告別苦悶的網頁測試生涯---自動化網頁測試工具 IEUnit

作者：李明儒

聽一位朋友抱怨外包的網站系統，都進入交付階段了，還被抓出一堆 Bug。更要命的是，廠商緊急修掉 Bug 後，許多已測試過的功能反而不能用了，搞得驗收主管火冒三丈，廠商的 RD 副總只得親自下海坐鎮，開發團隊則走了四個人… 問題出在哪裡？ 軟體測試！

軟體測試常見的困境

相信參與過軟體產品或專案開發的朋友都能體會測試有多重要，Bug 常是系統災難的源頭。即使測試的重要性不言而喻，實務上卻還是常常有意或無意地被忽視。不少團隊連開發人力都捉襟見肘了，更遑論建立正式的組織分工，由專人負責測試工作。所謂的系統測試，往往是指程式人員捧著剛出爐的網站程式，按照隔天驗收要跑的 Scenario 熬夜點點按按，確定整個 Demo 流程可以順利跑完就算交差。可以想見，體力透支的程式設計人員哈欠連連、睡眼惺忪之餘，常常就”不慎”跳過操作較繁瑣或罕用的介面。隔天驚險地通過驗收後，陰暗角落中的 Bug 甚至得等上一年(例如：藏在一年只用一次的結算功能)才會現身。

較豪華的開發團隊裡，有機會見到所謂的”測試美少女”(這似乎是軟體測試員的典型，大概是因爲小女生們細心、有耐性，較願意接受這類枯燥、待遇平平的工作吧！)。至少在專人測試下，可以減少程式撰寫與測試工作互相排擠的窘境，但是只要遇到開發進度延遲(幾乎 90%的專案都是如此吧？)，測試時程還是不免被壓縮。在專案經理的明示暗示下，簡略測試步驟、犧牲部分軟體品質似乎是及時交付的唯一解。

除了上線/出廠前的測試，還有另一項挑戰：通過完整測試的程式，日後仍免不了修修改改，而修改動作有可能引發一些副作用，使原來通過測試的項目發生問題。(改掉一個 Bug，冒出兩個新 Bug，相信許多人都有經驗吧？)

理論上，只要更動到較核心的程式碼，就應重跑一次完整測試。但這是對測試人員毅力最嚴酷的考驗，鮮少有人能在同樣的動作做了一百次後，仍保有與第一次相同的專注與謹慎。

追根究底，或許不該責怪測試人員因循苟且，一切只因人性使然。只是在測試工作的諸多環節中，”人”成了最弱的一環，固然情有可原，軟體品質下降卻是不爭的事實。因此，將枯燥無味的反覆動作交給電腦執行，看來是基於成本、效率、品質考量的最佳解。

自動化測試工具

軟體測試的自動化不是什麼新發明，業界已有不少成熟的產品[參考資料 1]。如果我們將自動化測試聚焦到”網站系統的功能測試”，可以看到網站測試自動化的概念可由以下三種角度切入：

1. 模擬滑鼠、鍵盤的操作

   錄製下使用者的滑鼠游標移動、按鍵動作，再加以播放。如果滑鼠座標、鍵盤輸入內容必須隨機應變，則需要修改 Test Script。但以座標數字、延遲時間長短爲主的 Script，撰寫起來十分繁瑣不直覺，也不易維護。另外，滑鼠/鍵盤只能處理輸入段，要檢查輸出結果是否符合預期，仍要採取其他途徑輔助。

2. 解析 HTTP 協定，直接模擬網路傳輸

   在網站系統中，不論如何複雜的瀏覽器互動操作，最後都還是要透過 HTTP 協定將特定的內容傳至伺服器。

測試軟體只要模擬送出同樣的內容，就等於取代了瀏覽器的複雜操作。但這有兩項缺點:

1) 測試重點放在伺服器端程式的處理，前端 Script 的程式邏輯被排除在測試範圍之外。(對壓力測試來說，這反而是優點。)

2) 需要額外的測試程序邏輯時，修改者得先理解前端動作與 POST 內容的關聯，再直接操控組合不同的 POST 內容，較爲複雜。

3. 解析並控制 HTML DOM 物件，模擬使用者操作

這種測試範圍包含了網頁前後端的程式邏輯，涵蓋度較高。而直接操作 HTML DOM 物件，遠比組裝 POST 內容或計算滑鼠座標來得直覺化，大幅降低 Test Script 的撰寫難度。如果要說缺點，DOM 無法涵蓋所有的瀏覽器互動，例如: 連上網站時輸入帳號/密碼的動作、選取檔案上傳的操作、元件是否被擋住或超出視窗範圍…等等。

解決方案有很多種，測試團隊可以依功能要求、開發技巧需求、採購預算決定合適的產品或工具。若以中小型軟體公司測試部門或小型專案測試團隊的需求爲例: 測試人員以執行功能性測試爲主，測試範圍要能涵蓋前端的網頁 Script。因本身從事前端網頁的開發，不乏 Javascript 開發能力，而由於公司或團隊規模不大，測試工具的採購預算十分有限。

