體驗 IE8 Activity 開發

作者：李明儒

在 IE8 推出的諸多新特性裡，除了支援新標準、安全性及效能提升，最受矚目的莫過於 Activity 與 WebSlice 了。本文將以一個簡單實例帶領讀者體驗 IE8 Activity 的開發。

IE8 Activity 是什麼?

簡單來說，IE8 Activity 賦與滑鼠右鍵更多的功能，透過簡單的介面規格讓開發者可以擴充各式網頁整合服務。Activity 的基本操作模式是以滑鼠選取網頁上的文字或停留在特定連結元素上，按滑鼠右鍵即帶出可用的 Activity 服務選單，移動滑鼠至 Activity 項目時，IE8 會將選取內容及相關資訊送交 Activity 指定的網頁，呈現預覽畫面；使用者如有需要，點選 Activity 項目還可開啓 Activity 網頁，以正常網頁大小進行完整操作。

用一個淺顯易懂的英文翻譯例子來說明，英翻中的 Activity 可以讓我們在閱讀網頁時，順手選取一段文字後快速獲得中文翻譯結果。這是 IE8 Beta 1 預先內建的 Activity 之一，毋需額外安裝設定，只要在網頁上選取一段文字，IE8 會自動浮出一個綠色的斜箭頭圖示(如圖 1)，點選後可列出可用的 Activity 清單。將滑鼠移至"Translate with Windows Live" Activity 上，會浮現另一個預覽視窗，並將剛才選取的英文翻譯成中文(如圖 2)；點選 Translate with Windows Live，則會另開 IE Tab 顯示完整的翻譯操作界面(如圖 3)；還有第二種觸發方式，可在網頁上點選右鍵，叫出內容功能表(Context Menu)，一樣也會列出 Activity 清單。

