Nº15 Abril 2013

COMPARTIMOSS

Revista especializada en Tecnologías SharePoint

ENTREVISTA

LECCIONES APRENDIDAS DE WF

MANEJADORES DE EVENTOS REMOTOS

EL MOBILE OBJECT MODEL DE SHAREPOINT 2013

Nº15 abril 2013

COMPARTIMOSS

Revista especializada en Tecnologías SharePoint

STAFF

DIRECCIÓN GENERAL

- Gustavo Vélez
- Juan Carlos González
- Fabián Imaz

Contactate con nosotros

gustavo@gavd.net
jgonzalez@gruposodercan.es
fabiani@siderys.com.uy

BLOGS
http://www.gavd.net
http://geeks.ms/blogs/ciin
http://blog.siderys.com/

FACEBOOK
http://www.facebook.com/group.php?gid=128911147140492

VISÍTENOS:
http://www.compartimoss.com

Contenidos

03 **Editorial**
En esta nueva etapa de CompartiMOSS...

04. ASP.NET MVC para SharePointeros (I)

08. Integración con Mapas y Client-side Rendering en SharePoint 2013

13. Un nuevo CSOM
el Mobile Object Model de SharePoint 2013

16 **Manejadores de eventos remotos**
SharePoint 2013 introduce el concepto de manejadores de eventos remotos como un mecanismo para notificar a sistemas externos.

20. Entrevista
a Pablo Pussacq Laborde

22. Paso a paso:
Metodología de Customización de Data Views con SharePoint Designer 2010.

24. El desafío empresarial de la Gestión de Procesos
SharePoint Team Services.

26 **RunWithElevatedPrivileges**
Workaround para permitir actualizar un término

28. ¿Cómo se hizo CompartiMOSS?
Sitios de publicación con SharePoint 2013

32. SharePoint como Sistema Colaborativo

34. Governance Q&A con Jeremy Thake

36. Cargar documentos
En SharePoint utilizando Servicios Web

40. Workflow en Project Server 2010

43. Notificaciones Push a APPS de Windows Phone

03 Editorial

En esta nueva etapa de **CompartiMOSS**, iniciada con el número 14, y con la nueva versión de SharePoint completamente asentada en el mercado y disponible de forma global desde el pasado mes de enero, les presentamos la edición 15 de la revista, cargada de artículos de calidad en los que nuestros autores y colaboradores han realizado un excelente trabajo transmitiendo sus conocimientos y su experiencia en los distintos temas tratados.

El tirón y popularidad de la revista sigue creciendo gracias a una gran base de autores qué número a número aumenta, el trabajo coordinado de la dirección y la continua evolución de nuestra web (www.compartimoss.com) realizada por nuestros compañeros Alberto Díaz y Santiago Porras, sin olvidar el soporte de nuestros patrocinadores, por supuesto. Como siempre, en este nuevo número se ha intentado combinar artículos de carácter técnico (sobre todo en torno a SharePoint 2013) con artículos de negocio, de buenas prácticas y de casos de la vida real que exponen las posibilidades y alcance qué tiene una de las plataformas estrella de Microsoft.

Esperamos que disfruten de este número y que los artículos contenidos en el mismo sean de su agrado e interés.

Abril, 2013

JUAN CARLOS GONZÁLEZ
jgonzalez@gruposodercan.es

GUSTAVO VÉLEZ
gustavo@gavd.net

FABIÁN IMAZ
fabiani@siderys.com.uy

04 ASP.NET MVC para SharePointeros (I)

Resumen

En este artículo veremos las bases de la programación de aplicaciones web con ASP.NET MVC, con un ejemplo sencillo de conexión a SharePoint.

Artículo

Los programadores SharePoint estamos tan acostumbrados a programar con ASP.NET Web Forms que no nos solemos preocupar por aprender otras tecnologías de programación ya que no se pueden aplicar directamente a SharePoint. Sin embargo, desde la versión 2010 de SharePoint, con el modelo de objetos de cliente (CSOM) podemos acceder a SharePoint desde cualquier aplicación .NET, incluyendo también el framework ASP.NET MVC, que vamos a introducir en este artículo.

Lo básico de MVC

En el mundo de ASP.NET Web Forms la base de todo es la página (instancia de clase Page). La página está mapeada físicamente a un fichero con extensión ASPX y dispara una serie de eventos (Init, Load, PreRender, Render etc) según vamos progresando de la petición HTTP a la respuesta HTML al cliente. Podemos decir que en cierto modo Web Forms implementa el patrón "Page Controller" .

En el mundo MVC esto ya no es así. La responsabilidad de servir la respuesta a una petición está dividida entre tres componentes que dan nombre al patrón de diseño MVC: modelo, vista y controlador.

El modelo son nuestras clases de negocio junto con su lógica de negocio. La vista es la renderización HTML de la respuesta al cliente mientras que el controlador es la lógica de la aplicación web que procesa el modelo y actualiza la vista. En el mundo Web Forms, la página hacía de vista y de controlador a la vez. Ahora estas dos cosas están bien separadas.

Las peticiones, rutas, acciones y otras cosas de nombre extraño

Vamos a ver como procesa ASP.NET MVC una petición y lo vamos a comparar con Web Forms para ver las diferencias. Supongamos que queremos ver un producto cuyo ID es 5. En Web Forms haríamos una página DisplayProduct.aspx y recogeríamos un parámetro desde Request.QueryString con el nombre ID. Es decir, la URL de la petición sería DisplayProduct.aspx?ID=5.

En MVC, la URL de la petición no apuntará a ningún fichero físico sino que nos indicara el controlador (de todos los controladores que hay), la acción (dentro de ese controlador) y posiblemente algún parámetro de esa acción. Una URL probable sería /Products/Display/5, indicando el controlador Products, acción Display con el parámetro 5.

Seguro que ahora os estáis preguntando como sabe MVC a que controlador tiene que enviar la petición y que vista se tiene que mostrar. Resulta que MVC por defecto interpreta la URL en el formato "controlador/acción/parámetro" y de esta manera la URL /Products/Display/5 equivale al invocar ProductsController.Display(5). ¿Y la vista? MVC la busca dentro de la subcarpeta Products de las vistas y buscará la que tiene como nombre Display. Como podéis ver, se favorece el uso de convenciones comunes de nombres para ahorrar código innecesario de infraestructura.

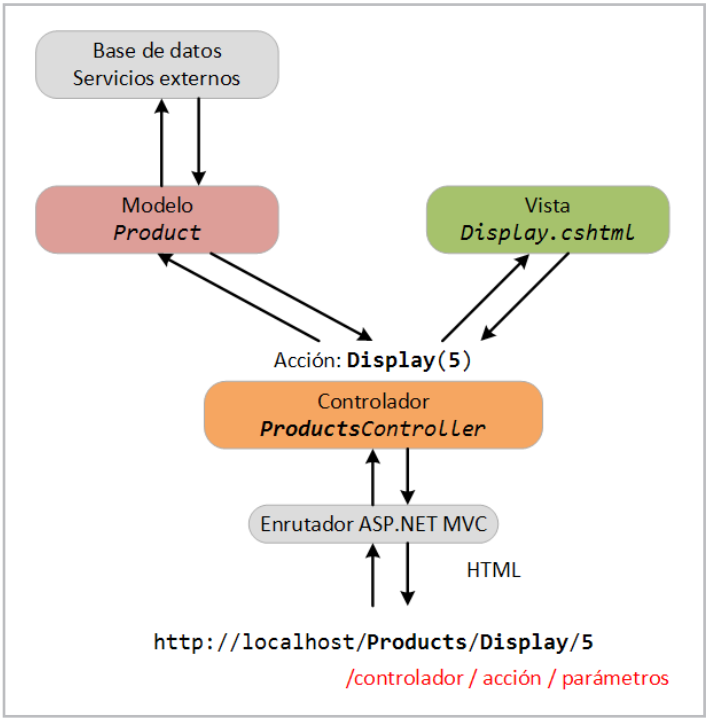


Imagen 1.- El esquema de funcionamiento de una petición MVC.

Una acción en el controlador será un método que devuelve

un resultado, que casi siempre será una vista. La acción puede recibir parámetros (sencillos o incluso complejos) y puede pasarle los parámetros a la vista. Lo más habitual es que la acción realice alguna comprobación de validez de los parámetros y que use otras clases y servicios para llamar a la lógica puramente de negocio. Se suele utilizar inyección de dependencias para desacoplar aún más los controladores (lógica de la capa web de presentación) de la funcionalidad de negocio (capa de dominio o de negocio).

Las vistas en ASP.NET MVC se pueden hacer en Web Forms (ficheros ASPX) o con la nueva tecnología llamada Razor. Se recomienda Razor por ser mucho más sencillo y rápido. Razor tiene una sintaxis más minimalista y cuesta un cierto tiempo acostumbrarse a él. En contraprestación, se corresponde más al HTML final.

```
@model MvcHelloWorld.Models.Product
@{
    ViewBag.Title = "Display";
}
<h2>Display</h2>
<p>@Model.Name <em>(@Model.ProductId)</em></p>
```

Fijaos que Razor usa el prefijo @ para introducir su marcado de servidor. En este ejemplo le estamos diciendo a la vista que va a usar un tipo de datos como modelo (la clase MvcHelloWorld.Models.Product) y luego en el contenido de la página se usan las propiedades de ese modelo mediante la sintaxis @Model.

Hello, world en MVC

Para acabar de sentar el concepto de una aplicación MVC, vamos a hacer un ejemplo sencillo: una aplicación que permite ver la lista de productos y el detalle del producto. Para simplificar esta primera toma de contacto con MVC, vamos a usar la lista de productos fija implementada en la clase DBHelper. Más adelante usaremos la API de SharePoint para obtener la lista desde un sitio de SharePoint.
public class DBHelper

```
public class DBHelper
{
    public static Product GetProduct(int
productId)
    {
        var product = GetProducts().Where(x =>
x.ProductId == productId).FirstOrDefault();
        return product;
    }

    public static IEnumerable<Product>
GetProducts()
    {
        List<Product> result = new
List<Product>();
```

```
        result.Add(new Product() { Name = "ACME
Gadget", ProductId = 1 });
        result.Add(new Product() { Name = "XYZ
Widget", ProductId = 2 });
        result.Add( new Product() { Name =
"Product ABC", ProductId = 4 });
        result.Add( new Product() { Name = "Foo
DEF", ProductId = 5 });

        return result;
    }
}
```

Abrimos Visual Studio y creamos una aplicación web MVC 3 con la plantilla “Empty Application”. Dejamos las opciones por defecto y nos encontramos con la siguiente estructura de un proyecto MVC.

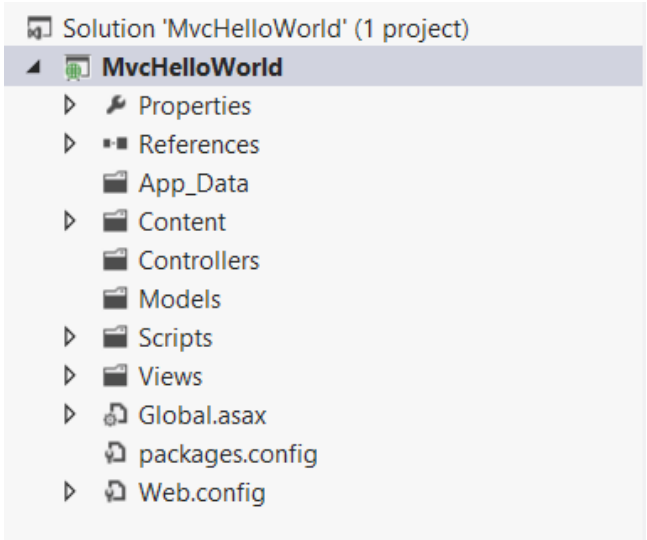


Imagen 2.- Estructura de un proyecto web MVC.

> *“... podemos acceder a SharePoint desde cualquier aplicación .NET, incluyendo también el framework ASP.NET MVC.”*

LAS CARPETAS MÁS IMPORTANTES SON:

- **Content:** contenido estático (HTML, ficheros auxiliares, documentos, etc).
- **Controllers:** aquí irán nuestros controladores.
- **Models:** las clases que componen el modelo de la aplicación irán aquí. En nuestro caso allí tendremos la clase Product y la clase DBHelper que simula una llamada a base de datos.
- **Scripts:** los ficheros de JavaScript para nuestra aplicación. De serie ya tenemos jQuery y Modernizr.
- **Views:** aquí van las vistas, en carpetas que se corresponden con el nombre del controlador.

Ahora vamos a agregar el controlador. Clicamos con el botón derecho en la carpeta Controllers y elegimos Add / Add Controller. En el cuadro de diálogo, escribimos en nombre del controlador (ProductsController) y elegimos la plantilla “Empty Controller”. Obtenemos el código siguiente:

```
public class ProductsController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

La acción por defecto que se va a ejecutar si no especificamos ninguna es la acción Index(). En esta acción vamos a mostrar la lista de los productos, con el enlace a ver el detalle de cada uno de ellos. Para ello hay que cambiar el código ligeramente y también vamos a aprovechar para crear una acción nueva llamada Display:

```
public ActionResult Index()
{
    var products = DBHelper.GetProducts();
    return View(products);
}

public ActionResult Display(int id)
{
    var product = DBHelper.GetProduct(id);
    return View(product);
}
```

Todas las acciones devuelven un ActionResult que indica el resultado de una acción. En la cláusula return devolvemos el resultado de una vista, pasándole un objeto (la lista de productos o un solo producto). Pero, ¿cómo sabe MVC que vista tiene que devolver? Otra vez, lo hace siguiendo la nomenclatura de los nombres de métodos y controladores. Ahora no hay ninguna vista asociada pero lo vamos a corregir añadiendo una vista. Para ello, clicamos por el botón derecho sobre el nombre de la acción y elegimos “Add View”. En el diálogo que nos sale elegimos la opción que vincula a una clase del modelo (“Generate strong-typed view”).

Vamos a modificar la vista para que genere una lista de productos:

```
@model
IEnumerable<MvcHelloWorld.Models.Product>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<ul>
    @foreach (var p in Model) {
        <li>
            @Html.ActionLink(p.Name, "Display"
new { id=p.ProductId })
        </li>
    }
</ul>
```

Como podéis ver, ahora la vista está ligada a un IEnumerable de productos (que es lo que le pasamos desde el controller). Además, creamos una lista de elementos que se generan usando una sintaxis de MVC llamada @Html. El método ActionLink nos genera en enlace a una acción, pasándole el nombre a mostrar, el nombre de la acción y un objeto con los parámetros de la acción (en nuestro caso sólo el ID del producto).

Nos falta crear la vista de la acción Display(), que generamos igual que la de Index() y le ponemos el código siguiente:

```
@model MvcHelloWorld.Models.Product

@{
    ViewBag.Title = "Display";
}

<h2>@Model.Name</h2>

<p>
    <em>Product Id: @Model.ProductId</em>
</p>
```

Antes de ejecutar el ejemplo, tenemos que decirle a MVC cuál es el controlador por defecto y para ello editamos el método RegisterRoutes() en el fichero Global.asax para apuntar a nuestro ProductsController en vez al controlador Home que tiene la plantilla de MVC.

```
routes.MapRoute(
    "Default", // Route name
    "{controller}/{action}/{id}", // URL with
parameters
    new { controller = "Products", action =
"Index", id = UrlParameter.Optional }
);
```

¡Ya estamos listos! Nuestra primera aplicación MVC está sólo a un F5 de distancia:

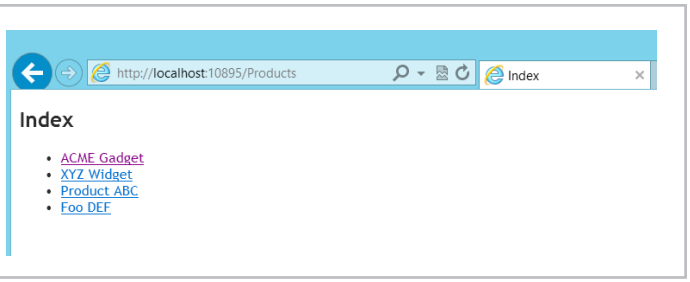


Imagen 3.- Nuestra primera aplicación MVC.

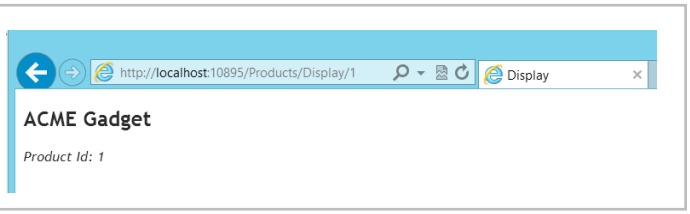


Imagen 4.- El resultado de la acción Display.

Hello, world en MVC y SharePoint

En este momento tenemos la aplicación web en MVC pero no usa SharePoint para nada. Para convertirla en aplicación de SharePoint vamos a agregar las referencias a la librería cliente

de SharePoint para .NET (Microsoft.SharePoint.Client.dll y Microsoft.SharePoint.Client.Runtime.dll).

Tenemos un sitio de SharePoint que tiene la lista de productos. En mi ejemplo la lista está en un Office 365 pero podría estar perfectamente en cualquier SharePoint 2010 o 2013.

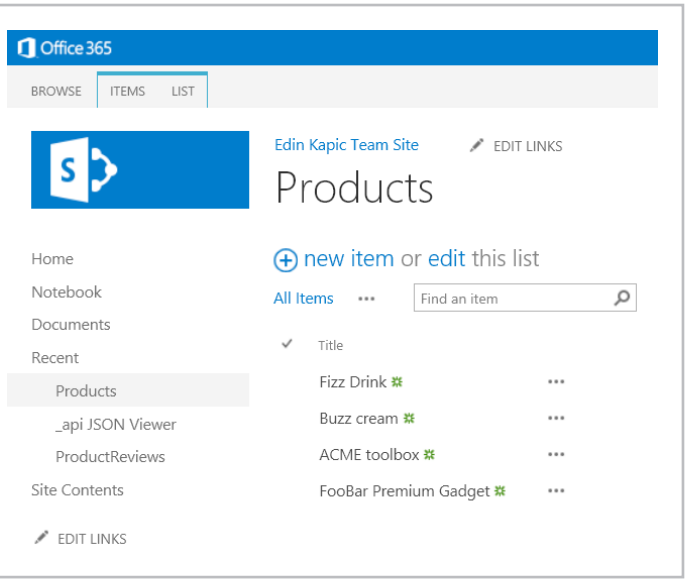


Imagen 5.- La lista de productos en SharePoint.

Ahora vamos a cambiar el método GetProducts() del DBHelper para que use SharePoint. Usaremos el modelo de objetos cliente de .NET para SharePoint. Para autenticarnos contra el servidor de SharePoint, tenemos que especificar la propiedad Credentials del ClientContext. Si usamos Office 365 con SharePoint 2013 tenemos unas credenciales específicas encapsuladas en la clase SharePointOnlineCredentials. Si es un SharePoint on-premise, usamos la clase NetworkCredential como siempre. Si tenemos la mala suerte de usar un SharePoint 2010 Online, podemos aprovechar la aportación de Chris Johnson para autenticarnos .

```
public static IEnumerable<Product>
GetProducts()
{
    ClientContext clientContext = new
ClientContext("http://misharepoint.com");
    SecureString password = new SecureString()
    foreach (char c in
"miPassword".ToCharArray())
    {
        password.AppendChar(c);
    }
            clientContext.Credentials = new
NetworkCredential("miUsername", password,
"miDominio");

    List list =
clientContext.Web.Lists.GetByTitle("Products")
    clientContext.Load(list);
    CamlQuery camlQuery = new CamlQuery();
    camlQuery.ViewXml = "<View/>";
    ListItemCollection listItems =
list.GetItems(camlQuery);
    clientContext.Load(listItems, items =>
items.Include( item=>item["Title"], item =>
item.Id));
    clientContext.ExecuteQuery();
    List<Product> result = new List<Product>()
    foreach (ListItem listItem in listItems)
    {
        var product = new Product() { ProductI
= listItem.Id, Name =
listItem["Title"].ToString() };
        result.Add(product);
    }
    return result;
}
```

Como podéis ver, la consulta a SharePoint es directa: obtenemos los datos de la lista Products y mapeamos los objetos ListItem a instancias de la clase Product, usando las propiedades Id y Title de SharePoint como las correspondientes a ProductId y Name de nuestro modelo.

Podemos ver que con este pequeño ejemplo ya tenemos nuestra aplicación MVC conectada a SharePoint, pulsando F5 para ejecutarla.

El código fuente de esta aplicación de ejemplo está disponible en la dirección http://sdrv.ms/XRmW0O.

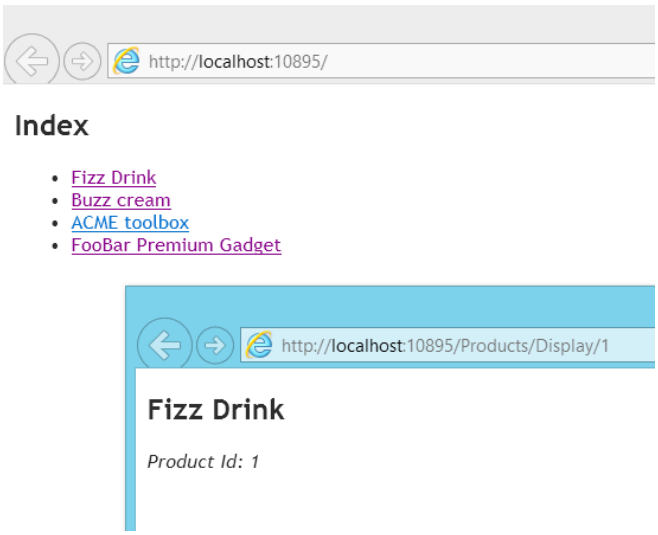


Imagen 6.- Nuestra primera aplicación MVC conectada a SharePoint.

¿Y ahora qué?

Espero que esta pequeña introducción sirva para que los desarrolladores de SharePoint exploren el mundo de ASP.NET MVC y pierdan el miedo a una plataforma a la que no están tan acostumbrados. En las próximas entregas veremos más detalles sobre las aplicaciones MVC aplicadas a SharePoint como por ejemplo como guardar datos, hacer validaciones y usar inyección de dependencia para desacoplar nuestra aplicación y poder realizar pruebas unitarias. ¡Nos vemos en el próximo número!

EDIN KAPIĆ
Arquitecto SharePoint
ekapic@pasiona.com
@ekapic
http://www.pasiona.com

08 Integración con Mapas y Client-side Rendering en SharePoint 2013

Resumen

SharePoint 2013 viene con novedades en lo que a la integración con mapas se refiere. Cuando digo integración me refiero a que en SharePoint vamos a poder crear campos o columnas de tipo coordenadas y de forma automática esto se nos va a representar en la UI de SharePoint 2013 como un mapa. Además, existe un nuevo método para modificar cómo se renderizan o representan los campos en SharePoint 2013, llamado representación del lado del cliente "client-side rendering". Este artículo se divide en dos partes: en la primera comentaremos como se activan y utilizan los campos de tipo GeoLocation y en la segunda parte veremos cómo personalizar la representación gráfica de los campos.

Artículo

SharePoint a lo largo de su historia ha ido añadiendo funcionalidades nuevas, concretamente con cada nueva versión se añaden aquellas funcionalidades que, previo estudio por parte del equipo de producto, creen que tienen más sentido de acuerdo a las necesidades actuales. Y es cierto, que actualmente las tendencias tecnológicas y de mayor adopción son los servicios web, HTML5, JavaScript, AJAX, REST/Odata, Mapas, y todo lo referente a que el procesamiento quede cada vez más del lado de "la nube". Tenemos que tener constancia de la importancia a nivel de desarrollo que están adquiriendo estos conceptos a lo largo de toda la plataforma Microsoft: Windows 8 tiene nueva API basada en JavaScript y permite desarrollar Apps con HTML5, CSS3 y JavaScript; Office y SharePoint 2013 entran en el modelo de "Marketplace" proporcionando las tecnologías citadas como medio de desarrollo de las Apps; y sin dejar de lado que estos lenguajes de desarrollo son multiplataforma y soportado por todos los sistemas operativos del momento. Con todo esto, quiero concientizar a los desarrolladores de la importancia que va adquiriendo lo que hasta ahora era el desarrollo web. Y de la importancia de los servicios externos como puede ser poner en nuestras aplicaciones un mapa de Bing o Google.

Este artículo se va a dividir en dos partes: primero explicaremos a nivel conceptual el nuevo tipo de columna de SharePoint que permite almacenar valores espaciales; y después veremos cómo añadir un nuevo tipo de campo modificando su representación gráfica con HTML y JavaScript, característica nueva en SharePoint 2013, que se ha nombrado "client-side rendering" (a lo que le doy la traducción de representación del lado del cliente).

Geolocalización en SharePoint 2013

En SharePoint 2013 tenemos de forma nativa un tipo de columna (field type) para la geolocalización. Concretamente en un campo de tipo Geolocation podemos introducir coordenadas como valores decimales, tanto para la latitud como longitud, aunque también podemos configurarlo de modo que obtenga las coordenadas del usuario actual directamente desde el navegador (siempre que implemente la API W3C de Geolocalización).

Este tipo de campo nuevo se implementa en SharePoint 2013 en sus dos capas:

- **Almacenamiento de coordenadas.** La forma en la que SharePoint almacena la información de las coordenadas es a través de un tipo de columna nuevo y esto a nivel de SQL Server queda representado con los nuevos tipos de datos geometry, geography y hierarchy ID, que requieren tener instalado SQLSysClrTypes.msi en el servidor de SQL Server. Para descargarlo, podemos usar este paquete: Microsoft® SQL Server® 2008 R2 SP1 Feature Pack [1].
- **Representación gráfica de mapas:** En cuanto a la representación gráfica de los puntos geográficos, SharePoint 2013 utiliza los mapas de Bing, concretamente "Bing Maps Ajax control V7". Además de representarse el mapa en la propiedad en sí, también tenemos un nuevo tipo de vista que es de tipo Mapa.

Esto es la teoría, pero ahora, ¿cómo hacemos para crear este campo en SharePoint?

Lo primero que tenemos que hacer es activar y configurar la clave de Bing Maps para que se enlace nuestro SharePoint con algunos credenciales aptos para usar el API de Bing. Para obtener tu clave de Bing Maps, puedes entrar en esta dirección http://bingmapsportal.com [2], requiere registro con un Live ID y la creación de una clave donde te pedirán la URL de tu aplicación donde vas a utilizar el mapa. Una vez tengamos activado esto a nivel de granja (en SharePoint OnPremise) o a nivel de sitio / colección de sitios (en SharePoint Online 2013), el siguiente paso es crear un campo de tipo Geolocation en cualquier lista que lo requiera. Como en todo, siempre hay varias formas de hacer lo mismo, a continuación veremos cómo hacer todos estos pasos de varias formas tanto para SharePoint 2013 OnPremise como para SharePoint 2013 Online, que como bien sabemos lo lanzaron dentro del paquete Office 365 [3] el pasado 27 de Febrero de 2013.

Configurar la clave de Bing Maps en SharePoint 2013

Tal como comentaba, para habilitar el mapa de Bing en nuestra granja de SharePoint 2013, debemos establecer la "clave" de Bing Maps o bien mediante modelo de objetos cliente o bien mediante PowerShell:

Set-SPBingMapsKey –BingKey "<clave de Bing Maps>"

No obstante, si nuestro SharePoint 2013 es Online, es decir, pertenece a un Office 365, no podremos activar la clave de Bing con este método y tendremos que ejecutar un programa de consola que acceda a la propiedad

