EBook Gratis

APRENDIZAJE sharepoint

Free unaffiliated eBook created from
Stack Overflow contributors.

#sharepoint

Tabla de contenido

Acerca de ... 1

Capítulo 1: Empezando con sharepoint ... 2

Observaciones ... 2

Versiones ... 2

Examples ... 3

Instalación de SharePoint 2016 para Single Server Farm ... 3

Introducción ... 3

Requerimientos ... 3

Instalación ... 3

Configuración ... 3

Configuración de la granja ... 4

Construye un elemento web con SharePoint Framework ... 5

SharePoint ULS Registros y registro ... 5

Estampación ... 6

Identificador de correlación ... 6

Agregando SPMonitoredScope a mi código ... 6

Capítulo 2: Aplicación de SharePoint ... 7

Introducción ... 7

Observaciones ... 7

Examples ... 7

SharePoint 2013: acceso a los datos del servicio de perfiles de usuario utilizando JSOM en ... 7

Capítulo 3: Creación de una aplicación alojada por el proveedor ... 9

Examples ... 9

Entorno de desarrollo de ajuste ... 9

Preparación para el sitio de desarrollador ... 10

Crear una aplicación en Visual Studio ... 11

Vamos a empezar a codificar ... 15

Creación de la página del artículo completo ... 18

Capítulo 4: Principales lanzamientos ... 21

Examples 21
SharePoint 2016 21
SharePoint 2013 21
Capítulo 5: Representación del lado del cliente de SharePoint 2013 23
Introducción 23
Examples 23
Cambiar el hipervínculo de campos / columnas dentro de la vista de lista usando CSR 23
Ocultar columna de la vista de lista de SharePoint usando CSR 24
Aplicar validaciones en el formulario Nuevo / Editar artículo usando CSR 24
Cambiar el nombre de visualización de la columna en la vista de lista usando CSR 29
Capítulo 6: Servicios de descanso 31
Observaciones 31
URL de punto final de servicio REST 31
Enviando solicitudes de REST 31
Sintaxis de XMLHttpRequest 31
jQuery AJAX Sintaxis 32
Examples 32
Trabajando con listas 32
Obtener elementos de lista con columnas de búsqueda 33
Tabla de listado de animales 33
Tabla de tipos de animales 33
Código de ejemplo 33
Agregar selecciones a un campo de búsqueda multivalor 35
Elementos de la lista de paginación devueltos de una consulta 36
Recuperar una ID del elemento recién creado en la lista de SharePoint 37
Cómo realizar operaciones CRUD mediante la interfaz REST de SharePoint 2010 38
Capítulo 7: Trabajar con cuadros de diálogo modales con JavaScript 42
Sintaxis 42
Parámetros 42
Observaciones 43
Examples 43

Realizar una acción cuando un cuadro de diálogo está cerrado 43
Mostrar una página existente en un diálogo 43
Mostrar un diálogo personalizado 44
Capítulo 8: Trabajar con JavaScript Client Object Model (JSOM) 45
Observaciones 45
Examples 46
Obtención de tipos de contenido de la biblioteca utilizando el nombre de la biblioteca 46
Eliminar un elemento en una lista 46
Creación de elementos o carpetas 46
Creación de elementos de lista 46
Creación de carpetas 47
Obtener información actual del usuario 48
Obtener un elemento de lista por ID 48
Obtener elementos de lista por consulta CAML 49
Ejemplo básico 49
Paginar los resultados de una consulta CAML 49
Capítulo 9: Trabajar con Managed Client Side Model Model (CSOM) 51
Observaciones 51
Examples 51
Hola mundo (obteniendo titulo del sitio) 51
Web. Recuperando las propiedades de un sitio web 52
Web. Recuperar solo las propiedades especificadas de un sitio web 52
Web. Actualización del título y descripción de un sitio web 52
Web. Creando un sitio web 52
Lista. Recuperar todas las propiedades de todas las listas en un sitio web 53
Lista. Recuperando solo propiedades especificadas de listas 53
Lista. Almacenar listas recuperadas en una colección 53
Lista. Recuperar campos de lista de un sitio web 54
Lista. Creando y actualizando una lista 54
Lista. Añadiendo un campo a una lista 55
Lista. Borrando una lista 55
Ít. Recuperar elementos de una lista 55

Ít. Recuperar elementos (utilizando el método Include) 55
Ít. Recuperar campos específicos de un número específico de elementos 56
Ít. Recuperar elementos de todas las listas en un sitio web 56
Ít. Recuperar elementos utilizando la posición de colección de elementos de lista 57
Ít. Crear un elemento de lista 57
Ít. Actualización de un elemento de la lista 58
Ít. Eliminar un elemento de la lista 58
Los grupos Recuperar todos los usuarios de un grupo de SharePoint 58
Los grupos Recuperando propiedades específicas de los usuarios 59
Los grupos Recuperar todos los usuarios en todos los grupos de una colección de sitios 59
Los grupos Agregar un usuario a un grupo de SharePoint 59
Roles Crear una definición de rol 60
Roles Asignar un usuario a un rol en un sitio web 60
Roles Crear un grupo de SharePoint y agregar el grupo a un rol 61
Permisos. Rompiendo la herencia de seguridad de una lista. 61
Permisos. Rompiendo la herencia de seguridad de un documento y agregando un usuario como l 61
Permisos. Rompiendo la herencia de seguridad de un documento y cambiando los permisos de u 62
Acción personalizada. Agregar una acción personalizada de usuario para los elementos de la 62
Acción personalizada. Modificar una acción personalizada del usuario 63
Acción personalizada. Agregar una acción personalizada del usuario a las acciones del siti 63
Parte web Actualización del título de un elemento web 63
Parte web Agregar un elemento web a una página 64
Parte web Eliminar un elemento web de una página 65
Contexto. Usando un caché de credenciales para la ejecución elevada de código 65

Capítulo 10: Trabajar con Managed Server Side Object Model (plena confianza) 67

Observaciones 67

Jerarquía conceptual 67

Advertencias del lado del servidor 67

Examples 67

Hola Mundo (obteniendo el título del sitio) 67
Recorriendo todo el conjunto de SharePoint 68
Recuperar los elementos de la lista 68
Recuperar elementos utilizando la paginación 68

Obtener la lista por url 69
Crear un elemento de lista 69
Creditos 70

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: sharepoint

It is an unofficial and free sharepoint ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official sharepoint.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con sharepoint

Observaciones

SharePoint puede referirse a uno o más productos de la familia Microsoft SharePoint.

- **SharePoint Foundation** : esta fue la tecnología subyacente para todos los sitios de SharePoint y ya no está disponible para SharePoint 2016
- **SharePoint Server** : esta es la versión local de SharePoint. Puede implementar uno o más servidores de SharePoint. Ofrece características adicionales sobre SharePoint Foundation, como las capacidades de BI, Enterprise Content Management y más
- **SharePoint Online** : versión de SharePoint basada en la nube. El cliente no necesita preocuparse por la infraestructura del servidor o la escalabilidad.

Office 365 es una oferta separada de Microsoft que incluye el servicio de SharePoint Online, aunque no todos los planes son compatibles con todas las características de SharePoint.

Los siguientes enlaces brindan amplias comparaciones de funciones entre las versiones de SharePoint disponibles:

- SharePoint 2013 en las instalaciones frente a SharePoint 2016 en las instalaciones: disponibilidad de funciones en los planes locales de SharePoint
- Características de SharePoint en Office 365: disponibilidad de características en los planes de SharePoint
- Funciones de SharePoint en SharePoint Online (sin Office 365): disponibilidad de funciones en los planes independientes de SharePoint
- Comparación combinada de características entre SharePoint 2013 y SharePoint Online: http://www.buckleyplanet.com/2014/06/sharepoint-online-vs-onprem-feature-comparison.html

Versiones

Versión	Nombre oficial	Fecha de lanzamiento
Pre-2003	SharePoint Portal Server	2002-07-09
2003	SharePoint Portal Server 2003	2003-11-23
2007	SharePoint Server 2007	2007-01-27
2010	Microsoft SharePoint Server 2010	2010-07-15
2013	Microsoft SharePoint Server 2013	2013-01-09
2016	Microsoft SharePoint Server 2016	2016-05-01

Examples

Instalación de SharePoint 2016 para Single Server Farm

Introducción

SharePoint 2016 es la versión 16 de la versión de la familia de productos de SharePoint. Se lanzó el 4 de mayo de 2016. Este ejemplo cubre la instalación de SharePoint 2016 mediante la configuración de la granja de servidores únicos. Esta configuración cubre los conceptos básicos de la configuración de una granja de servidores de SharePoint sin la necesidad de tener varios servidores. Tenga en cuenta que los escenarios cubiertos por una granja de servidores únicos generalmente se limitan al desarrollo y los escenarios de producción muy pequeños.

Requerimientos

Antes de instalar SharePoint, se debe configurar el entorno básico. SharePoint almacena documentos, así como metadatos, registros, aplicaciones personalizadas, personalizaciones y mucho más. Asegúrese de tener suficiente espacio en disco y RAM disponible por encima de los requisitos de la línea base.