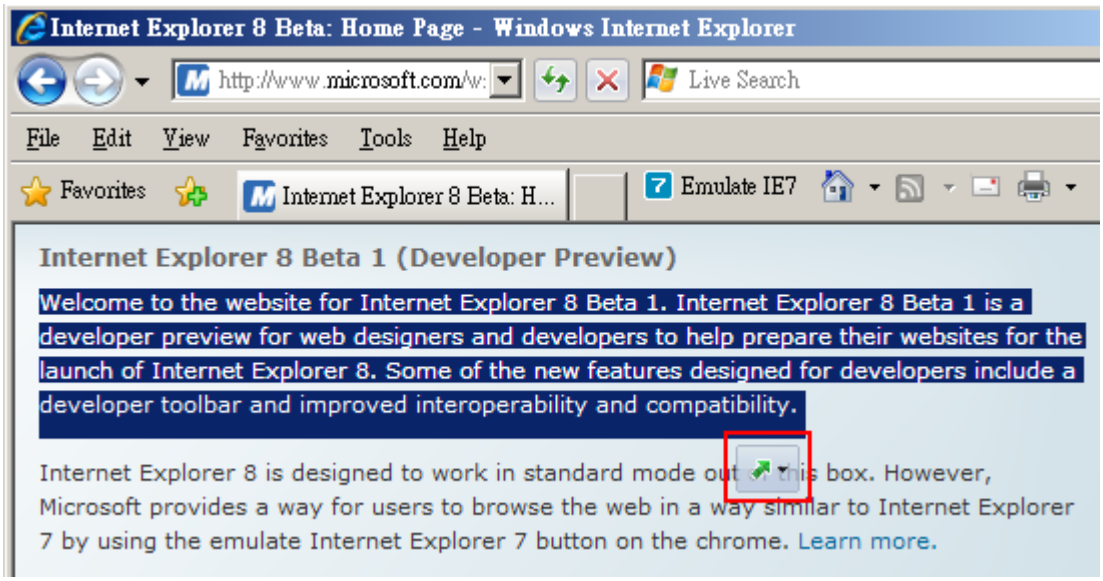

圖 1 選取文字後浮現綠色斜箭頭圖示

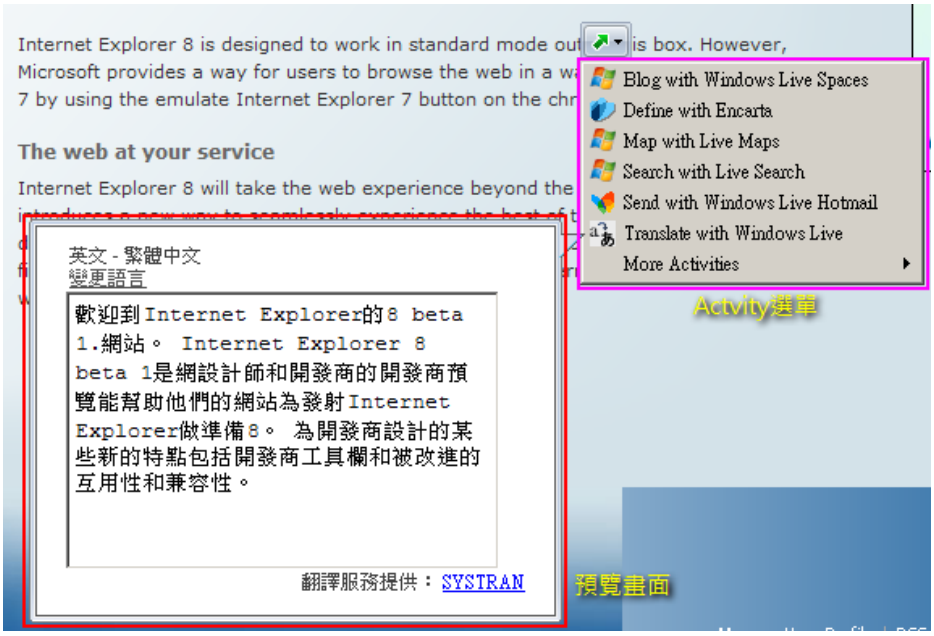

圖 2 語言翻譯 Activity 預覽畫面

圖 3 點選 Activity 後可使用完整網頁介面操作

由以上的說明應可以體會出 Activity 的設計理念: Activity 的終極目標是要簡化原本選取、複製、開啓服務網頁、

貼上、取得結果的過程，讓服務更貼近使用者，透過兩三個滑鼠操作就得到結果，提高服務的便利性。對於服務提供者來說，可視爲一個提供高度網頁整合服務的管道。

開發相關資源

除了 IE8 內建的幾個 Activity 外，還有一些額外的 Activity 可供下載安裝，透過 Context Menu 中的 More Activities/Find More Activities 選單會引導到 IE Team 所維護的 Activity Providers 目錄 (http://ie.microsoft.com/activities/en-en/default.aspx)。目前可選擇的 Activity 數量還不算太多，但隨著 IE 上市腳步的逼近，Activity 項目應會逐漸豐富起來。

對網頁開發者來說，要開發自己的 Activity 亦非難事，MSDN 上可以找到 Activity 的開發指南 (http://msdn.microsoft.com/en-us/library/cc289775(VS.85).aspx)，按照其中的指引，不需太多額外功夫即可做出 Activity。以下我們就來動手實作一個 Tiny URL 網址簡碼轉換服務，體驗一下快快樂樂開發 Activity 的感覺。

運作原理

在捲起袖子動手前，先來解析一下 IE8 Activity 的運作原理。Activity 的核心服務以標準網頁方式提供，並無其他高深學問，以先前所提的英翻中爲例，我們看到了兩個網頁，第一個是圖 2 中的預覽網頁，第二個則是圖 3 的完整操作網頁，使用者選取的內容透過 Query String 或是 Form POST 的方式傳至網頁，這部分跟一般網頁運作並無不同，對網頁開發者來應非難事。

但要如何讓 IE8 知道將使用者選取的內容送到我們寫好的網頁呢? IE8 採用了名爲 Open Service Description 的 XML 文件規格，藉以定義 Activity 的各項細節。