```
Uri oUri = new Uri("<url>");
MsOnlineClaimsHelper claimsHelper = new
MsOnlineClaimsHelper(oUri, "<id>", "<password>");
using (ClientContext context = new ClientContext(oUri))
{
        context.ExecutingWebRequest += claimsHelper.
clientContext_ExecutingWebRequest;
        Web web = context.Web;
        web.AllProperties["BING_MAPS_KEY"] = "<clave>";
        web.Update();
        context.ExecuteQuery();
}
```

Sección 1.- Cargando la Key de Bing vía código.

Para ver más detalles sobre configurar la clave de Bing en SharePoint Online, véase el post de Sundar Narasiman [4]. Además con el fin de facilitar la tarea adjunto el código que me he montado para probar esta asignación de clave en SharePoint 2013 Online. Lo podéis descargar desde el vínculo dado en la referencia [5]. (ConfigurarClaveBingSharePointOnline.zip).

Una vez que tenemos configurada la clave el siguiente paso es crear la columna de tipo GeoLocation.

Añadir Campo de tipo GeoLocation en SharePoint 2013

La mala noticia es que no se puede añadir este tipo de campo desde la interfaz gráfica de SharePoint 2013. Por el contrario si se podrá añadir mediante código, por lo que la forma más rápida y común usada por un desarrollador es arremangarse y decir, ¡lo tengo!, esto es una aplicación de consola :). Entonces abrimos Visual Studio 2012 (por supuesto, la última versión) y añadimos este sencillo código en un proyecto de tipo "Console Application":

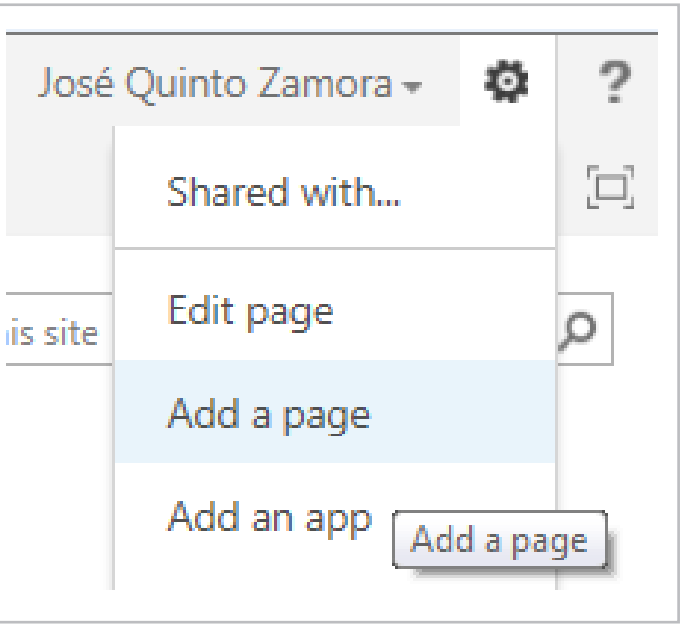
```
class Program
    {
        static void Main(string[] args)
        {
            AddGeolocationField();
                Console.WriteLine("Location field added
successfully");
        }
        private static void AddGeolocationField()
        {
            // Reemplazar la url y los nombres de lista
y columna.
            ClientContext context = new ClientContext("http://
sitio");
            List oList = context.Web.Lists.GetByTitle("Lista");
                oList.Fields.AddFieldAsXml("<Field
Type='Geolocation' DisplayName='Columna'/>", true,
AddFieldOptions.AddToAllContentTypes);
            oList.Update();
            context.ExecuteQuery();
        }
    }
```

Sección 2.- Creando una columna de Geolocation por código.

Pero claro, ahora nos queda la duda de si cada vez que queramos crear un campo de este tipo vamos a tener a mano este programa para ejecutarlo, además de la duda de si este código va a funcionar bien en SharePoint Online o no. Entonces es cuando pasamos a la segunda fase de superación y decidimos hacer la creación de esta columna con modelo de objetos cliente, pero de JavaScript con toda la idea de poner esto en una página web (con un Content Editor o un Code Embed) y ya tenemos una página desde la cual crear este tipo de columnas.

Veamos como ejemplo la creación de esta página en un SharePoint 2013 Online:

- Creamos una nueva página en nuestro sitio:



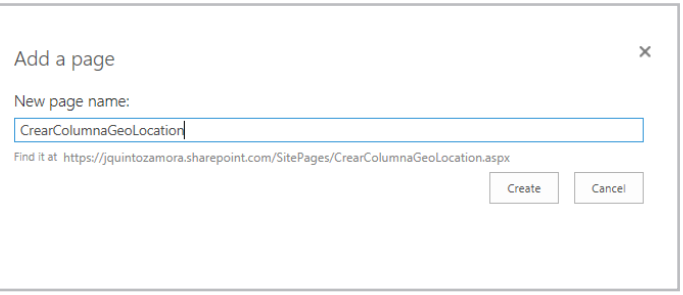


Imagen 1.- Creación de una página desde la interfaz de usuario de SharePoint.

- Editamos la página y le insertamos un código embebido:

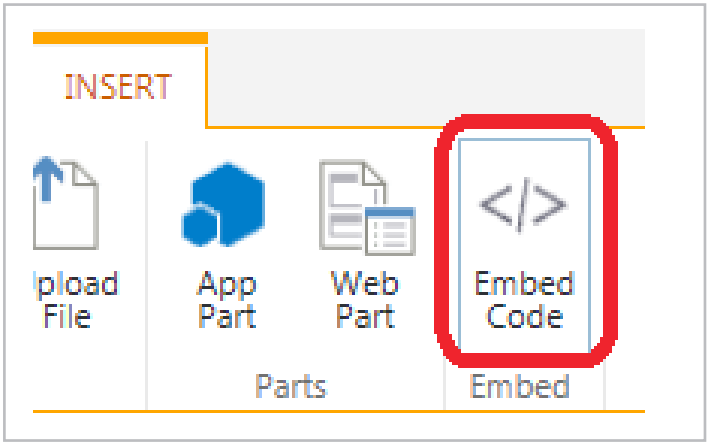


Imagen 2.- Inserción de código embebido en una página.

- Copiamos este código:

```
<script type="text/javascript">
        var contexto;
        function CrearColumna()
        {
                var nombreLista = document.
getElementById('nombreLista').value;
                var nombreColumna = document.
getElementById('nombreColumna').value;
                contexto = new SP.ClientContext.get_
current();
                var lista = contexto.get_web().get_
lists().getByTitle(nombreLista);
                contexto.load(lista);
                this.newField = lista.get_fields().
addFieldAsXml(
                                "<Field Type='Geolocation' DisplayName='" + nombreColumna +
"'/>",
                                true,
                                SP.AddFieldOptions.
defaultValue
                        );
                lista.update();
                contexto.executeQueryAsync(Function.
createDelegate(this, this.CreadaCorrectamente),Function.
createDelegate(this, this.ErrorAlCrear));

        }

        function CreadaCorrectamente()
        {
                alert('Columna GeoLocation creada.');
        }

        function ErrorAlCrear(sender, args)
        {
                alert('Error: ' + args.get_message() +
'\n' + args.get_stackTrace());
        }
</script>

<div>
        <div>Crear columna de tipo GeoLocation:</div>
        <span>Nombre de Lista: </span><input type="text"
value="" id="nombreLista" />
        <span>Nombre de Columna: </span><input type="text"
value="" id="nombreColumna" />
        <input type="button" value="CrearColumna"
onClick="CrearColumna();" />
</div>
```

Sección 3.- Creando una columna del tipo GeoLocation en JavaScript

- Clic en Insert.
- Guardamos la página y ya tenemos algo parecido a esto:

Imagen 3.- Aspecto visual del código insertado.

Desde donde podemos indicar el nombre de la lista y de la columna a crear para hacer lo propio. Debemos de tener en cuenta que existe un límite de 2 columnas de tipo GeoLocation por lista, por lo que si intentamos crear más de dos nos dará un mensaje de error parecido a este:

"There are too many columns of the specified data type. Please delete some other columns first. Note that some column types like numbers and currency use the same data type".

Una vez tenemos el campo añadido en nuestra lista, solamente nos queda saber cómo utilizarlo.

Añadiendo coordenadas a listas de SharePoint 2013 y viéndolas en mapas

Hasta ahora, hemos creado una lista y le hemos añadido una columna de tipo GeoLocation. Ahora si vamos a añadir un nuevo elemento en esta lista veremos como la columna de coordenadas se representa así:

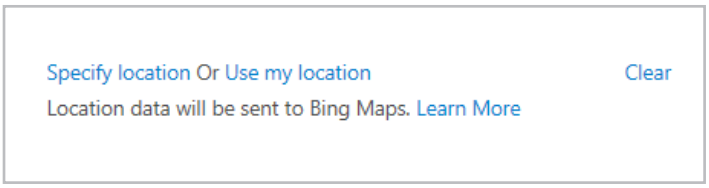


Imagen 4.- Representación de una columna de coordenadas en la lista.

"... poder crear campos o columnas de tipo coordenadas y de forma automática esto se nos va a representar en la UI de SharePoint 2013 como un mapa..."

Desde donde podremos o bien introducir unas coordenadas o bien utilizar las nuestras actuales (que se obtienen desde el navegador). Para especificar nuevas coordenadas podemos utilizar Bing Maps por ejemplo, ya que poniendo el nombre de un lugar nos saca automáticamente sus coordenadas, fijaos cuando busco el pueblo donde vivo (Albatera):

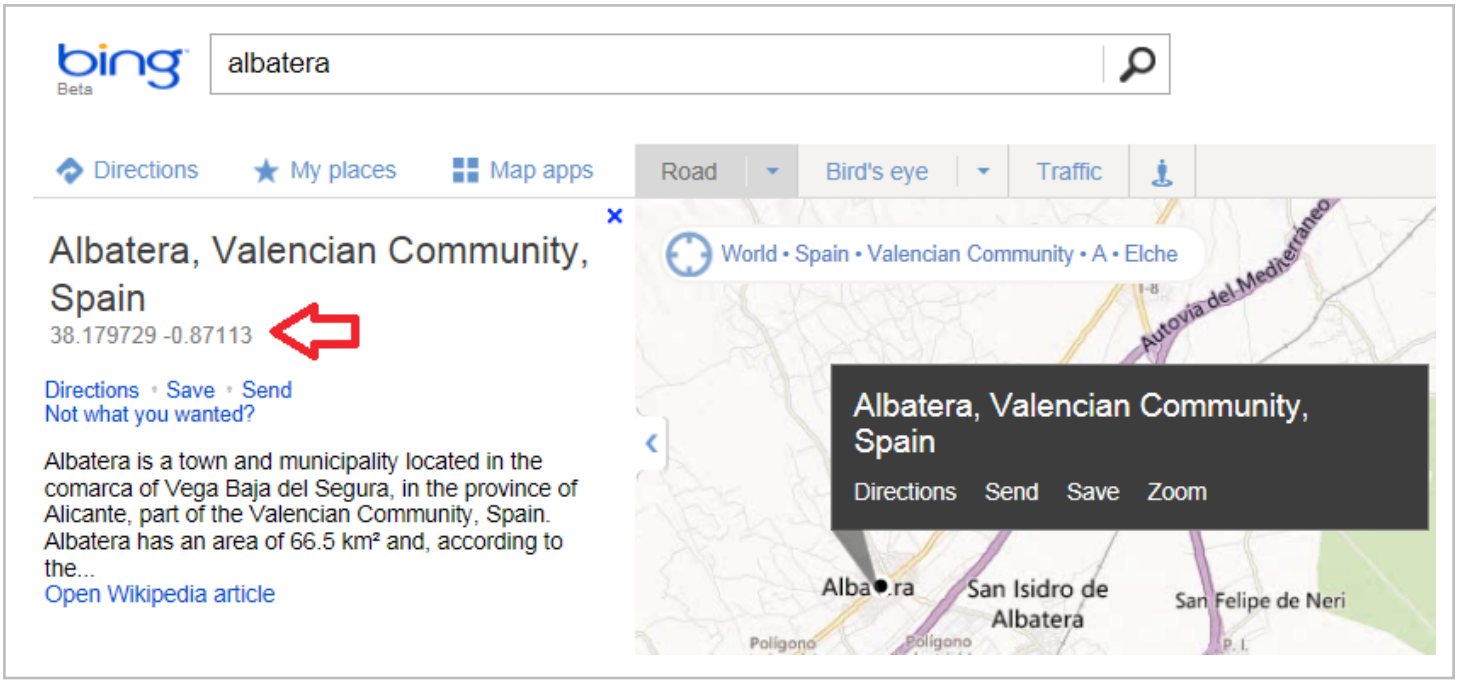


Imagen 5.- Obtención de información de coordenadas con Bing Maps.

Si pongo las coordenadas:

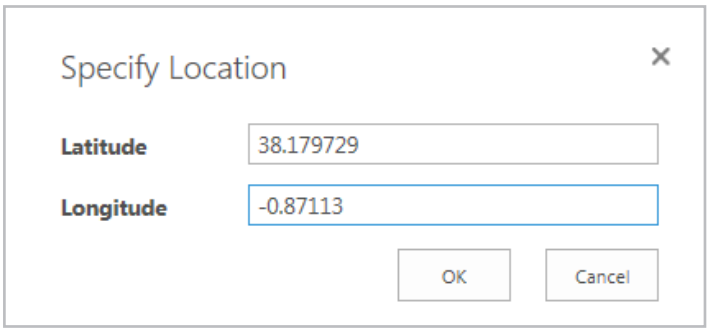


Imagen 6.- Configurando los valores de las coordenadas en el campo de geolocalización.

Vemos como SharePoint automáticamente nos representa este punto en el mapa:

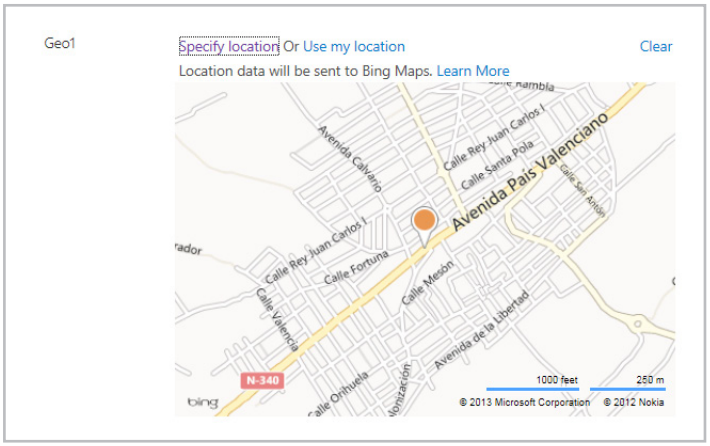


Imagen 7.- Representación de la información de coordenadas en un mapa de Bing Maps.

Además si guardo el elemento y voy a la vista de todos los elementos de la lista, veré como se representa el tipo de columna con un nuevo icono, que cuando lo clicas, se abre el mapa:

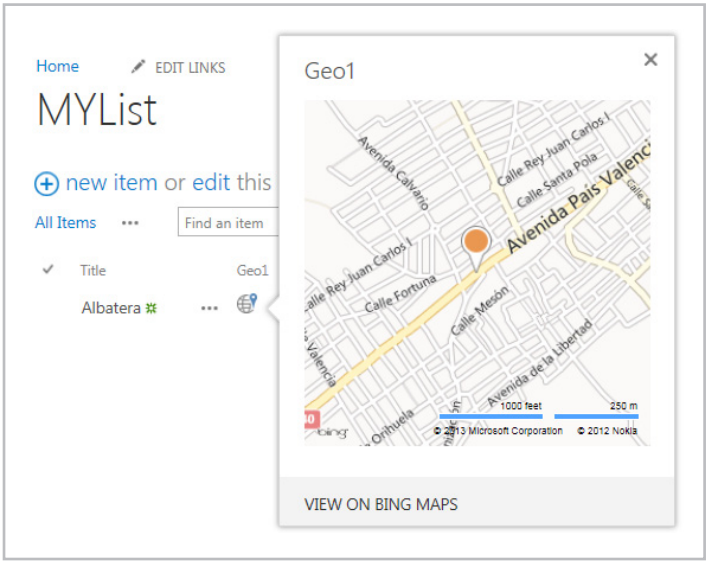


Imagen 8.- Acceso a la información geolocalizada desde la vista de lista.

¿A qué esta guapo?

Pues aún hay más, y es que imaginar el escenario en que tenemos una lista de puntos de interés, y en cada punto de interés decidimos guardar una coordenada. Está claro que tarde o temprano vamos a querer ver un mapa y todos estos puntos representado en él. Pues para esto hay una tipo de vista nueva, que se llama “Mapa” y si creamos esta vista sobre una lista que contiene columna de tipo GeoLocation, nos representará en el mapa todos los puntos:

- Creamos la vista:

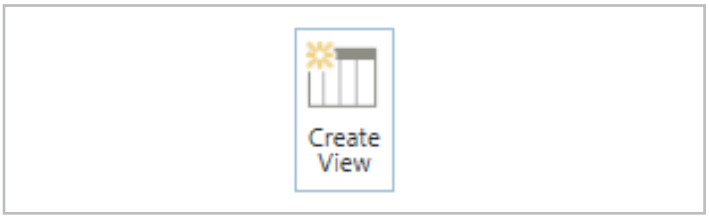


Imagen 9.- Acción para crear una vista de lista en SharePoint.

- De tipo Map View:

Imagen 10.- Opción para crear una vista de tipo “Map View”.

- Y aquí tenemos nuestros puntos de interés:

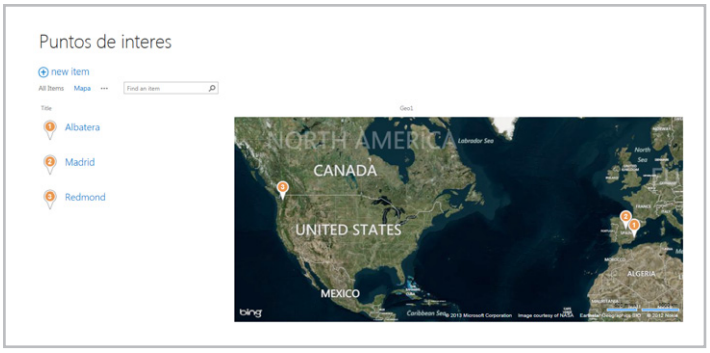


Imagen 11.- Vista de puntos de interés.

Esta forma en la que se representan los mapas, se puede cambiar mediante código JavaScript, es más, se puede implementar para que estos campos se representen en mapas de Google o de Nokia. Esta parte es la referida a “client-side rendering” que la abordaremos en la segunda parte del artículo.

Referencias

[1] Microsoft® SQL Server® 2008 R2 SP1 Feature Pack: http://www.microsoft.com/en-us/download/details.aspx?id=26728

[2] Registrarse y obtener clave de Bing Maps: http://bingmapsportal.com

[3] Office 365: http://office.microsoft.com

[4] How to add geolocation fields to SharePoint 2013 site: http://msmvps.com/blogs/sundar_narasiman/archive/2012/11/22/how-to-add-geolocation-fields-to-sharepoint-2013-site.aspx

[5] Office 365 - SharePoint 2013 Online. Establecer clave de Bing Maps para el campo de tipo Geolocation: http://bit.ly/BingMapsSharePointOnline

[6] eBook Gratuito de SharePoint 2013 Apps: http://bit.ly/SharePoint2013AppsEbook

JOSÉ QUINTO ZAMORA
MCPD y MCITP en SharePoint 2010
jquinto@solidq.com
@jquintozamora
http://blogs.solidq.com/sharepoint

13

Un nuevo CSOM, el Mobile Object Model de SharePoint 2013

Resumen

Un punto diferenciador de SharePoint, con respecto a otras plataformas de colaboración, es la capacidad de ofrecer una capa de abstracción en cliente, permitiendo el desarrollo de aplicaciones o integraciones externas. Desde el primer día, los servicios Web de SharePoint han permitido interactuar con la plataforma de una forma sencilla y universal, permitiendo, sin tener que desarrollar ninguna API de servidor, acceder a cualquier tipo de contenido en SharePoint.

En este artículo, haremos una introducción al modelo de objetos de cliente para Windows Phone, una plataforma que empieza a ofrecer funcionalidades empresariales, de las que no podemos olvidarnos trabajando con SharePoint en las empresas.

Artículo

Un punto diferenciador de SharePoint, con respecto a otras plataformas de colaboración, es la capacidad de ofrecer una capa de abstracción en cliente, permitiendo el desarrollo de aplicaciones o integraciones externas. Desde el primer día, los servicios Web de SharePoint han permitido interactuar con la plataforma de una forma sencilla y universal, permitiendo, sin tener que desarrollar ninguna API de servidor, acceder a cualquier tipo de contenido en SharePoint.

Con SharePoint 2010, se incluyeron dos opciones nuevas, el Cliend-Side Objet Model (CSOM) y la interfaz REST, que complementaban a los servicios Web de la plataforma, ofreciendo más posibilidades de desarrollo. Mientras que en el último SharePoint, la versión 2013, se han añadido nuevas funcionalidades al CSOM, así como una nueva y más limpia interfaz REST.

Dentro de las múltiples versiones del modelo de objetos de cliente de SharePoint 2013, nos encontramos con un nuevo SDK para el desarrollo de aplicaciones móviles, el Mobile Object Model.

- Versión para Windows Phone 7:
http://www.microsoft.com/en-us/download/details.aspx?id=35475
- Versión para Windows Phone 8:
http://www.microsoft.com/en-us/download/details.aspx?id=36818&WT.mc_id=rss_office_allproducts
- Versión REST:
http://msdn.microsoft.com/en-us/library/fp142385.aspx

Desde el Mobile Object Model, podemos trabajar con listas, bibliotecas de documentos, las nuevas notificaciones push (artículo de Adrián Díaz en CompartiMOSS) de SharePoint 2013, Perfiles de usuarios, Social, y casi todos los elementos que SharePoint expone en cliente.

Una nueva oportunidad para mejorar la movilidad de las empresas que usan SharePoint, ya que pueden extender sus procesos empresariales de forma universal a todos los usuarios, estén donde estén y tanto con un ordenador como con un móvil. Pensemos en aplicaciones que permitan aprobar flujos de trabajo en SharePoint, consultar datos de clientes en listas o almacenar fotos geo-localizadas en una biblioteca de documentos en SharePoint.

En este artículo, haremos una introducción a la versión de Windows Phone, dejando para el siguiente la versión REST, que nos permitiría hacer un cliente más universal para Windows 8, iPhone o Android.

El CSOM de Windows Phone usa el servicio WCF, client.svc, que es el encargado de analizar y ejecutar las consultas del cliente, usando el modelo de objetos de servidor, y devolver el resultado en JSON al cliente para que pueda trabajar con los objetos que necesite. Siguiendo el siguiente diagrama de arquitectura.

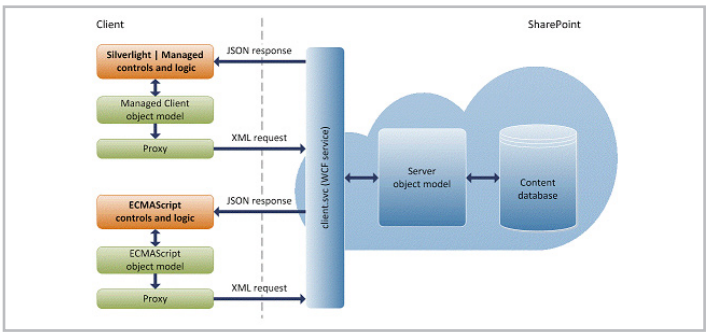


Imagen 1.- Arquitectura del CSOM en SharePoint 2013.

Si ya hemos trabajado con la implementación Silverlight del CSOM, trabajar con esta versión para Windows Phone es prácticamente lo mismo, básicamente hay que tener presente las llamadas asíncronas y que los objetos no se cargan hasta que ejecutemos la consulta en el servidor.

Autenticación

Hasta ahora, el principal problema que teníamos con Windows Phone y la integración con SharePoint es el de la autenticación del usuario frente a SharePoint. Para esto, tenemos una clase, Authenticator, que se encarga de realizar la autenticación por nosotros, aunque podemos realizarla manualmente.

