Programando WebParts – Parte 4
WebParts con código de cliente

Autor : Gustavo Velez
Para : www.gavd.net/servers/
Fecha : 02-19-2005
Versión : 1.0.0

Tal vez la actividad mas común cuando hablamos de programar para SharePoint, es hacer WebParts ("Elementos Web", según la traducción de Microsoft). WebParts nos permiten añadir funcionalidad y personalizar una instalación de SharePoint/WSS de una manera fácil y rápida.
Esta serie de artículos describen las diferentes posibilidades en cuanto a programación de WebParts:
Parte 1 – Programando una WebPart básica
Parte 2 – Propiedades y Panel de Herramientas
Parte 3 – Distribución, archivos dwp y gallerías de WebParts
Parte 4 – WebParts con código de cliente
Parte 5 – Interconectando WebParts
Parte 6 – WebPart asincrónicas

Requisitos

- WSS o SharePoint instalados y funcionando
- Conocimientos de cómo usar WebParts in SharePoint
- Visual Studio DotNet 2003
- Plantillas para la creación de WebParts en Visual Studio. Se pueden bajar del sitio de Microsoft (http://www.microsoft.com/downloads/details.aspx?FamilyID=14D5D92F-C3A6-407C-AAD7-B8C41A4991BE&displaylang=en)
- Conocimientos de programación y alguna experiencia sobre como usar Visual Studio. Todo el código en esta serie de artículos es escrito en C#, pero es fácilmente adaptable a VB.
- Conocimientos básicos del Modelo de Objetos (API) de SPS

El código de ejemplo de este articulo continua con el código que se puede encontrar en la primera y segunda parte de la serie.

Maneras de incluir código cliente en una WebPart

El principal problema de código cliente en una WebPart es que si la WebPart se usa más de una vez en una página, el código cliente se repetirá en la página generada, produciendo los consecuentes conflictos. Más adelante se explicara como evitar esta situación.
Hay dos maneras de incluir código de cliente en una WebPart: en un archivo separado del código fuente ("Script vinculado") o incluyendo el código de script en el código fuente de la WebPart ("Script incluido").

Código Vinculado
El código cliente se puede incluir en un archivo localizado en el servidor que ejecuta SharePoint. Las ventajas de este sistema son que es más eficiente si el código es utilizado por otras WebParts y que es más fácil de darle mantenimiento pues no hay que recompilar la WebPart si es necesario cambiar el código cliente.
(Para saber más sobre la construcción básica de esta WebPart, vea las partes uno y dos de esta serie de artículos)

1 - Para hacer la vinculación es necesario tener primero el código en un archivo separado. Haga un nuevo archivo "MyCodigoCliente.js" con los siguientes renglones:

```
function HolaDesdeArchivo()
{
        alert('Hola desde un archivo');
}
```

2 – Si el ensamblado esta localizado en el GAC, localice el directorio

C:\Archivos de programa\Archivos comunes\Microsoft Shared\web server extensions\wpresources

Y cree un nuevo directorio con el nombre del ensamblado. En el caso del ejemplo, será:

C:\Archivos de programa\Archivos comunes\Microsoft Shared\web server extensions\wpresources\MyWebPart

Ahora es necesario crear otro directorio bajo el directorio anterior con la forma:

[versión ensamblado]_[cultura]_[clave publica]

Si la cultura es la cultura por defecto ("neutral") no se debe añadir. En el ejemplo, la ruta será:

C:\Archivos de programa\Archivos comunes\Microsoft Shared\web server extensions\wpresources\MyWebPart\1.0.0.0_ d1310637471ff006

3 – Si el ensamblado esta localizado en el directorio Bin del directorio virtual de IIS, simplemente copie el archivo en el directorio

C:\inetpub\wwwroot\wpresources\[nombre del ensamblado]

En el caso del ejemplo de este artículo será:

C:\inetpub\wwwroot\wpresources\MyWebPart

4 – Incluya los siguientes renglones en la sección de declaración de variables del código de la WebPart

```
private const string ScriptFileNombre = "MyCodigoCliente.js";
private const string IncluyeScriptClaveVin = "myIncluyeScript";
private const string IncluyeScriptFormatoVin = @"<script language=""{0}"" src=""{1}{2}""></script>";
```

5 – Añada un manejador de eventos para cargar el script en el evento "PreRender"