換句話說，我們只要寫好兩個網頁(一個預覽用、一個全瀏覽器操作用)，再多定義一個 Open Source Description XML 檔，一個全新的 Activity 就大功告成了，十分簡單。

Get TinyUrl Link Activity 構想

Tiny URL 是很通用的網址簡碼服務，它可以將冗長的網址變成幾個文數字組成的簡短 URL，看起來簡潔許多，也便於轉述或抄寫。例如: http://blog.darkthread.net/blogs/darkthreadtw/archive/2008/06/04/minicsharplab12.aspx 會轉成 http://tinyurl.com/6j7v9s， 當場由 85 個字元簡化成 25 個字元。

Tiny URL 雖然已經提供了 Toolbar 協助快速轉換網址，但是使用 IE8 Activity 操作的流暢度顯然更上一層樓。因此，就假設我們是 Tiny URL 開發團隊的一員，試著製作一個可快速轉換 TinyURL Actvitiy Provider。

首先，我們要準備兩個網頁，一個供做預覽用，命名爲 TinyURLPreview.aspx，以 320*200 的尺寸顯示選取網址所轉出的 Tiny URL，並且爲了趣味起見，我們再加上 websnapr 提供的網頁縮圖顯示(http://www.webnapr.com)。至於點選 Activity 後顯示的全尺寸操作網頁，理論上借用 Tiny URL 網站本身的查詢結果網頁即可，但基於安全考量，OpenService XML 中的預覽與執行等<activityAction>宣告的 action URL 必須與宣告的首頁(homepageUrl)同屬一個網域，所以預覽與執行無法放在不同的網站。這裡打算採行的解決方法是做一個簡單的轉接網頁，接受使用者傳來的參數，原封不動地轉 POST 給 http://tinyurl.com/create.php。當然，這層額外的手續是因爲我們"借用"了 Tiny URL 服務進行展示，如果本身就是服務提供者時，將預覽、執行網頁放在同一個 Web Site 是很自然且合理的設計，不需考量此一限制。

預覽網頁

依據開發指南的說明，預覽網頁的大小不應超過 320x240，超出的部分會被裁掉，且建議不要動用捲軸，以可以直接檢視爲原則。我們做一個很簡單的預覽網頁(如圖 4)，顯示觸發來源(ltrType: Document、Link 或 Selection)、連結標題(ltrTitle: Document 或 Link 時適用)、原始網址(ltrUrl 及轉換後的 Tiny URL(ltrTinyUrl)以及網頁縮圖(imgSnapshot)。

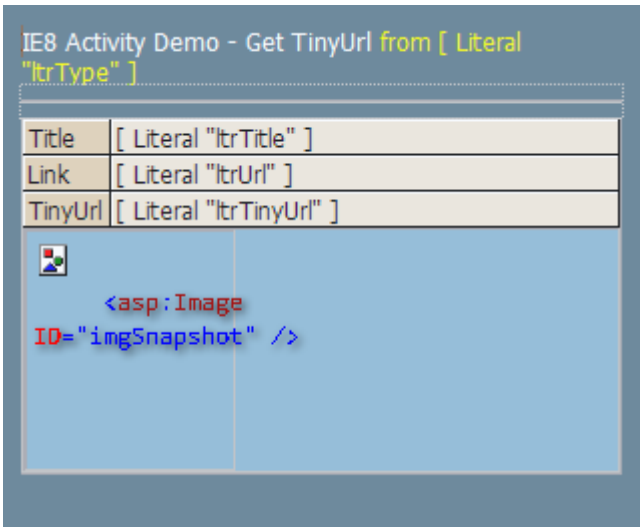

圖 4 預覽網頁的配置

後端的程式碼很簡單，我們透過呼叫 http://tinyurl.com/api-create.php?url= 取得 Tiny URL，至於網頁縮圖的部分則透過 http://images.websnapr.com/?size=s&nocache=36&url= 的方式顯示 websnapr 提供的網頁縮圖。程式碼如程式 1 所示:

