**使用選擇器(https://www.runoob.com/cssref/css-selectors.html)來抓出我們想要的內容囉**。

如果有用過 JQuery 或是整天寫 CSS 的朋友應該不會陌生，大致上是這樣的：

- `div`：就是抓 `<div>`
- `#wow`：抓 id 是 wow 的元素
- `.hello`：抓 class 是 hello 的元素

當然也可以加以組合，例如 `div#wow`, `div > p.hello` 等，有興趣的朋友可以再看一眼 菜鳥教程的 CSS 選擇器說明 (https://www.runoob.com/cssref/css-selectors.html)，其他狀況就等需要的時候再查表即可。

首先讓我們好好觀察 HTML 結構，可以發現標題框是放在 `class="r-ent"` 的 div 裡，那我們就可以這樣下 Selector： `div.r-ent`

接著我們就能用 `QuerySelectorAll()` 這個方法來用 Selector 抓取我們想要的元素：

```
document
    .QuerySelectorAll("div.r-ent")
    .Select(node => node.InnerHtml)
    .Dump();
```

可以看到我們成功抓了一排標題資訊回來囉！

當然 AngleSharp 也提供了 `QuerySelector` 來抓取單個元素，這兩個方法用起來和 JavaScript (https://developer.mozilla.org/zh-TW/docs/Web/API/Document/querySelector) 的體驗應該是差不多啦。

確定我們有把要的內容抓回來，也不會再用到網頁內容的話，就可以補一行 `document.Close();` 把開啟的網頁內容順手清掉囉。

> 註：現在都２０２２年了，瀏覽器當然也有提供直接抓 Selector 的功能了
>
> 這邊以 Edge 為例，讓我們到目標網頁按下Ｆ１２打開開發人員工具，直接選取目標：
>
> 
>
> 選取目標後就會告訴我們這是哪個元素，我們只需要對該元素右鍵，然後複製它的 Selector 就行啦
>
> 不過這樣抓到的 Selector 語法通常會比較囉嗦一點，例如這個頁面就是 `#main-container > div.r-list-container.action-bar-margin.bbs-screen` ，這部份就再自己調整一下囉

整理抓取到的內容

現在讓我們建立一個類別用來處理標題資訊吧，這邊我只需要名稱、推噓數和文章連結，其他像是作者什麼的不太需要：

```
public class Post
{
        public string Title { get; set; }
        public int Push { get; set; }
        public string Link { get; set; }
}
```

接著就來把我們剛剛抓到的每一則文章標題轉換成我們要的物件吧！

觀察上面抓到的標題資訊，可以發現標題的文字和文章連結都放在 `<div class="title">` 裡的 `<a>`，這邊我們就可以利用 QuerySelector 來抓到這個元素

而為了拿到連結，這邊會需要**使用 `GetAttribute` 來抓取元素的 `href` 屬性**。這樣標題和連結就搞定了。

另外要注意的是：如果文章被刪掉了，可是抓不到這些東西的！所以可以在 `QuerySelector` 之後用 `?.` 的方式來做個 Null 時的防呆，最後也可以再用 `Where` 來過濾掉無效的文章。

```
var titleElement = post.QuerySelector("div.title > a");
var title = titleElement?.InnerHtml;
var link = titleElement?.GetAttribute("href");
```

剩下的推噓數可以看到是放在 `<div class="nrec">` 裡面，這邊我希望能轉換成數字，方便我們後續如果要用推噓數做篩選。所以讓我們額外處理一下：

- 只顯示「爆」，這時候我們就視作 100
- 如果有明確的數字，轉換為 Int

- 沒有數字的話就當成 0。

```
var pushString = post.QuerySelector("div.nrec > span")?.InnerHtml;

var pushCount =
    pushString == "爆" ? 100 :
    Int16.TryParse(pushString, out var push) ? push : 0;
```