若以這些條件爲前題，IEUnit[參考資料 2]將會是相當適合的解決方案。本文就來介紹這個能讓測試人員脫離苦海、專案經理眉開眼笑的自動化網頁測試工具---IEUnit。

註: 使用 Visual Studio 2005 Team Suite 或 Team Edition For Software Developer, Team Edition For Software Tester 的開發者還有一個選擇---可以考慮採用 Visual Studio 2005 內建的 Web Test Project，不過它操控的重點放在送回網站的 POST 內容上，就如前述的第二種方式，以 FormPostParameters 物件更動傳送給伺服器的資料[參考資料 3]，與直接操作 HTML DOM 相比，畢竟隔了一層，較爲不便。即便忽略 Visual Studio 2005 的成本，針對單純的自動化網頁功能測試，IEUnit 還是較具優勢。

IEUnit 概述

IEUnit 是個開放源碼(Open Source)專案，是著名的 xUnit Test Framework[參考資料 4]在 IE 及 Javascript 上的實作。簡單來說，IEUnit 使用 Javascript 在 Windows 平台上開啓 IE 連至特定網頁，並執行 Test Case 存取、操作網頁的 HTML DOM 物件，模擬出網頁互動，取代人工操作瀏覽器。而執行結果的檢查，一樣可透過檢查 HTML DOM 達成。

在 IEUnit 中，Test Case 以.jst 檔案爲單位，Test 的程式邏輯被寫在 Test Case 檔案的 Javascript 函數中；同一目錄下的多個 Test Case 則組成一個 Test Suite。

由於 Test Case 的本質就是用 Javascript 存取 HTML DOM，對網頁開發人員來說，跟撰寫前端的 Script Code 沒什麼兩樣。Test Case 開發人員不需要學習就能上手，是 IEUnit 的迷人特點之一。而開放源碼的 IEUnit 沒有採購成本，在特殊狀況下也可修改核心程式碼更符合實際需求，算是很出色的解決方案。若硬要說缺點，一來它跟 IE 的核心綁得很緊，限定 IE+Windows 的組合，有平台上的限制，也無法測試不同瀏覽器間的差異。二則以 HTML DOM 爲操作對象的做法，無法涵蓋一些常見的瀏覽器操作(例如: 挑選檔案上傳、輸入網站登入帳號密碼…等)，IEUnit 雖提供了解決方式，但仍不夠簡便。

IEUnit 初體驗

網站上有現成的 MSI 安裝檔可供下載[參考資料 5]，執行後即可完成安裝(需要.NET Framework 1.1)。IEUnit 並沒有像 Visual Studio 一樣華麗的 IDE 環境，Test Case 的開發可以使用一般的文字編輯器(當然，萬用程式編輯器--Notepad 也可以勝任)。IEUnit 2.0 以後的版本附了一個由 VisualMap Technology 公司開發的輔助工具 QuickFocus，讓開發人員可以透過視覺化的方式操作網頁，產生 Script 的雛型。對初學者來說，這是逐步熟悉 IEUnit Test Case 寫法的捷徑，所以我們就用 QuickFocus 來產生第一個 Test Case。