程式 1

```
protected void Page_Load(object sender, EventArgs e)
{
    string url = Request.QueryString["u"];
    string tinyUrl = "";
    if (!string.IsNullOrEmpty(url))
        tinyUrl = getTinyUrl(url);
    //記得加上 HtmlEncode 防止 XSS 攻擊
    ltrUrl.Text = HttpUtility.HtmlEncode(url);
    ltrTinyUrl.Text = HttpUtility.HtmlEncode(tinyUrl);
    ltrTitle.Text =
        HttpUtility.HtmlEncode(Request["t"] ?? "");
    //區別 Activity 的觸發來源
    switch (Request["o"] ?? "")
```

```
    {
        case "s": ltrType.Text = "Selection"; break;
        case "d": ltrType.Text = "Document"; break;
        case "l": ltrType.Text = "Link"; break;
    }
    if (tinyUrl.Length > 0)
        imgSnapshot.ImageUrl =
            "http://images.websnapr.com/?size=s&nocache=36&url="
            + HttpUtility.UrlEncode(url);
    else
        imgSnapshot.Visible = false;
}

public string getTinyUrl(string url)
{
    //只接受以 http://起首的 URL
    if (!Regex.IsMatch(url, "(?i)^http://")) return "";
    //利用 TinyUrl API 取得 TinyUrl
    HttpWebRequest req = (HttpWebRequest)WebRequest.Create(
        "http://tinyurl.com/api-create.php?url=" + url);
    StreamReader sr = new StreamReader(
        req.GetResponse().GetResponseStream());
    return sr.ReadToEnd();
}
```

除了滑鼠移至 Activity 項目上可以直接預覽外，點選 Activity 可另外開啓頁籤以全網頁方式操作。原本的構想是模擬輸入網址按下”Make TinyURL!”所送出的 POST 請求直接連至 http://tinyurl.com 網站看結果。基於安全上限制，預覽與執行網頁必須隸屬同一個網域，這裡用了點技巧，我們開發一個 GotoTinyUrl.aspx，用 POST 方式接入 IE8 傳來的參數，並將其重新 Submit 轉接至 Tiny URL 正式網站。程式碼如程式 2:

程式 2

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(@"
<html><body>
Redirecting...
<form action=""http://tinyurl.com/create.php"" method=""POST""
id=""frm"" >
<input type=""hidden"" name=""url"" value=""" +
HttpUtility.HtmlEncode (Request["url"] ?? "") + @"""/>
```

```
</form>
<script type=""text/javascript"">
document.getElementById(""frm"").submit();
</script>
</body></html>");
    Response.End();
}
```

OpenService Activity XML

現在預覽與執行網頁都完成了，我們可以直接輸入 URL QueryString 看看它的效果。接下來的重點工作便是宣告一份 OpenService Activity XML 將網頁變成 Activity 服務，XML 文件的內容如程式 3。

程式 3

```
<?xml version="1.0" encoding="UTF-8"?>
<openServiceDescription
   xmlns="http://www.microsoft.com/schemas/openservicedescription/1.0">
  <homepageUrl>http://www.darkthread.net/TinyUrlActivity</homepageUrl>
  <display>
    <name>Get TinyUrl Link</name>
    <icon>http://www.darkthread.net/TinyUrlActivity/web.ico</icon>
  </display>
  <activity category="Url">
    <activityAction context="document" >
      <preview action="
http://www.darkthread.net/TinyUrlActivity/TinyUrlPreview.aspx">
        <parameter name="u" value="{documentUrl}" />
        <parameter name="t" value="{documentTitle}" />
        <parameter name="o" value="d" />
      </preview>
      <execute action="
http://www.darkthread.net/TinyUrlActivity/GoToTinyUrl.aspx"
method="post">
        <parameter name="url" value="{documentUrl}"/>
      </execute>
    </activityAction>
    <activityAction context="link" >
      <preview
action="http://www.darkthread.net/TinyUrlActivity/TinyUrlPreview.aspx">
        <parameter name="u" value="{link}" />
```