- 4 núcleos en procesadores compatibles de 64 bits
- 12 - 24 GB de RAM (dependiendo de la prueba o la implementación de prod)
- Disco duro de 80GB para sistema
- Disco duro de 100GB como segundo disco
- Servidor con Windows Server 2012 R2 de 64 bits o vista previa técnica "Umbral"
- SQL Server 2014 o SQL Server 2016
- .NET Framework 4.5.2 o .NET Framework 4.6
- Dominio unido a la computadora y cuentas de servicio de granja delegada

Todos los demás requisitos previos pueden instalarse manualmente o realizarse utilizando el instalador de requisitos previos de SharePoint incluido con la instalación de SharePoint.

Instalación

- Ejecute el instalador de prerrequisitos; Puede solicitar reiniciar el servidor antes de continuar.
- Ejecute Setup.exe desde la instalación de SharePoint
- Ingrese la clave de licencia
- Aceptar el acuerdo de licencia.
- Seleccione "Completo" en la pestaña Tipo de servidor
- La instalación debe completarse con éxito
- En la página completa, deje la casilla de verificación marcada junto al Asistente para ejecutar la configuración del producto y haga clic en Cerrar

Configuración

Si continúa desde el paso anterior, el Asistente de configuración de productos de SharePoint 2016 se abrirá automáticamente. Si el cuadro no aparece o si está ejecutando la configuración más adelante, abra el asistente de configuración yendo a Inicio -> Productos de SharePoint 2016 -> Asistente de configuración de productos de SharePoint 2016.

- Haga clic a continuación en la página de bienvenida.
- Aparecerá un cuadro de diálogo modal que indica que algunos servicios se reiniciarán durante la configuración; aún no se ha instalado nada, así que haz clic en sí
- Agregue el servidor de base de datos para la granja.
    - Introduzca el nombre de la máquina que ejecuta SQL Server; En este caso, es la máquina local.
    - Ingrese el nombre de la base de datos de configuración o mantenga el nombre predeterminado SharePoint_Config
    - Ingrese el nombre de usuario del usuario del servicio de dominio que accederá a la base de datos (en forma de DOMINIO \ usuario) * Ingrese la contraseña para el usuario del dominio
    - Haga clic en siguiente cuando haya terminado
- Introduzca la contraseña de la granja; Esto se usará al unir servidores adicionales a la nueva granja
- Seleccione el rol de la granja de servidores únicos
- Configure la aplicación web de administración central (donde SharePoint será administrado por los administradores de la granja) seleccione el número de puerto y seleccione el tipo de federación de autenticación (NTLM o Negotate (Kerberose))
- Revise la configuración en las páginas finales y haga los cambios necesarios.
- Cuando esté listo, ejecute la configuración que puede tardar unos minutos
- Al finalizar, abrirá el asistente que le permitirá abrir el sitio de Administración Central
- En caso de error, puede investigar los registros en la carpeta% COMMONPROGRAMFILES% \ Microsoft Shared \ Web Server Extensions \ 16 \ LOG

Configuración de la granja

Una vez que la aplicación web central, la base de datos de configuración y el administrador central estén configurados, estará listo para configurar la granja de servidores para su uso para los usuarios o el desarrollo. Puede marcar la ubicación del sitio de administración central o acceder a ella a través de un acceso directo en la misma ubicación que el Asistente de configuración del producto.

- Si está iniciando la configuración más adelante, haga clic en Inicio rápido -> Asistentes de configuración -> Asistente de configuración de la comunidad
- Si está iniciando el Asistente desde el paso de instalación, haga clic en Iniciar el Asistente
- Elija si desea formar parte del programa de mejora del cliente haciendo clic en Sí o No
- En la página de configuración de la granja, seleccione la cuenta de dominio que ejecutará

los servicios en segundo plano en la granja.
    - Si bien esta cuenta puede ser la misma que la cuenta de la base de datos, también puede ser diferente para la separación de roles y privilegios
    - Ingrese la cuenta como DOMINIO \ usuario
- Valide los servicios que desea que estén disponibles en la granja en la página Servicios
- Cree la primera colección de sitios en la granja (este paso se puede omitir y realizar más adelante)
    - Ingrese el título, la descripción, la dirección web de la colección de sitios (por lo general, el primer sitio se encuentra en la raíz del servidor) y la plantilla
    - La mayoría de las cosas se pueden cambiar (título, descripción) se pueden cambiar fácilmente, pero otras como la URL de la web pueden requerir mucho más trabajo para cambiar; la plantilla tampoco se puede revertir fácilmente, pero SharePoint permite una gran cantidad de personalizaciones que le permiten tomar cualquier plantilla base y convertir el estilo y el diseño del sitio
- Cuando hayas completado la configuración, haz clic en finalizar.

La granja de servidores y la primera colección de sitios ahora están configuradas para su uso.

Construye un elemento web con SharePoint Framework

dev.office.com/sharepoint es un excelente lugar para comenzar a usar SharePoint Framework.

SharePoint Framework es un enfoque moderno del lado del cliente para el desarrollo de SharePoint dirigido inicialmente a SharePoint Online en Office 365. Las partes web creadas con SharePoint Framework son un nuevo tipo de elemento web y pueden estar disponibles para agregarse en las páginas existentes de SharePoint y Nuevas páginas de SharePoint.

Hay un gran ejemplo de "hola mundo" para este proceso alojado en " Construya su primera parte web del lado del cliente de SharePoint" (Hola mundo parte 1) . Todos los ejemplos en dev.office.com están disponibles para contribuciones de la comunidad a través de github.

Los pasos básicos de Hello World en SharePoint Framework son:

1. Genere el esqueleto del proyecto con Yeoman SharePoint Generator .

   yo @ microsoft / SharePoint

2. Edite el código generado en el editor de su elección. El soporte para Visual Studio Code es fuerte en todas las plataformas.

3. Previsualice el elemento web utilizando Gulp y el SharePoint Workbench local.

   un buen trago

4. Vista previa en su entorno de SharePoint Online

   Vaya a la siguiente URL: ' https: //your-sharepoint-site/_layouts/workbench.aspx '

SharePoint ULS Registros y registro

El servicio de registro unificado de SharePoint (ULS) ofrece capacidades de soporte y depuración tanto para operadores como para operadores. Comprender cómo leer los registros es un primer paso importante para resolver problemas.

Estampación

Microsoft proporciona el Visor ULS para ayudar a leer los registros antiguos y los registros que actualmente se están escribiendo mientras se ejecuta la granja. También puede filtrar y aplicar formato a los registros para ayudar a reducir un problema.

Identificador de correlación

Para aislar un problema, es útil mirar solo un ID de correlación particular. Cada identificador de correlación está asociado con una acción de solicitud o de extremo a extremo del sistema (como un trabajador de tiempo). Si hay un problema con la representación de una página web, ubicar la solicitud en los registros de ULS y aislarla a la identificación de correlación específica elimina todo el ruido de los otros registros, lo que ayuda a identificar el problema.

Agregando SPMonitoredScope a mi código

Una forma de calcular el registro de agregar y algo de monitoreo de rendimiento es agregar SPMonitoredScope a su código.

```
using (new SPMonitoredScope("Feature Monitor"))
{
    // My code here
}
```

Este código registrará el principio y el final de sus solicitudes, así como algunos datos de rendimiento. La creación de su propio monitor personalizado que implementa ISPScopedPerformanceMonitor le permite establecer el nivel de seguimiento o el tiempo máximo de ejecución para un conjunto de códigos.

Lea Empezando con sharepoint en línea:
https://riptutorial.com/es/sharepoint/topic/950/empezando-con-sharepoint

Capítulo 2: Aplicación de SharePoint

Introducción

Aplicación alojada de SharePoint

Observaciones

Referencia requerida del sitio: http://www.letsharepoint.com/what-is-user-information-list-in-sharepoint-2013/

Examples

SharePoint 2013: acceso a los datos del servicio de perfiles de usuario utilizando JSOM en SharePoint 2013

SharePoint 2013: acceso a los datos del servicio de perfiles de usuario utilizando JSOM en SharePoint 2013

En este artículo, aprenderemos a administrar o acceder a la Aplicación del Servicio de Perfil de Usuario (UPS) utilizando JSOM (Modelo de Objeto de Javascript) y crearemos una Aplicación básica. Antes de comenzar, veamos primero la terminología básica de UPS.

Perfil de usuario: tiene toda la información de las personas en una organización de manera organizada. Muestra todas las propiedades como AccountName, FirstName, LastName, WorkEmail, etc. relacionadas con un usuario.

Aplicación de servicio de perfiles de usuario: se considera una ubicación centralizada para almacenar todos los perfiles de usuario y también permite a los administradores configurar o administrar perfiles, sincronización de perfiles, Mi sitio, etiquetas sociales, etc. También puede obtener información de servicios de directorio como Active Directory.

Mi sitio: un sitio personalizado para que el usuario individual administre su información y almacene documentos, enlaces, etc. Proporciona una amplia red y características sociales al permitir que los usuarios compartan información sobre ellos mismos o sus actividades. Se puede acceder a Mi sitio haciendo clic en Nombre de usuario en la esquina superior derecha de la página de SharePoint.