```
    var context = new ClientContext("https://xxx.s
harepoint.com");
    var authenticator = new Authenticator();
    authenticator.CookieCachingEnabled = true;
    context.Credentials = authenticator;
```

Consultas a lista

No tenemos cambios significativos cuando realizamos consultas a listas, básicamente obtenemos la lista, con o sin el CAML query, trabajando siempre desde el contexto de cliente y en el ExecuteQueryAsync obtenemos los elementos de esa lista o consulta.

```
    var lista = new ObservableCollection<Announcem
ent>();

    var query = GetAnnouncementQuery();
    var items = Context.Web.Lists.GetByTitle("Anno
uncements").GetItems(query);
    Context.Load(items);
    Context.Load(items, listItems => listItems.Inc
lude(item => item.FieldValuesAsText));

    Context.ExecuteQueryAsync(
        delegate(object sender, ClientRequestSucce
ededEventArgs args)
        {
            foreach (var item in items)
            {
                var anuncio = new Announcement();
                anuncio.ID = item.Id.ToString();
                anuncio.Title = item.FieldValuesAs
Text["Title"];
                anuncio.Body = item.FieldValuesAsT
ext["Body"];
                anuncio.Expires = item.FieldValues
AsText["Expires"];
                anuncio.Created = item.FieldValues
AsText["Created"];

                lista.Add(anuncio);
            }

        },
        delegate(object sender, ClientRequestFaile
dEventArgs args)
        {
            //Manejamos el error de la consulta
        });
```

Crear elementos en lista

Como siempre, nos creamos un objeto del tipo ListItemCreationInformation y lo envíanos al contexto.

```
    var subscriptionList = Context.Web.Lists.GetBy
Title("HubSubscribers");
    Context.Load(subscriptionList);
    deviceItem = subscriptionList.AddItem(new List
ItemCreationInformation());
    deviceItem["Title"] = displayName;
    deviceItem["UserAccount"] = acccountName;
    deviceItem["ChannelUri"] = pushChannel;
    deviceItem["ChannelUriDate"] = System.DateTime
.Now;
    deviceItem["DeviceId"] = deviceId;
    deviceItem.Update();
    Context.ExecuteQuery();
```

Obtener un perfil de usuario

En el CSOM de SharePoint 2013 tenemos acceso a los perfiles de usuario, y en Windows Phone también. Para esto, nos creamos un PeopleManager y obtenemos las propiedades que necesitemos.

```
    var peopleManager = new PeopleManager(Context)
;
    var personProperties = peopleManager.GetProper
tiesFor(userAccount);

    context.Load(personProperties, p => p.AccountN
ame, p => p.DisplayName, p => p.Email, p => p.User
ProfileProperties,
p => p.DirectReports, p => p.Peers, p => p.Picture
Url);
    context.ExecuteQueryAsync(
        delegate(object sender1, ClientRequestSucc
eededEventArgs args)
        {
            var profile = new Profile();
            profile.AccountName = personProperties
.AccountName;
            profile.DirectReports = personProperti
es.DirectReports.ToList();
            profile.Peers = personProperties.Peers
.ToList();
            profile.DisplayName = personProperties
.DisplayName;
            profile.PictureUrl = personProperties.
PictureUrl;
            profile.Email = personProperties.Email
;
            profile.Manager = personProperties.Use
rProfileProperties["Manager"];
            profile.Status = personProperties.User
ProfileProperties["SPS-StatusNotes"];
            profile.WorkPhone = personProperties.U
serProfileProperties["WorkPhone"];
            profile.Department = personProperties.
UserProfileProperties["Department"];

            loadProfileCompletedCallback(new LoadP
rofileCompleteEventArgs { Profile = profile });
        },
        delegate(object sender1, ClientRequestFail
edEventArgs args)
        {
            //Manejamos el error de la consulta
        });
```

Conclusiones

¿A qué esperamos para desarrollar aplicaciones de negocio para Windows Phone? Personalmente, creo que nos puede abrir un nuevo abanico de posibilidades y de aplicaciones empresariales que no han terminado de llegar con SharePoint 2010. Por cierto, aunque no lo he probado al 100%, este modelo de objetos de cliente también funciona con SharePoint 2010, salvo las funcionalidades nuevas, como los perfiles, que no tenían soporte de cliente en SharePoint 2010.

ALBERTO DIAZ MARTIN
MVP SharePoint
adiazcan@hotmail.com
@adiazcan
http://geeks.ms/blogs/adiazmartin

AvePoint®

SharePoint User Group Spain
SUGES

SHAREPOINT SIN RODEOS

Una Serie **Exclusiva de Webcasts** presentada por AvePoint y SUGES

Office 365 Social
Protección de SharePoint
Nube **Flujos de trabajo**
Azure Gestión de contenido **SharePoint 2013**
Flujos de trabajo
Sitios de Publicación

En nuestro esfuerzo contínuo para mantenerle al tanto de las noticias y los eventos que suceden en el mundo de Microsoft® SharePoint®, AvePoint y SUGES lanzarán una serie de 6 webcasts sobre 6 temas actuales de SharePoint 2013 y la Nube, exclusivamente presentada por MVPs de SharePoint, empezando el 25 de abril.

¡No se lo pierda!

Regístrese hoy y reciba información de primera mano a través de la experiencia de MVPs. https://eu.avepoint.com/resources/webinars

16

Manejadores de eventos remotos en SharePoint 2013

Resumen

SharePoint 2013 introduce el concepto de manejadores de eventos remotos como un mecanismo para notificar a sistemas externos de sucesos qué se hayan producido en el propio SharePoint como por ejemplo la creación, actualización o borrado de elementos en una lista. De esta forma, sistemas externos al propio SharePoint pueden reaccionar a estos cambios. Como ocurre con los manejadores de eventos clásicos, los manejadores de eventos remotos permiten reaccionar a eventos de naturaleza síncrona (“ing”) y/o asíncrona (“ed”).

Artículo

En este artículo vamos a ver cómo crear un manejador de eventos remoto que reaccione ante los cambios qué tengan lugar en una lista personalizada desplegada como parte de una aplicación del nuevo modelo de aplicaciones de SharePoint 2013.

Creación del manejador de eventos remoto

Para crear un manejador de eventos remoto que responda a eventos que suceden en los elementos de una lista personalizada, seguiremos los siguientes pasos:

- Iniciamos Visual Studio 2012 y creamos un proyecto de tipo “Aplicación para SharePoint 2013”.
- En el asistente de configuración especificamos el nombre de la aplicación, la Url que se va a utilizar para cuestiones de depuración y el tipo de aplicación. Como se aprecia en la Imagen 1, en este caso se están indicando una Url de un sitio de Office 365 y “Autohospedado” como tipo de aplicación.

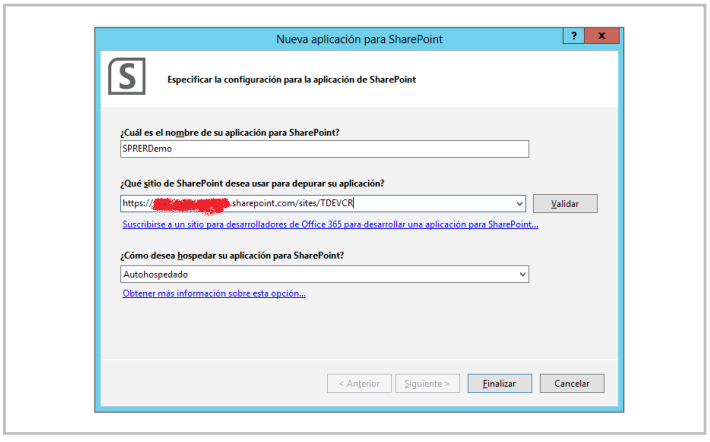


Imagen 1.- Asistente de configuración de la aplicación.

- Una vez qué finaliza el asistente de configuración, el explorador de soluciones de Visual Studio 2012 muestra qué se ha creado una solución qué dispone de dos proyectos:
  - El proyecto de aplicación en sí mismo que permite genera un archivo “.app” qué posteriormente facilita la distribución de la aplicación.
  - Un proyecto de tipo web qué contiene la definición de la aplicación y en el que hay elementos como páginas, scripts JavaScript, hojas de estilo y clases auxiliares que facilitan la interactuación entre la aplicación y SharePoint.
- Creamos una lista de tipo personalizado en el contexto de la aplicación. Para ello, seleccionamos el proyecto de aplicación y añadimos un elemento de tipo “Lista”. A continuación se inicia un nuevo asistente que nos guiará por el proceso de definición de la lista. Por simplicidad, creamos una lista de tipo personalizado (Imagen 2).

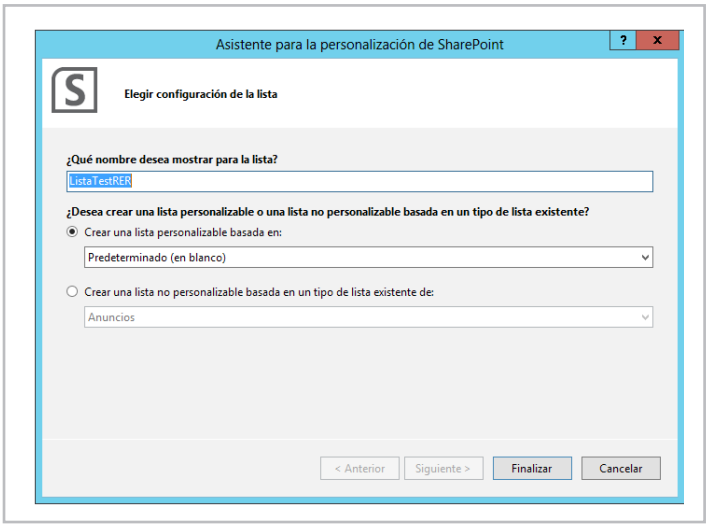


Imagen 2. Asistente para la creación de la lista en la aplicación.

- Como resultado de la creación de la lista, por un lado se muestra el diseñador visual de listas de Visual Studio 2012. Por otro, en el proyecto de la aplicación se añade el correspondiente SharePoint Project Item (SPI) que contiene la definición de la lista y la correspondiente instancia.
- El siguiente elemento que vamos a añadir al proyecto de aplicación es manejador de eventos remoto utilizando la plantilla “Receptor de eventos remoto” disponible. Para configurar el manejador, se dispone de un asistente similar

al ya conocido para manejadores de eventos “clásicos” qué permite configurar:

- El tipo de receptor de eventos, siendo los valores posibles “Eventos web”, “Eventos de lista” y “Eventos de elementos de lista”.
- Los eventos a controlar.

En nuestro caso, elegimos la opción “Eventos de lista” y marcamos como eventos a controlar los siguientes:

- “Se va a agregar un elemento” y “Se va actualizar un elemento” (eventos síncronos).
- “Se agregó un elemento” (evento asíncrono).

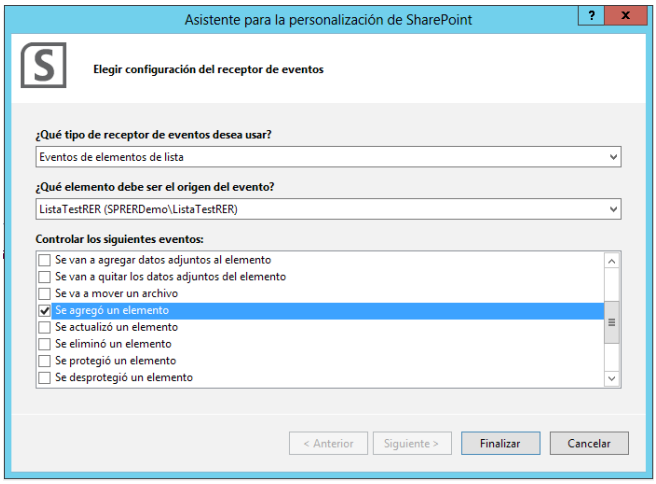


Imagen 3.- Asistente de configuración de un manejador de eventos remoto.

- Como resultado de la creación del manejador, por un lado se añade al proyecto web de la aplicación un servicio WCF en el que se tiene que implementar la lógica correspondiente. Por otro lado, en el proyecto de la aplicación se añade un SPI que contiene el archivo de manifiesto relativo al manejador. El Listado 1 muestra el contenido de dicho manifiesto. Como se puede apreciar, refleja los tipos de eventos a controlar y la Url dónde se registrará el servicio WCF qué contiene la lógica de procesado correspondiente en función del evento que se produzca en la lista.

```
</Receiver>
<Receiver>
<Name>CompartiMOSSRERItemUpdating</Name>
<Type>ItemUpdating</Type>
<SequenceNumber>10000</SequenceNumber>
<Url>~remoteAppUrl/CompartiMOSSRER.svc</Url>
</Receiver>
<Receiver>
<Name>CompartiMOSSRERItemAdded</Name>
<Type>ItemAdded</Type>
<SequenceNumber>10000</SequenceNumber>
<Url>~remoteAppUrl/CompartiMOSSRER.svc</Url>
</Receiver>
</Receivers>
</Elements>
```

Listado 1.- Definición del manejador de eventos remoto en la aplicación.

Programación del manejador

Una vez qué tenemos creados todos los elementos necesarios para evaluar el funcionamiento de los manejadores de eventos remotos, sólo resta codificar los métodos correspondientes en el servicio WCF:

- El servicio WCF consta de una clase que hereda de la interfaz IRemoteEventService. Esta interfaz define dos métodos decorados con el atributo OperationContact que son ProcessEvent() y ProcesOneWayEvent(). El primero de los eventos permite qué el servicio reaccione a eventos de tipo síncrono (“ing”) ocurridos en el sitio. El segundo, garantiza respuesta por parte del manejador a eventos de naturaleza asíncrona (“ed”).
- Por ejemplo, podemos codificar el método ProcessEvent() de acuerdo al Listado 2.

```
publicSPRemoteEventResultProcessEvent(SPRemoteEventP
roperties properties)
{
SPRemoteEventResult result =
newSPRemoteEventResult();
switch (properties.EventType)
    {
caseSPRemoteEventType.ItemAdding:
result.ChangedItemProperties.Add(
"Title",
properties.ItemEventProperties.AfterProperties["Titl
e"] +=
" - AñadiendoElemento");
break;
caseSPRemoteEventType.ItemUpdating:
result.ChangedItemProperties.Add(
"Title",
properties.ItemEventProperties.AfterProperties["Titl
e"] +=
" - ActualizandoElemento");
break;
default:
break;
    }
return result;
}
```

Listado 2.- Ejemplo de codificación del método ProcessEvent().

Como se puede apreciar, por una parte el objeto properties (de tipo SPRemoteEventProperties) contiene la información relativa al evento qué ha ocurrido (propiedad EventType). Por otra parte, para cada tipo de evento síncrono simplemente se accede a la columna Title (cuyo valor a cambiado) del elemento de la lista y se actualiza su valor concatenando una cadena de texto. Esto es posible gracias al objeto result que es de tipo SPRemoteEventResult lo que permite acceder a la(s) columna(s) cuyo valor ha(n) cambiado en el elemento y modificarlas a posteriori y antes de que se guarde el elemento en la lista.

- El Listado 3 muestra un ejemplo de codificación para el método PorccessOneWayEvent():