安裝好 IEUnit 後，可以從程式集中找到 IEUnit/QuickFocus，啓動後會看到如圖 1 的操作介面。

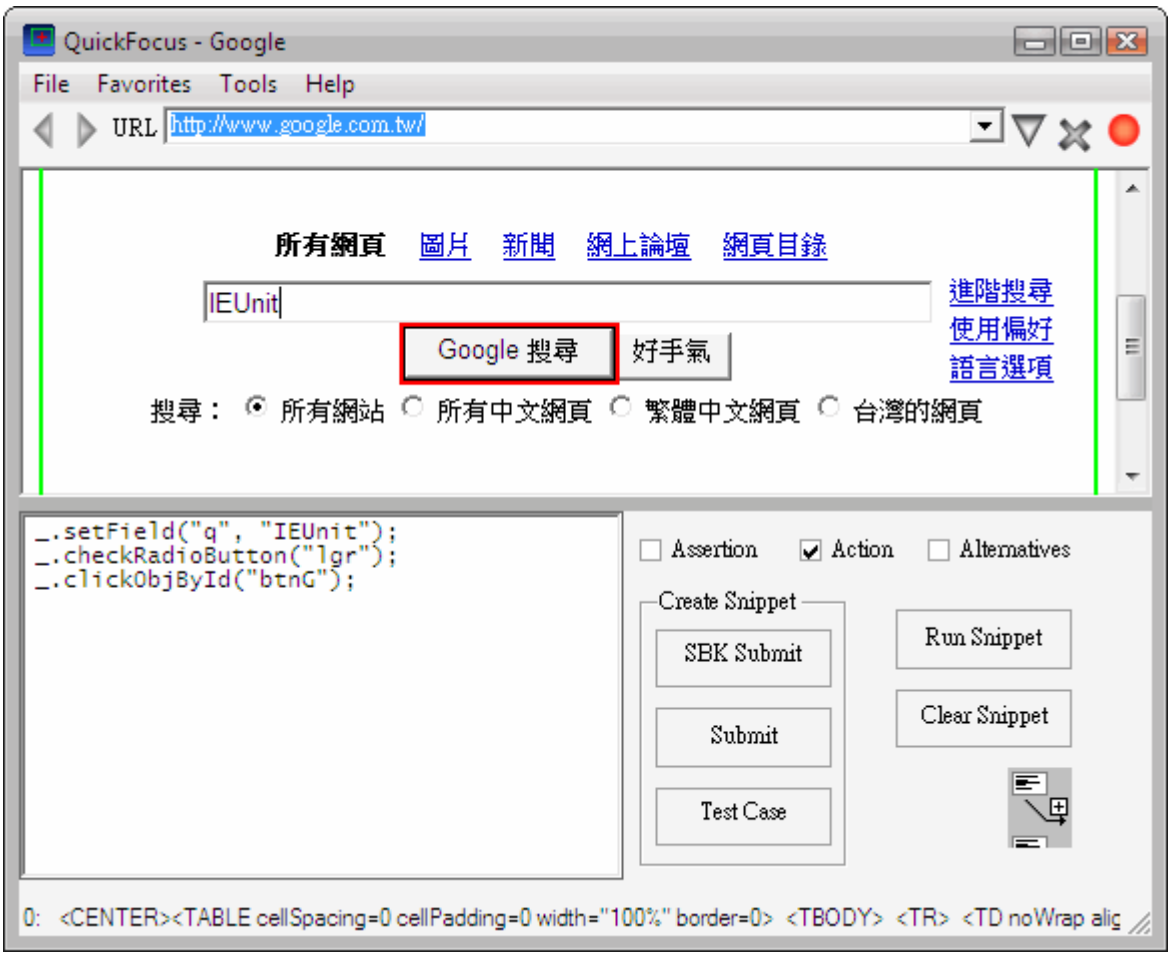

圖 1 QuickFocus 的操作畫面

QuickFocus 的操作流程如下:

1. 輸入 URL 後，中央視窗是一個 IE 瀏覽器，會顯示網頁內容。
2. URL 輸入框右方有個圓鈕，可以切換操作模式。圓鈕呈灰色時(Detach 模式)，中央視窗就化身爲標準的 IE 瀏覽器；圓鈕呈紅色則代表 Attach 模式，在 Attach 模式下，使用滑鼠在網頁上的各元素間移動，下方會顯示出該元素的 HTML Tag 內容(相當於 OuterHTML 屬性)，協助我們解析頁面上的各元素。
3. 在Attach模式下,若你點選網頁上的元素,下方就會出現類似_.setField("q","IEUnit");的Javascript程式碼。這些程式碼的目標便是模擬剛剛在網頁上的操作。我們可以透過這種方式學習 IEUnit 針對各種元素操作所提供的現成函數。例如: 指定 HTML Text Input 可用_.setField()，點選 RadioButton 則可以用_.checkRadioButton()。(註: 在 Test Case 中，底線符號_.等於 this.)

當然，我們也可以不依賴這些現成函數，使用_.doc 取得該網頁的 document 物件後，用 _.doc.all(“inputName”)去找元素，再進行操作。只是 IEUnit 的現成函數中有不少額外的應變容錯邏輯，一方面更嚴謹，一方面也可減少我們的程式量，值得善加利用。IEUnit 的完整 API 參考文件可以由程式集 IEUnit 目錄下的 API Reference 網頁找到。

4. 在 QuickFocus 中，有個 Run Script 鈕可以用來測試 Script 的執行效果。而 Create Snippet 的三個按鈕可以用來產生樣版，其中按下 Test Case 鈕產生的就是完整 Test Case(.jst)的樣版(如程式 1 所示)。在程式 1 中，assimilate 是 IEUnit 用來模擬物件導向概念的做法，assimilate(this, new IeUnit())會讓目前的物件擁有 IeUnit 物件的所有函數、屬性，相當於物件導向中的繼承。setUp 及 tearDown 函數則會在每個測試函數開始與結束時被呼叫。testCaseOne 內含的即爲測試的內容，代表了 Test Case 中的一項 Test，一個 Test Case 可以包含多個 Test，Test Function 的名稱(例如: testCaseOne)可以任意自訂，但必須以 test 起首。

程式1 Test Script範本

```
function SampleCase() {
  assimilate(this, new IeUnit());
  this.setUp = function() {
    _.openWindow("http://www.google.com.tw/");
  };
  this.tearDown = function() {
    _.closeWindow();
  };
  this.test1 = function(){
    _.setField(“q”,”IEUnit”);
    _.clickObjById(“btnG”);
    _.assertPageHasText("ieunit.sourceforge.net/");
  };
  this.test2 = function(){
    _.setField("q","單元測試");
    _.clickObjById("btnG");
    _.assertPageHasText("單元測試及先行測試開發");
  };
}
```

註: QuickFocus 輔助產生的程式碼中，不包含網頁結果的比對，這部分需要手動開發。最笨的做法是取得 document 物件後尋找出特定的網頁元素做比對，發現不正確時 throw new IeError(errCode,errMsg)宣告未通過測試。IEUnit 提供了一些函數可以簡化這些檢查，例如: assertPageHasText、assertSelectHasOption、assertTagHasText、assertTextContains 等，另外，若要由一般的邏輯比對觸發測試失敗，則可利用 assertTrue、assertCondition..等函數。