Administrar y acceder a datos de perfil de usuario

Ya que vamos a trabajar utilizando JSOM, solo podemos realizar operaciones de 'Lectura' con la excepción de que la imagen del perfil puede cambiarse utilizando JSOM (o CSOM o REST)

* El código del lado del servidor permite leer / escribir ambas operaciones.

Recuperar propiedades de perfil de usuario utilizando JSOM

Permite crear una aplicación alojada de SharePoint y recuperar información del usuario en esa aplicación:

Inicie Visual Studio 2013 y seleccione "Aplicación para SharePoint 2013" desde Nuevo proyecto. Después de seleccionar el tipo de proyecto anterior, aparece una ventana para conectarse al sitio de SharePoint y seleccionar el tipo de aplicación que se implementará (ver la captura de pantalla a continuación). Aquí proporcioné la URL del sitio del desarrollador de SharePoint en línea y seleccioné la aplicación hospedada de SharePoint. Haga clic en Finalizar.

3.) Después de crear el proyecto, verá un conjunto de carpetas / archivadores agregados en el Explorador de soluciones agregado al proyecto de manera predeterminada.

4.) Si abre la página "Default.aspx", encontrará algunas bibliotecas de JavaScript ya agregadas a la página.

Aquí necesitamos agregar una biblioteca más para comenzar a trabajar con los perfiles de usuario

Lea Aplicación de SharePoint en línea: https://riptutorial.com/es/sharepoint/topic/9876/aplicacion-de-sharepoint

Capítulo 3: Creación de una aplicación alojada por el proveedor

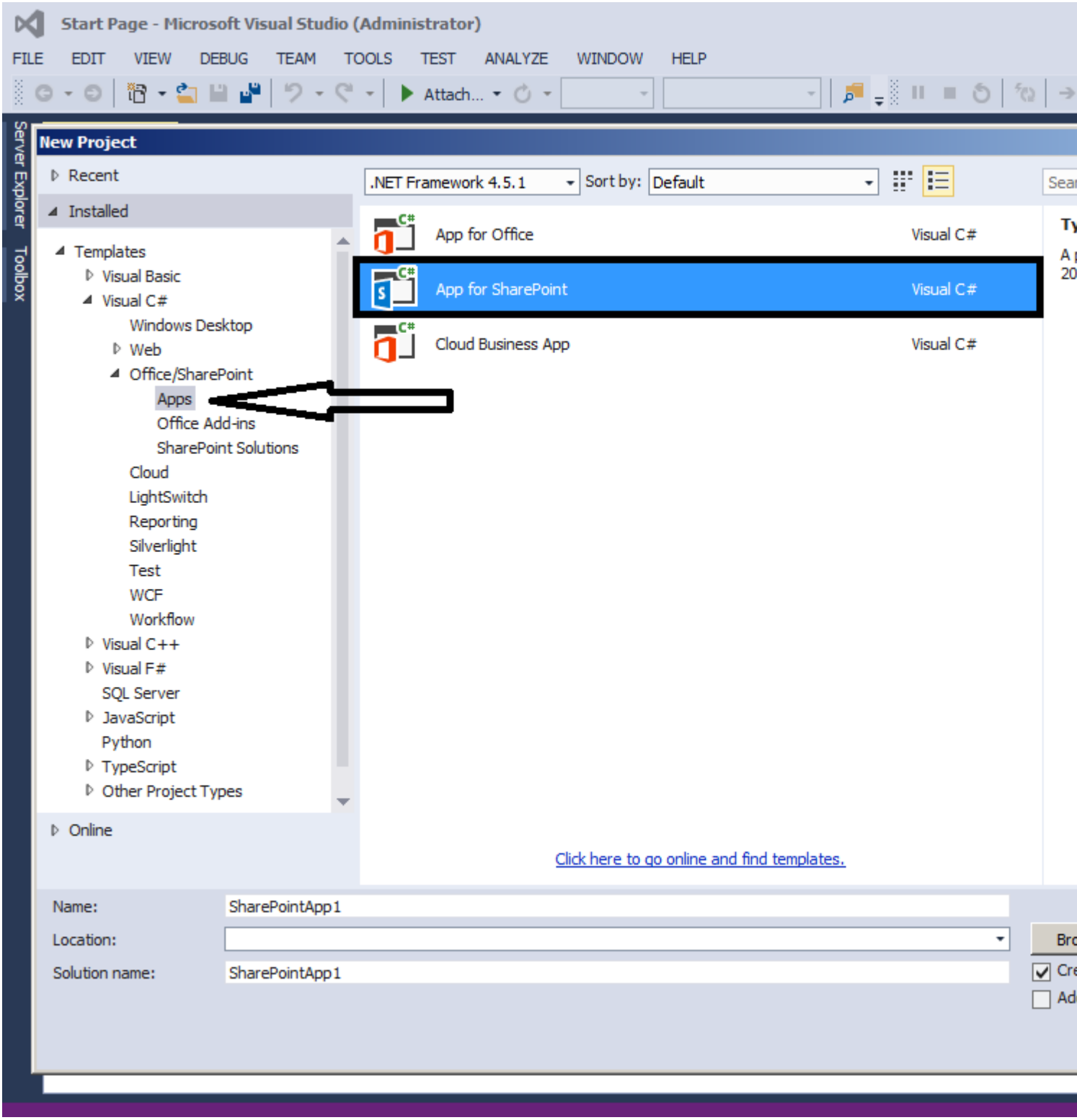
Examples

Entorno de desarrollo de ajuste

Para comenzar con el desarrollo de aplicaciones necesitamos Visual Studio 2013 o una versión superior. Descargue la última edición de la comunidad o expresión desde aquí> https://www.visualstudio.com/products/free-developer-offers-vs

Una vez descargado e instalado

> Abrir y hacer **clic crear nuevo proyecto**
>
> expandir la sección Office / SharePoint debería ver una opción para la aplicación como se muestra a continuación.



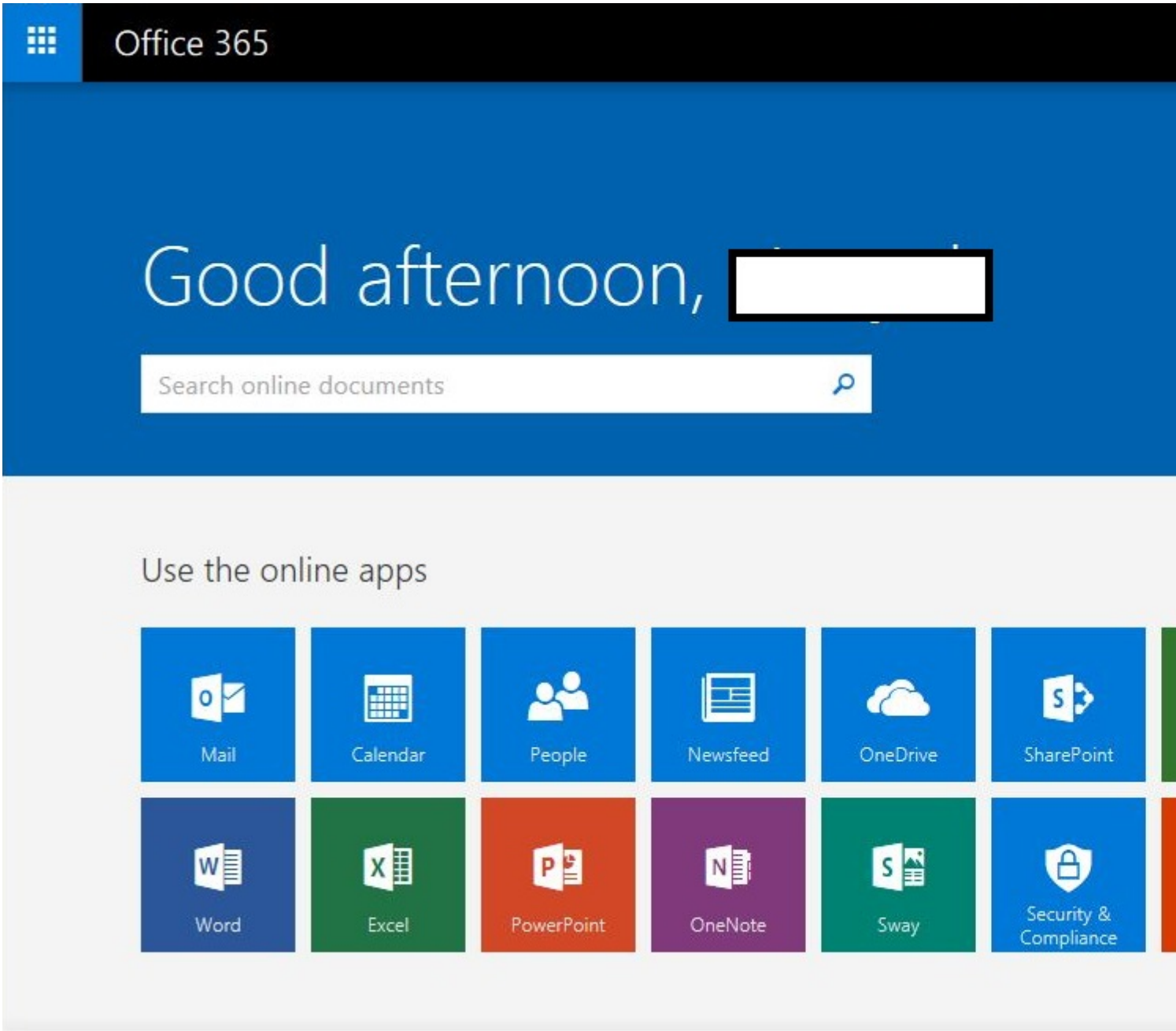
Si la opción de aplicación no está disponible Cierre el VS, descargue e instale **Microsoft Office Developer Tools** https://www.visualstudio.com/en-us/features/office-tools-vs.aspx