```
publicvoidProcessOneWayEvent(SPRemoteEventProperties
properties)
{
if (properties.EventType ==
SPRemoteEventType.ItemAdded)
    {
using (ClientContextctx = newClientContext(
properties.ItemEventProperties.WebUrl))        {
ListlList =
ctx.Web.Lists.GetByTitle(
properties.ItemEventProperties.ListTitle);
ctx.Load(lList);
ListItemliItem =
lList.GetItemById(
properties.ItemEventProperties.ListItemId);
ctx.Load(liItem);
ctx.ExecuteQuery();
liItem["Title"] +=
" - ElementoAñadido";
liItem.Update();
ctx.ExecuteQuery();
}
    }
}
```

Listado 3.- Ejemplo de codificación del método ProcessOneWayEvent().

En este caso como se trata de procesar un evento síncrono, si queremos hacer algún tipo de operación con el elemento de la lista tendremos que acceder al mismo una vez añadido (para el caso de ejemplo). Como el acceso se realiza de forma remota desde un servicio WCF, tenemos que recurrir al modelo de objetos en cliente para poder recuperar el elemento y realizar el procesado requerido. De nuevo, el objeto properties nos da toda la información necesaria (a través de la propiedad ItemEventProperties) para poder crear instancias de objetos ClientContext, List y ListItem. Cada vez que necesitemos información del sitio de SharePoint, es necesario definir la operación a realizar y posteriormente llamar al método ExecuteQuery del objeto ClientContext creado.

Despliegue y prueba del manejador

El despliegue y prueba del manejador de eventos remotos para una aplicación de tipo "Autohosted" es realmente sencillo:

- En Visual Studio 2012, seleccionamos el nombre del proyecto de la aplicación, hacemos clic con el botón derecho del ratón y presionamos la opción "Implementar" de forma que se inicia el proceso de despliegue de la aplicación en el sitio de Office 365 configurado inicialmente. El proceso concluye ejecutando el explorador web donde tendremos que introducir las credenciales de acceso a Office 365.
- Una vez introducidas, se muestra la página de la Imagen 4 en la que se nos solicita confirmación para que la aplicación desplegada pueda acceder a información del sitio.

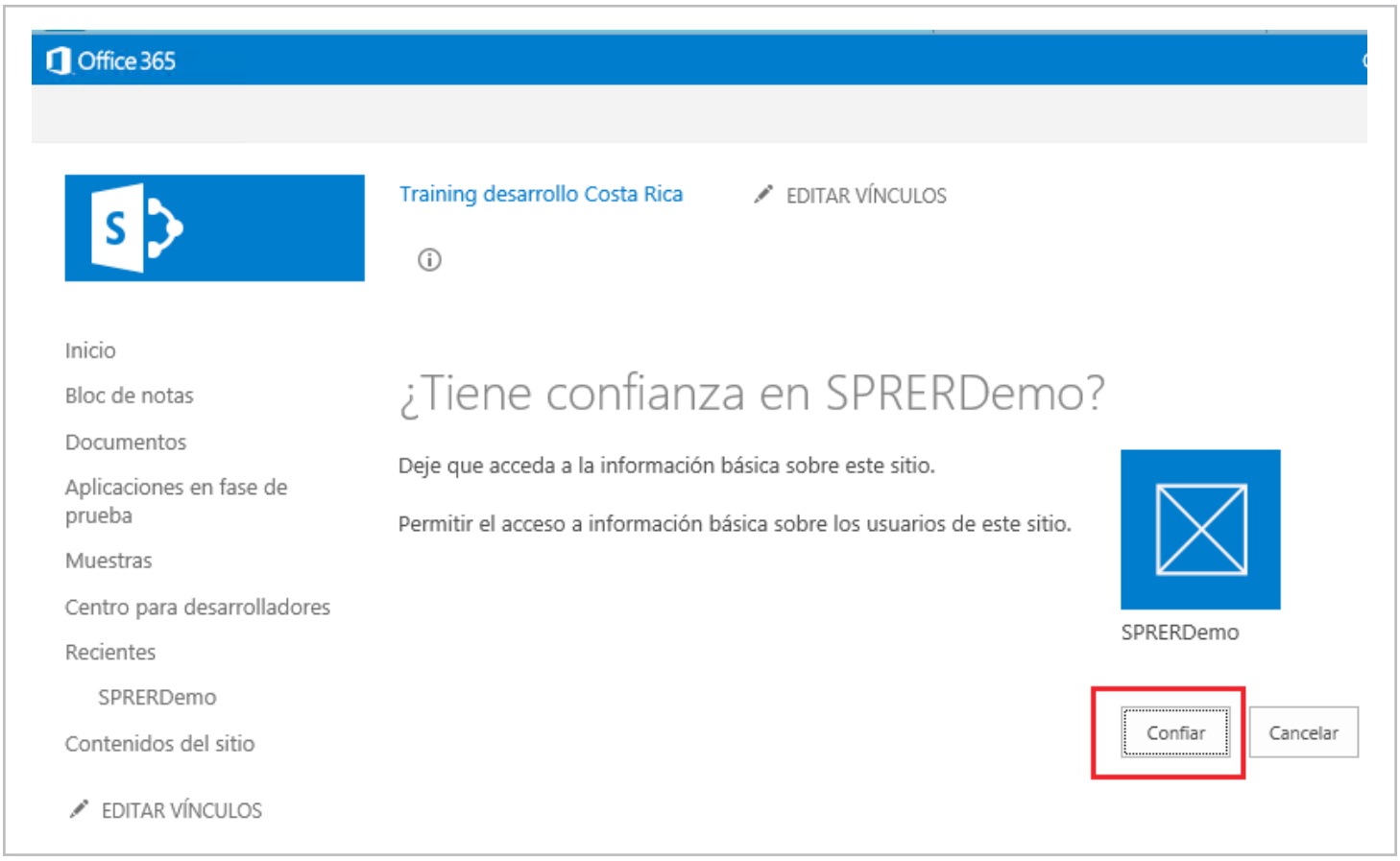


Imagen 4. Página de confirmación de acceso de la aplicación a información del sitio.

"... un mecanismo para notificar a sistemas externos de sucesos qué se hayan producido en el propio SharePoint."

- Pulsamos el botón “Confiar”, de manera que accedemos a la página principal de la aplicación. A partir de aquí, podemos comprobar por un lado qué el servicio WCF qué implementa la lógica del manejador se ha publicado de forma correcta y por otro que la lista de ejemplo está disponible para probar el manejador.
- Para verificar que el servicio se ha publicado correctamente, simplemente en la Url de la aplicación eliminamos todo el contenido innecesario y lo reemplazamos por el nombre del servicio WCF:

URL DE LA APLICACIÓN
https://defe8b3e-8dbf-4afe-9ca2-274728027618.o365apps.net/Pages/Default.aspx?SPHostUrl=https%3A%2F%<DominioOffice365>%2Esharepoint%2Ecom%2Fsites%2FTDEVCR&SPLanguage=es%2DES&SPClientTag=0&SPProductNumber=15%2E0%2E4454%2E1011&SPAppWebUrl=https%3A%2F%2F<DominioOffice365>%2Dfe7f0e9d237f43%2Esharepoint%2Ecom%2Fsites%2FTDEVCR%2FSPRERDemo

URL DEL SERVICIO
https://defe8b3e-8dbf-4afe-9ca2-274728027618.o365apps.net/CompartimossRER.svc

- Para visualizar la lista (Imagen 5) y probar el manejador, modificamos la Url del sitio de Office 365 en el que se ha agregado la aplicación de la siguiente forma:
https://<URL_Sitio_Office365>/<NombreAplicacion>/Lists/<NombreLista>

Al introducir esta Url en el navegador, automáticamente se realiza una redirección a una Url de la forma:
https://<DominioOffice365>-fe7f0e9d237f43.sharepoint.com/sites/TDEVCR/SPRERDemo/Lists/ListaTestRER/AllItems.aspx

Como se puede deducir, el proceso de despliegue de la aplicación implica en este caso la creación de un subsitio en el qué se va a crea la instancia lista en base a la definición incluida en el proyecto.

Adicionalmente, este sitio se encuentra en un dominio completamente aislado del sitio en el que agrego originalmente la aplicación.

Imagen 5.- Acceso a la lista desplegada con la aplicación.

Imagen 6.- Prueba del manejador de eventos remoto en la lista.

- Finalmente, probamos que el manejador de eventos remotos está cumpliendo con su cometido.

Nota:

Como se puede apreciar en la Imagen 6, el manejador de eventos remoto está respondiendo de forma correcta a eventos de naturaleza síncrona, pero no a eventos de naturaleza asíncrona.

Como en la fecha de realización de este artículo (febrero de 2012) Microsoft se encuentra actualizando el servicio de SharePoint Online en Office 365, este puede ser el motivo de qué el manejador no esté respondiendo de forma correcta a eventos de naturaleza asíncrona. Adicionalmente, la versión de las herramientas de desarrollo para SharePoint y Office de Visual Studio 2012 sigue siendo la Preview 2.

Conclusiones

Los manejadores de eventos remotos vienen a cubrir una capacidad muy demandada por los desarrolladores de SharePoint desde hace años: poder reaccionar a eventos que ocurren en el sistema desde aplicaciones de negocio externas. Estos manejadores se implementan en la forma de servicios WCF que reaccionan ante eventos que se produzcan en los ámbitos de sitio web, lista y elemento de lista.

Adicionalmente, las listas externas y los tipos de contenido externos soportan manejadores de eventos aunque su tratamiento se sale del ámbito de este artículo.

JUAN CARLOS GONZÁLEZ MARTÍN
MVP de SharePoint Server, Arquitecto de soluciones en el CIIN
jgonzalez@gruposodercan.es
@jcgm1978
http://geeks.ms/blogs/ciin

20

Entrevista a Pablo Pussacq Laborde

Mi nombre es Juan Pablo Pussacq Laborde, nací en Buenos Aires, Argentina en 1974. Soy Licenciado en Análisis de Sistemas, recibido en la Facultad de Ingeniería de la Universidad de Buenos Aires de Argentina. He sido docente de algunas materias en esa facultad: Modelos y Optimización I, Bases de Datos y Administración y Control de Proyectos informáticos I y II.

He tenido también el honor de ser orador en algunas conferencias entre las que destaco el congreso del SEPGLA en México y las jornadas del PMI en Argentina. En los últimos dos años he recibido el premio MVP de Microsoft por mi especialidad en SharePoint, lo cual me ha llenado de orgullo y confirma mi frase de cabecera: "Todo lo que eres capaz de soñar eres capaz de conseguirlo, siempre que te acompañe el esfuerzo y la convicción"

¿Por qué y cómo empezaste en el mundo de la tecnología?

Por culpa del inglés. Mientras estudiaba en la secundaria, también hacía cursos de inglés por la tarde. A diferencia del colegio, en inglés era el peor alumno y no le encontraba la vuelta. Entonces decidí abandonarlo y mis padres me dijeron que me anotara en otro curso. Tímidamente y sin saber lo que era una computadora, me anoté en un curso de computación, en donde aprendí Basic en una Commodore y realmente me voló la cabeza. Mis calificaciones eran las mejores, es que para mí era un juego, realmente disfruté esos dos años de programación en Basic, en donde esperaba ansioso pasar mi programa del papel a la máquina: eran tiempos de GOTO, los sprites y los números de línea. Cuando terminé el curso, quise hacer otro, pero ahí me dijeron que lo próximo era la carrera, y así me pasé a la Universidad de Buenos Aires, hasta que me recibí, con una compra de una AT 286 en mi segundo año y mi primer disco rígido de 40 MB un tiempo después, necesario porque Turbo C no se podía ejecutar desde la disquetera...

¿Cuáles son tus principales actividades tecnológicas hoy en día?

Trabajo fuertemente en Project Server, que une mi especialidad técnica en SharePoint con mis años de experiencia en la rama de administración de proyectos y mejora de procesos. Es un tema de nicho realmente.

También hago desarrollos en SharePoint, mi plataforma preferida. Me gusta extender SharePoint, convencido de que se pueden hacer buenas aplicaciones con 80% de configuración y 20% de desarrollo. SharePoint es un producto increíble.

Además me he especializado en sitios web avanzados de la mano del CMS WordPress y también de BuddyPress, el plugin de WordPress para redes sociales. Esto me ha dado un complemento muy bueno a mis conocimientos previos, especialmente en lo relacionado con la parte web como: CSS, SEO, herramientas para community manager (en este tema he profundizado), integración y automatización entre diferentes componentes web y algo de Seguridad.

¿Cuáles son tus principales actividades NO tecnológicas hoy en día?

La familia principalmente. Además he heredado el 1% de la profesión de ebanista de mi abuelo y otro 1% de la de ingeniero civil de mi padre. Esto me permite cambiar lamparitas, podar árboles, arreglar ventanas y hasta modificar muebles usando un serrucho. Esta actividad me distiende bastante, aunque extraño el ctrl-z y la posibilidad de prueba y error que te da el software.

Soy además un buen deportista, al menos eso dicen los que me conocieron en otra época y entrenaba tres veces por semana. Ya volveré...

¿Cuáles son tus hobbies?

Hoy mi cable a tierra, mi hobbie, está muy relacionado con la tecnología, los automóviles y las redes sociales. Estoy haciendo un experimento juntando esas tres patas. Es un hobbie excelente, apasionante y atrapador. En algún momento les comentaré más detalles...

Describir las particularidades de cada una de las secciones que la componen y analizar sus funciones y ventajas

¿Cuál es tu visión de futuro en la tecnología de acá a los próximos años?

Creo que la nube y la movilidad están cambiando la forma de trabajar. Yo estoy respondiendo estas preguntas en mi teléfono inteligente mientras viajo en un tren. Eso se guarda en la nube y se sincroniza luego con mi laptop, en donde terminaré de darle forma. Es un ejemplo mínimo. Particularmente creo que la nube permite a las empresas invertir mejor sus recursos y tener mejores servicios en menos tiempo. Eso junto a los dispositivos móviles de todo tipo, ha hecho que en último tiempo sea mucho más sencillo integrar componentes. Casi no importa dónde esté la información ni con que dispositivo te conectes.

Por otro lado, creo que las redes sociales están democratizando el poder en Internet. Hoy le puedes enviar un tweet a un presidente o juntarte virtualmente con personas de cualquier parte del mundo para lograr un objetivo común. Para mi es una revolución que aún está en pañales, pero cambiará estructuras fuertemente arraigadas. Respecto a Internet, esperemos que siga siendo libre...

Los viajes interestelares aún los veo lejanos, pero quizá en breve tengamos algo parecido a la teletransportación. ¿O creen que pasará mucho tiempo hasta que nuestra imagen viaje a cualquier parte y en conjunto con algún tipo de sensores podamos abrazar a un familiar del otro lado del mundo? No creo que falte mucho para este nuevo teléfono...

JUAN PABLO PUSSACQ LABORDE
SharePoint MVP
Blog: http://surpoint.blogspot.com/
Facebook: http://facebook.com/surpointblog/
Twitter: http://twitter.com/jpussacq/

22

Paso a Paso: Customización de Data Views con SharePoint Designer 2010 y Visual Studio 2010

Resumen

Como sabemos, cada vez que hacemos una personalización con SharePoint Designer de cómo debería visualizarse un elemento web con los datos procedentes de una lista resulta engorroso, y a veces complicado, realizar el pasaje entre ambientes, ya que no tenemos más remedio que repetir cada uno de los pasos en el otro servidor, pero: ¿Qué sucede si en el sitio productivo no tenemos permiso? ¿Y si el sitio que tiene nuestra customización es exportado/importado en un nuevo servidor?

En el último caso, la lista tendrá un nuevo ID y como resultado nuestro elemento no se visualizará ya que no encuentra el ListID con el que fue creado.

Artículo

Cuando nos encontramos ante esta necesidad de crear una vista con cierto look & feel diferente de los elementos de una lista generalmente terminamos utilizando SharePoint Designer 2010, lo que nos resuelve el requerimiento de "ese sitio" de "tal servidor", voy a mencionar 2 problemas que es bastante común que se nos puedan presentar:

PROBLEMA CONOCIDO 1:
Cuando aprueban el comportamiento de la personaliación en desarrollo y dan el OK para el pasaje al ambiente de QA: ¿Qué hacemos? ¿Repetimos todo desde cero de nuevo?

PROBLEMA CONOCIDO 2:
La customización está en un sitio que se migrará haciendo un "export". El efecto que causa esta metodología es que se recrean los ID de listas, elementos de lista, etc. Por lo que como nuestra customización está atada a cierto ListID que ya no existe, tendremos como resultado que el elemento web desapareció de nuestra página.

SOLUCIÓN A LOS PROBLEMAS:
Para resolver el primer problema tenemos que pensar en que nuestra customización sea desplegable en otros ambientes. Para resolver el segundo problema nuestra customización NO tiene que depender del ListID sino del ListName.

Para solucionar estos problemas vamos a apoyarnos en el Visual Studio 2010 para que nuestra solución sea deployable y configurable a través del ListName.

Paso a Paso

A modo de ejemplo, personalizaremos una vista que se muestre en 4 columnas.

PASO 1:
Con SharePoint Designer insertamos en una página un Data View, configuramos el Data Source y el campo a mostrar. Para este ejemplo, insertamos un Empty Data View, seleccionamos una lista de tipo "Links" llamada Topics como Data Source y mostramos la columna correspondiente a la URL.

PASO 2:
Cambiar el diseño de la vista a la vista 2 columnas: "Two columns repeating form with border" como primer paso.

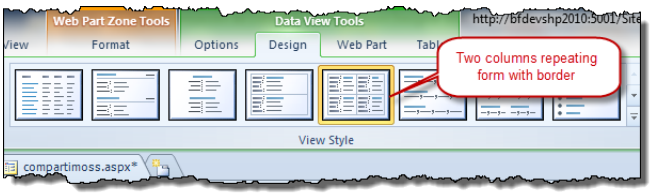


Imagen 1.- SharePoint Designer 2010 - Design Tab para cambiar el estilo de la vista

PASO 3:
Para visualizar el contenido en 4 columnas:
a. Buscar las líneas y reemplazar el valor "50%" por "25%" como muestra el código de ejemplo:

```
<xsl:template name="dvt_1.rowview">
<td valign="top" width="25%">
    <table border="0" cellspacing="0" width="100%">
    <tr>
```

Imagen 2.- Modificación del width para visualizar 4 columnas.

b. Para cambiar a 4 columnas, buscamos en el código de la página las líneas que contienen "mod 2", como muestra la imagen, y lo reemplazamos por "mod 4".

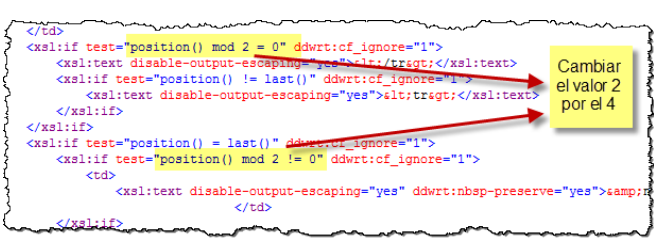


Imagen 3.- Cambiar la visualización de 2 columnas a 4 columnas.

PASO 4:

Una vez que terminamos de editar el Data View y modificar el look & feel del mismo, guardamos los cambios y visualizamos los resultados en el browser.

Si el resultado es el esperado entonces estamos listos para exportar el elemento web y guardar el archivo ".webpart" generado.

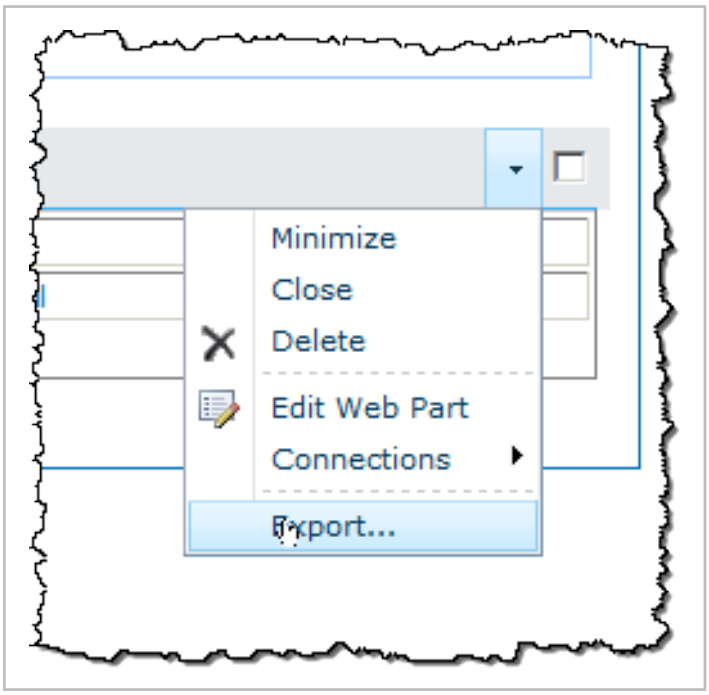


Imagen 4.- Exportar un elemento web a través del browser.

PASO 5

Ya estamos listos para realizar las modificaciones al Data View y hacerlo desplegable a otro ambiente donde reconozca la lista del origen de datos. Ejecutamos el Visual Studio 2010 como Administrador (Nota: Cuando se requiere depurar un desarrollo contra un sitio de SharePoint, es necesario iniciar el IDE con un usuario Administrador).

Creamos un nuevo proyecto del tipo "Empty SharePoint Project" y elegimos la opción "deploy as a farm solution".

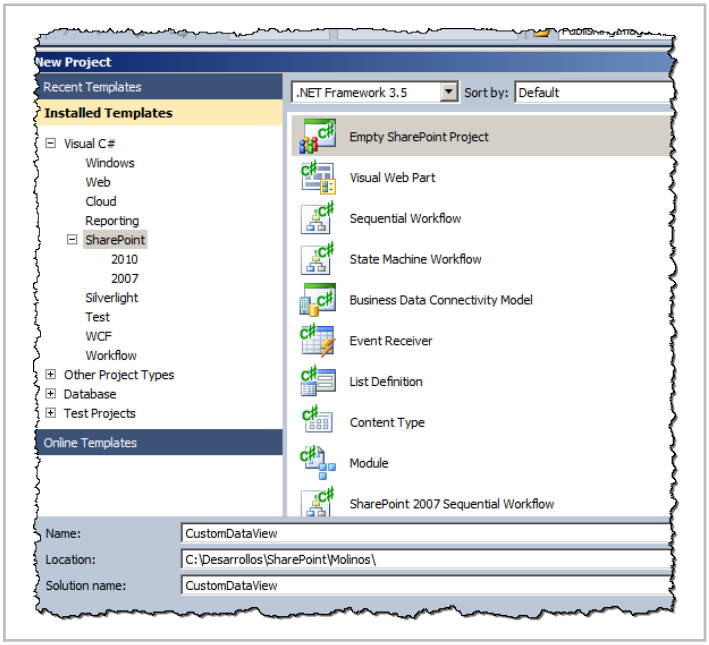


Imagen 5.- Nuevo Proyecto en Visual Studio 2010

PASO 6

Agregamos un nuevo ítem al proyecto y seleccionamos el tipo Web Part, le damos un nombre en mi ejemplo, "myCustomDataView," y luego hacemos clic en el botón "Add"

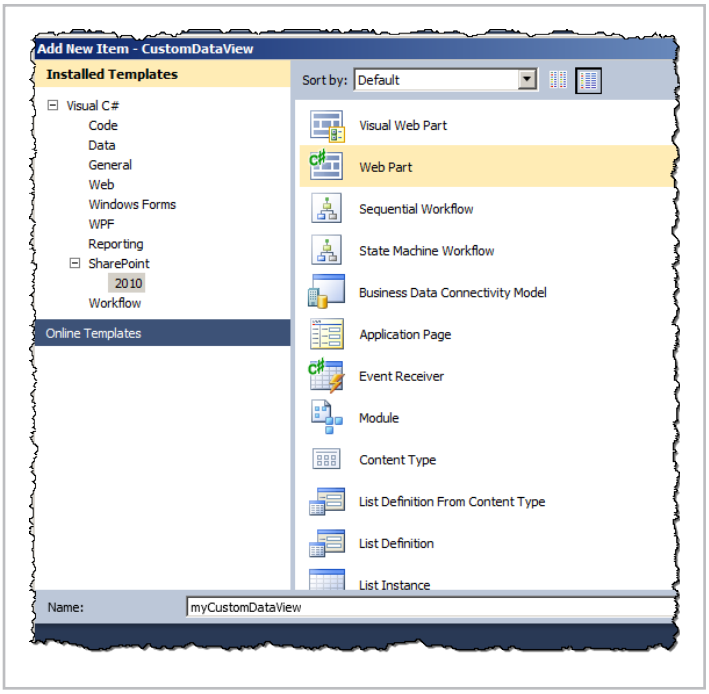


Imagen 6.- Agregar un Web Part al proyecto creado.

Una vez agregado, eliminamos el archivo "myCustomDataView.cs" del proyecto

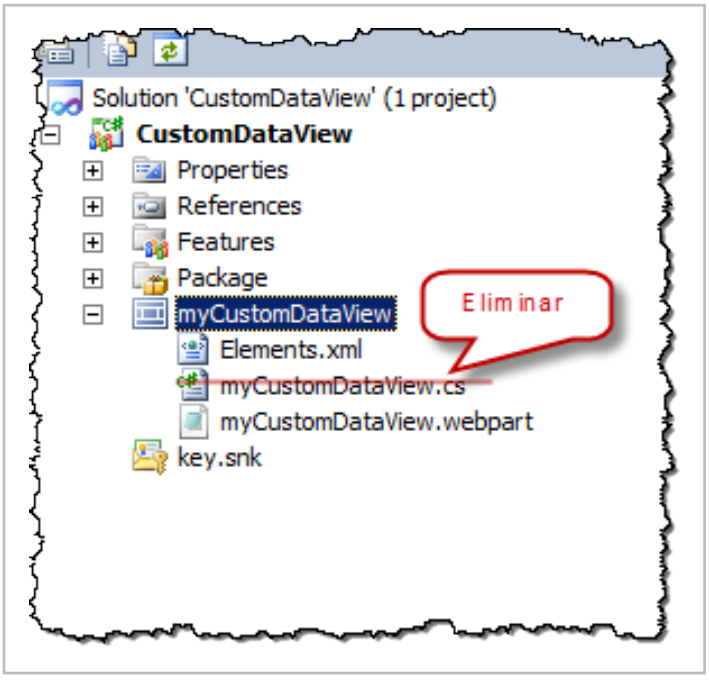


Imagen 7.- Eliminación del archivo ".cs" del web part.

"... cada vez que hacemos una personalización con SharePoint Designer... resulta engorroso, y a veces complicado, realizar el pasaje entre ambientes..."

PASO 7

Buscamos el archivo ".webpart" que exportamos en el Paso 4, copiamos el contenido y lo pegamos en el archivo de nuestro proyecto "myCustomDataView.webpart".

myCustomDataView.webpart*

```
<webParts>
  <webPart xmlns="http://schemas.microsoft.com/WebPart/v3">
    <metaData>
      <type name="Microsoft.SharePoint.WebPartPages.DataFormWebPart, Microsoft.SharePoint, V
      <importErrorMessage>Cannot import this Web Part.</importErrorMessage>
    </metaData>
    <data>
      <properties>
        <property name="InitialAsyncDataFetch" type="bool">False</property>
        <property name="ChromeType" type="chrometype">Default</property>
        <property name="Title" type="string">Topics</property>
        <property name="Height" type="string" />
        <property name="CacheXslStorage" type="bool">True</property>
        <property name="XslLink" type="string" null="true" />
        <property name="AllowZoneChange" type="bool">True</property>
        <property name="AllowEdit" type="bool">True</property>
        <property name="XmlDefinitionLink" type="string" />
        <property name="DataFields" type="string">@URL,URL;@Comments,Notes;@FileLeafRef,Name
        <property name="AutoRefreshInterval" type="int">60</property>
        <property name="NoDefaultStyle" type="string">TRUE</property>
        <property name="ListDisplayName" type="string" null="true" />
        <property name="AutoRefresh" type="bool">False</property>
        <property name="ViewFlag" type="string">8</property>
        <property name="ListUrl" type="string" null="true" />
        <property name="AllowConnect" type="bool">True</property>
        <property name="Description" type="string" />
        <property name="AllowClose" type="bool">True</property>
        <property name="ShowWithSampleData" type="bool">False</property>
        <property name="ParameterBindings" type="string">
          &lt;ParameterBinding Name="ListID" Location="None" DefaultValue="{D92E95D7-A67C-4B
```

Imagen 8.- Parte del código resultante de la exportación del Web Part.

PASO 8:

Para que en el despliegue reconozca a la lista a través de su nombre y no de su ID lo que tenemos que hacer ciertas modificaciones en las propiedades y en el código xsl generado por el SharePoint Designer.

a. Buscar la línea donde aparece la siguiente propiedad ListDisplayName

```
<property name="ListDisplayName" type="string" null="true" />
```

Imagen 9.- Propiedad ListDisplayName inicial.

b. Y cambiarla a:

```
<property name="ListDisplayName" type="string" null="true">Topics</property>
```

Imagen 10.- Propiedad ListDisplayName modificada.

c. En la propiedad ListName, cambiar el ID de la lista por el nombre de la Lista:

```
<property name="Hidden" type="bool">False</property>
<property name="ListName" type="string">{D92E95D7-A67C-4B2A-A798-450F951F6489}</property>
<property name="DataSourceID" type="string" />
```

Imagen 11.- Propiedad ListName Original.

d. Cambiar el ParameterBinding “ListID” como se muestra abajo, por el valor “ListName” y completar el DefaultValue con el nombre de la lista:

```
<property name="ParameterBindings" type="string">
  &lt;ParameterBinding Name="ListID" Location="None" DefaultValue="{D92E95D7-A67C-4B2A-A798-450F951F6489}"/&gt;
  &lt;ParameterBinding Name="dvt_apos" Location="Postback;Connection"/&gt;
  &lt;ParameterBinding Name="ManualRefresh" Location="WPProperty[ManualRefresh]"/&gt;
  &lt;ParameterBinding Name="UserID" Location="CAMLVariable" DefaultValue="CurrentUserName"/&gt;
  &lt;ParameterBinding Name="Today" Location="CAMLVariable" DefaultValue="CurrentDate"/&gt;
  &lt;ParameterBinding Name="dvt_startposition" Location="Postback" DefaultValue=""/&gt;
</property>
```

Imagen 12.- Propiedad ParameterBindings.

e. Eliminar la línea:

```
<property name="ListId" type="System.Guid, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
  d92e95d7-a67c-4b2a-a798-450f951f6489
</property>
```

Imagen 13.- Propiedad ListID.

f. Reemplazar cada ListID por ListName y el ID por el nombre de la lista respectivamente dentro de la propiedad DataSourcesString:

```
<property name="DataSourcesString" type="string">
  &lt;%@ Register TagPrefix="sharepoint" Namespace="Microsoft.SharePoint.WebControls" Assembly="Microsoft.SharePoint, Version=14.0.0.0, Culture=neutral, Publick
  &lt;webpartpages:DataFormParameter ParameterKey="ListID" PropertyName="ParameterValues" DefaultValue="{D92E95D7-A67C-4B2A-A798-450F951F6489}" Name="ListID"&g
  &lt;asp:Parameter DefaultValue="0" Name="StartRowIndex"&gt;&lt;/asp:Parameter&gt;
  &lt;asp:Parameter DefaultValue="0" Name="nextpagedata"&gt;&lt;/asp:Parameter&gt;
  &lt;asp:Parameter DefaultValue="10" Name="MaximumRows"&gt;&lt;/asp:Parameter&gt;
  &lt;/SelectParameters&gt;
  &lt;UpdateParameters&gt;
  &lt;webpartpages:DataFormParameter ParameterKey="ListID" PropertyName="ParameterValues" DefaultValue="{D92E95D7-A67C-4B2A-A798-450F951F6489}" Name="ListID"&g
  &lt;/UpdateParameters&gt;
  &lt;InsertParameters&gt;
  &lt;webpartpages:DataFormParameter ParameterKey="ListID" PropertyName="ParameterValues" DefaultValue="{D92E95D7-A67C-4B2A-A798-450F951F6489}" Name="ListID"&g
  &lt;/InsertParameters&gt;
  &lt;DeleteParameters&gt;
  &lt;webpartpages:DataFormParameter ParameterKey="ListID" PropertyName="ParameterValues" DefaultValue="{D92E95D7-A67C-4B2A-A798-450F951F6489}" Name="ListID"&g
  &lt;/DeleteParameters&gt;
  &lt;/sharepoint:SPDataSource&gt;
</property>
```

Imagen 14.- Propiedad DataSourcesString.

PASO 9:
Una vez realizados los cambios, le damos una descripción apropiada al Web Part para que se muestre al usuario, al feature correspondiente y hacemos el despliegue de la solución para probar.

CÓMO LO REUSAMOS:
Si tenemos una lista de links que queremos mostrar con ese mismo look & feel podemos reusar el elemento web. Simplemente, lo agregamos a nuestra página, editamos el elemento web, hacemos clic en el botón "Parameter Editor..." y cuando se abra la ventana modal editamos y modificamos el valor del ListName existente por el nombre de la lista de links que queremos visualizar.

Aceptamos los cambios y veremos la nueva lista de links con el mismo diseño que se creó para la lista de links inicial.

Resumiendo, usamos el SharePoint Designer una vez, customizamos el look & feel de cómo se verá la lista en la página, exportamos el xsl del webpart y lo agregamos a un nuevo proyecto en Visual Studio para eliminar toda referencia al ID de la lista y que ésta se referencie a través de su nombre.

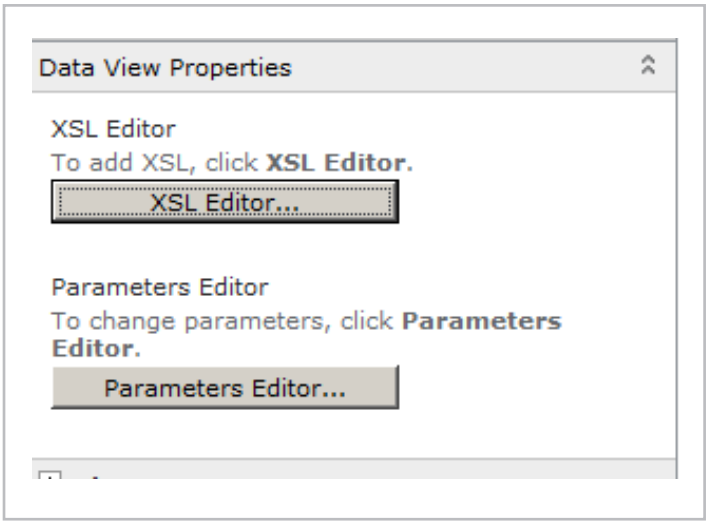


Imagen 15.- Edición del web part desde el browser

SILVINA ANDREA PIZZARULLI
Especialista SharePoint en Baufest
http://silvinapizzarulli.blogspot.com

26

Workaround para permitir actualizar un término usando RunWithElevatedPrivileges

Resumen

En este artículo vamos a mostrar un workaround para resolver un pequeño bug de SharePoint 2010, que aparece cuando necesitamos actualizar un término del Servicio de Metadatos Administrados, y el usuario del contexto no tiene permisos sobre el Almacén de términos.

Artículo

Partimos del siguiente escenario. Tenemos un conjunto de términos TermSet abierto, donde utilizando el control estándar de SharePoint, se han creado varios términos. Además, tenemos la necesidad desde código, de poder acceder a un término Term, y actualizar alguna de sus propiedades, por ejemplo para añadirle una propiedad personalizada, que podría indicarnos la última fecha/hora en la que se utilizó el término, o el número de veces que ese término se ha utilizado.

Para acabar con la descripción del escenario, tenemos una última restricción, y es que el usuario del contexto, con el que se ejecutará nuestro código, no tiene permisos de administración sobre el conjunto de términos TermSet.

Para ilustrar mejor el escenario, observemos la Imagen 1, donde tenemos un control TaxonomyWebTaggingControl que está enlazado a un conjunto de términos. Además, tenemos un botón "Update Term", que intentará añadir una Custom Property al Term seleccionado.

Nota: En el número de Septiembre de 2012 de CompartiMOSS, tenemos un artículo que explica como configurar y enlazar el control TaxonomyWebTaggingControl.

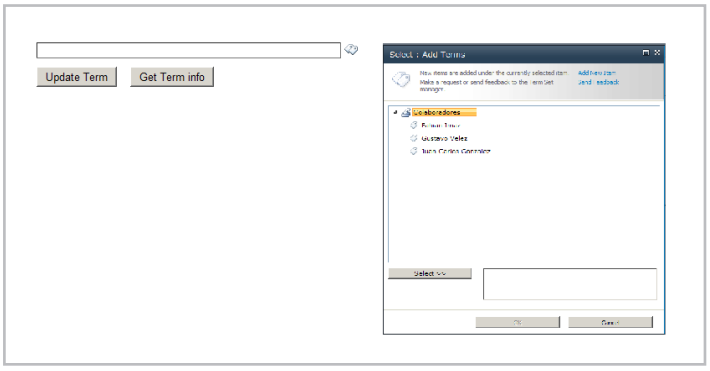


Imagen 1.- Escenario de partida.

Como en nuestro escenario, estamos logados con un usuario que no es administrador del Almacén de términos, si intentamos actualizar el Term, tendremos un error de permisos. Para que nuestro código se ejecute con los permisos suficientes, usaremos la función `SPSecurity.RunWithElevatedPrivileges`

Como ya sabemos, al usar *RunWithElevatedPrivileges*, nuestro código se ejecutará con la cuenta del sistema de SharePoint. Debemos asegurarnos que la cuenta del sistema, sí es administrador del Almacén de términos (por lo general, lo será).

El Listado 1 contiene el código necesario para recuperar el término seleccionado, e intentar actualizar una propiedad, usando *RunWithElevatedPrivileges*.