最後就把這些資訊拿來組裝我們的物件，那麼現在的程式碼就會像這樣：

```
var posts = postSource.Select(post =>
{
    var titleElement = post.QuerySelector("div.title > a");
    var title = titleElement?.InnerHtml;
    var link = titleElement?.GetAttribute("href");

    var pushString = post.QuerySelector("div.nrec > span")?.InnerHtml;
    var pushCount =
        pushString == "爆" ? 100 :
        Int16.TryParse(pushString, out var push) ? push : 0;

    return new Post
    {
        Title = title,
        Link = link,
        Push = pushCount
    };
})
.Where(post => post.Title != null)
.Dump();
```

到這邊我們就成功把網頁的內容抓下來、篩選出我們要的內容囉！

> 註：除了我們前面使用的 QuerySelector、GetAttribute 以外，AngleSharp 還提供了 `GetElementsByTagName` 、 `Children` 等屬性和方法讓我們能方便地在 DOM 中到處抓取元素。
>
> 有興趣的朋友再自己摸索一下吧，我自己是比較習慣無腦 `QuerySelector` 了啦哈哈。

搭配遞迴來多撈幾頁

其實我們前面已經把 AngleSharp 的基本操作跑完一輪了，基本上不外乎是「**打開目標網頁 → 找到目標元素 → 用篩選器抓出來**」這樣的 Loop，這節只是單純讓這個腳本完善一點而已。

因此不感興趣的朋友也可以直接跳過這一段，直接前往，準備出發去動手抓自己想要的網頁囉。

現在讓我們回到文章列表來，只抓第一頁實在沒什麼搞頭。如果我們想要換頁，那麼首先就要先抓出這個換頁按鈕：

利用前面提到的 F12 大法，我們可以拿到這個按鈕的 Selector 語法，讓我們直接丟到 `QuerySelector` 裡，並且取得它的 `href` 屬性：

```
var nextPageLink = document
        .QuerySelector("div.btn-group.btn-group-paging > a:nth-child(2)"
)
        .GetAttribute("href")
        .Dump();
```

現在我們有了往前一頁的連結了，後續只需要把站台的 Url 和下一頁的相對 Url 就可以拼湊出換頁的連結了。

如此一來就可以做出「抓文章 → 下一頁 → 抓文章...」的循環。像這種場合直接使用遞迴，寫起來會快一點。

首先讓我們把抓取文章的處理過程抽出去當作方法，當然會重複用到我們的 Browser，還有站台 Url 以及每一頁的 Url，最後再丟個數字控制要抓幾頁，大概像這樣：

```
private async Task<IEnumerable<Post>> GetPosts(
        IBrowsingContext browser,
        string baseUrl,
        string pageUrl,
        int remainingPages)
{
}
```

現在讓我們把上面的處理步驟逐一搬移到方法中，預期會需要：

- 先組裝 Url、設定 Cookie 然後 Open 抓回網頁內容
- 抓取這一頁的所有文章標題
- 取得下一頁的連結
- 遞迴取得下一頁及往後頁數的文章列表

- 把這一頁和下一頁往後的文章列表組裝起來回傳

```
private async Task<IEnumerable<Post>> GetPosts(
        IBrowsingContext browser,
        string baseUrl,
        string pageUrl,
        int remainingPages)
{

        var url = new Url(baseUrl + pageUrl);
        browser.SetCookie(url, "over18=1'");
        var document = await browser.OpenAsync(url);


        var postSource = document.QuerySelectorAll("div.r-ent");

        var posts = postSource.Select(post =>
        {
                var titleElement = post.QuerySelector("div.title > a");
                var title = titleElement?.InnerHtml;
                var link = titleElement?.GetAttribute("href");

                var pushString = post.QuerySelector("div.nrec > span")?.InnerHtml;
                var pushCount =
                        pushString == "爆" ? 100 :
                        Int16.TryParse(pushString, out var push) ? push : 0;

                return new Post
                {
                        Title = title,
                        Link = link,
                        Push = pushCount
                };
        })
        .Where(post => post.Title != null);


        var nextPageUrl = document
                .QuerySelector("div.btn-group.btn-group-paging > a:nth-child(2)")
                .GetAttribute("href");

        document.Close();


        remainingPages--;
        if (remainingPages == 0)
        {
                return posts;
        }


        var nextPagePosts = await GetPosts(browser, baseUrl, nextPageUrl, remainingPa
```

```
ges);
        return posts.Concat(nextPagePosts);
}
```