Preparación para el sitio de desarrollador

Una vez que tengamos Visual Studio, necesitamos un sitio para desarrolladores para implementar aplicaciones en SharePoint. La forma más sencilla de obtener es> Registrarse para obtener una cuenta de desarrollador de Office 365 gratuita en un año https://profile.microsoft.com/RegSysProfileCenter/wizardnp.aspx?wizid=14b845d0-938c-45af-

b061ffb4d170&lcid=1033

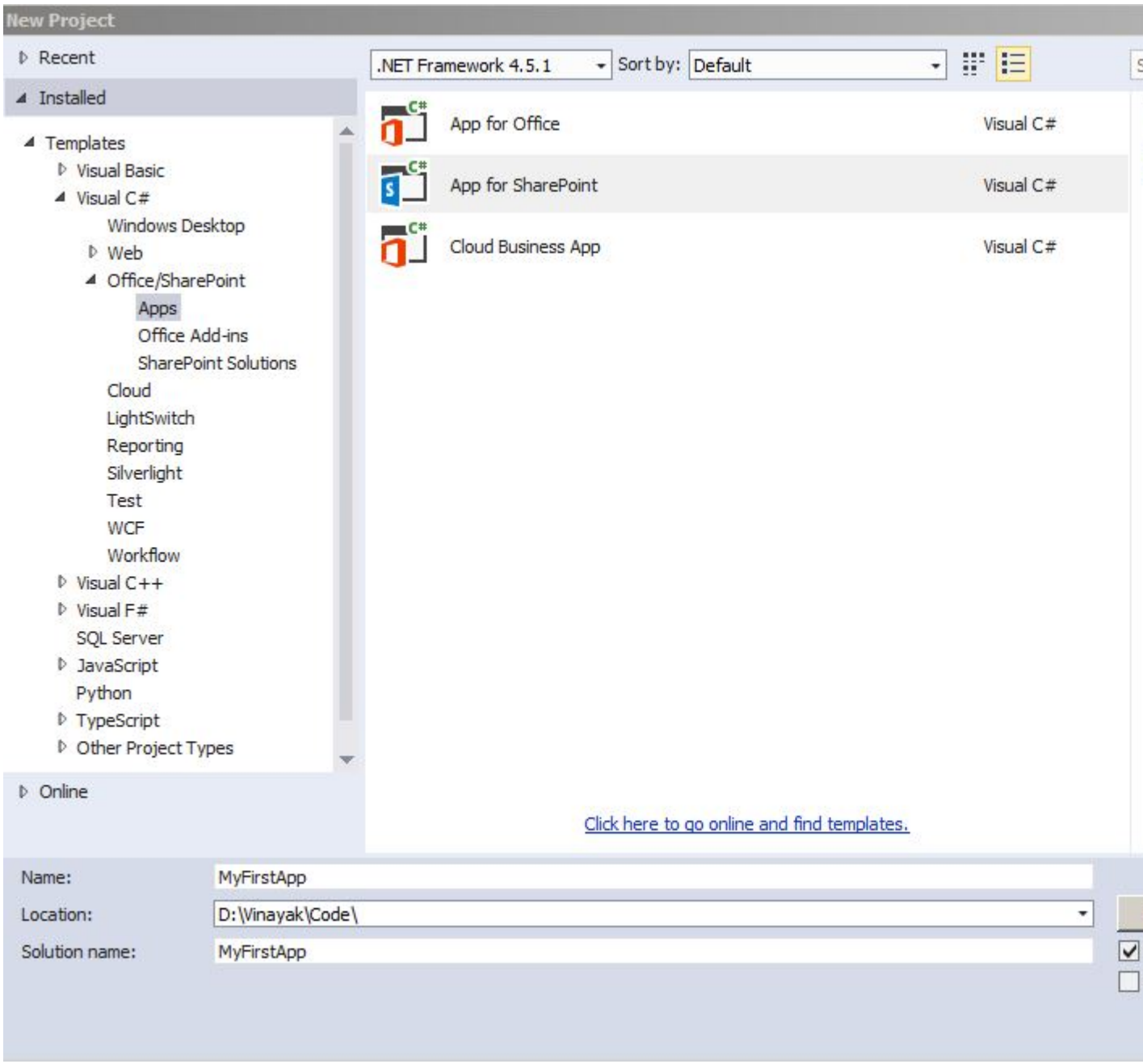
Una vez que finalice el proceso de registro https://www.office.com/ center URL para toda su aplicación



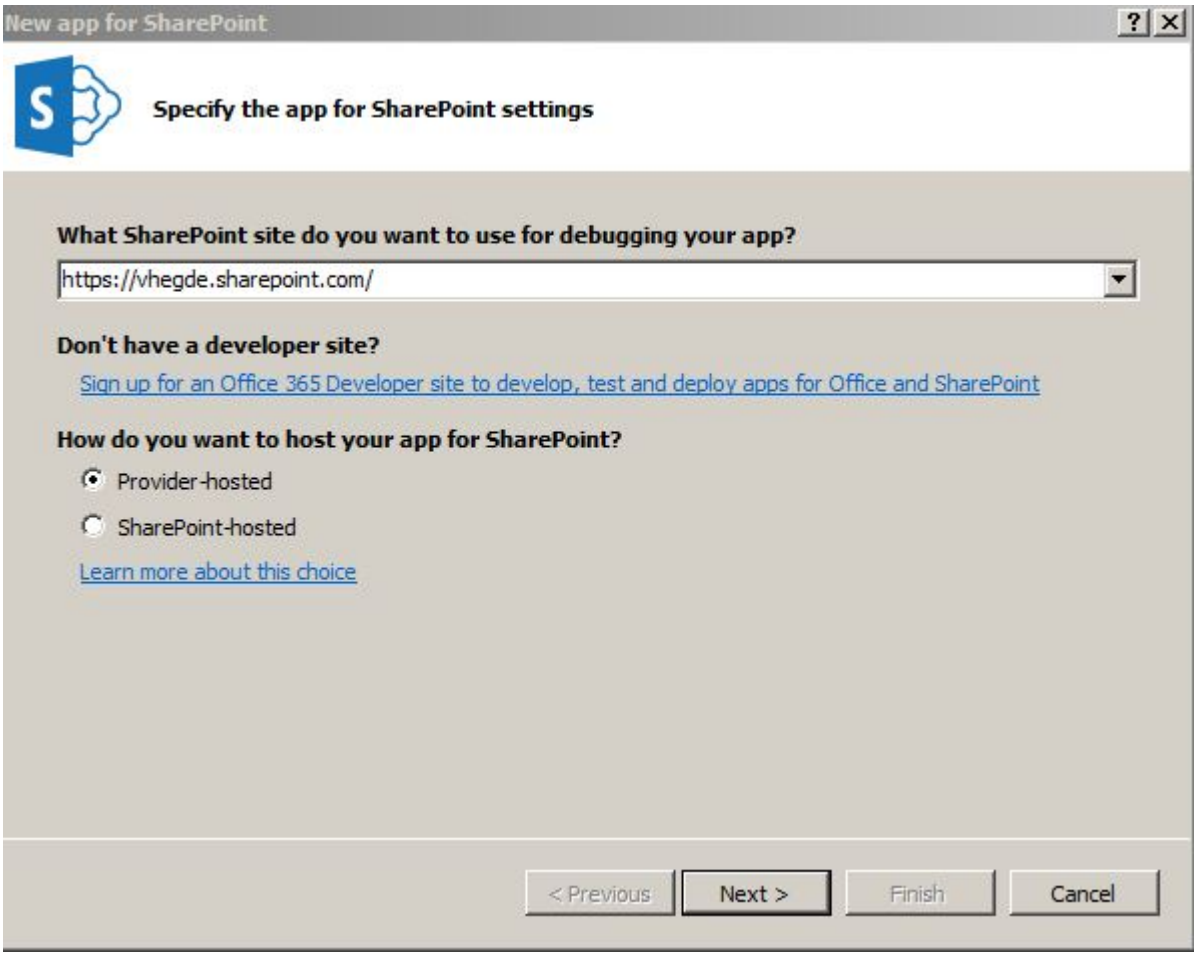
Crear una aplicación en Visual Studio

Comencemos con la creación de nuestra primera aplicación.