```
SPSecurity.RunWithElevatedPrivileges(delegate
{
    using (SPSite site = new SPSite(SPContext.Current.Site.ID))
    {
        using (SPWeb web = site.OpenWeb())
        {
            string windowsUserName =
                WindowsIdentity.GetCurrent().Name; // DOMAIN\sp_apppool (It's my system account)
            string appPoolAccountName = web.CurrentUser.Name;  // system account

            //Get Term guid of Term picked
            TaxonomyFieldValueCollection tagsCollection =
                TaxonomyFieldControl.GetTaxonomyCollection(myTaxonomyControl.Text);
            string termGuid = tagsCollection[0].TermGuid;

            //Get Term object from its Guid
            TaxonomySession taxonomySession = new TaxonomySession(site);
            TermStore termStore = taxonomySession.DefaultSiteCollectionTermStore;
            Term term = termStore.GetTerm(new Guid(termGuid));

            string customPropertyValue = DateTime.Now.ToString();
            term.SetCustomProperty("Foo", customPropertyValue);

            termStore.CommitAll();
        }
    }
});
```

Listado 1.- Código para actualizar el Term seleccionado usando *RunWithElevatedPrivileges*.

Si lanzamos ese código (con algunas líneas más de debug), con un usuario que no es administrador del almacén de términos, obtendremos el resultado de la Imagen 2.

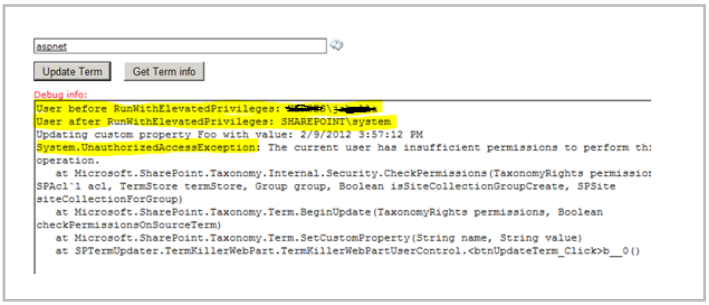


Imagen 2.- Error de permisos al lanzar el código del listado 1.

Si nos fijamos en las 2 primeras líneas de la caja de texto, vemos como en la primera, nos muestra el usuario logado, antes de ejecutar el RunWithElevatedPrivileges. En la siguiente línea, observamos el usuario logado, una vez dentro del RunWithElevatedPrivileges, que, como cabría de esperar, es el usuario de la cuenta del sistema.

Sin embargo, obtenemos la misma excepción de permisos insuficientes, al intentar actualizar el término. Esto es debido a un pequeño bug de SharePoint 2010 que hace que el nuevo contexto que se crea al usar RunWithElevatedPrivileges, no tenga correctamente actualizados los permisos relativos al almacén de términos.

Por suerte, tenemos un sencillo workaround, que forzará a que el nuevo contexto se genere refrescando también los permisos relativos al almacén de términos.

El concepto del workaround es sencillo, y se resume en estos 4 pasos:

1. Hacemos una copia del HttpContext actual.
2. Limpiamos el HttpContect actual, lo que limpiará también el SPContext.
3. Creamos el nuevo contexto, con el New del SPSite y el SPSite.OpenWeb. Este paso sería el código anterior del Listado 1.
4. Restauramos el contexto con el HTTPContext que hemos guardado en el paso 1.

El Listado 2 contiene el código completo, excepto parte del paso 3, que ya está en el listado 1.

```
HttpContext backupCtxt = HttpContext.Current;
Guid backupCurrentSiteID = SPContext.Current.Site.ID;

SPSecurity.RunWithElevatedPrivileges(delegate
{
    if (SPContext.Current != null)
    {
        HttpContext.Current = null;
    }

    using (SPSite site = new SPSite(backupCurrentSiteID))
    {
        using (SPWeb web = site.OpenWeb())
        {
            //Get and Update Term
        }
    }
});