```
      <parameter name="t" value="{linkText}" />
      <parameter name="o" value="l" />
    </preview>
    <execute
action="http://www.darkthread.net/TinyUrlActivity/GoToTinyUrl.aspx"
method="post">
      <parameter name="url" value="{link}"/>
    </execute>
  </activityAction>
  <activityAction context="selection" >
    <preview
action="http://www.darkthread.net/TinyUrlActivity/TinyUrlPreview.aspx">
      <parameter name="u" value="{selection}" />
      <parameter name="o" value="s" />
    </preview>
    <execute
action="http://www.darkthread.net/TinyUrlActivity/GoToTinyUrl.aspx"
method="post">
      <parameter name="url" value="{selection}"/>
    </execute>
  </activityAction>
 </activity>
</openServiceDescription>
```

OpenService XML 的規格細節很多，MSDN 文件(http://tinyurl.com/448dn4)裡已有詳細的說明，在此不多贅述。只補充幾個重點提示:

1. <activity>的 category 屬性可爲 Activity 設定分類，除了既有的 map, blog, define, add, translate 類別外，開發人員也可以自訂，這裡我們自訂了一個新分類--Url。
2. 由網頁中取得 URL的來源有三種: 網頁本身的URL、網頁中出現的連結<a href="..">、使用者選取的 URL 文字。在此宣告了三個<activityAction>，context 對象依序是 document、link 跟 selection，分別適用於前述的三種場合。
3. {documentUrl}, {documentTitle}, {link}, {linkText}, {selection}在實際執行時，會分別被取代爲網頁 URL、網頁標題、連結 URL、連結文字顯示、使用者選取的文字。
4. 每個<activityAction>都有<preview>及<execute>宣告預覽與執行時的網址，但注意必須與 homepageUrl 同屬一個網域。
5. method 屬性可以指定使用 GET 或 POST 方式傳遞參數給預覽或執行網頁。
6. method="get"時，也可以將參數直接寫在 action 中，例如: action="http://web/activity.aspx?u={documentUrl}&t={documentTitle}&o=d"。

部署測試

將 GetTinyUrl.xml 及預覽、執行網頁都部署到網站後，下一步便是提供使用者安裝 Activity 的管道。IE8 提供了一組新的 API，可以協助安裝 Activity。

爲此，我們寫一個安裝網頁，先以 window.external.IsServiceInstalled 檢測使用者是否已安裝該 Activity。若尚未安裝，則提供一個按鈕，可觸發 window.external.AddService 進行安裝，安裝網頁的片段如程式 4。

另外，這裡還用了點小技巧，利用 window.XDomainRequest 物件是否存在檢測使用者的瀏覽器版本是否爲 IE8，當非 IE8 時，就停用安裝功能並顯示 IE8 Only 的提示。由於 window.external.AddService 是非同步作業，無法預測使用者裝好 Activity 的時機，此處用了 window.setInterval 的方式持續監控是否已安裝完成，以利顯示切換。

程式 4

```
<body onload="onLoadJob();" style="text-align:left;">
<script type="text/javascript">
var xmlUrl = null;
function onLoadJob() {
    if (window.ActiveXObject &&
        window.XDomainRequest)
    {
        //檢測版本爲 IE8
        document.getElementById("spnIE8Only")
        .style.display="none";
        var url = location.href;
        xmlUrl = url.substr(0, url.lastIndexOf("/") + 1)
                + "GetTinyUrl.xml";
        //偵測是否已安裝
        if (window.external.IsServiceInstalled(xmlUrl, "Url"))
            //顯示已安裝的資訊
            document.getElementById("spnInstalled")
            .style.display="block";
        else
            //顯示安裝鈕
            document.getElementById("spnInstallButton")
            .style.display="block";
    }
}
function installActivity() {
    window.external.AddService(xmlUrl);
    //AddService 爲非同步作業，故使用 setInterval 觀察是否安裝完成
    window.setInterval("if
```