1. Abrir estudio visual y crear nuevo proyecto.
2. Ingrese Nombre y Ubicación

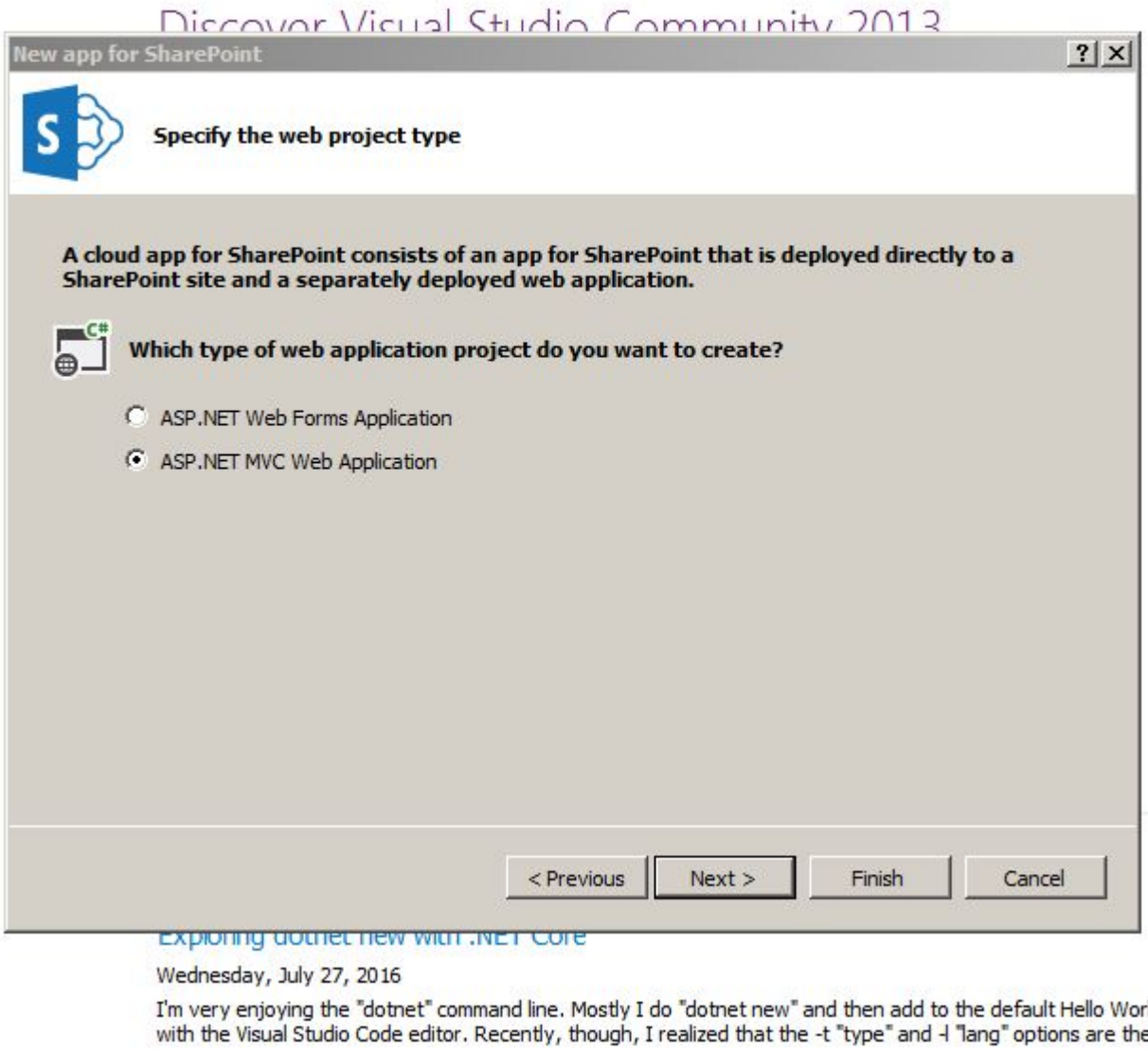


3. Ingrese la URL de su sitio de desarrollador creada en el paso anterior y seleccione Hospedado por el proveedor



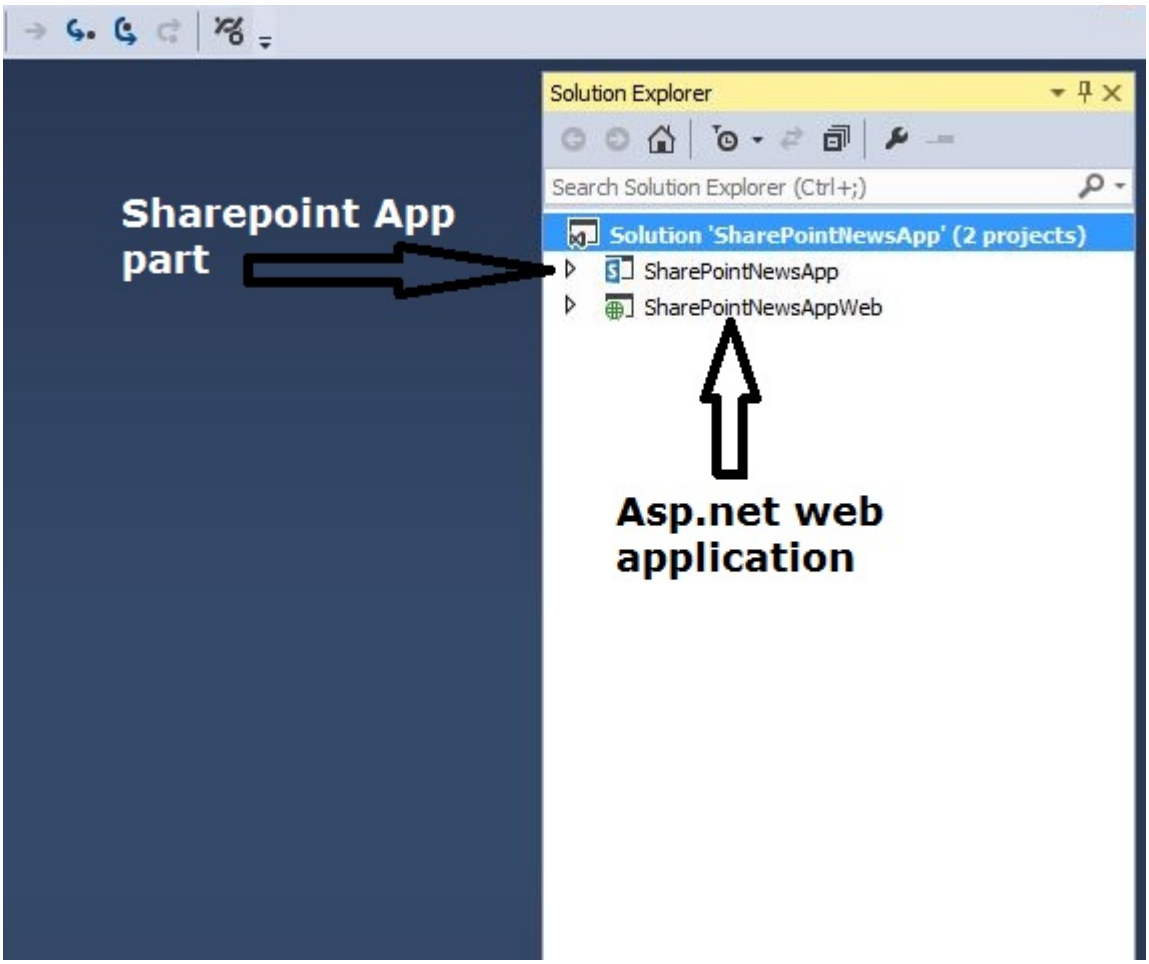
4. Se abrirá una ventana emergente que abrirá para iniciar sesión

5. El siguiente paso será en cuanto al tipo de aplicación, ya sea seleccionar MVC o Webform. Estoy seleccionando MCV aquí



6. En ¿Cómo desea que su complemento se autentique?, Elija Usar el Servicio de control de acceso de Windows Azure y haga clic en Finalizar