if (SPContext.Current == null)
{
    HttpContext.Current = backupCtxt;
}
```

Listado 2.- Código completo solución al problema planteado.

"... cuando necesitamos actualizar un término del Servicio de Metadatos Administrados, y el usuario del contexto no tiene permisos sobre el Almacén de términos."

LUIS MÁÑEZ
MCPD SharePoint 2010 /
Microsoft Active Professional 2012
http://geeks.ms/blogs/lmanez/
http://twitter.com/luismanez

28

Sitios de publicación con SharePoint 2013: ¿Cómo se hizo CompartiMOSS? – Parte I

Resumen

SharePoint 2013 facilita la creación de sitios de publicación habilitando que los diseñadores puedan crear prácticamente cualquier tipo de diseño qué pueda ser "consumido" por la plataforma utilizando para ello herramientas de diseño clásicas en lugar del tradicional SharePoint Designer.

Artículo

En SharePoint 2013 se ha potenciado la presencia de los sitios de publicación dado el radical cambio que ha sufrido la plataforma en su actualización.

El primero y más importante de estos cambios lo podemos encontrar en el núcleo de los sitios de SharePoint, HTML + JavaScript componen la estructura sea cual sea la plantilla, con lo que no será tan complicado realizar Branding en una Intranet al igual que en los sitios de Publicación.

El segundo de los cambios tiene efecto en la libertad que se otorga a los diseñadores para usar el programa que deseen para modelar con HTML, JavaScript y CSS el diseño de los sitios de SharePoint. De esta forma, no tendrán que aprender a usar SharePoint Designer ni hacer un desarrollo en su programa preferido para después irlo trasladando a SharePoint paso a paso. A partir de ahora, podrán controlar sus páginas maestras, hojas de estilo, scripts de JavaScript, etc., desde su programa habitual con sólo conectarse a la dirección de los ficheros de diseño del sitio.

Finalmente, desde Microsoft se han afanado en facilitar la vida al diseñador, así que han dispuesto un nuevo lugar de administración sólo para diseño denominado Administrador de diseños (Design Manager) desde donde los diseñadores podrán perfilar sus páginas maestras, los diseños de página y las plantillas para elementos de contenido.

Comenzar un sitio de publicación de SharePoint 2013

Para mostrar un ejemplo claro, explicaré paso a paso la construcción del nuevo portal de CompartiMOSS (http://www.compartimoss.com) para el que seguimos los siguientes pasos.

1. Diseño gráfico
2. Traducción del diseño gráfico a una plantilla HTML
3. Exportar boceto con Design Manager
4. Completar página maestra con Design Manager y programa de edición web
5. Crear diseños de página (Parte II)
6. Definición de plantillas de elementos de contenido (Parte II)

Diseño gráfico

Para empezar, establecimos diferentes ideas de cómo debía ser el portal de CompartiMOSS para que resultara atractivo pero que sobre todo, sirviera a su cometido, que no es otro que facilitar la lectura a los usuarios. Además, también teníamos que tomar como referencia la aplicación que ya hemos desarrollado para Windows 8. Para ello, usamos un programa de edición gráfica con el que se creó el diseño definitivo que podéis apreciar.

Imagen 1.- Diseño de CompartiMOSS.

Traducción del diseño gráfico a una plantilla HTML

Una vez tenemos el diseño decidido, el siguiente paso fue traducirlo como plantilla HTML, con las imágenes, hojas de estilo y scripts. No es necesario que se programe la navegación, ni siquiera el contenido, sino tan sólo una página principal, porque lo que vamos a realizar en este caso, sería el paso previo para generar la página maestra con sus estilos, scripts y recursos personalizados. El resto de elementos los veremos más adelante. Antes de continuar, es necesario introducir

la estructura de directorios en la que me he basado y que recomiendo encarecidamente:

- **Nombre de proyecto:**
  - CSS.
  - Images.
  - JavaScript-
  - Fichero HTML principal con el nombre del proyecto.

Para realizar esta tarea, podemos usar el programa de edición que más nos guste, lo que facilitará notablemente el trabajo. Para CompartiMOSS hice uso de WebMatrix que funciona a las mil maravillas con este tipo de trabajos.

Exportar boceto con el Administrador de diseños

Los chicos de Microsoft nos han puesto las cosas bastante fáciles gracias al Administrador de diseños desde donde podremos realizar todos los cambios necesarios en cuestiones de diseño, desde administrar los canales de los diferentes dispositivos que accedan a nuestro sitio hasta exportar un paquete con nuestras personalizaciones.

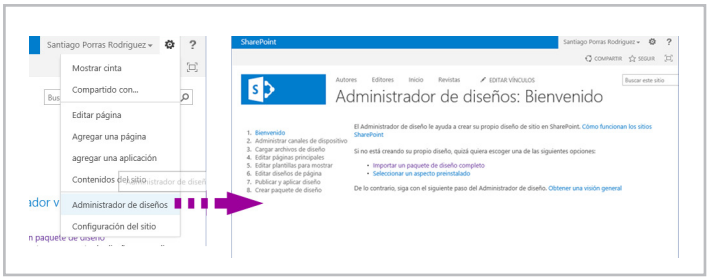


Imagen 2.- Acceso al administrador de diseños.

En nuestro caso, debemos empezar por cargar la plantilla HTML que hemos creado, para lo que al seleccionar la opción Cargar archivos de diseño, SharePoint nos ofrece una dirección que debemos abrir con un explorador de archivos (recomendado que se establezca como unidad de red), donde podremos ubicar la plantilla que generamos en el paso anterior.

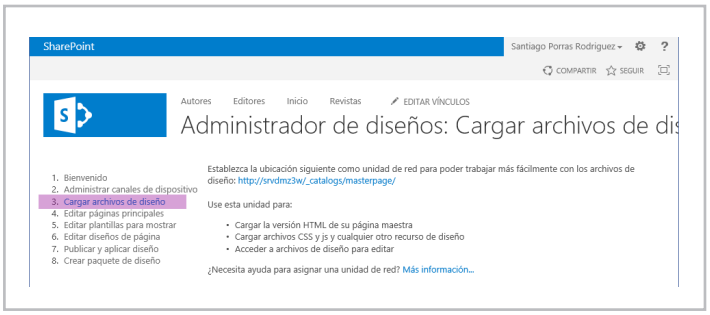


Imagen 3.- Adjuntado del diseño en el Administrador de diseños.

Tan solo tendríamos que copiar la estructura de directorios (completa), que os propuse en el paso anterior, dentro de la carpeta *nombre-servidor/_catalogs/masterpage* para tener la Página Maestra creada.

NOTA: *Esta acción generará un archivo .master con el nombre del fichero HTML de la plantilla que se regenerará cada vez que modifiquemos el fichero .html.*

IMPORTANTE: *A partir de ahora, trabajaremos con los ficheros copiados y no los originales, es decir, debemos conectar nuestro programa de edición web a la estructura de directorios de SharePoint.*

Completar página maestra con Design Manager y programa de edición web

Llegados a este punto, ya tenemos creada nuestra Página Maestra sin haber hecho más esfuerzos que podremos centrar en agregar los elementos de SharePoint que necesitamos, tales como el menú.

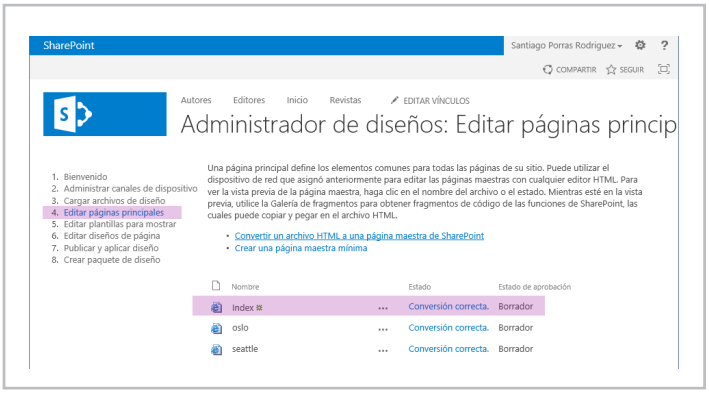
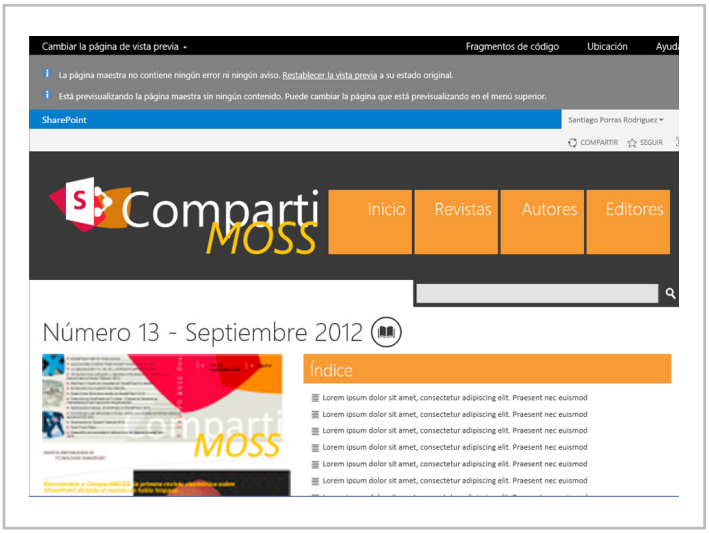


Imagen 4.- Edición de la Página Maestra en el Administrador de diseños.

Como podéis comprobar en la imagen, aparece una nueva Página Maestra llamada index que se corresponde con la que acabo de crear. Si seguisteis mi recomendación de ponerle el nombre del proyecto al archivo HTML de la plantilla, os saldría en este caso una Página Maestra con el nombre del proyecto. Ahora debemos proceder a añadir los elementos de SharePoint que necesitamos, tales como el menú. Para ello, desde la opción Editar páginas principales, sólo tenemos que seleccionar la Página Maestra que acabamos de crear, lo que nos llevará a la página de vista previa de la misma.



magen 5.- Vista previa de la página.

Desde está página podremos seleccionar la opción Fragmentos de código que nos permitirá generar el código necesario

para cada uno de los elementos que necesitemos insertar en nuestra Página Maestra. Menú de navegación, título, logotipo, etc. pueden ser generados desde aquí y, posteriormente copiados a nuestra plantilla HTML.

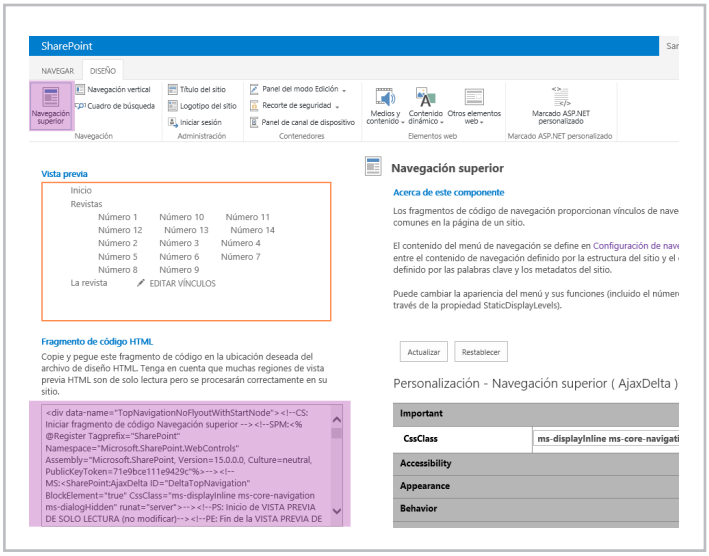


Imagen 6.- Inserción de fragmentos de código en el Administrador de diseños.

Como se puede ver en la imagen, si se selecciona uno de los elementos que aparecen en la Cinta, se podrá ver debajo una vista previa, el Fragmento de código HTML generado y, en la derecha de la página, una zona para definir las propiedades del elemento al estilo de Visual Studio.

Para la Página Maestra de CompartiMOSS se ha hecho uso de los siguientes elementos:

- Navegación superior.
- Cuadro de búsqueda.
- Logotipo del sitio.

"... En SharePoint 2013 se ha potenciado la presencia de los sitios de publicación..."

Imagen 7.- Vista previa del HTML insertado.

Estos elementos se han de insertar en el fichero HTML de nuestra plantilla, pero como avisé anteriormente, deben ser hechos en la plantilla que hemos copiado dentro de la carpeta *_catalog/masterpages*. Al guardar los cambios en este fichero HTML, automáticamente se regenerará el fichero .master que es realmente la Página Maestra que podrá usar SharePoint.

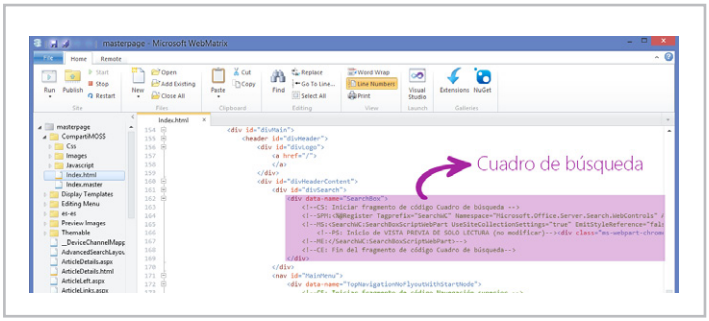


Imagen 8.- Archivo .html a partir del qué se va a generar la página maestra.

Una vez hayamos completado la Página Maestra con los elementos de SharePoint vamos a publicarla y aplicarla a nuestro sitio por medio de la opción Publicar y aplicar diseño desde el Administrador de diseños.

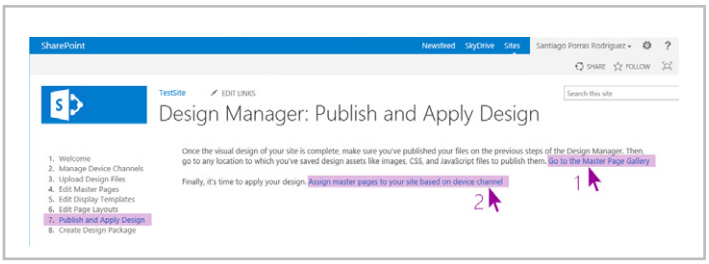


Imagen 9.- Aplicación de la Página Maestra por medio del Administrador de diseños.

Publicarla significa publicar todos y cada uno de los elementos que componen nuestra plantilla HTML, esto es, archivos html, hojas de estilo, imágenes y scripts. Para ello, seleccionando el vínculo marcado como 1 en la imagen anterior, llegaremos a la página de Páginas maestras y diseños de página a la que también podemos llegar desde Configuración de sitio. Una vez aquí deberemos ir publicando todos los elementos.

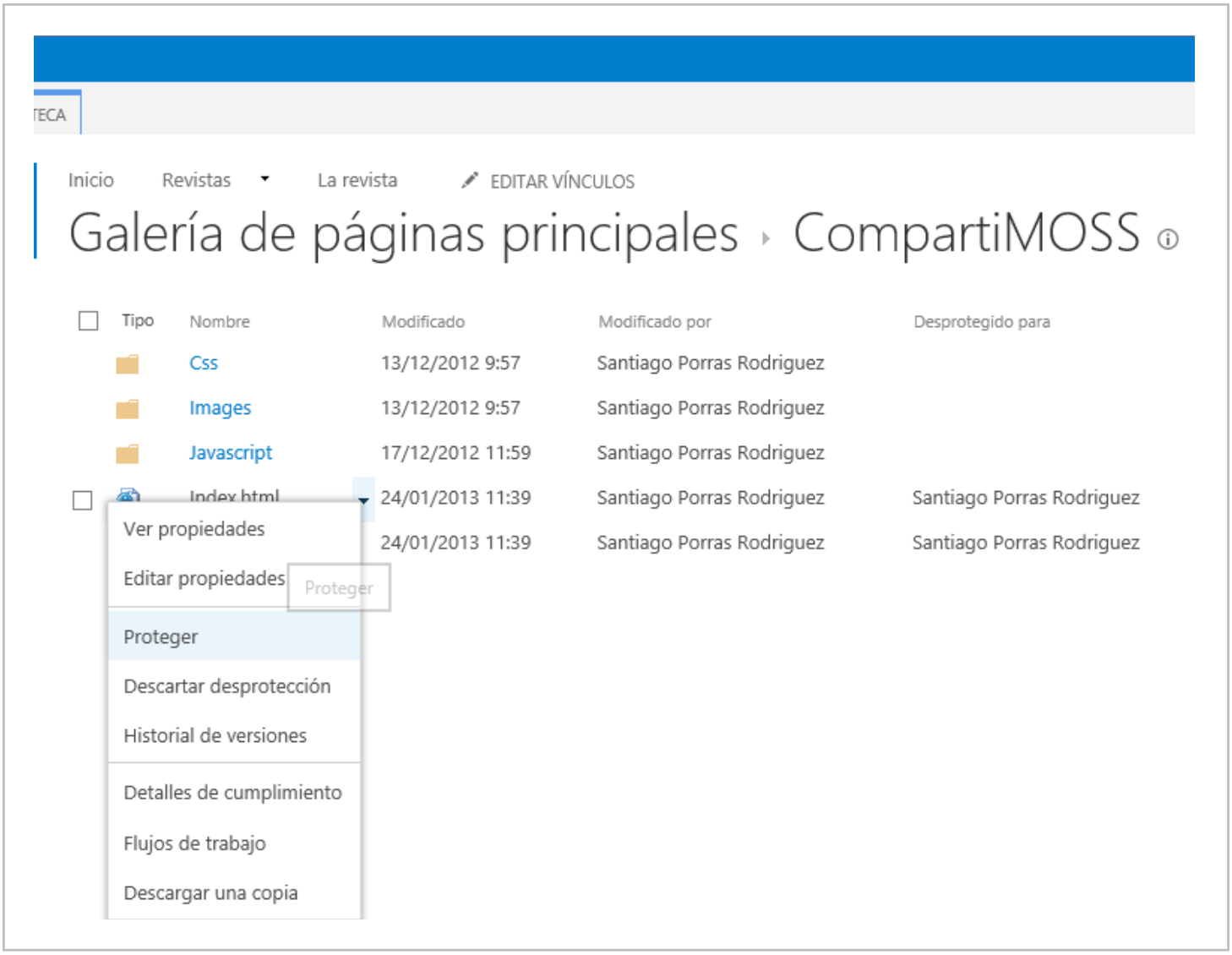


Imagen 10.- Publicación de los elementos de diseño.

Ahora que tenemos todo publicado, sí que podemos aplicar la Página Maestra a nuestro sitio, acción que podemos lograr seleccionando el vínculo marcado como 2 en la imagen de la opción *Publicar y aplicar diseño*.

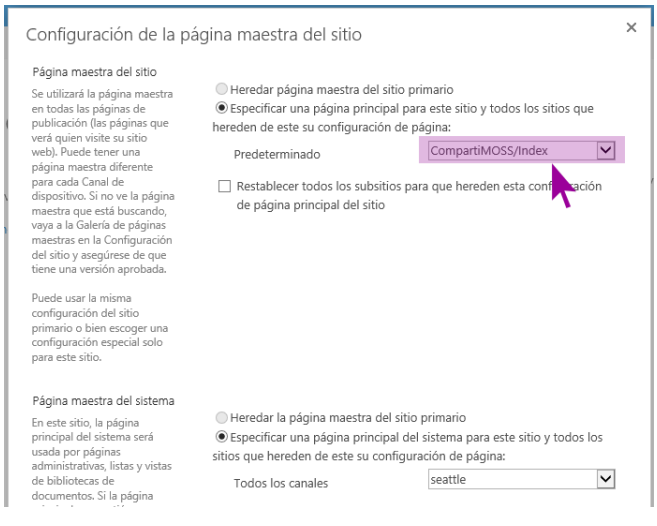


Imagen 11.- Configuración de la página maestra del sitio.

Finalmente, ya tenemos aplicada la Página maestra como podréis comprobar y estamos un paso más cerca del resultado final que completaremos en un próximo artículo donde trataremos los diseños de página y las plantillas de elementos de contenido.

Imagen 12.- Página maestra aplicada al sitio.

SANTIAGO J. PORRAS RODRÍGUEZ
UX Developer
http://geeks.ms/blogs/santypr
@saintwukong

32 SharePoint como Sistema Colaborativo

Resumen

Los tiempos en la actualidad han cambiado y esto le ha exigido a las organizaciones modernizar su gestión de negocios, para ello han tenido que buscar tiempos de respuesta más ágiles, mejor comunicación y sobretodo una mejor colaboración. SharePoint viene a ser una herramienta completa que abarca todos estos aspectos que actualmente buscan las organizaciones.

Artículo

Hoy en día los Sistemas Colaborativos son primordiales en la gestión de procesos de las organizaciones. Tiempo atrás estos sistemas eran asincrónicos, por ejemplo: los envíos de correos electrónicos, pero las compañías en este momento necesitan cierto dinamismo en su proceso de negocios para poder rendir ante el mundo tan competitivo al que nos enfrentamos.

La colaboración en estos tiempos se ha vuelto necesaria, y las herramientas como SharePoint vienen a facilitar nuestros procesos en todo momento, desde la construcción, pasando por la preparación (resistencia al cambio) y hasta en la utilización en el día a día.

Los Sistemas basados en Entornos Colaborativos, son ideales para grupos de personas dentro de distintas actividades en las que existen necesidades importantes de comunicación y colaboración.

En la actualidad, existen en el mercado diversas herramientas colaborativas, pero entre todas estas herramientas se opta por elegir Microsoft SharePoint por diversos motivos:

1. Los productos Microsoft Office que son altamente utilizados en las organizaciones. Por esto SharePoint es una herramienta atractiva, debido a que son productos conocidos y utilizados por los usuarios.
2. El integrar herramientas propias de la organización o de otros proveedores, al entorno de Microsoft SharePoint se realiza de manera fácil.
3. Microsoft SharePoint está orientada a la organización, no se necesita ser Ingeniero en Informática para poder manejar la herramienta, de modo que los usuarios van a poder abordar sus necesidades de forma autosuficiente y con menos “Resistencia al Cambio”.
4. Es una tecnología adecuada para la colaboración, no solo colaboración documental sino en el más amplio significado de colaboración en una organización (tareas, eventos, reuniones, discusiones...).

Imagen 1.- SharePoint cuenta con una diversidad de listas y aplicaciones dentro del Contenido del Sitio, para ser utilizado en el momento que se necesiten.

5. La herramienta cuenta con bastante documentación por lo que el capacitarse para la utilización de SharePoint resulta una tarea fácil.

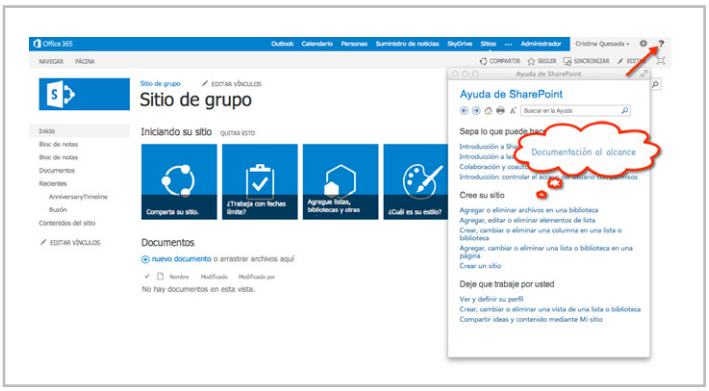


Imagen 2.- SharePoint cuenta con documentación al alcance.

La herramienta colaborativa Microsoft SharePoint se basa en Microsoft SharePoint Server tanto la versión Standard Como la Enterprise y Microsoft SharePoint Foundation. SharePoint Server es un servidor de portales Web que va a permitir a los usuarios integrar diversas aplicaciones, personalizar el contenido, y hacer búsquedas avanzadas.

Microsoft SharePoint Foundation permite agregar, organizar y ofrecer sitios para facilitar el compartir documentos y la colaborar en proyectos y reuniones, crear y utilizar plantillas, y

gestionar el control de versiones y publicaciones.

SharePoint conecta los sitios de trabajo y los distintos usuarios que conforman los equipos, proporcionando organizaciones más eficientes.

"... buscar tiempos de respuesta más ágiles, mejor comunicación y sobretodo una mejor colaboración."

De manera natural al comenzar a utilizar cualquier herramienta informática se comienza por estudiar el manual de usuario de la misma. En el caso concreto de Microsoft SharePoint, el sistema debe ser adaptado a las necesidades de los usuarios cada vez que se implementa.

Microsoft SharePoint es una herramienta dirigida al usuario final, por lo que al utilizarla para potenciar los procesos de negocios la resistencia al cambio es mucho menor que cuando se utiliza alguna otra herramienta programada para este fin.

La buena y fácil capacitación al usuario hace que se trabaje de manera independiente.

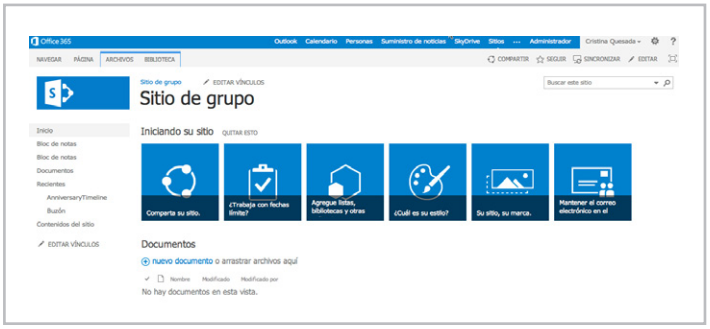


Imagen 3.- El trabajo en SharePoint es fácil y dinámico.

Recomendaciones a la hora de implementar un sistema de colaboración con SharePoint

1. Saber que un sistema de colaboración no se desarrolla solamente con la experiencia de un conocedor de la herramienta o un técnico, sino lo principal es conocer bien el proceso de negocio a desarrollar, para esto necesitamos una toma de requerimientos bien robusta con la ayuda de múltiples áreas (mercadeo, recursos humanos, gerencia, etc.).
2. SharePoint por su naturaleza web, tiene la flexibilidad de conformar jerarquías de sitios, por lo cual hay que tener mucho cuidado con este aspecto, ya que se recomienda no navegar a más de tres niveles, debido a que el usuario busca encontrar la información en la menor cantidad de clics.
3. Saber traducir la necesidad tanto funcional como la administración de la información para un buen acople del trabajo en la organización.
4. Tener en cuenta la buena formación que debe tener el usuario final para el buen uso de la herramienta y la seguridad de que SharePoint como sistema de colaboración cumple las expectativas.

Conclusiones

SharePoint es una herramienta de colaboración orientada al usuario final. Al saber bien que procesos de negocios tenemos que potenciar en la organización se podrá sacar el máximo de provecho a la herramienta.

Se debe de tener presente todas las áreas involucradas en el proceso. No es cosa solo de una persona el poder diseñar e implementar un sistema colaborativo como SharePoint.

Se deben tener siempre presente las 3 C's:

- Comunicación: esta es la función más importante donde se establece el flujo de información.
- Coordinación: asegura que el equipo de trabajo es eficiente y se alcanza el objetivo de manera conjunta.
- Colaboración: proporciona ventajas para resolver problemas a tiempo.

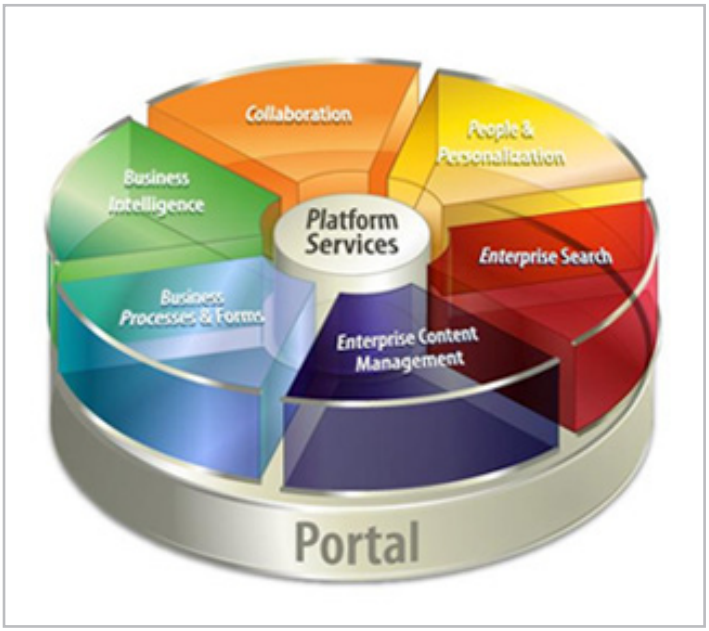


Imagen 4.- Todo un mundo de colaboración en una sola herramienta.

CRISTINA QUESADA CALDERÓN
Consultora en tecnologías SharePoint
Profesora de Sistemas Colaborativos Universidad Latina de Costa Rica
cristi_q@hotmail.com
@cris_quesada
http://cristina-quesada.blogspot.com/

34 Governance Q&A con Jeremy Thake

Resumen

En esta entrevista, Jeremy Thake, arquitecto jefe en AvePoint y Microsoft SharePoint MVP, da su opinión sobre una de las áreas más importantes para que su implementación de SharePoint sea un éxito - las políticas de governance.

Artículo

¿POR QUÉ UNA FUERTE POLÍTICA DE GOVERNANCE TIENE TANTA IMPORTANCIA PARA ASEGURAR EL CONTROL DE SHAREPOINT?

Según el uso de SharePoint se incrementa y se desarrolla en una empresa, es fácil generar grandes cantidades de datos no estructurados que dan como resultado una plataforma voluminosa e ineficiente. Por esta razón, es necesario tener una política de governance para controlar el crecimiento de los datos de una manera que sea fácil de manejar para los equipos de IT.

Uno de los mayores retos dentro de SharePoint es la responsabilidad. Mientras que una persona puede ser responsable de la creación de un sitio de SharePoint hoy, si cambian de puesto de trabajo, existe el riesgo de que el sitio se quede sin nadie asignado para hacerse cargo de la responsabilidad y de su mantenimiento. Esto puede traer como consecuencia que los sitios pierdan rápidamente su utilidad, obligando a los usuarios a encontrar otras formas de compartir información.

Por lo tanto, una parte esencial dentro de las políticas de Governance en SharePoint es la de asignar responsabilidades y asegurar que esta tarea se mantiene. Esto es algo que puede ser difícil de controlar utilizando las funcionalidades nativas de SharePoint, lo que hace que sean atractivas las herramientas de terceros destinadas al Governance.

¿CÓMO DEBERÍA UNA EMPRESA ELABORAR UNA POLÍTICA DE GOVERNANCE DE SHAREPOINT? ¿EN QUÉ MEDIDA SE DEBE ALINEAR CON LA ESTRATEGIA GENERAL DE LA EMPRESA?

Cada política de governance en la organización debe alinearse con la estrategia de negocio. Un plan de SharePoint Governance, al igual que un plan para cualquier otro software, debe coincidir con los objetivos del negocio. Por lo tanto, una ayuda seria tener un representante corporativo o ejecutivo que pueda apoyar y ayudar a implementar el plan. Nadie va a prestar atención a la hora de hacer cumplir las reglas y regulaciones si se aplican desde abajo hacia arriba. La manera más eficaz es hacer que el consejo de administración y el departamento de TI implementan esas reglas desde arriba hacia abajo.

¿CUÁLES SERÍAN LOS PRIMEROS PASOS MÁS IMPORTANTES A LA HORA DE INTRODUCIR UNA ESTRATEGIA DE GOVERNANCE?

A la hora de elaborar una política, hágalo teniendo en cuenta que el plan definirá la relación entre el equipo de negocio que quiere soluciones y el equipo de TI que tiene que superar retos. Establecer expectativas y límites en el plan de governance por lo tanto puede ayudarle a gestionar la relación entre ambos.

Cuando se introduzca una estrategia de governance, es importante definir cuánto tiempo se necesitará para construir la introducción - no solo la creación, sino cómo el plan se comunicará y cómo la empresa va a hacer cumplir las políticas incluidas en el plan. Al establecer las expectativas para lo que ambos equipos pueden y no pueden hacer, los problemas se podrán tratar más rápida y fácilmente cuando aparezcan.

¿CUÁL ES UNO DE LOS PUNTOS PROBLEMÁTICOS MÁS FRECUENTES AL QUE SE ENFRENTAN LAS ORGANIZACIONES AL IMPLEMENTAR O MANTENER UNA IMPLEMENTACIÓN DE SHAREPOINT, Y CÓMO PUEDE AYUDAR UNA POLÍTICA DE GOVERNANCE?

Muchas empresas se enfrentan a retos que hacen referencia al backup y recovery de SharePoint. Mientras que SharePoint tiene integradas las capacidades de protección de datos, las empresas pocas veces pasan por un escenario de disaster recovery para comprobar que se puede volver a la normalidad dentro de un plazo determinado, en caso de una interrupción.

El propósito de un plan de governance en este caso sería definir cuánto tiempo sería necesario para la recuperación utilizando la capacidad del hardware y software actual. Si la empresa piensa que tardará demasiado tiempo, entonces una discusión puede tener lugar para resolver lo que necesita para acelerar la recuperación. Luego, cuando lo inesperado ocurre, el departamento de TI conoce el proceso para evitar la interrupción y puede fijar un plazo para restaurar la funcionalidad con la confianza de que la empresa va a apoyarlos.

¿CON QUÉ FRECUENCIA DEBE UNA EMPRESA REVISAR SU ESTRATEGIA DE GOVERNANCE?

Muchas organizaciones dicen que van a revisar su política cada trimestre, pero en realidad es un periodo demasiado corto de tiempo para saber lo que hay que cambiar. En cambio, las empresas suelen revisar su política una vez o dos veces al año.

Sin embargo, la clave para medir el éxito de una política es establecer métricas por adelantado sobre las que la empresa pueda realizar un seguimiento. Sin esto, no hay suficientes evidencias como para apoyar ninguna conclusión. Dicho esto, la medición de métricas dentro de SharePoint puede ser difícil. Es por eso, que la mayoría de las empresas adquieren soluciones de terceros para ayudar en este proceso.

¿QUÉ FUNCIONALIDADES DEBEN BUSCAR LOS CLIENTES A LA HORA DE IMPLEMENTAR UNA SOLUCIÓN DE GOVERNANCE PARA MICROSOFT SHAREPOINT?

Al comprar una solución de terceros, se hace necesario que los empleados puedan usarla intuitivamente sin necesidad de formación. Por ejemplo, con nuestra plataforma DocAve, hemos construido nuestro catálogo de servicios para que los usuarios finales puedan ver aquellas áreas que necesitan para realizar una función específica.

Si una política de gobierno establece que alguien que trabaja en ventas puede crear un sitio para un cliente, pero necesita la aprobación de dos supervisores para hacerlo, los empleados que trabajan en Recursos Humanos no tendrán por qué ver dicha petición. Hemos creado interfaces dinámicas que se basan en el perfil de la persona lo que hace que cumplimentar los formularios de solicitud sea muy simple porque solo muestra los campos que se necesitan completar.

¿CÓMO PUEDE UNA POLÍTICA DE GOVERNANCE AYUDAR A LAS COMPAÑÍAS TRATAR CON LAS FUTURAS TENDENCIAS DEL SECTOR?

Espero que las futuras versiones de SharePoint tengan diferentes estados en los que las personas puedan trabajar. Tendremos el habitual estado en la oficina; el estado en remoto en el que trabajaremos cuando tengamos una hora disponible desde un aeropuerto o desde cualquier otra localización; y un estado de trayecto, en el que sólo tendremos unos minutos para realizar una tarea mientras estamos de paso. La colaboración entre estos tres estados requerirá diferentes formas de trabajar con SharePoint. En ese sentido, una política de governance puede ayudar a definir los procedimientos para estas diferentes áreas, tales como la aplicación de los dispositivos móviles con estrictas medidas de seguridad, ayudando al departamento de IT a entender el lugar de trabajo como algo fragmentado y móvil.

Si desea obtener más información sobre cómo implementar un plan de governance, puede registrarse en nuestro próximo webcast presentado por AvePoint y Encamina. El webcast incluye:

Las mejores prácticas a la hora planificar y escribir sus políticas organizacionales de governanza

Una demo que presentará cómo se puede utilizar la plataforma de DocAve de AvePoint para soportar las políticas de gestión destinadas al contenido y la disponibilidad del mismo, y a la vez, cumpliendo con el acceso de políticas de gestión del ciclo de vida de contenido.

Si desea saber más sobre el próximo webcast, por favor póngase en contacto con ventas@avepoint.com

JEREMY THAKE
MVP SharePoint
@jthake
http://www.jeremythake.com/
http://www.avepoint.com

"... es fácil generar grandes cantidades de datos no estructurados que dan como resultado una plataforma voluminosa e ineficiente."

36 Cargar documentos en SharePoint utilizando Servicios Web

Resumen

Este artículo pretende aproximarnos al funcionamiento de algunos servicios web que expone SharePoint a terceros para copiar programáticamente un documento desde una carpeta local hasta una biblioteca de SharePoint.

Artículo

Existen varias maneras de cargar documentos en una biblioteca de documentos de SharePoint. Algunas son procesos manuales como el que ofrece la propia herramienta o a través del "Guardar como" si estamos hablando de documentos Office. También existen procesos automáticos para este propósito. Uno de ellos son los servicios web.

Sólo debe considerarse como una aproximación ya que no se establecen mecanismos para controlar los errores que se puedan producir por denegación de permisos, tamaño del archivo, carga de archivos ejecutables, etc.

Trabajaremos entonces con el servicio web copy.asmx (operación CopyIntoItems), aunque será fácilmente extensible a los servicios Imaging.asmx (operación Upload) y Lists.asmx (operación AddAttachment). Mientras que el primero nos permite cargar documentos en una biblioteca de documentos el segundo nos permite cargar imágenes en una biblioteca de imágenes y el tercero añadir un adjunto a un elemento de una lista.

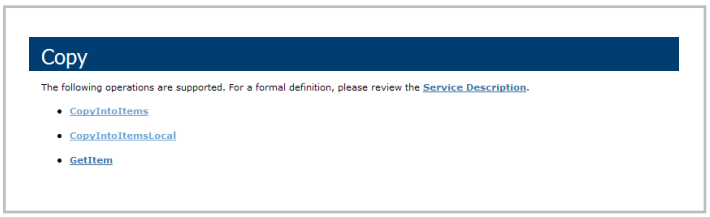


Imagen 1.- Servicio copy.asmx de SharePoint.

Herramienta

Utilizaremos en este caso Visual Basic Script Edition por la facilidad que tiene de ejecución en cualquier estación de trabajo o servidor.

Preliminares

Para conocer las operaciones disponibles del servicio copy.asmx deberemos ejecutar en nuestra instalación de SharePoint la URL: http://<<Misitio>>/_vti_bin/copy.asmx y si queremos conocer la sintaxis de la operación CopyIntoItems : http://<<Misitio>> /_vti_bin /copy.asmx?op=CopyIntoItems.

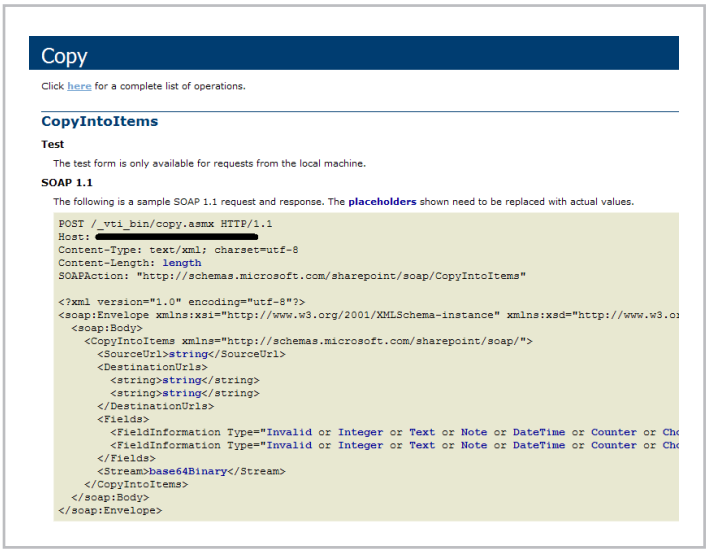


Imagen 2.- Detalle de la operación CopyIntoItems del servicio copy.asmx.

Es aquí donde nos damos cuenta que el archivo que carguemos necesita ir codificado en base64Binary. Crearemos objetos que nos permitirán realizar las cuatro funciones principales de las que consta esta aproximación:

- Leer el contenido del archivo en modo binario.
- Codificar dicho contenido a base64Binary.
- Construir y ejecutar una petición Http (SOAP 1.1) para cargar el documento.
- Construir y ejecutar dos peticiones Http (SOAP 1.1) que nos permitirán actualizar el documento cargado.

Definición de constantes y objetos

Utilizaremos dos constantes: Ruta y nombre del archivo local, URL de la biblioteca de SharePoint incluyendo nombre de archivo destino.

Utilizaremos además cuatro objetos: Un objeto Stream para la lectura del archivo, un elemento de un objeto XMLDOM para la codificación del documento y para la recuperación de la respuesta del servidor y un objeto XMLHTTP para la ejecutar las peticiones.

```
ConstArchivoLocal = "Ejemplo.pptx"
ConstURLDestino =
"http://.../Biblioteca/Ejemplo.pptx"


SetObjetoStream = CreateObject("ADODB.Stream")
SetObjetoDOM = CreateObject("Microsoft.XMLDOM")
SetObjetoElemento = ObjetoDOM.CreateElement("TMP")
SetObjetoHTTP = CreateObject("Microsoft.XMLHTTP")
```

Listado 1.- Objetos a utilizar para cargar un documento

Lectura del contenido del archivo y Codificación

A continuación obtenemos el archivo codificado en base64Binary

```
'Lectura del archivo en binario
ObjetoStream.Open
ObjetoStream.type= 1'Tipo Binario
ObjetoStream.LoadFromFile(ArchivoLocal)
ArchivoBinario = ObjetoStream.Read()
ObjetoStream.Close

'Conversion a Base64
ObjetoElemento.DataType = "bin.base64" 'Tipo Base64
ObjetoElemento.NodeTypedValue = ArchivoBinario
ArchivoCodificado = ObjetoElemento.Text
```

Listado 2.- Archivo a subir codificado en base64Binary

Construcción y ejecución de la Petición de carga del documento

Tal y como se ve en la Imagen 2, la operación SOAP CopyIntoItems se basa en un archivo XML que debemos construir con las constantes definidas y las variables que hemos obtenido. Analizaremos algunos elementos:

- <SourceURL> Contiene el origen del documento a copiar. Este valor es utilizado por SharePoint para recuperar el origen de la copia del documento. Este valor no puede estar en blanco. Como en nuestro caso el origen es un archivo local ajustaremos este dato a "C:/" aunque posteriormente deberemos realizar otras peticiones que nos permitan modificar el valor.
- <DestinationUrls> Una serie de Urls donde queremos copiar el archivo origen. Podemos hacer diferentes copias en diferentes bibliotecas desde el mismo origen.
- <Fields> Una serie de valores que se asignan a columnas definidas en las bibliotecas. En nuestro caso establecemos el valor: "Archivo cargado con SOAP" a la columna Título.
- <Stream> Documento a copiar debidamente codificado que hemos obtenido previamente.