執行測試

藉由 QucikFocus，我們已大致了解 Test Case 的結構及約略模樣，由 QuickFocus 右下方的 Icon 可以將產生的程式片段複製成 Test Case 檔案(.jst)。接著，來看看 IEUnit 如何執行 Test Case。

依循 xUnit Framework 的設計，IEUnit 提供了兩種 Test Runner，測試人員可以依需要選擇純文字或網頁式的 Test Runner；還有第三種方法是在.jst 檔上按右鍵，會看到 Open 與 Debug Script 兩個專屬選單項目，都可用來執行 Test Case。

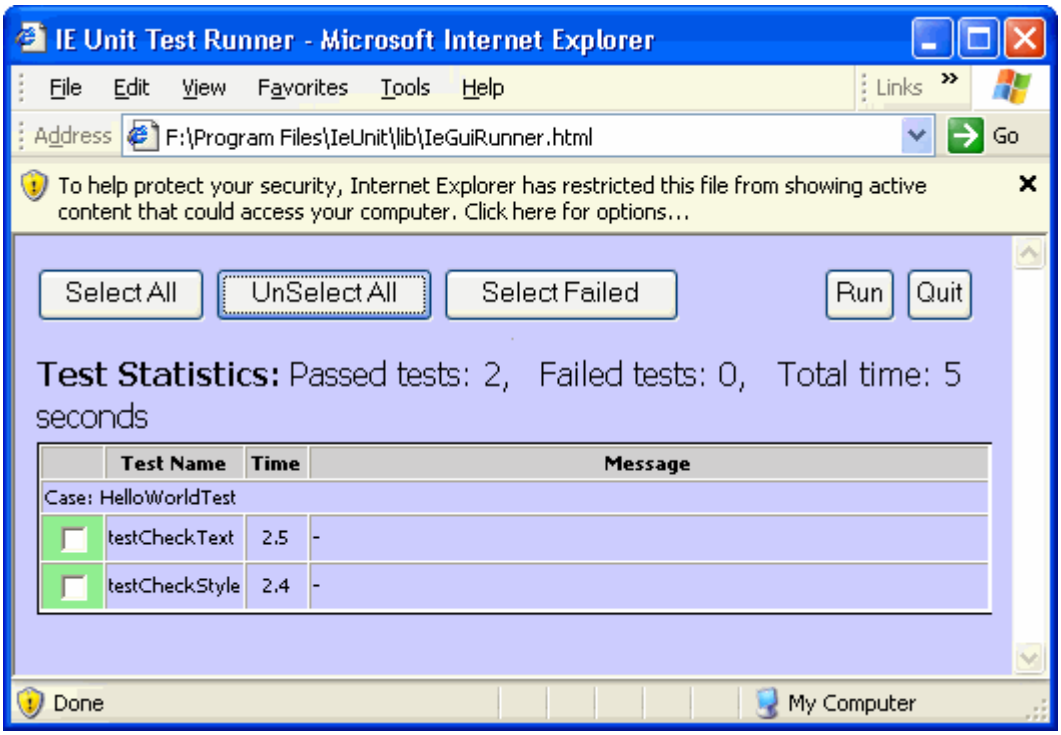

圖 2 IeGuiRunner 的執行畫面(上方會出現限制警告，經測試不致影響操作)

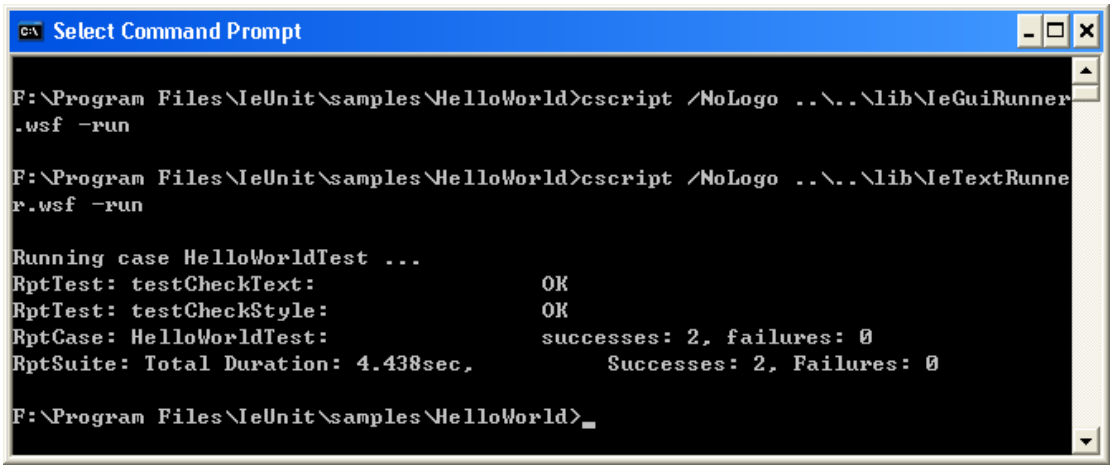

圖 3 GUI、Text Test Runner 執行指令及 IeTextRunner 執行結果範例

在軟體測試實務上，使用 GUI Test Runner 或直接開啓 Test Case 檔只適用於開發 Test Case 階段。一旦 Test Case 開發完成後，應排定成標準的軟體開發作業程序，只要修改到網站中的任一環節，就應找出影響