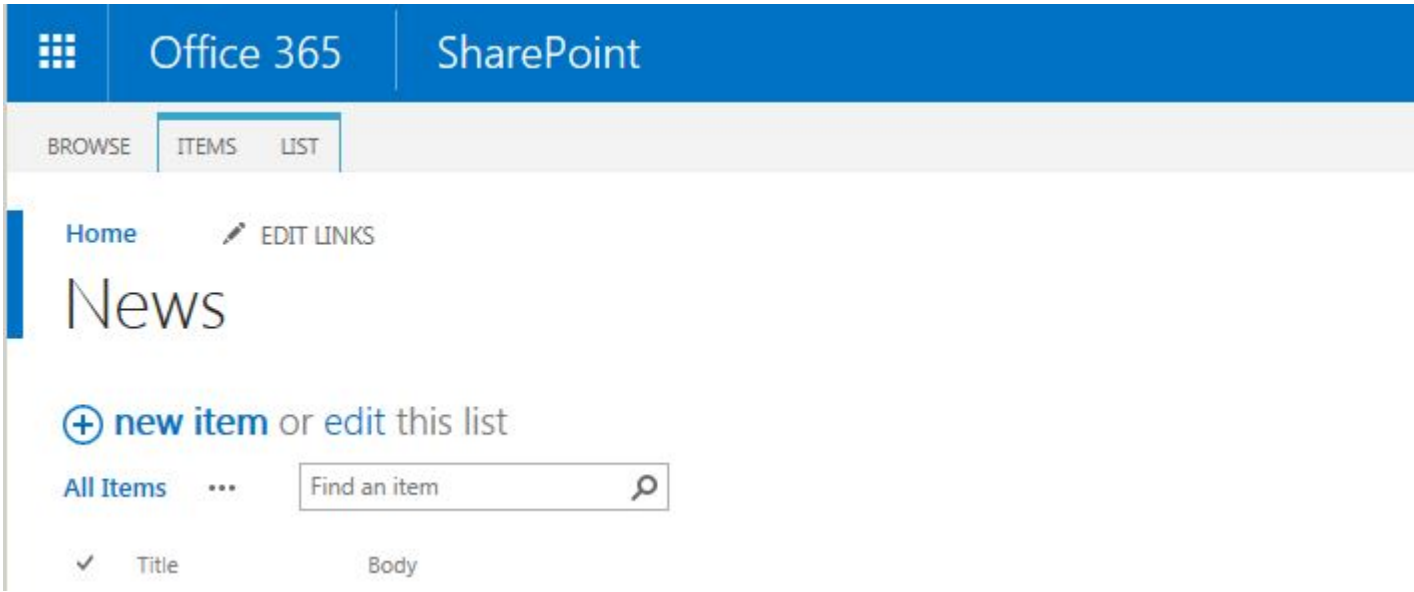
7. En explorador de soluciones podemos ver 2 proyectos creados. Una es la parte de la aplicación de SharePoint y la otra es la aplicación web asp.net



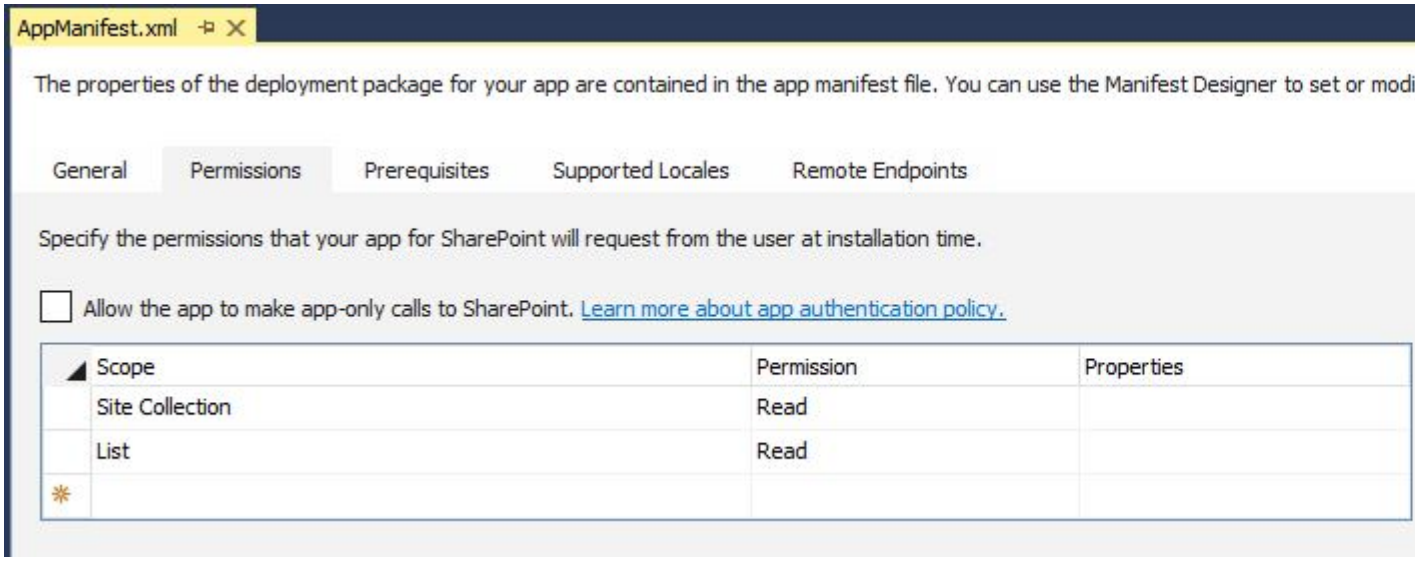
Vamos a empezar a codificar

Aquí estoy tomando el ejemplo de una aplicación de noticias básica.

1. Abra el sitio para desarrolladores de SharePoint y cree una lista para almacenar nuestros artículos de noticias
2. Cree una lista personalizada y agregue 3 columnas más Cuerpo, Summery, ThumbnailImageUrl



3. Regrese a nuestra aplicación de SharePoint, abra el archivo AppManifest.xml, haga clic en la pestaña de permisos y otorgue permiso de Lectura a la colección de sitios y guárdela.



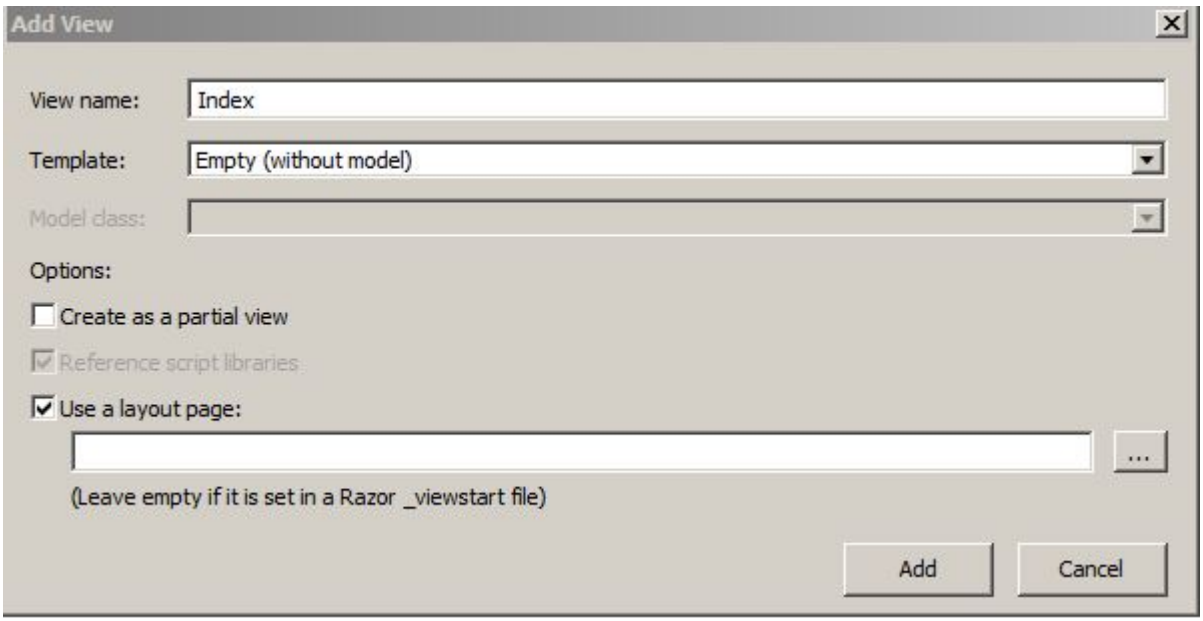
4. Abrir HomeController desde la aplicación web, en mi caso es una aplicación MVC. Si está creando una aplicación de formulario web, su código debería estar en la página default.aspx.cs

5. A continuación se muestra el fragmento de código para obtener las últimas noticias de la lista. Así debería ser nuestra página de índice.

```
[SharePointContextFilter]
public ActionResult Index()
{
    User spUser = null;

    var spContext = SharePointContextProvider.Current.GetSharePointContext(HttpContext);
    List<NewsList> newsList = new List<NewsList>();
    using (var clientContext = spContext.CreateUserClientContextForSPHost())
    {
        if (clientContext != null)
        {
            spUser = clientContext.Web.CurrentUser;

            clientContext.Load(spUser, user => user.Title);

            clientContext.ExecuteQuery();

            ViewBag.UserName = spUser.Title;

            List lst = clientContext.Web.Lists.GetByTitle("News");
            CamlQuery queryNews = CamlQuery.CreateAllItemsQuery(10);
            ListItemCollection newsItems = lst.GetItems(queryNews);
            clientContext.Load(newsItems, includes => includes.Include(i => i.Id, i =>
i.DisplayName, i => i["ThumbnailImageUrl"], i => i["Summery"]));

            clientContext.ExecuteQuery();

            if (newsItems != null)
            {
                foreach (var lstProductItem in newsItems)
                {
```

```
                    newsList.Add(
                        new NewsList
                        {
                            Id = Convert.ToInt32(lstProductItem.Id.ToString()),
                            Title = lstProductItem.DisplayName.ToString(),
                            Summery = lstProductItem["Summery"].ToString(),
                            Thumbnail = lstProductItem["ThumbnailImageUrl"].ToString()
                        });
                }
            }
        }
    }

    return View(newsList);
}
```