```
'Construye texto Peticion de carga del documento
URLServicio = "http://.../_vti_bin/copy.asmx"
AccionSOAP =
"http://schemas.microsoft.com/sharepoint/soap/CopyI
ntoItems"
Peticion="<?xmlversion='1.0' encoding='utf-8'?>"+_
"<soap:Envelopexmlns:xsi='http://www.w3.org/2001/XM
LSchema-instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soap='http://schemas.xmlsoap.org/soap/envelop
e/'>"+_
"<soap:Body>"+_
"<CopyIntoItemsxmlns='http://schemas.microsoft.com/
sharepoint/soap/'>"+_
"<SourceUrl>C:/</SourceUrl>"+_
"<DestinationUrls>"+_
"<string>"+URLDestino+"</string>"+_
"</DestinationUrls>"+_
"<Fields>"+_
"<FieldInformation Type='Text' InternalName='Title'
DisplayName='Titulo' Value='Archivocargado con
SOAP' />"+_
"</Fields>"+_
"<Stream>"+ArchivoCodificado+"</Stream>"+_
"</CopyIntoItems>"+_
"</soap:Body>
</soap:Envelope>"
```

Listado 3.- XML para pasarle al Servicio Web

EjecutaPeticion

Una vez construida la petición podemos enviarla al servidor por medio del procedimiento que utiliza el objeto "Microsoft. XMLHTTP"

```
Private Sub EjecutaPeticion
ObjetoHTTP.Open "Get", URLServicio, false
ObjetoHTTP.SetRequestHeader "Content-Type",
"text/xml; charset=utf-8"
ObjetoHTTP.SetRequestHeader "SOAPAction",
AccionSOAP
ObjetoHTTP.Send Peticion
End Sub
```

Listado 4.- Enviando archivo al servidor usando el método web.

El archivo se carga correctamente en la biblioteca de SharePoint, pero como comentamos antes, el documento almacena un enlace al archivo original. Se trata del parámetro <SourceURL> que se copia en el metadato _CopySource del documento.

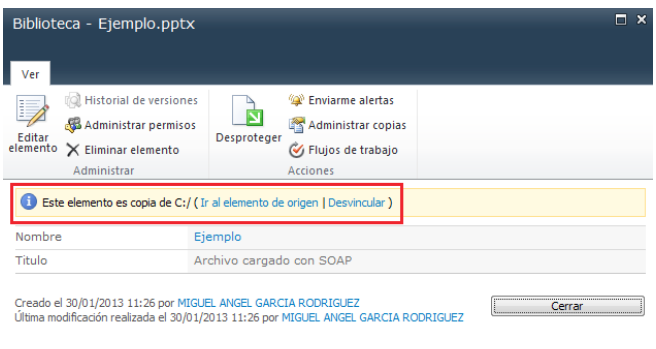


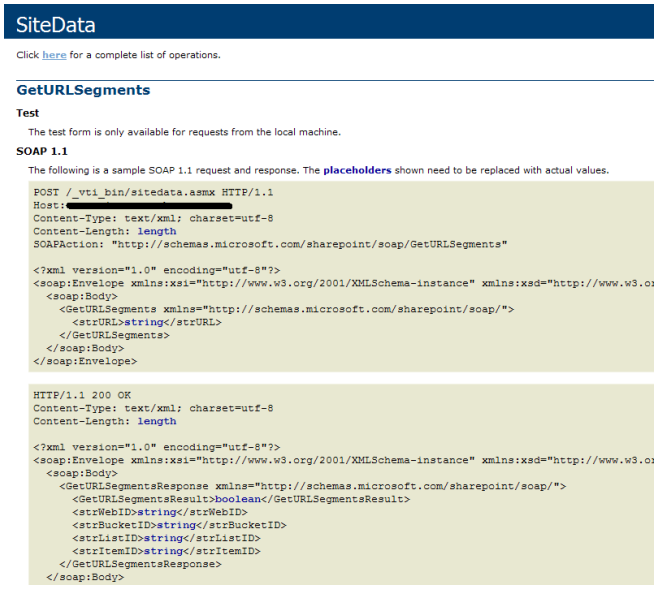
Imagen 3.- Detalle del documento cargado en la biblioteca de SharePoint.

Imagen 4.- Documento cargado en la biblioteca de SharePoint.

Construcción y ejecución de las peticiones adicionales

Podemos desvincular manualmente este documento ya que su origen es un archivo local y no se aplica a nuestro ejemplo. Pero podemos hacerlo programáticamente mediante dos nuevas peticiones, la primera recuperará el ID del elemento publicado y el GUID de la biblioteca y la segunda modificará el metadato _CopySource para que desaparezca el vínculo innecesario.

En la primera petición utilizaremos el servicio web http://<<Misitio>>/_vti_bin/sitedata.asmx con la operación GetURLSegments que incluye en el pedido la URL del documento y nos devuelve el GUID de la Lista y el ID del elemento.


SiteData

Click here for a complete list of operations.

GetURLSegments

Test

The test form is only available for requests from the local machine.

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /_vti_bin/sitedata.asmx HTTP/1.1
Host: [redacted]
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.microsoft.com/sharepoint/soap/GetURLSegments"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.o
  <soap:Body>
    <GetURLSegments xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <strURL>string</strURL>
    </GetURLSegments>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.o
  <soap:Body>
    <GetURLSegmentsResponse xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <GetURLSegmentsResult>boolean</GetURLSegmentsResult>
      <strWebID>string</strWebID>
      <strBucketID>string</strBucketID>
      <strListID>string</strListID>
      <strItemID>string</strItemID>
    </GetURLSegmentsResponse>
  </soap:Body>
```

Imagen 5.- Operación GetURLSegments.

```
'Peticion para recuperar los Ids del documento
cargado
URLServicio = "http://.../_vti_bin/sitedata.asmx"
AccionSOAP =
"http://schemas.microsoft.com/sharepoint/soap/GetUR
LSegments"
Peticion = "<?xml version='1.0' encoding='utf-
8'?>"+_
"<soap:Envelope
xmlns:xsi='http://www.w3.org/2001/XMLSchema-
instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soap='http://schemas.xmlsoap.org/soap/envelop
e/'>"+_
"<soap:Body>"+_
"<GetURLSegments
xmlns='http://schemas.microsoft.com/sharepoint/soap
/'>"+_
"<strURL>" + URLDestino + "</strURL>"+_
"</GetURLSegments>"+_
"</soap:Body>"+_
"</soap:Envelope>"
```

Listado 5.- XML para recuperar el ID del documento.

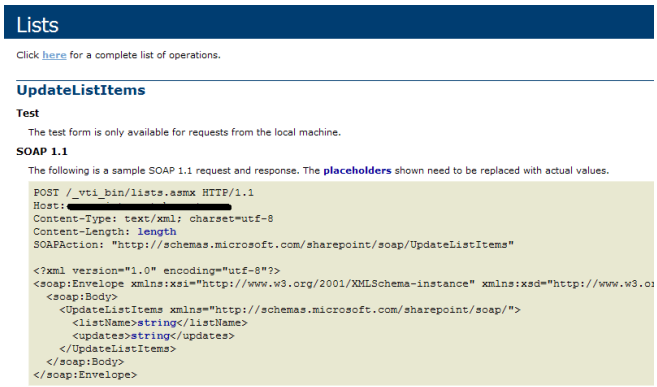
```
EjecutaPeticion

ObjetoDOM.loadXML(ObjetoHTTP.responseText)
Set nodeBook =
ObjetoDOM.selectSingleNode("//strItemID")
IDDocumento = nodeBook.text
Set nodeBook =
ObjetoDOM.selectSingleNode("//strListID")
IDBiblioteca = nodeBook.text
```

Listado 5.- Petición al servidor para cargar valores

"... servicios web que expone SharePoint a terceros para copiar programáticamente un documento..."

En la segunda petición actualizaremos el metadato _CopySource de este documento actualizándolo a Null. Con esto lograremos eliminar el vínculo al elemento origen que no es necesario en este caso. Utilizaremos el servicio web http://<<Misitio>>/_vti_bin/lists.asmx con la operación UpdateListItems que necesita como parámetros de entrada los dos datos que hemos conseguido con la petición anterior.


Lists

Click here for a complete list of operations.

UpdateListItems

Test

The test form is only available for requests from the local machine.

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /_vti_bin/lists.asmx HTTP/1.1
Host: [redacted]
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://schemas.microsoft.com/sharepoint/soap/UpdateListItems"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.o
  <soap:Body>
    <UpdateListItems xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <listName>string</listName>
      <updates>string</updates>
    </UpdateListItems>
  </soap:Body>
</soap:Envelope>
```

Imagen 6.- Operación UpdateListItems.

```
'Peticion para actualizar _CopySource en el
documento
URLServicio = "http://.../_vti_bin/lists.asmx"
AccionSOAP =
"http://schemas.microsoft.com/sharepoint/soap/Updat
eListItems"
Peticion = "<?xml version='1.0' encoding='utf-
8'?>"+_
"<soap:Envelope
xmlns:xsi='http://www.w3.org/2001/XMLSchema-
instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'
xmlns:soap='http://schemas.xmlsoap.org/soap/envelop
e/'>"+_
"<soap:Body>"+_
"<UpdateListItems
xmlns='http://schemas.microsoft.com/sharepoint/soap
/'>"+_
"<listName>" + IDBiblioteca + "</listName>"+_
"<updates>"+_
"<Batch OnError='Return'><Method ID='1'
Cmd='Update'><Field Name='ID'>" + IDDocumento +
"</Field><Field Name='MetaInfo'
Property='_CopySource'></Field></Method></Batch>"+_
"</updates>"+_
"</UpdateListItems>"+_
"</soap:Body>"+_
"</soap:Envelope>"

EjecutaPeticion

WScript.Echo "Documento publicado"
```

Listado 6.- XML para modificar el documento cargado

Conclusiones

En este artículo hemos visto cómo utilizar uno de los servicios expuestos por la plataforma SharePoint, copy.asmx, junto con las operaciones disponibles para facilitar el trabajo remoto con información almacenada en un sitio de SharePoint.

Referencias

[1] http://msdn.microsoft.com/en-us/library/ee705814(v=office.14).aspx
[2] http://msdn.microsoft.com/en-us/library/dd955870.aspx

MIGUEL ÁNGEL GARCÍA RODRÍGUEZ
Analista SharePoint
miganggarrod@hotmail.com

40

Lecciones aprendidas de un proyecto de Workflow en Project Server 2010

Resumen

En este breve artículo voy a resumir algunas lecciones aprendidas en un proyecto de implementación de flujo de trabajo en Project Server 2010. A pesar de que estos proyectos deben desarrollarse en Visual Studio (excepto que usen Nintex), no voy a centrar el artículo en cuestiones técnicas, sino en aspectos funcionales y de arquitectura. Esto se debe a que muchas veces no sabemos cuál es el mejor enfoque para resolver un problema en esta tecnología, debido fundamentalmente a la falta de información. A continuación, mis experiencias en casos reales, que intentan poner un granito más de arena a este mundo, en donde una búsqueda en Google arroja tan pocos resultados que nos hace sentir cierto temor.

Artículo

En este breve artículo voy a resumir algunas lecciones aprendidas en un proyecto de implementación de flujo de trabajo en Project Server 2010. A pesar de que estos proyectos deben desarrollarse en Visual Studio (excepto que usen Nintex), no voy a centrar el artículo en cuestiones técnicas, sino en aspectos funcionales y de arquitectura. Esto se debe a que muchas veces no sabemos cuál es el mejor enfoque para resolver un problema en esta tecnología, debido fundamentalmente a la falta de información. A continuación, mis experiencias en casos reales, que intentan poner un granito más de arena a este mundo, en donde una búsqueda en Google arroja tan pocos resultados que nos hace sentir cierto temor.

Imagen 1.- Project Server 2010.

Introducción

La funcionalidad de flujos de trabajo en Project Server se utiliza muchas veces para manejar el proceso de aprobación de los proyectos antes de su ejecución. Si bien la arquitectura de flujos de trabajo de Project Server está montada sobre la de SharePoint, posee muchos aspectos propietarios que nos dirigen con mucha fuerza hacia un formato de solución. Estos lineamientos principales se pueden resumir en los siguientes puntos:

- A través de configuración se define un conjunto de fases y etapas que constituyen los pasos de nuestro flujo de trabajo. Las etapas son importantes porque pueden definir detalles como la obligatoriedad de los campos de empresa o la posibilidad de definirlos como sólo lectura. También pueden definir qué páginas de empresa pueden estar visibles. Y por último no debe olvidarse que servirán de filtros en nuestras vistas de Project Server.
- La arquitectura de las PDPs nos permite crear páginas de SharePoint que se muestran dentro del contexto de uno o varias etapas de nuestro flujo de trabajo. Al ser páginas de SharePoint, nos permiten agregar cualquier tipo de elemento web, no es necesario usar elementos web exclusivos de Project Server. Esto nos brinda una posibilidad enorme de extender nuestros flujos de trabajo, con configuración y/o desarrollo.
- Por último, los campos de empresa clásicos de Project Server, forman parte del corazón del flujo de trabajo. Constituyen la manera más sencilla de capturar información en cada uno de los pasos. Pero no es la única forma y tiene algunas limitaciones.

Lección 1: Maestro detalle

Es casi imposible escaparle a este requerimiento. En algún momento vamos a necesitar que en alguno de los pasos se cargue o visualice información de detalle. Ejemplos: productos, documentos, notas, etc. La forma más sencilla que se puede utilizar es creando una PDP que contenga varios elementos web: un elemento de la lista de SharePoint en donde guardaremos el detalle; un elemento de formulario InfoPath que sirva para crear elementos de detalle asociados al maestro (el proyecto); y un elemento de filtro de URL para pasar el dato de ID del Proyecto a los otros elementos web. Este esquema no requiere programación y es muy potente. Y puede ser mejorado con Client Object Model.

Más información en: http://surpoint.blogspot.com/2012/12/workflow-en-project-server-2010-como.html

Lección 2: Valores predeterminados en campos de empresa

Con el uso de las pdps y toda su estructura para manejo de campos de empresa, seguramente necesitarás completar valores predeterminados en los campos e incluso ocultarlos. Esta característica no funciona como se espera con las opciones fuera de la caja, en particular con la configuración del valor predeterminado del campo en la configuración de Project Server. Sin embargo, siempre es posible usar algo de código jQuery para ayudar. La siguiente porción de código, que pueden incluir en una CEWP muestra cómo resolver esta problemática:

```
$('input[title="'+id_campo+'"]').attr("value",texto_valor);
$('input[title="'+id_campo+'"]').attr("LTValue",guid_valor);
$('input[title="'+id_campo+'"]').parent().parent().
parent().parent().parent().parent().css("display","none");
```

Más información en: http://surpoint.blogspot.com/2013/01/workflow-en-project-server-2010-valores.html

Lección 3: Manejo de rechazos en un paso del flujo de trabajo

Manejar vuelta a pasos anteriores siempre es algo complicado en un flujo de trabajo. Un requerimiento muy común, es que ante un rechazo, se pueda modificar la información y relanzar el proceso. Una forma sencilla de resolver esto en Project Server es:

- Asignar tareas a los distintos aprobadores, en la que puedan elegir entre Aprobar o Rechazar.
- Ante una aprobación, pasar a la siguiente etapa.
- Ante un rechazo terminar el flujo de trabajo.
- Si el iniciador quiere volver a iniciar el proceso, deberá hacer uso de la opción Restar Workflow, para lo cual habrá que haberle asignado el permiso correspondiente.
- Lo bueno es que la información de campos de empresa no se pierde, así que sólo debe modificar lo que cambió.
- Una posible mejora es crear una lista en SharePoint que muestre un log de aprobaciones y rechazos histórico, para que el usuario pueda conocer en cada caso las razones de los rechazos.

Es el proceso de optimización de nuestro sitio web con el objetivo de conseguir un buen posicionamiento en los motores de búsqueda

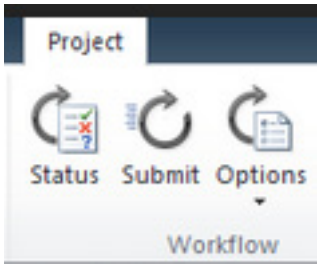


Imagen 2.- Reinicio del WF desde Options.

Lección 4: Asignación de tareas basada en roles

Un requerimiento típico es que las tareas de aprobación de cada paso deban ser asignadas a diferentes personas, dependiendo de una condición, basada en algún campo completado en algún paso. Una forma de resolver esto es crear una lista en SharePoint que maneje las reglas de asignación. El usuario configura en esta lista la regla, por ejemplo: "cuando el país es Argentina y el sector es Marketing, entonces el grupo de asignación es Gerentes de Marketing de Argentina."

Internamente, el flujo de trabajo consulta la lista con el fin de obtener el grupo de asignación para cierta condición. Ese grupo, no es más que un grupo de SharePoint que puede incluir uno o varios miembros. Cuando el flujo de trabajo asigna la tarea al grupo, SharePoint envía el mail en forma automática. Este tipo de reglas le dan enorme flexibilidad al flujo de trabajo.

Lección 5: Visibilidad de PDPs

Project Server nos permite definir qué PDP puede estar visible en cada etapa del flujo de trabajo. Esto nos da mucho poder con poco esfuerzo. A continuación enumero sólo algunos ejemplos, como para entender el alcance funcional:

- Diferentes campos de empresa en cada etapa.
- Habilitar la PDP de Schedule sólo a partir de una determinada etapa.
- Mostrar información de una lista de SharePoint de forma distinta en diferentes etapas. Por ejemplo con opciones de creación y edición en una etapa, y con opciones de sólo lectura en otras etapas.
- Diferentes páginas de estado en diferentes etapas.

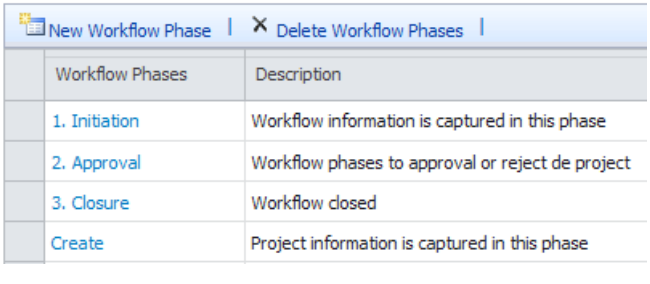


Imagen 3.- Ejemplos de fases.

Estos fueron sólo algunos ejemplos y nunca debemos olvidar la innumerable cantidad de opciones que tenemos al poder personalizarlas con diferentes elementos:

- Varios elementos web de Project Fields, que nos permiten agrupar la información.
- InfoPath.
- SQL Reporting Services.
- Listas de SharePoint.
- CEWP (WebPart de editor de contenido) con código JavaScript y con Client Object Model.
- Librearías de documentos.
- Estado visual del flujo de trabajo.
- Elementos de filtro por URL.
- Etc.

Más información en:

- Fases y etapas: http://surpoint.blogspot.com/2012/11/workflow-en-project-server-2010-como_3147.html
- PDPs: http://surpoint.blogspot.com/2012/11/workflow-en-project-server-2010-como.html
- PDP de estado: http://surpoint.blogspot.com/2012/11/workflow-en-project-server-2010-como_30.html

Lección 6: Sobre el uso de campos de empresa

Los campos de empresa constituyen la alternativa natural para capturar información en un flujo de trabajo.

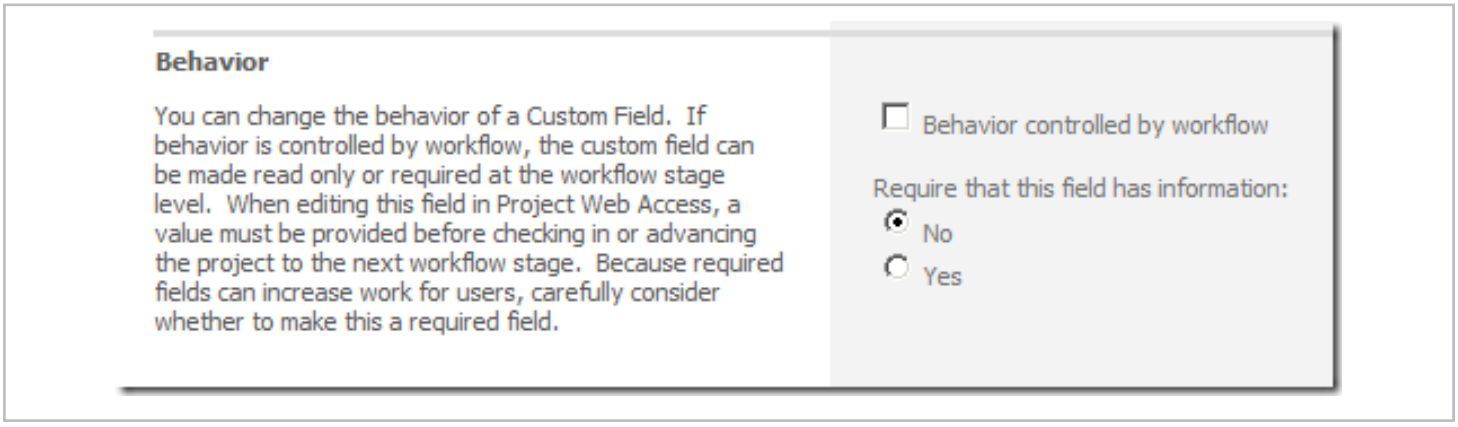


Imagen 4.- Opción de campo de empresa controlado por WF.

Esto está muy bien y es recomendable, pero conviene tener en cuenta algunas cuestiones:

- La cantidad de campos puede afectar el rendimiento de Project Server. De hecho es una de las variables para realizar un dimensionamiento de la arquitectura.
- Los campos aparecen en Project Pro y la única forma de no mostrarlos es usando la funcionalidad de departamentos.
- Modificar un campo desde un flujo de trabajo implica operaciones costosas como la desprotección y la protección del proyecto. Y lo más importante es que nadie verá los cambios hasta que no se publique el proyecto.
- Los campos de empresa no manejan información repetitiva como las relaciones maestro detalle.
- Los campos de empresa no tiene flexibilidad en el manejo de tipos de datos, ni permiten validaciones sofisticadas.

Es por ello que en algunos casos, la alternativa de usar listas de SharePoint nos permite soluciones más livianas y flexibles. Es absolutamente recomendable usar esta alternativa en muchas situaciones, no en todas por supuesto.

Lección 7:Seguridad

A diferencia de la mayoría de las implementaciones de Project Server, en donde la configuración estándar suele cubrir muchos requerimientos, cuando implementamos un flujo de trabajo, aparecen algunas necesidades que a continuación enumero:

- La necesidad de crear un grupo y una categoría para los iniciadores de flujos de trabajo. Este grupo no suele coincidir con los líderes de proyecto y puede necesitar permisos especiales, por ejemplo para reiniciar un flujo de trabajo.
- La necesidad de crear un grupo para los que aprueban pasos del flujo de trabajo.
- La necesidad de crear grupos en SharePoint para poder acceder a listas como la de tareas, pero también a listas especiales que hayamos creado para capturar información durante el proceso.
- Por último, es posible que necesitemos crear un grupo de administración de la configuración del flujo de trabajo.

Más información en: http://surpoint.blogspot.com/2013/01/Workflow-ProjectServer-Seguridad.html

Conclusiones

En este breve artículo he intentado presentar algunas lecciones aprendidas en proyectos de gestión de la demanda en Project Server 2010. Lamentablemente es complicado encontrar suficiente información sobre este tema y a veces no es sencillo saber si estamos tomando la decisión correcta. Por ello este artículo: para compartir mi experiencia.

¿¿Y cuál ha sido tu experiencia???

¡Hasta la próxima!

JUAN PABLO PUSSACQ LABORDE
SharePoint MVP
Blog: http://surpoint.blogspot.com/
Facebook: http://facebook.com/surpointblog/
Twitter: http://twitter.com/jpussacq/

43 Notificaciones Push a APPS de Windows Phone desde SharePoint 2013-2010 Parte II

Resumen

Este artículo es la continuación del artículo del número pasado en el explicábamos como enviar notificaciones PUSH desde SharePoint. En este número vamos a explicar cómo tenemos que desarrollar una APP de Windows Phone 7.5 y 8 para poder recibir las notificaciones desde SharePoint tanto en su versión 2010 como en la nueva versión 2013.

Requisitos Previos

Descargar el SDK Client SharePoint para Windows Phone 7.1 que facilita la comunicación de la aplicación móvil con SharePoint

Desarrollo de la APP de Windows Phone

La APP que vamos a desarrollar solamente incluye dos partes. Por un lado vamos a explicar cómo habilitar nuestra aplicación para poder recibir notificaciones PUSH y por otro lado como podemos leer los datos de las Listas que hay en SharePoint. Dados estos requerimientos podemos realizar una aplicación que funcione tanto para Windows Phone 7.5 y Windows Phone 8; primero tenemos que saber que tendremos que hacer dos aplicaciones distintas, pero no hace falta que dupliquemos todo el código. La cuestión es utilizar los elementos comunes que tienen ambas versiones y de esta forma hacer una aplicación que nos cueste mucho menos de mantener, sea escalable y de esta forma llegue a un número mayor de usuarios.

Para hacer que el código que realicemos sea compatible en ambas versiones lo que vamos a realizar en primer lugar es crearnos un proyectos de Bibliotecas portables en el que vamos a poner los elementos comunes que van a tener ambas APP. Que va a ser todo a excepción de las Vistas ya que no es lo mismo diseñar para una APP que tiene distinto tamaño de pantalla; el resto del código fuente va a ser igual.

Seleccionamos el tipo de SO que vamos a utilizar para crear la aplicación eligiendo la versión 7.1. Lo hacemos así porque en la versión 8 incluye todo lo que había antes y además las nuevas mejoras que conlleva.

A este proyecto le tenemos que añadir las siguientes referencias (Microsoft.Phone, Microsoft.Phone.Interop y Microsoft.SharePoint.Client.Phone)

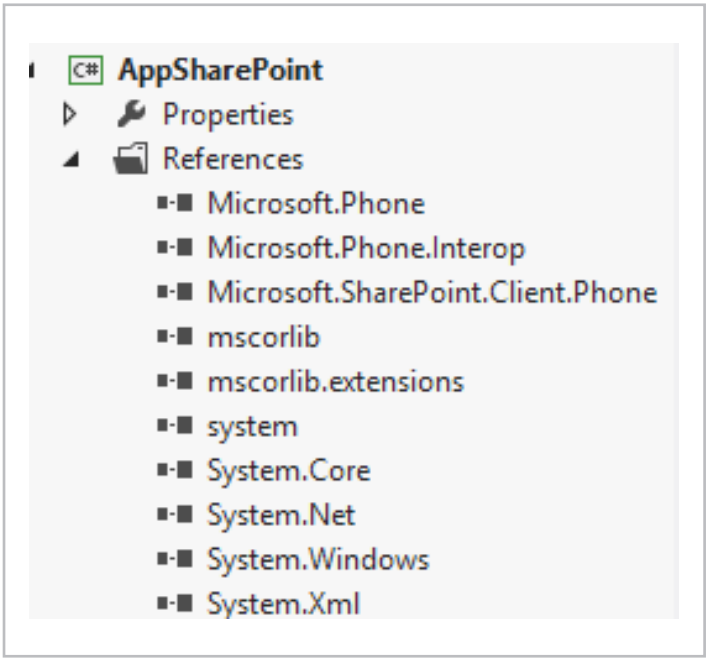


Imagen 1.- Referencias a añadir al proyecto.

En primer lugar lo que vamos a implementar es una clase "Notification" que tendrá una variable de tipo HttpNotificationChannel que es la que se encargara de comunicarse con el MPNS. En primer lugar nos crearemos un método OpenNotificationChannel cuya función es buscar si el canal de comunicación esta creado. En caso de que no esté creado se debe crear. Hay que saber que una aplicación de WP solo puede tener un único canal de comunicaciones abierto y este no se puede comunicar con el resto de aplicaciones que hay en el teléfono. Además se le añadirán manejadores a los siguientes eventos:

- ChannelUriUpdate que se lanzará si el MPNS cambia la dirección URI de notificaciones.
- ErrorOcurred se lanzará si ocurre un error en la aplicación.
- httpChannel_ShellToastNotificationReceived se lanzará en el momento que recibamos una notificación TOAST.

La primera vez que ejecutamos el canal no tenemos aún la dirección URI y solo tendremos una dirección URI valida en el momento que se lance el evento ChannelUriUpdate, por lo que hasta que no tengamos esta dirección valida no podremos registrar nuestro dispositivo en la aplicación.