```
(window.external.IsServiceInstalled(xmlUrl, 'Url'))
location.reload();", 2000);}
</script>
<form id="form1" runat="server">
<div>
<div style="margin:10px; color:Yellow;">
<span id="spnInstalled" style="display:none;">
    You have installed Darkthread GetTinyUrl Activity, thank you!
</span>
<span id="spnInstallButton" style="display:none;">
    <input type="button" onclick="installActivity();"
     value="Install Darkthread GetTinyUrl Activity" />
</span>
<span id="spnIE8Only">* This Activity demo is for IE8 only *</span>
</div>
```

將安裝網頁與 XML 放上網站後，便可以開啓 IE8 進行測試。連上安裝網頁，由於尚未安裝過該 Activity，會出現我們設計的安裝鈕；按下安裝鈕後，會啓動 IE8 的 Activity Provider 新增界面(如圖 5)，我們注意到有個 Make this my default provider for this type of activity 的選項。設定爲預設提供者的 Activity，會列在 Activity 的第一層清單中，否則需叫出 More activities 清單才看得到，每個分類只能有一個預設提供者。

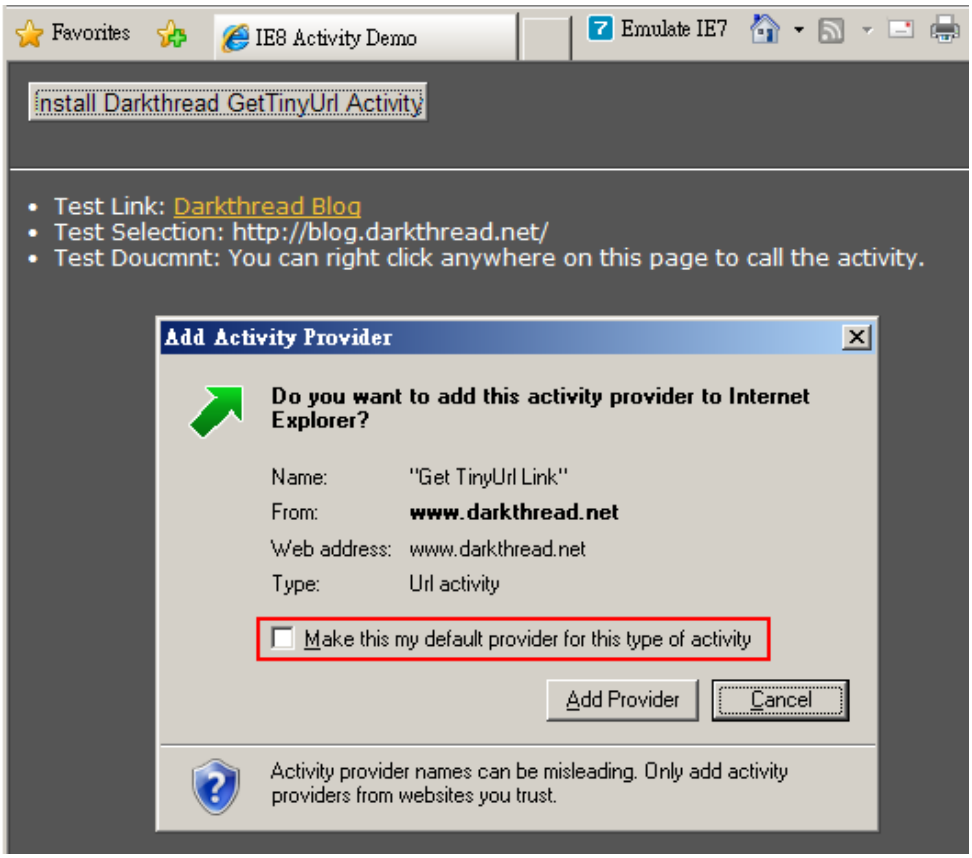

圖 5 Activity Provider 的安裝畫面

安裝完成，我們可以試著分別在網頁空白處、連結元素上或選取文字後按右鍵叫出內容選單，可看到我們製作的 Get TinyUrl Link 名列 Activity 清單中，將滑鼠停在其上方，即可看到 TinyURL 及網頁縮圖檢視。(如圖 6)

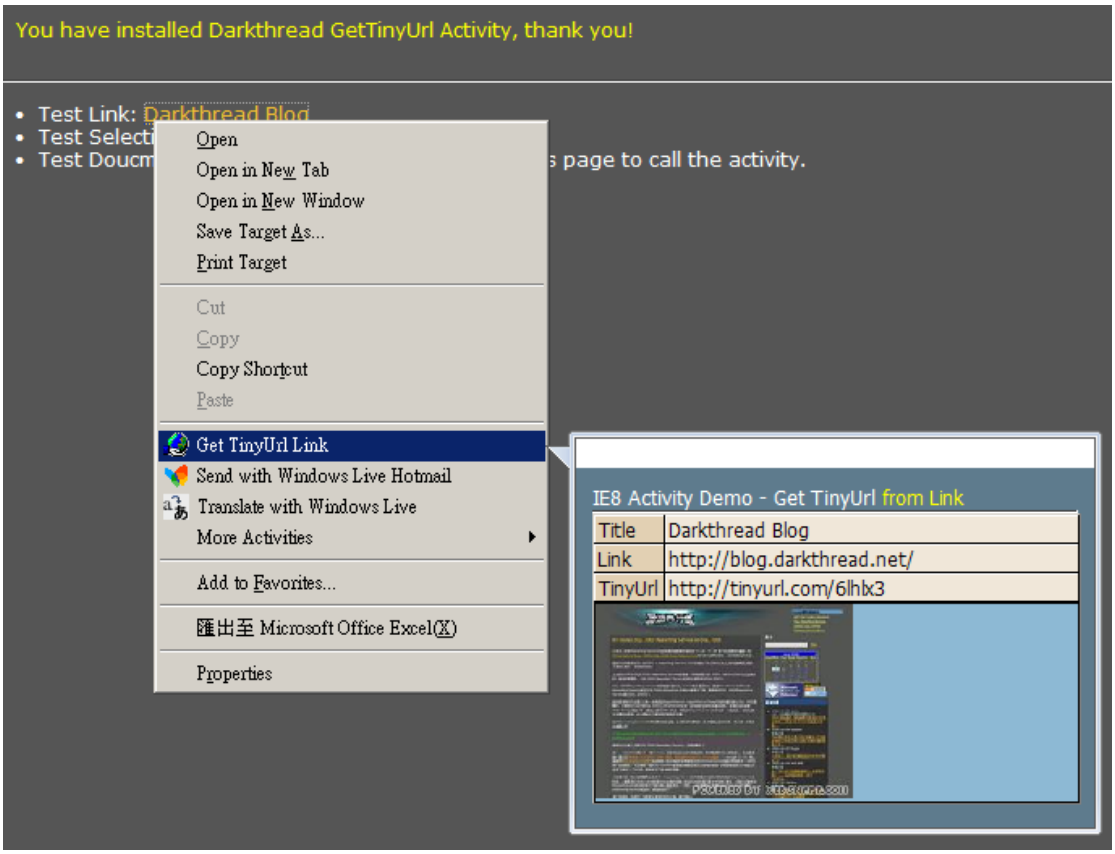