6. Ahora haga clic derecho en **Índice** y haga clic en **Agregar vista.** Luego haga clic en Agregar



7. Ahora abra el archivo **Index.cshtml** Desde el **directorio Vistas> Inicio**

8. A continuación se muestra el fragmento de código para el archivo index.cshtml

```
@model List<SharePointNewsAppWeb.Models.NewsList>
 @{
 ViewBag.Title = "My News - browse latest news";
}
 <br />
 @foreach (var item in Model)
 {
 <div class="row panel panel-default">
  <div class="col-xs-3">
    <a href="/home/aticle?ArticleId=@item.Id">
        <img class="img-responsive" style="max-height:200px;max-width:100%;"
src="@item.Thumbnail" alt="@item.Title" />
    </a>
</div>
<div class="col-xs-9 panel-default">
```

```
    <div class="panel-heading">
        <h4><a href="/home/aticle?ArticleId=@item.Id">@item.Title.ToUpper()</a></h4>
    </div>
    <div class="panel-body">
        <p>@item.Summery</p>
    </div>
</div>
```

9. Haga clic derecho en la carpeta Modelo de su solución y agregue un archivo de clase CS. Añadir a continuación las clases de modelos

```
 using System;
 using System.Collections.Generic;
 using System.Linq;
 using System.Web;

 namespace SharePointNewsAppWeb.Models
{
  public class NewsApp
   {

   }
 public class NewsList
  {

public int Id { get; set; }

public string Title { get; set; }

public string Summery { get; set; }

public string Thumbnail { get; set; }
 }
 public class FullArticle
 {

  public int Id { get; set; }

  public string Title { get; set; }

  public string Body { get; set; }

 }
}
```

10. Use la tecla F5 para implementar y ejecutar su complemento. Si ve una ventana de Alerta de seguridad que le pide que confíe en el certificado de Localhost autofirmado, elija Sí.

Y ahora la primera aplicación está lista.

Creación de la página del artículo completo

Ya hemos creado la primera página que mostrará todos los artículos de noticias. Esta página mostrará el artículo completo.

1. Añadir un método de acción más a HomeController

```
[SharePointContextFilter]
public ActionResult Aticle(int ArticleId)
{
    User spUser = null;

    var spContext = SharePointContextProvider.Current.GetSharePointContext(HttpContext);
    FullArticle article = new FullArticle();
    using (var clientContext = spContext.CreateUserClientContextForSPHost())
    {
        if (clientContext != null)
        {
            spUser = clientContext.Web.CurrentUser;

            clientContext.Load(spUser, user => user.Title);

            clientContext.ExecuteQuery();

            ViewBag.UserName = spUser.Title;

            List lst = clientContext.Web.Lists.GetByTitle("News");
            CamlQuery queryNews = new CamlQuery();
            queryNews.ViewXml = @"<View><Query><Where><Eq><FieldRef Name='ID'/>" +
"<Value Type='Number'>" + ArticleId + "</Value></Eq></Where></Query>" +
                "<ViewFields><FieldRef Name='ID'/><FieldRef Name='Title'/><FieldRef
Name='Body'/></ViewFields></View>";//
            ListItemCollection newsItems = lst.GetItems(queryNews);
            clientContext.Load(newsItems, includes => includes.Include(i => i.Id, i =>
i.DisplayName, i => i["Body"]));

            clientContext.ExecuteQuery();

            if (newsItems != null)
            {
                foreach (var lstProductItem in newsItems)
                {
                    article.Id = Convert.ToInt32(lstProductItem.Id.ToString());
                    article.Title = lstProductItem.DisplayName.ToString();
                    article.Body = lstProductItem["Body"].ToString();
                }
            }
        }
    }
    return View(article);
}
```

2. De nuevo, haga clic derecho en Acción y cree una vista con el mismo nombre Nombre del método de acción. En mi caso la vista se llamará **Aticle**

```
  @model SharePointNewsAppWeb.Models.FullArticle

@{
 ViewBag.Title = "Aticle";
}

<br />
<div class="panel panel-default">
 <div class="panel-heading"><a style="font-size:20px;" href="/"><i   class="glyphicon
glyphicon-chevron-left"></i> <i class="glyphicon glyphicon-home"></i>      </a></div>
 <div class="panel-heading"><h1 class="h2">@Model.Title.ToUpper()</h1></div>
```

```
 <div class="panel-body">@Html.Raw(@Model.Body)</div>
</div>
```

Este es el código de la página del artículo completo que muestra el cuerpo del artículo de noticias

Lea Creación de una aplicación alojada por el proveedor en línea: https://riptutorial.com/es/sharepoint/topic/6301/creacion-de-una-aplicacion-alojada-por-el-proveedor

Capítulo 4: Principales lanzamientos

Examples

SharePoint 2016

Número de compilación	Descripción	Producto
16.0.4366.1000	Actualización acumulativa abril 2016	SharePoint Server 2016
16.0.4336.1000	RTM	SharePoint Server 2016
16.0.4327.1000	Candidato de lanzamiento	SharePoint Server 2016
16.0.4266.1001	16.0.4306.1002 Beta 2	SharePoint Server 2016

SharePoint 2013

Número de compilación	Descripción
15.0.4623.1001	Junio de 2014
15.0.4631.1001	Julio de 2014
15.0.4641.1001	Agosto 2014
15.0.4649.1001	Septiembre 2014
15.0.4659.1001	Octubre 2014
15.0.4667.1000	Noviembre de 2014
15.0.4675.1000	Diciembre 2014
15.0.4693.1001	Febrero 2015
15.0.4701.1001	Marzo 2015
15.0.4711.1000	Abril 2015 (SP1 REQ)
15.0.4719.1002	Mayo 2015
15.0.4727.1001	Junio 2015
15.0.4737.1000	Julio 2015
15.0.4745.1000	Agosto 2015

Número de compilación	Descripción
15.0.4753.1003	Septiembre 2015
15.0.4763.1002	Octubre 2015
15.0.4771.1000	Noviembre 2015
15.0.4779.1000	Diciembre 2015
15.0.4787.1000	MS16-004
15.0.4787.1000	Enero 2016
15.0.4797.1001	Febrero 2016
15.0.4805.1000	Marzo 2016
15.0.4815.1000	Abril 2016
15.0.4823.1003	Mayo 2016
15.0.4833.1000	Junio 2016

Fuente: Números de compilación de SharePoint 2013 y CU

Lea Principales lanzamientos en línea: https://riptutorial.com/es/sharepoint/topic/2737/principales-lanzamientos

Capítulo 5: Representación del lado del cliente de SharePoint 2013

Introducción

Client Side Rendering (CSR) es un nuevo concepto que se introdujo en SharePoint 2013. Le proporciona un mecanismo que le permite usar su propio render de salida para un conjunto de controles que se alojan en una página de SharePoint (vistas de lista, formularios de lista y resultados de búsqueda). La representación del sitio del cliente es simplemente cuando los datos se transforman utilizando el cliente en lugar del servidor. Esto significa usar tecnologías del lado del cliente, como HTML y JavaScript en lugar de tener que escribir XSLT.

Examples

Cambiar el hipervínculo de campos / columnas dentro de la vista de lista usando CSR

El siguiente ejemplo muestra cómo cambiar el hipervínculo del campo " **ID** " y " **Título (Título de enlace)** " dentro de la vista de lista mediante CSR.

Paso 1: Crea un archivo JS y pega el siguiente código

```
(function () {

    function registerRenderer() {
        var ctxForm = {};
        ctxForm.Templates = {};

        ctxForm.Templates = {
            Fields : {
                'LinkTitle': { //------ Change Hyperlink of LinkTitle
                    View : function (ctx) {
                        var url = String.format('{0}?ID={1}',
"/sites/Lists/testlist/EditItem.aspx", ctx.CurrentItem.ID);
                        return String.format('<a href="{0}" onclick="EditItem2(event,
\'{0}\');return false;">{1}</a>', url, ctx.CurrentItem.Title);
                    }
                },
                'ID' : { //------ Change Hyperlink from ID field
                    View : function (ctx) {
                        var url = String.format('{0}?ID={1}',
"/IssueTracker/Lists/testlist/DisplayItem.aspx", ctx.CurrentItem.ID);
                        return String.format('<a href="{0}" onclick="EditItem2(event,
\'{0}\');return false;">{1}</a>', url, ctx.CurrentItem.ID);
                    }
                },

            }
        };
        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
```

```
    }
    ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js');

})();
```

Paso 2: Ir a las propiedades del elemento web de la Vista de lista y agregar la referencia de JS Link a este archivo js recién creado (por ejemplo, ~ sitecollection / SiteAssets / CSRCodeFile.js)

(Nota: consulte su JSlink solo en este formato. "~ Sitecollection / YourJSfFilePath".)