```
public static void OpenNotificationChannel(bool
isInitialRegistration)
        {
            try  {
                // Buscamos si el canal ya esta creado
                httpChannel =
HttpNotificationChannel.Find(ChannelName);
                // Si no se encuentra el canal creamos
uno
                if (httpChannel == null)
                {
                    httpChannel = new
HttpNotificationChannel(ChannelName);
                    //Añadimos los eventos asociados
al HttpNotificationChannel
                      // Hasta que no salte el evento
                      ChannelUriUpdate como no
                      tenemos URI
                    // no podemos subscribir el
dispositivo Sharepoint
                    AddChannelEventHandlers();
                    httpChannel.Open();
                }
                else
                {
                        //En caso de que el canal ya
                      exista asociamos los eventos al
                      HttpNotificationChannal
                    AddChannelEventHandlers();
                      // En caso de que sea la
                      primera vez que se ejecute
                      suscribiremos al Servicio
                    if (isInitialRegistration)
                    {
                            SubscribeToService();
                    }
                }
            }
            catch (Exception ex)
            {
               MessageBox.Show(ex.Message, "Error
               Abriendo el Canal",
               MessageBoxButton.OK);
             CloseChannel();
            }
        }
```

Una vez ya tenemos el canal abierto, el siguiente paso a realizar es implementar los manejadores:

CHANNELURIUPDATE: En este evento lo que vamos a realizar es que cada vez que se ejecute, subscribiremos este dispositivo. Pueden darse dos situaciones: la primera vez en la que registramos el dispositivo y para la que valdría el siguiente método. En la segunda, simplemente realizamos la subscripción.

```
    /// <summary>
        /// Suscribir el servicio en una lista de
Sharepoint
        /// </summary>
        private static void SubscribeToService()
        {
            Guid deviceAppInstanceId =
GetSettingValue<Guid>(DeviceAppIdKey, false);
            Context.Load(Context.Web, w => w.Title, w
=> w.Description);
            PushNotificationSubscriber pushSubscriber
=

Context.Web.RegisterPushNotificationSubscriber(deviceA
ppInstanceId, httpChannel.ChannelUri.AbsoluteUri);
            Context.Load(pushSubscriber);
            Context.ExecuteQueryAsync
            (
                (object sender,
ClientRequestSucceededEventArgs args) =>
                {
                SetRegistrationStatus(true);
                if (!httpChannel.IsShellTileBound)
{httpChannel.BindToShellTile();}
                if
(!httpChannel.IsShellToastBound)
{httpChannel.BindToShellToast();}

ShowMessage(string.Format("Subscripcion Correcta
registrada: {0}",
pushSubscriber.User.LoginName),"Realizado");
                },
                (object sender,
ClientRequestFailedEventArgs args) =>
                {

ShowMessage(args.Exception.Message, "Error
Subscribiendo");
                });
        }
```

3 Void SubscribeToService

Pero el problema es que la dirección URI puede cambiar, ¿y entonces que hacemos cuando se modifica un dirección? Pues bien primero llamaremos a la siguiente función que lo que hace es actualizar la Dirección URI en SharePoint, en caso de que no exista no hace nada. Posteriormente, si procede, llamamos a la función de arriba.

```
///<summary>
/// Actualizo el Canar Uri en el Servidor
///</summary>
privatestaticvoidUpdateChannelUriOnServer()
        {
Guiddeviceappinstanceid =
GetSettingValue<Guid>(DeviceAppIdKey, false);
Context.Load(Context.Web, w =>w.Title, w
=>w.Description);
PushNotificationSubscriber subscriber =
Context.Web.GetPushNotificationSubscriber(deviceAppIns
tanceId);
Context.Load(subscriber);
Context.ExecuteQueryAsync(
            (object sender1,
ClientRequestSucceededEventArgs args1) =>
                {
subscriber.ServiceToken =
httpChannel.ChannelUri.AbsolutePath;
subscriber.Update();
Context.ExecuteQueryAsync(
            (object sender2,
ClientRequestSucceededEventArgs args2) =>
            {
ShowMessage("Channel URI updated on server.",
"Success");
            },
            (object sender2,
ClientRequestFailedEventArgs args2
            ) =>
            {
ShowMessage(args2.Exception.Message, "Error Upating
Channel URI");
});
            },
            (object sender1,
ClientRequestFailedEventArgs args1) =>
            {

            });
        }
```

ERROROCURRED: en el caso de que se produzca un error mostraremos un mensaje legible para el usuario en la pantalla (esto es una de las condiciones de certificación de aplicación en el MarketPlace de Windows Phone)

HTTPCHANNEL_SHELLTOASTNOTIFICATIONRECEIVED: en el momento que recibamos una notificación lo único que vamos a realizar en este ejemplo es mostrar un mensaje en pantalla para que el usuario pueda visualizar la notificación

```
///<summary>
/// En el momento que recibimos una notificacionToast
///</summary>
///<param name="sender"></param>
///<param name="e"></param>
staticvoidhttpChannel_ShellToastNotificationReceived(o
bject sender, NotificationEventArgs e)
        {
if (e.Collection != null)
            {
Dictionary<string, string> collection =
(Dictionary<string, string>)e.Collection;
ShellToast toast = newShellToast();
toast.Title = collection["wp:Text1"];
toast.Content = collection["wp:Text2"];

ShowMessage(string.Format("Titulo: {0}\r\nAutor: {1}",
toast.Title, toast.Content), "Toast Notification");
}
        }
```

Para almacenar el GUID de la aplicación, y si hemos registrado o no la aplicación, hemos utilizado el almacenamiento local que nos ofrece Windows Phone ya que nos proporciona un grado de seguridad altísimo, puesto que ninguna otra aplicación puede acceder a zonas de memoria usadas por el sistema o por otras aplicaciones.

El siguiente paso que le añadiremos a esta biblioteca es crearnos el ViewModel que vamos a realizar en primer lugar nos creamos una clase ItemArticleViewModel que va a contener las propiedades que queremos mostrar en nuestro caso “Titulo” del articulo y “Autor” la clase quedaría de la siguiente forma (hay que añadirle la parte de notificación para que se modifiquen los valores en los bindings de la capa de vista); el resto es como si fuera una clase normal:

```
publicclassItemArticleViewModel :
INotifyPropertyChanged
    {
publicstring Title { get; set; }
publicstring Author { get; set; }

publiceventPropertyChangedEventHandlerPropertyChanged;
privatevoidNotifyPropertyChanged(StringpropertyName)
        {
PropertyChangedEventHandler handler = PropertyChanged;
if (null != handler)
            {
handler(this,
newPropertyChangedEventArgs(propertyName));
            }
        }
    }
```

Una vez creado la definición del modelo nos creamos la capa de donde se obtienen los datos y ahí es donde entra en juego SharePoint. Antes de empezar a ver cómo hacerlos. Hay que saber las opciones que tenemos para obtener los datos de un SharePoint:

1. A través del SDK de SharePoint para Windows Phone 7.5 (que es compatible también para la versión 8).
2. Utilizando el API REST de SharePoint.

En este ejemplo vamos a realizarlo a través la opción 1, entonces nos creamos una clase MainArticleViewModel que va a tener una esctructura como la siguiente:

```
public class MainArticleViewModel :
INotifyPropertyChanged
    {
        public MainArticleViewModel()
        {
            this.Items = new
ObservableCollection<ItemArticleViewModel>();
        }

        /// <summary>
        /// Colección para objetos ItemViewModel.
        /// </summary>
        public
ObservableCollection<ItemArticleViewModel> Items {
get; private set; }

        public string SampleProperty
        {
            get;
            set;
        }

        public bool IsDataLoaded
        {
            get;
            private set;
        }
        public event PropertyChangedEventHandler
PropertyChanged;
        private void NotifyPropertyChanged(String
propertyName)
        {
            PropertyChangedEventHandler handler =
PropertyChanged;
            if (null != handler)
            {
                handler(this, new
PropertyChangedEventArgs(propertyName));
            }
        }
```

Y para completar esta clase le falta la función LoadData( ). En esta función es donde vamos a realizar la llamada a SharePoint, y se hace de una forma muy similar a la que se utiliza en las llamadas Cliente en 2010 y convertiremos los elementos de la lista en objetos de nuestro ViewModel. La función quedaría así:

```
public void LoadData()
        {
            ClientContext context = new
ClientContext("http://compartimoos");
                context.Credentials =  new Authentica
        tor();
            List articlesList =
context.Web.Lists.GetByTitle("Articulos");
            CamlQuery query = new CamlQuery();
            query.ViewXml = @"<Query><Eq>
                                        <FieldRef
Name='Revista' />
                                        <Value
Type='Text'>'Numero13'</Value>
```

```
                                    </Eq>
                                </Where></Query>";
            ListItemCollection itemsList =
articlesList.GetItems(query);
            context.Load(itemsList);
            context.ExecuteQuery();
            if (itemsList.Count > 0)
            {
                foreach (ListItem item in itemsList)
                {
                    ItemArticleViewModel itemArticle=
new ItemArticleViewModel();

itemArticle.Author=item["Author"].ToString();

itemArticle.Title=item["Title"].ToString();
                    this.Items.Add(itemArticle);
                }

            }
```

Creando las aplicaciones

Una vez tenemos la clase con toda la funcionalidad disponible vamos a crear la aplicación de Windows Phone. Lo primero que tenemos que hacer es agregar la referencia a la librería que habíamos creado anteriormente.

A continuación nos vamos a la página MainPage.xaml y a la página le añadimos un objeto de tipo “Pivots “ y en los “Bindings” tendremos que poner los nombres de los campos de nuestro modelo en nuestro caso Article y Phone

```
<controls:Pivot Title="MI APLICACIÓN">
            <!--Elemento Pivot uno-->
            <controls:PivotItem Header="primero">
                <!--Lista de líneas dobles con ajuste
de texto-->
                <ListBox x:Name="FirstListBox"
Margin="0,0,-12,0" ItemsSource="{Binding Items}">
                    <ListBox.ItemTemplate>
                        <DataTemplate>
                          <StackPanel
Margin="0,0,0,17" Width="432" Height="78">
                            <TextBlock
Text="{Binding Article}" TextWrapping="Wrap"
Style="{StaticResource PhoneTextExtraLargeStyle}"/>
                            <TextBlock
Text="{Binding Author}" TextWrapping="Wrap"
Margin="12,-6,12,0" Style="{StaticResource
PhoneTextSubtleStyle}"/>
                          </StackPanel>
                        </DataTemplate>
                    </ListBox.ItemTemplate>
                </ListBox>
            </controls:PivotItem>
        </controls:Pivot>
```

El siguiente paso es irnos a al APP.CS y añadir este propiedad:

```
        public static MainArticleViewModel ViewModel
        {
            get
            {
                // Retrasar la creación del modelo de
vista hasta que sea necesario
                if (viewModel == null)
                    viewModel = new
MainArticleViewModel();

                return viewModel;
            }
        }
```

Y dentro del evento cuando se active añadir esta llamada:

```
if (!App.ViewModel.IsDataLoaded)
            {
App.ViewModel.LoadData();
            }
```

Ahora nos dirigimos a la MainPage.cs y aquí tendremos que establecer el contexto de esta pantalla que con el View Model quedaría de la siguiente forma:

```
// Constructor
publicMainPage()
        {
InitializeComponent();

// Establecer el contexto de datos del control ListBox
control en los datos de ejemplo
DataContext = App.ViewModel;
this.Loaded += newRoutedEventHandler(MainPage_Loaded);
}

// Cargar datos para los elementos ViewModel
privatevoidMainPage_Loaded(object sender,
RoutedEventArgs e)
        {
if (!App.ViewModel.IsDataLoaded)
            {
App.ViewModel.LoadData();
            }
}
```

Una vez realizado esto si ejecutamos nuestra aplicación se visualizará la siguiente pantalla:

“... como poder enviar notificaciones Push desde SharePoint tanto en su versión 2010 como en la nueva versión 2013 a una aplicación Windows Phone ...”

Imagen 2.- Previsualización de la aplicación en el emulador de Windows Phone.

Ahora lo que falta es añadir que esta aplicación pueda recibir notificaciones PUSH dentro de nuestra aplicación, para ello en el main llamamos al método OpenNotificacion de la clase Notificacion. Como hemos visto anteriormente, lo que hace es lo siguiente:

- Crear el Canal de comunicaciones.
- En el momento que tenemos una dirección URI valida subscribimos la aplicación a SharePoint.
- Cuando recibimos una notificación la mostramos en la aplicación (en caso de que tengamos la aplicación cerrada se visualizara la notificación en la parte superior de la pantalla como en las siguientes imágenes):

Imagen 3.- Ejemplo de notificación.

Imagen 4.- Notificación PUSH con la APP en segundo plano.

Esto solo es un ejemplo de como se pueden enviar notificaciones desde una lista de Sharepoint a aplicaciones Windows Phone. Para una aplicación "profesional" tendriamos que crear una pantalla de configuración de estas notificaciones para que el usuario tenga la potestad de elegir si quiere o no quiere recibir notificaciones. Tambien quedaria pendiente la forma en la que se deja de recibir notificiaciones (hay un metodo unsubscriber tambien) . Otra cosa a tener en cuenta es que con aplicaciones de Windows Phone, no hay notificación de eventos cuando una aplicación se desinstala por lo que en tendremos que implementar alguna función que nos haga saber si todos los dispositivos móviles que tenemos registrados estan "activos". Pero esto ya se deja a la intriga del lector.

¿Cómo implementamos la aplicación si estamos trabajando con un SharePoint 2010?

El desarrollo es al 90% igual para un sistema que para el otro, la unica diferencia es que en 2010 no tenemos el metodo de subscribir al dispositivo a una lista y por lo tanto lo tendremos que hacer de forma manual. Viendo la solución que habia planteado con anterioridad esta claro lo que tenemos que hacer. Hay que insertar un elemento en una lista de SharePoint y esto utilizando las librerias clientes de SharePoint es muy sencillo de hacer.

Lo que va a realizar la siguiente función es: en primer lugar consultamos si tenemos este elemento en la lista y en caso que afirmativo actualizamos la dirección URI y sino existe insertamos el elemento en la lista.

```
privatevoidSubscribeToService(stringGuid,
stringChannelUri)
        {
ClientContext context =
newClientContext("http://compartimoos");

if (!existGuid(Guid, context))
                {
InsertData(Guid, ChannelUri, context);
                }
else
                {
UpadteData(Guid, ChannelUri, context);
                }

        }
```

```
privateboolexistGuid(stringGuid, ClientContext
context)
        {
bool res = false;

ListmensajesList =
context.Web.Lists.GetByTitle("Subscriptores");
CamlQuery query = newCamlQuery();
query.ViewXml = string.Format(@"<Query><Eq>
<FieldRef Name='Guid' />
<Value Type='Text'>{0}</Value>
</Eq>
</Where></Query>"
,Guid);
ListItemCollectionitemsList =
mensajesList.GetItems(query);
context.Load(itemsList);
context.ExecuteQuery();
if (itemsList.Count> 0) { res = true; }
return res;
        }
privatestaticvoidInsertData(stringGuid,
stringChannelUri, ClientContext context)
        {
Listmensajeslist =
context.Web.Lists.GetByTitle("Subscriptores");
ListItemCreationInformationitemCreateInfo =
newListItemCreationInformation();
ListItemoListItem =
mensajesList.AddItem(itemCreateInfo);
oListItem["Guid"] = Guid;
oListItem["ChannelUri"] = ChannelUri;
oListItem.Update();
context.ExecuteQuery();
        }
```

Conclusiones

En este articulos hemos visto lo sencillo que es crear Apps para Windows Phone tanto en la versión 7.5 y versión 8 con el origen de datos almacenados en SharePoint tanto en la versión 2010 como en la 2013. Ademas se le puede añadir complementos para que la interacción APP con SharePoint este presente y de esta forma el usuario pueda estar al tanto de lo que se realiza. Con lo que es un muy buen punto para incluir dentro de nuestros desarrollos profesionales.

Referencias

- Windows Phone 7.5 Desarrollo de aplicaciones en Silverlight- Josué Yeray Julian.
- Notificaciones de empuje para el desarrollo de Windows Phone http://blogs.ligasilverlight.com
- Push Notifications with SharePoint 2013-based Windows Phone apps http://www.deviantpoint.com/post/2012/07/27/Push-Notifications-with-SharePoint-2013-based-Windows-Phone-apps.aspx

ADRIÁN DÍAZ CERVERA

SharePoint Developer at Encamina

MCPD SharePoint 2010 MAP y MCC 2012

http://blogs.encamina.com/desarrollandosobresharepoint

adiaz@encamina.com @AdrianDiaz81

49 Nosotros

Fabian Imaz

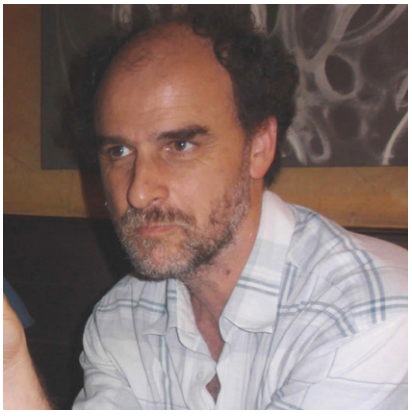
Fabián Imaz, MVP de SharePoint Server trabaja en el mundo del desarrollo de software desde hace más de 10 años, teniendo la suerte de trabajar en distintas arquitecturas y tecnologías Microsoft. Pertenece a la firma Siderys, http://www.siderys.com, empresa de desarrollo de Software especializada en SharePoint 2007/2010 y en desarrollo de soluciones inteligentes.

Desde los comienzos Fabián ha trabajado en distintitas comunidades como MOSSCA dónde es uno de los directores y CUMUY donde organiza y promueve eventos locales para la difusión de tecnología dentro de los miembros de la misma. Es director de la carrera SharePoint 2010 en Microsoft Virtual Academy, http://www.mslatam.com/latam/technet/mva2/Home.aspx y cuenta con un sitio en CodePlex con varios desarrollos http://siderys.codeplex.com. Podéis contactar con Fabián a través de su @fabianimaz o su blog http://blog.siderys.com

Juan Carlos González Martín

Juan Carlos González, es Arquitecto de Soluciones en el CIIN (www.ciin.es) de Cantabria, uno de los Microsoft Innovation Centers de España. Ingeniero de Telecomunicaciones por la Universidad de Valladolid, cuenta con más de 9 años de experiencia en tecnologías y plataformas de Microsoft diversas (SQL Server, Visual Studio, .NET Framework, etc.), aunque su trabajo diario gira en torno a la plataforma SharePoint. Juan Carlos es MVP de SharePoint Server desde el año 2008, coordinador del grupo de usuarios .NET de Cantabria (Nuberos.Net, www.nuberos.es) y co-fundador del Grupo de Usuarios de SharePoint de España (SUGES, www.suges.es) y del Grupo de Usuarios de Cloud Computing de España (CLOUDES). Desde el año 2011 participa junto con Gustavo Vélez y Fabián Imaz en la dirección de CompartiMOSS.

o Twitter: @jcgm1978.
o Blogs: http://geeks.ms/blogs/ciin & http://jcgonzalezmartin.wordpress.com/



Gustavo Velez

Gustavo Velez es Ingeniero Mecánico y Electrónico; trabaja en el diseño e implementación de sistemas de IT basados en tecnologías de Microsoft, especialmente SharePoint, para Avanade (http://www.avanade.com), una compañía multinacional de IT.

Propietario del sitio especializado en información sobre SharePoint en español http://www.gavd.net y autor de seis libros sobre SharePoint y sus tecnologías.

Sitio web:
http://www.gavd.net

Email:
gustavo@gavd.net

¿Desea colaborar con CompartiMOSS?

La subsistencia del magazine depende de los aportes en contenido de todos. Por ser una revista dedicada a información sobre SharePoint en español, todo el contenido deberá ser directamente relacionado con Microsoft SharePoint Services (WSS) y/o Microsoft Office SharePoint Server (MOSS) y escrito en castellano. No hay limitaciones sobre el tipo de articulo o contenido, lo mismo que sobre el tipo de versión.

Si desea publicar algo, por favor, utilice uno de los siguientes formatos:

- Artículos de fondo: tratan sobre un tema en profundidad. Normalmente entre 2000 y 3000 palabras y alrededor de 4 o 5 figuras. El tema puede ser puramente técnico, tanto de programación como sobre infraestructura, o sobre implementación o utilización.
- Artículos cortos: Máximo 1000 palabras y 1 o 2 figuras. Describen rápidamente una aplicación especial de SharePoint, o explica algún punto poco conocido o tratado. Experiencias de aplicación de SharePoint en empresas o instituciones puede ser un tipo de artículo ideal en esta categoría.
- Ideas, tips y trucos: Algunos cientos de palabras máximo. Experiencias sobre la utilización de SharePoint, problemas encontrados y como solucionarlos, ideas y trucos de utilización, etc.

Los formatos son para darle una idea sobre cómo organizar su información, y son una manera para que los editores le den forma al magazine, pero no son obligatorios. Los artículos deben ser enviados en formato Word (.doc o .docx) y las figuras por separado en un formato de alta resolución (.tif), todo comprimido en un archivo (.zip o .rar) con el nombre del autor y del artículo.

Si desea escribir un artículo de fondo o corto, preferiblemente envíe una proposición antes de escribirlo, indicando el tema, aproximada longitud y número de figuras. De esta manera evitaremos temas repetidos y permitirá planear el contenido de una forma efectiva.

Envíe sus proposiciones, artículos, ideas y comentarios a la siguiente dirección:
fabiani@siderys.com.uy
jgonzalez@gruposodercan.es
gustavo@gavd.net

COMPARTIMOSS