圖 6 Get TinyURL Link Activity 的操作畫面

Activities for Firefox

Activity 的方便性，顯然引起了 Firefox 開發社群的注意，很快地便有人推出在 Firefox 上使用 Activity 的 Firefox Extension(http://tinyurl.com/2lab74)。配合 Firefox 3，甚至連預覽功能都可原味重現(圖 7)。看來，Activity 已可視爲跨平台的技術，對於考量是否要投入 Activity 開發的廠商，無疑是一項有利因素。

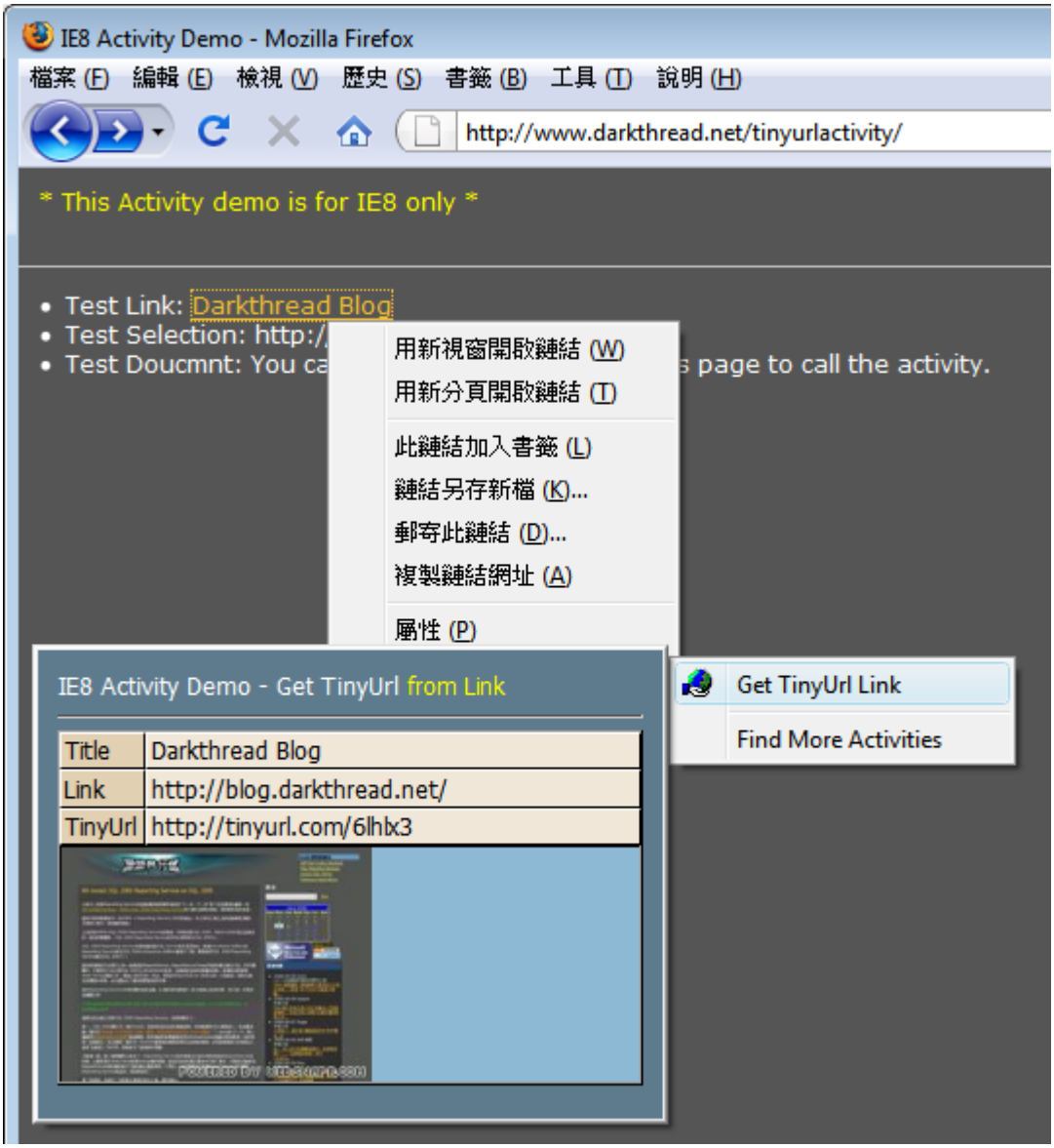

圖 7 Activity 在 Firefox 3+Activity Extension 上的呈現效果

結論

透過本文的示範，我們可以發現 Activity 的運作原理十分簡單，不需額外安裝擴充程式、也沒有太多安全疑慮，網頁開發者幾乎不費吹灰之力就可將既有的服務包裝成 Activity，再加上 Firefox 也積極投入支援，這塊新市場，勢必成爲網路內容提供廠商積極拓展通路的新藍海，讀者們也可以動動腦，想出各式有趣的應用。