的模組，重新執行該模組的完整單元測試，以確保所有的功能正常無誤。因應這類常規作業需求，多半會使用 IeTextRunner.wsf，而它也提供了一些參數，例如：

- 可要求不顯示 IE(-v false)，適用於批次執行
- 可輸出成 XML 檔案(-xml filename)，格式與 NUnit 相容，可配合 Nunit2Report 產生 HTML 格式的報表(如圖 4 所示)
- IeTextRunner 預設會執行目錄下所有的 Test Case 檔，但也可以使用 -run 參數指定只執行特定 Test Case，或某 Test Case 中的特定 Test

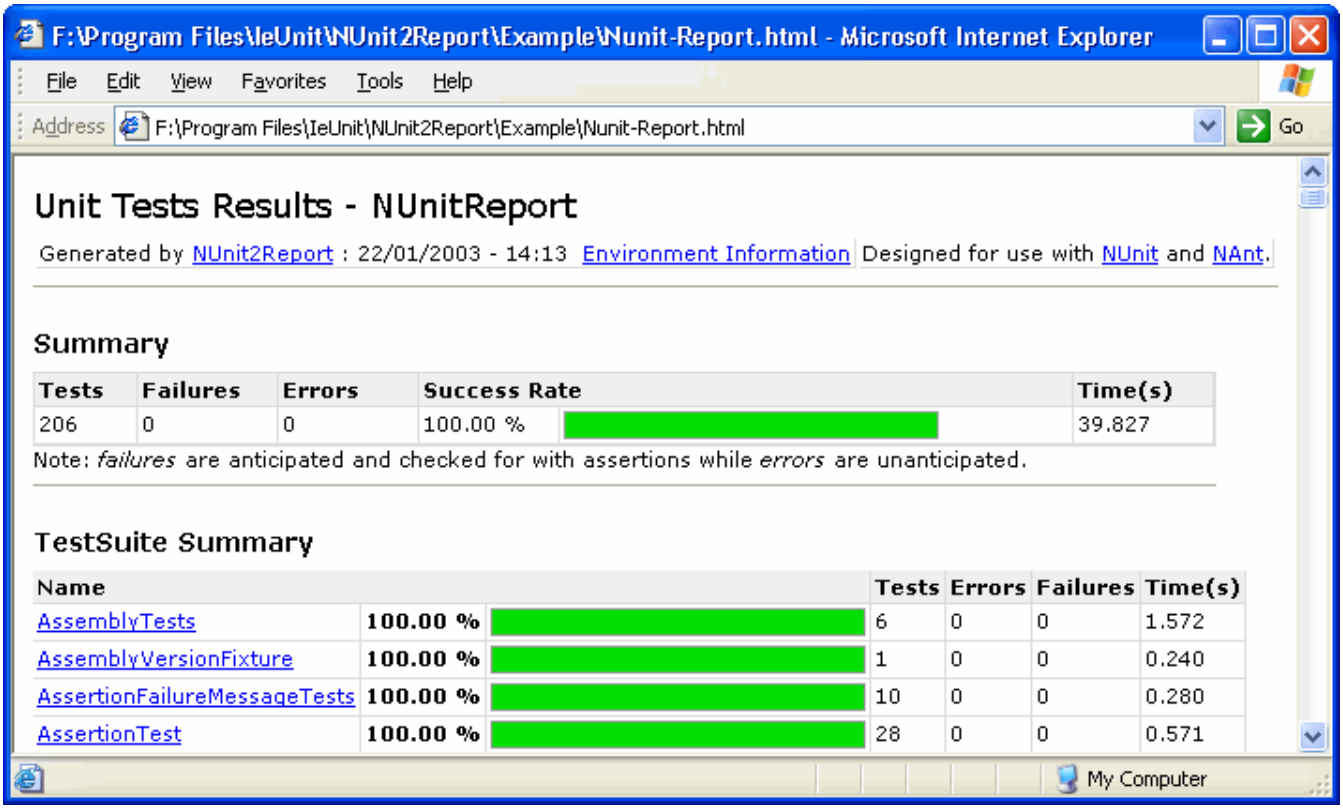

圖 4 IEUnit 結合 NUnit2Report 產生的測試報表

Test Case 開發提示

Test Case 說穿了就是 Javascript+HTML DOM，難不倒寫網頁的老手，但有點讓人爲難的是，IEUnit 提供了不少尋找、操作、比對網頁物件的共用函數，都可以使用傳統 document.all(objId)等方式取代，到底要不要學習及應用 IEUnit 的函數呢? 我的看法是，應用 IEUnit 提供的函數讓 Test Case 更簡潔，同時程式風格比較不會因人而異，形成其他人解讀與維護時的額外困擾。

