<http://fecbob.pixnet.net/blog/post/38087701-c%23-%E6%8A%93%E5%8F%96%E7%B6%B2%E9%A0%81%E6%95%B8%E6%93%9A%E3%80%81%E5%88%86%E6%9E%90%E4%B8%A6%E4%B8%94%E5%8E%BB%E9%99%A4html%E6%A8%99%E7%B1%A4>

首先將網頁內容整個抓取下來，這個我就不說了，不是本次話題的重點。假設抓取的資料放在RecvBuffer這個byte[]中（資料從網路上傳輸過來時不是字串的形式而是byte），那麼我們的第一步就是將RecvBuffer轉化為String，以便於對其操作，實例如下：

// 將接收到的資料增加到回應字串中

strResponse += Encoding.ASCII.GetString(RecvBuffer, 0, nBytes);

strResponse即是保存資料的字串，此處用系統自帶的System.Text.Encoding的方法轉化RecvBuffer，GetString的第一個參數RecvBuffer就是我們的原始資料，即包含需要解碼的位元組序列的位元組陣列；第二個參數0代表第一個要解碼的位元組的索引，一般就從0開始；第三個參數nBytes為要解碼的位元組數，可以自己調整。

得到了資料的字串形式，然後可以對網頁進行解析了（其實就是對字串的各種操作和正則運算式的應用）。下面我以幾個例子來說明對網頁數據的解析：

// 解析頁面，查找連結

// 此處尚需擴展，還有某些形式的連結不被識別

string strRef = @"(href|HREF|src|SRC|action|ACTION|Action)[ ]\*=[ ]\*[""'][^""'#>]+[""']";

MatchCollection matches = new Regex(strRef).Matches(strResponse);

strStatus += "找到: "+matches.Count+" 個連結\r\n";

上面的例子將網頁中的連結解析出來，strRef變數表示了正則運算式的模式，變數matches表示符合匹配的專案的集合，後面的Regex(strRef).Matches(strResponse)就是創建正則規則使得strResponse裡符合strRef模式的字串都返回。然後調用matches的變數就可以取得各種資訊了。

當然，這裡只能識別一些基本的連結形式，像script中的連結和一些不帶「」的連結都沒有被支援，這個的擴展還是蠻簡單的。

再舉幾個更簡單點的解析的例子，大家學習學習：

//獲取標題

Match TitleMatch = Regex.Match(strResponse, "<title>([^<]\*)</title>", RegexOptions.IgnoreCase | RegexOptions.Multiline);

title = TitleMatch.Groups[1].Value;

//獲取描述資訊

Match Desc = Regex.Match(strResponse, "<Meta name=\"DESCRIPTION\" content=\"([^<]\*)\">", RegexOptions.IgnoreCase | RegexOptions.Multiline);

strdesc = Desc.Groups[1].Value;

//獲取網頁的大小

size = strResponse.Length;

—————————————————————割—————————————————————————

好了，下面說一下如何去除Html標籤，這個想必有很多初學者很需要。其實還是正則運算式和字串基本操作的應用，由於這個功能還是比較常用的，所以例子寫成了函數，便於調用：

/\*\*//// <summary>

/// 將Html標籤轉化為空格

/// </summary>

/// <param name="strHtml">待轉化的字串</param>

/// <returns>經過轉化的字串</returns>

private string stripHtml(string strHtml)

{

Regex objRegExp = new Regex("<(.|\n)+?>");

string strOutput = objRegExp.Replace(strHtml, "");

strOutput = strOutput.Replace("<", "&lt;");

strOutput = strOutput.Replace(">", "&gt;");

return strOutput;

}

ok，這樣一來Html標籤就基本沒了，但是有些例外會使得去除不幹淨，所以建議連續兩次轉化，這樣就搞定了。但是還沒結束，如果你留意的話，可以看到上面的函數其實是將Html標籤轉化為了空格。太多連續的空格會影響之後對字串的操作。所以再加入這樣的語句：

//把所有空格變為一個空格

Regex r = new Regex(@"\s+");

wordsOnly = r.Replace(strResponse, " ");

wordsOnly.Trim();

好了，大功告成，這裡的wordsOnly便是我們最終的成果----去除了Html標籤，並去除了多餘空格的字串。