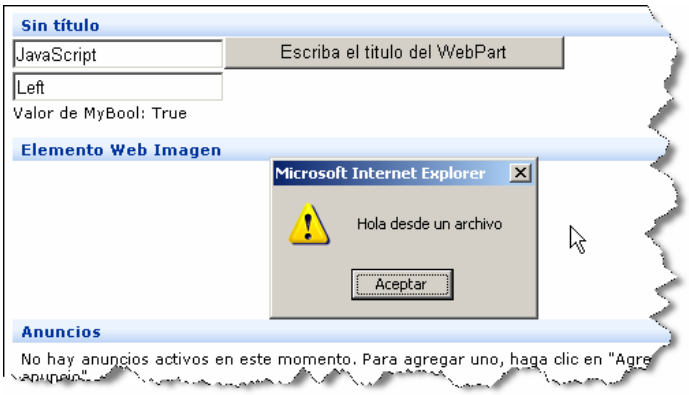
```
        public WebPart1()
        {
                this.PreRender += new EventHandler(WebPart1_PreRender);
        }
```

6 – Añada el evento "PreRender"

```
private void WebPart1_PreRender(object sender , System.EventArgs e )
{
        if (!Page.IsClientScriptBlockRegistered(IncluyeScriptClaveVin))
        {
                String strLocalizacion = this.ClassResourcePath + "/";
                string incluyeScript = String.Format(IncluyeScriptFormatoVin, "javascript",
strLocalizacion, ScriptFileNombre);
                Page.RegisterClientScriptBlock(IncluyeScriptClaveVin, incluyeScript);
        }
}
```

7 – Añada el siguiente renglón en el método “CreateChildControls()” para ejecutar el JavaScript cuando se aprieta el botón:

```
        objButton.Attributes.Add("OnClick","javascript:HolaDesdeArchivo();");
```



Código Incluido

El código cliente se puede incluir directamente en el código fuente de la WebPart. La ventaja principal de este sistema es que es más rápido para cargar que el código vinculado, pero con cualquier cambio en el código cliente hay que recompilar la WebPart de nuevo.

1 - Incluya los siguientes renglones en la sección de declaración de variables del código de la WebPart

```
private const string IncluyeScriptClaveInc = "myByeByeIncludeScript";
private const string ScriptFormatoInc =
"<script language=javascript>function HolaDesdeCodigo(){alert('Hola desde el codigo');}</script>";
```

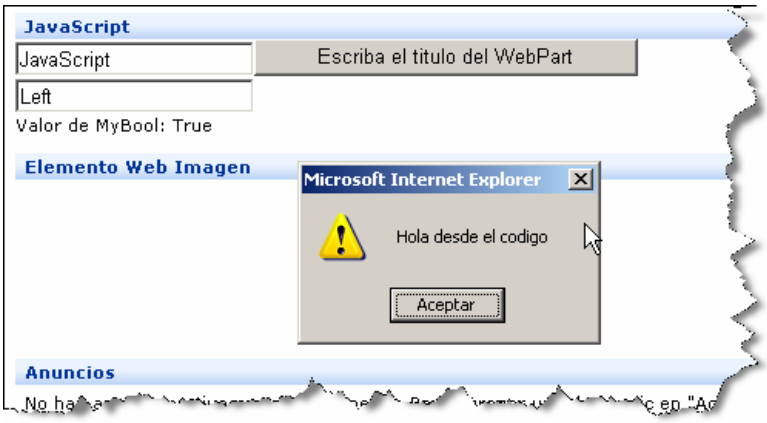
2 – En el evento PreRender, incluya el siguiente código:

```
if(!Page.IsClientScriptBlockRegistered(IncluyeScriptClaveInc))
        Page.RegisterClientScriptBlock(IncluyeScriptClaveInc, ScriptFormatoInc);
```

3 – Para activar el JavaScript con el botón, modifique sus atributos de la siguiente forma:

```
objButton.Attributes.Add("OnClick","javascript:HolaDesdeArchivo();HolaDesdeCodigo();");
```

4 – Ahora, cuando se utilice el botón, primero aparecerá el script vinculado y luego el incluido



Reemplazo de símbolos

SPS\WSS utiliza algunos símbolos para identificar los componentes de una WebPart, controlar el funcionamiento de JavaScripts e identificar componentes de cada página (como la forma, por ejemplo).

Algunas veces es necesario conocer los símbolos para utilizarlos con código cliente. Utilice el método “ReplaceTokens” para ver sus valores. Para verlos en acción, incluya el siguiente código en el evento “RenderWebPart” del ejemplo:

```
output.Write("<br><br><b>Reemplazo de simbolos</b>");
output.Write("<br><b>Usuario: </b>" + this.ReplaceTokens("_LogonUser_"));
output.Write("<br><b>Ruta Clase Recursos: </b>" + this.ReplaceTokens("_WPR_"));
output.Write("<br><b>Cualificador: </b>" + this.ReplaceTokens("_WPQ_"));
output.Write("<br><b>WebPart ID: </b>" + this.ReplaceTokens("_WPID_"));
```