以下簡單提示一些常用的函數，詳細的函數用法及原始碼，可以在 IEUnit 的 API 參考文件中找到。

1. 尋找網頁元素：
   - findByObjId(): 適用於已知 Name 或 Id 屬性時，效果與 document.all()相同
   - findButton、findCheckBox、findField、findImage、findLink、findSelect、findTextArea: 顧名思義，即找尋特定類型的元素
   - findTableCell、findRowByText: 找尋特定 TD 或 TR
   - findParent: 依 Parent 關係向上找尋特定 Tag 的元素

- findByTag、findByTagAndAttr：找尋特定 Tag 的元素，支援以 attrName~attrValue 的語法限定屬性
- findByText：包含特定文字的元素

當搜尋結果爲陣列時，部份函數還可傳入索引數字，指定傳回陣列中的第幾個。另外，findScope 可以指定搜索的範圍，findTimeout 則可指定尋找逾時的期限。

2. 網頁元素操作：
   - checkRadioButton、clickButton、clickImage、clickLink、clickLinkRange、clickObj、clickObjById、setCheckBox、setField、setTextArea、setSelectOption：點選或設定各式網頁元素

   部分元素在點選後，會觸發網頁送出(Postback)或連至其他網頁的動作(例如：Submit Button、Link)，此時應等待新網頁完全載入後再進行後續操作。有個 checkSubmit 函數會持續檢查，直到 document 狀態變成 complete 爲止。clickButton、clickLink、clickObj 等函數已內含 checkSubmit 程序。

   另外，有兩個參數與 checkSubmit 相關：submitPause 是指觸發送出動作後，等待一小段時間讓 IE 將內容送至 Server 端，之後才開始檢查 document 狀態；若等待超過 findTimeout，則會產生錯誤。
3. IE 操作
   - openWindow、openWindowAsync：以同步或非同步方式開啓一個新的 IE 視窗連向指定的 URL
   - navigateTo、navigateToAsync：以同步或非同步的方式連向指定的 URL
   - seekWindow、seekAndSetWindow：找尋具有特定標題的 IE 視窗
   - win、doc：相當於 Javascript 中的 window 及 document 物件
   - comWin：IE 視窗的 COM 物件
   - closeWindow：關閉目前的 IE 視窗
4. 比對函數：
   - assertPageHasText、assertSelectHasOption、assertTagHasText、assertTextContains：網頁相關檢查
   - assertContains、assertEquals、assertFail、assertFalse、assertMustFail、assertNotNull、assertNull、assertTrue：通用檢查
5. 其他函數：
   - setTime：設定 submitPause 及 findTimeout
   - _sleep：等待一段時間，再進行後續動作
   - _contains：比對字串中是否包含另一字串

進階應用議題

以 HTML DOM 及 Javascript 爲基礎的 IEUnit，可以應付絕大部份的網頁操作需求，但也存在一些先天上無法涵蓋的域領，例如：

1) 連上網站時，由網站伺服器發出的的身份認證要求(HTTP Authentication)。由於 HTML 根本尚未載入，自然無 DOM 可用。
2) 對網頁以 showModalDialog 方式叫出的新視窗，HTML DOM 並不提供任何存取新視窗的管道。
3) 上傳檔案時，挑選檔案的動作，基於安全的考量，不允許 Javascript 控制。(試想，若網頁中的 Javascript

可以偷偷決定上傳你本機的某個機密檔案，多可怕!)

4) alert、prompt、confirm 等要求 User 確認或輸入的動作，原本即非 Javascript 所能干涉的。

以上所提的種種限制，雖然都有其存在的理由，卻會影響測試自動化的流暢度。對此，IEUnit 提供了解決方案。

基本原理

深入了解，IEUnit 之所以能跨出 HTML DOM 的領域，依賴的是一顆另外撰寫的 COM+元件 --Win32Dom.Desktoop。

在安裝 IEUnit 時，會一併註冊這顆以 C#開發，並以 CCW(COM Callable Wrapper)包裝的元件。它主要封裝了一些需要呼叫 Wndows API 方能達成的功能，例如: 列舉或搜尋桌面上所有 Window 的清單、取得 IE 視窗中的 Document 物件、甚至還能控制 Window Form 上的 Button、TextBox! 除此之外，它也支援對特定 Window 模擬滑鼠、按鍵的功能，要做些簡單的 Window Form 程式測試也是可行的。IEUnit\samples\Win32DomTest\ 下有幾個範例可以參考，如果你對這顆神奇的元件有興趣，SourceForge 上也找得到 C#原始碼。[參考資料 6]

有了 Win32Dom.Desktoop，Test Script 就能找尋桌面上特定標題的 Window，進一步設定其中 TextBox 的值、按下特定按鈕；若找到的是載入網頁的 IE 視窗，還可以與其中的 HTML DOM 搭上線!

具備了這些能力，要找到身份認證對話框，輸入帳號密碼後按下登入鈕，便不再是難事。(動動腦筋，它還可以用來玩出更多好玩的應用呢!)