雖然也可以把詳細的步驟再拆得更細，像是把組裝 `Post` 的部分拆出去私有方法，或是多加一層迴圈進到文章內抓取照片等等，不過現在我們只需要穩定拿到文章資訊就行

接著再稍微修改一下，外面呼叫方法的部份只需要負責建立 Broswer 和定下第一頁的 Url 就好了

```
async Task Main()
{
        var config = AngleSharp.Configuration.Default
                .WithDefaultLoader()
                .WithDefaultCookies();

        var browser = BrowsingContext.New(config);

        var baseUrl = "https://www.ptt.cc";
        var indexUrl = "/bbs/Beauty/index.html";

        var pages = 10;
        var posts = await GetPosts(browser, baseUrl, indexUrl, pages);
}
```

這樣就大功告成，一次抓它個十頁都沒有問題囉！

最後就可以按照我們的要求來處理 `posts` 的文章啦，例如：

```
posts.Where(post => post.Push > 90).Dump();
```

馬上就可以抓出十頁內 90 推以上的文，所以說 Linq 就是方便哪。

到這邊我們已經可以很靈活地去運用這個列表了，像是把撈出來的文章連結搭配 LineNotify (https://igouist.github.io/post/2020/04/bandon-3-line-notify/) 做個推播通知啦，還是乾脆掛到排程服務去定時爬資料啦，都是很彈性很自由的了。

小結

最後再一次整理本篇的操作流程：

- 從 `AngleSharp.Configuration.Default` 建立組態
- 使用 `BrowsingContext.New(config)` 來建立 Browser
- 根據需求調整組態和 Browser，例如 `SetCookie`
- 使用 `browser.OpenAsync(url)` 把網頁內容抓回來
- 使用 篩選器 搭配 `QuerySelector` 等方法，從網頁內容中抓出我們要的資訊
- 網頁爬完之後可以順手 `Close()`
- 自由發揮

步驟其實非常簡單，各位朋友可以出發去動手抓自己想要的網頁囉！

例如說抓一下股票資訊 (https://gist.github.com/Igouist/039148b1aaaa8e3073f5e135ff689f9b)啦、稽查朋友在 PTT 的留言 (https://gist.github.com/Igouist/ebfc29be9e350bb7c289f05df694535b)啦，都蠻有趣（？）的呢

總之，當你需要在 C# 能簡單使用的小爬蟲，就是 AngleSharp 出場的時候啦！

參考資料

- 使用類似 javascript selector 來爬網站的工具 - AngleSharp | kinanson的技術回憶 (https://dotblogs.com.tw/kinanson/2017/08/30/085049)
- 用 .NET Core 做網頁爬蟲抓取資料 - 使用 HttpClicent 與 AngleSharp - 長庚的作業簿 (https://dannyliu.me/%E7%94%A8-net-core%E5%81%9A%E7%B6%B2%E9%A0%81%E7%88%AC%E8%9F%B2%E6%8A%93%E5%8F%96%E8%B3%87%E6%96%99-%E4%BD%BF%E7%94%A8httpclicent%E8%88%87anglesharp/)
- AngleSharp - Documentation (https://anglesharp.github.io/docs/01-articles)
- CSS 選擇器 | 菜鳥教程 (runoob.com) (https://www.runoob.com/cssref/css-selectors.html)
- c# - Parsing CSS with AngleSharp - Stack Overflow (https://stackoverflow.com/questions/59219106/parsing-css-with-anglesharp)

igouist.github.io (https://igouist.github.io/post/2022/06/angle-sharp/)