網站身份

當 IIS 設定基本驗證或 NT 整合式驗證，IE 在連上網站載入網頁前，會彈出身份認證對話框，必須輸入有效的帳號、密碼，並按下確定後，才能瀏覽網頁。Win32Dom.js 中提供了一個 openWindowAsUser 函數，可以指定 URL、Username 及 Password，就能用指定的身份連上網站並開啓網頁，如程式 2 所示。

程式 2

```
function WebLoginTest() {
    assimilate(this, new IeUnit());
    this.testLogin = function() {
        var url =
"http://ieunit.sourceforge.net/samples/LoginDemo/Helloworld.html";
        this.openWindowAsUser(url, "ieunit", "DemoPwd");
        this.assertPageHasText("Hello World!");
        this.closeWindow();
    };
}
```

很遺憾地，這段 IEUnit 提供的標準範例可能會讓很多人失望，IE 會卡在輸入密碼的對話框不動，未如預期做到自動登入。仔細探究，原來是 openWindowAsUser 函數利用密碼輸入對話框上的標題作爲搜索條件，以鎖定對話框 Window 輸入帳號密碼。而這部分並未考量到不同語系下標題文字的不同，於是在英文版以外的 Windows 執行時，openWindowAsUser 函數便失效了。所幸，IEUnit 是個 Open Source 專案，我們只需更改

Win32Dom.js 兩處即可解決這個問題(如程式 3)。當然，更嚴謹的做法應該要判斷 Windows 的語言版本自動決定比對的文字內容，但在固定使用特定語言版本 Windows 的測試環境，我們只需稍做修改就夠了。

程式 3 IEUnit\lib\Win32Dom.js 第 117 列

```
//loginWinCap = "Connect to " + loginWinCap;
loginWinCap = "連線到 " + loginWinCap;

this.openWindowAsync(url);
…省略…
this.findTextBox(loginWin, 2).text = userPwd;
//this.findWinButton(loginWin, "OK").Click();
this.findWinButton(loginWin, "確定").Click();
```

Modal Dialog

或許有些人對 Modal Dialog 不是很熟悉，基本上它就是由某個網頁另外開啓的一個網頁，但新開啓的網頁會強制取得焦點，直到該視窗關閉後才歸還，它特別適用於彈出另一個網頁強迫使用者檢視、選擇或輸入後再返回的場合。要套用這種強制使用者選擇、操作後再返回的模式，可以透過 window.showModalDialog(sURL [, vArguments] [, sFeatures])函數，以 Modal Dialog 方式開啓指定的網頁即可。

Modal Dialog 在網頁 UI 設計實務上還挺常見的，例如：寫信時叫出連絡人清單視窗供使用者挑選收信人、在清單中選取一筆資料編輯後返回清單…等等。使用 Modal Dialog 可以避免另開的新視窗跑到背景、被最小化後不知去向的混淆。

在 DOM 中，使用 Iframe、Frame、window.open 等開啓的網頁，原網頁仍具有控制權(前題是要符合新網頁也屬於同一 DNS 網域的安全限制)，獨獨對 showModalDialog 開出的新網頁，DOM 並未提供任何存取管道。要溝通只能透過 dialogArguments 將原網頁的 DOM 相關物件當成參數傳過去，由新網頁操作原網頁或傳回數值、元件，原網頁對新網頁則是全然無法可管。

這點特性在撰寫 Test Script 時會造成很大的困擾。舉例來說：模擬發信測試時，叫出連絡人選取視窗後的下一步，當然是要挑選某個連絡人。我們可以用 Javascript Click 網頁上的某個 Button 觸發事件，以 showModalDialog 顯示連絡人選取清單，但接下來便無法存取 Modal Dialog 網頁的 DOM 進行後續的連絡人選取操作。

爲了克服這個問題，IEUnit 再次借重了 Win32Dom.Desktoop 尋找桌面上特定標題 Window 的能力，以 Modal Dialog 特有的”Modal Popup Window – 網頁對話”(聰明的你應該想到了，愛台灣、重本土的中文版使用者們又得動手改比對字樣了!)標題找到新開的 Dialog IE 視窗，接著就能跟新網頁 DOM 搭上線。不過由於 IEUnit 中，每個 Test Case 以存取單一 IE 視窗的 DOM 爲主，要操作 Modal Dialog 時，得預先啓用另一個獨立的.sbk(Smart-Bookmark Script)，以守株待兔的方式等待 Modal Dialog 出現，再進行預先設計好的操作。

IEUnit\samples\Win32DomTest 目錄下有四個 Modal Dialog 的測試檔案，ModalDialog.html 以 showModalDialog 方式叫出 ModalPopup.html 要求使用者輸入姓名後傳回 ModalDialog.html，ModalDialogTest.jst 負責主導整個測試過程，並以 WScript.Shell Run 函數啓動 ModalEnterName.sbk，由 ModalEnterName.sbk 負責監測 ModalPopup.html、輸入姓名並按下 submit 將姓名傳回

ModalDialog.html。

對大部分的人來說，這一組測試需要做些修改才能執行：

第一，較新版本的 IE 預設會停用本機 HTML 檔案中的 Javascript，導致 showModalDialog 函數失效。建議將兩個 HTML 放到本機的 IIS 下並修改 ModalDialogTest 中的 URL，以避開 Javascript 被停用的問題。

第二，跟登入對話框的問題一樣，IEUnit 未考量到不同版本的標題差異，所以我們也得變更一下 ModalEnterName.sbk 中要搜尋的標題文字。

修改後的程式如程式 4、5。

程式 4 IEUnit\Samples\Win32DomTest\ModalDialogTest.jst

```
function ModalDialogTest() {
assimilate(this, new IeUnit());
this.testModalDialog = function() {
    //html 檔改放 IIS 時，URL 要修改
    this.openWindow("http://localhost/ModalDialog.html");

    // 開啓 Modal Dialog 前, 先啓動攔截程式
    var cmdShell = new ActiveXObject("WScript.Shell");
    cmdShell.Run("ModalEnterName.sbk", 0, false);

    this.clickButton("Open Modal Dialog");
    this.assertPageHasText("Hi, Jeffrey");
    this.closeWindow();
  };
}
```

程式 5

IEUnit\Samples\Win32DomTest\ModalEnterName.sbk

```
this.waitForModalDialog("Modal Popup Window -- 網頁對話");
this.setField("yourName", "Jeffrey");
this.clickButton("submit");
```

完成以上修改，利用 RunTest ModalDialogTest 指令就可以看到測試成功的訊息了。

上傳檔案

基於安全考量，想透過 Javascript 決定上傳檔案的路徑與檔名，是不被允許的。而在模擬測試時，要克服這個障礙，解決方法類似 Modal Dialog，也是利用攔截特定標題的 Window 完成。IEUnit 的 CVS 中可以找到相關的範例[參考資料 7]，以下是一個指定上傳檔案的簡單測試：(程式 6、7、8)

程式 6 Upload.html，放在 Web 上

```
<html><body>
Select File: <input type="file" name="uploadFile">
```

```
</body></html>
```

程式 7 UploadTest.jst

```
function UploadTest() {
  assimilate(this, new IeUnit());
  this.testUpload = function() {
    this.openWindow("http://localhost/Upload.html");
    this.clickObjById("uploadFile");
    var fname = "C:\\Temp\\Test.txt";
    var cmdShell = new ActiveXObject("WScript.Shell");
    cmdShell.Run("EnterFileName.sbk " + fname, 0, false);
    this.assertEquals(this.findObjById("uploadFile").value, fname);
    this.closeWindow();
  };
}
```

程式 8 EnterFileName.sbk

```
var fpath = " " + WScript.Arguments(1);
var popupWin = _.waitForWindow("選擇檔案", 30000);
_.findWindow(popupWin, "Edit").sendText(fpath);
_.findWinButton(popupWin, "開啓(&O)").click();
```

Popup Window

在網頁中，有時會用到 alert、confirm、prompt 等簡單的 Popup Window，等待使用者按下確定、決定 Yes/No 或輸入特定文字後傳回。在自動測試的過程中，如果要省略掉這段互動，IEUnit 建議的作法是以自訂的函數取代內建功能。例如: this.doc.parentWindow.confirm = function() { return true; }就會覆寫系統原本的確認功能，不彈出任何對話框，直接傳回 true 給呼叫程式。以此類推，alert、prompt 都可以比照辦理。IEUnit\Samples\ApiTest 下的 SupressPopupWindow.jst 中有完整的範例。

Debug Test Script

最後補充一點，撰寫 Test Case Script，免不了需要 Debug。要 Debug .jst 檔，有兩種選擇:

第一種是透過 IeTextRunner 的/X 參數，例如:

```
cscript /X %IEUNIT_HOME%/IeTextRunner.wsf -run yourTestScript.jst
```

第二種方法，是在檔案總管中找到.jst 檔按右鍵，選單中會有一個 Debug Script 的選項。

以上兩種方式，會在啓動.jst 的同時就叫出 Windows Script Debugger，並停在第一個指令上，若你希望在程式碼的特定位置加上中斷點，可以在 Script 中加入一列 debugger;指令。

關於 Debug 的進一步細節，可以參考[參考資料 8]。

結論

以上的介紹，算是爲大家引見了 IEUnit 這個沒沒無聞，卻威力十足的免費網頁測試工具。原來網頁測試自動化的實踐，並非遙不可及的夢想，也未必得仰賴昂貴的自動化測試工具。或許我們該認真思考，別再用各種藉口省略、逃避繁瑣的測試工作了；開始學著善用工具，以完善的測試，讓網站品質更上一層樓!

參考資料

參考資料 1 Web Functional Test Tool http://www.aptest.com/webresources.html

參考資料 2 IEUnit 網站 http://ieunit.sourceforge.net/

參考資料 3 VS 2005 Web Test 介紹 http://www.myfaq.com.cn/2005Nov/2005-11-25/209461.html

參考資料 4 xUnit http://en.wikipedia.org/wiki/XUnit#A_Partial_List_of_xUnit_Frameworks

參考資料 5 IEUnit 下載 http://www.visumap.net/index.aspx?p=Resources/WebTesting

參考資料 6 Win32Dom 元件原始碼
http://ieunit.cvs.sourceforge.net/ieunit/IeUnit/tool/Win32Dom/

參考資料 7 上傳檔案 Test Script 參考 http://ieunit.cvs.sourceforge.net/ieunit/IeUnit/contrib/

參考資料 8 IEUnit FAQ http://ieunit.sourceforge.net/FAQ.html