Paso 3: Appy y Hecho

Ocultar columna de la vista de lista de SharePoint usando CSR.

Este ejemplo muestra cómo ocultar un campo "Fecha" de la vista de lista de SharePoint mediante CSR.

```
(function () {

    function RemoveFields(ctx) {
        var fieldName = "Date"; // here Date is field or column name to be hide
        var header = document.querySelectorAll("[displayname=" + fieldName +
"]")[0].parentNode;
        var index = [].slice.call(header.parentNode.children).indexOf(header) + 1;
        header.style.display = "none";
        for (var i = 0, cells = document.querySelectorAll("td:nth-child(" + index + ")"); i <
cells.length; i++) {
            cells[i].style.display = "none";
        }
    }

    function registerRenderer() {
        var ctxForm = {};
        ctxForm.Templates = {};
        ctxForm.OnPostRender = RemoveFields;
        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
    }
    ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js');

})();
```

Aplicar validaciones en el formulario Nuevo / Editar artículo usando CSR

Supongamos que tenemos una lista de SharePoint y tiene cuatro campos a saber. Título, nombre completo, correo electrónico, número de teléfono móvil, etc. Ahora, si desea aplicar la validación personalizada en el formulario Nuevo / Editar elemento, puede hacerlo fácilmente con el código CSR. Lo siguiente mencionado puede validar las siguientes condiciones en los formularios:

- Valores en blanco en los campos
- Comprobación de formato de correo electrónico con expresión regular
- Formato de número de móvil Compruebe con expresión regular
- El campo Nombre completo no debe contener valores numéricos

Paso: 1 Cree un archivo JS, diga `CSRValidations.js` y copie y pegue el siguiente código en el archivo JS

```
        (function () {

        // Create object that have the context information about the field that we want to
change it's output render
        var fieldContext = {};
        fieldContext.Templates = {};
        fieldContext.Templates.Fields = {
            // Apply the new rendering for Email field on New and Edit Forms
            "Title": {
                "NewForm": titleFieldTemplate,
                "EditForm":  titleFieldTemplate
            },
            "Full_x0020_Name": {
                "NewForm": fullNameFieldTemplate,
                "EditForm":  fullNameFieldTemplate
            },
            "Email": {
                "NewForm": emailFieldTemplate,
                "EditForm":  emailFieldTemplate
            },
            "Mobile_x0020_Phone": {
                "NewForm": mobilePhoneFieldTemplate,
                "EditForm":  mobilePhoneFieldTemplate
            }
        };

        SPClientTemplates.TemplateManager.RegisterTemplateOverrides(fieldContext);

    })();

    // This function provides the rendering logic
    function emailFieldTemplate(ctx) {

        var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);

        // Register a callback just before submit.
        formCtx.registerGetValueCallback(formCtx.fieldName, function () {
            return document.getElementById('inpEmail').value;
        });

        //Create container for various validations
        var validators = new SPClientForms.ClientValidation.ValidatorSet();
        validators.RegisterValidator(new emailValidator());

        // Validation failure handler.
        formCtx.registerValidationErrorCallback(formCtx.fieldName, emailOnError);

        formCtx.registerClientValidator(formCtx.fieldName, validators);

        return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpEmail' class='ms-long'> \ <br><span id='spnEmailError' class='ms-
formvalidation ms-csrformvalidation'></span></span>";
    }

    // Custom validation object to validate email format
    emailValidator = function () {
        emailValidator.prototype.Validate = function (value) {
```

```
            var isError = false;
            var errorMessage = "";

            //Email format Regex expression
            //var emailRejex = /\S+@\S+\.\S+/;
            var emailRejex = /^(([^<>()[\]HYPERLINK
"\\.,;:\s@\"\\.,;:\s@\"]+(\.[^<>()[\]HYPERLINK "\\.,;:\s@\"\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

            if (value.trim() == "") {
                isError = true;
                errorMessage = "You must specify a value for this required field.";
            }else if (!emailRejex.test(value) && value.trim()) {
                isError = true;
                errorMessage = "Please enter valid email address";
            }

            //Send error message to error callback function (emailOnError)
            return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

        };
    };

    // Add error message to spnError element under the input field element
    function emailOnError(error) {
        document.getElementById("spnEmailError").innerHTML = "<span role='alert'>" +
error.errorMessage + "</span>";
    }

    // This function provides the rendering logic
    function titleFieldTemplate(ctx) {

        var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);
        // Register a callback just before submit.

        formCtx.registerGetValueCallback(formCtx.fieldName, function () {
            return document.getElementById('inpTitle').value;
        });

        //Create container for various validations
        var validators = new SPClientForms.ClientValidation.ValidatorSet();
        validators.RegisterValidator(new titleValidator());

        // Validation failure handler.
        formCtx.registerValidationErrorCallback(formCtx.fieldName, titleOnError);

        formCtx.registerClientValidator(formCtx.fieldName, validators);

        return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpTitle' class='ms-long'> \ <br><span id='spnTitleError' class='ms-
formvalidation ms-csrformvalidation'></span></span>";
    }

    // Custom validation object to validate title format
    titleValidator = function () {
        titleValidator.prototype.Validate = function (value) {
            var isError = false;
            var errorMessage = "";

            if (value.trim() == "") {
                isError = true;
```

```
                errorMessage = "You must specify a value for this required field.";
            }

            //Send error message to error callback function (titleOnError)
            return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

        };
    };

    // Add error message to spnError element under the input field element
    function titleOnError(error) {
        document.getElementById("spnTitleError").innerHTML = "<span role='alert'>" +
error.errorMessage + "</span>";
    }

    // This function provides the rendering logic
    function mobilePhoneFieldTemplate(ctx) {

        var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);

        // Register a callback just before submit.
        formCtx.registerGetValueCallback(formCtx.fieldName, function () {
            return document.getElementById('inpMobilePhone').value;
        });

        //Create container for various validations
        var validators = new SPClientForms.ClientValidation.ValidatorSet();
        validators.RegisterValidator(new mobilePhoneValidator());

        // Validation failure handler.
        formCtx.registerValidationErrorCallback(formCtx.fieldName, mobilePhoneOnError);

        formCtx.registerClientValidator(formCtx.fieldName, validators);

        return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpMobilePhone' class='ms-long'> \ <br><span id='spnMobilePhoneError'
class='ms-formvalidation ms-csrformvalidation'></span></span>";
    }

    // Custom validation object to validate mobilePhone format
    mobilePhoneValidator = function () {
        mobilePhoneValidator.prototype.Validate = function (value) {
            var isError = false;
            var errorMessage = "";

            //MobilePhone format Regex expression
            //var mobilePhoneRejex = /\S+@\S+\.\S+/;
            var mobilePhoneRejex = /^[0-9]+$/;

            if (value.trim() == "") {
                isError = true;
                errorMessage = "You must specify a value for this required field.";
            }else if (!mobilePhoneRejex.test(value) && value.trim()) {
                isError = true;
                errorMessage = "Please enter valid mobile phone number";
            }

            //Send error message to error callback function (mobilePhoneOnError)
            return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

        };
```

```
    };

    // Add error message to spnError element under the input field element
    function mobilePhoneOnError(error) {
        document.getElementById("spnMobilePhoneError").innerHTML = "<span role='alert'>" +
error.errorMessage + "</span>";
    }

    // This function provides the rendering logic
    function fullNameFieldTemplate(ctx) {

        var formCtx = SPClientTemplates.Utility.GetFormContextForCurrentField(ctx);

        // Register a callback just before submit.
        formCtx.registerGetValueCallback(formCtx.fieldName, function () {
            return document.getElementById('inpFullName').value;
        });

        //Create container for various validations
        var validators = new SPClientForms.ClientValidation.ValidatorSet();
        validators.RegisterValidator(new fullNameValidator());

        // Validation failure handler.
        formCtx.registerValidationErrorCallback(formCtx.fieldName, fullNameOnError);

        formCtx.registerClientValidator(formCtx.fieldName, validators);

        return "<span dir='none'><input type='text' value='" + formCtx.fieldValue + "'
maxlength='255' id='inpFullName' class='ms-long'> \ <br><span id='spnFullNameError' class='ms-
formvalidation ms-csrformvalidation'></span></span>";
    }

    // Custom validation object to validate fullName format
    fullNameValidator = function () {
        fullNameValidator.prototype.Validate = function (value) {
            var isError = false;
            var errorMessage = "";

            //FullName format Regex expression
            var fullNameRejex = /^[a-z ,.'-]+$/i;

            if (value.trim() == "") {
                isError = true;
                errorMessage = "You must specify a value for this required field.";
            }else if (!fullNameRejex.test(value) && value.trim()) {
                isError = true;
                errorMessage = "Please enter valid name";
            }

            //Send error message to error callback function (fullNameOnError)
            return new SPClientForms.ClientValidation.ValidationResult(isError, errorMessage);

        };
    };

    // Add error message to spnError element under the input field element
    function fullNameOnError(error) {
        document.getElementById("spnFullNameError").innerHTML = "<span role='alert'>" +
error.errorMessage + "</span>";
    }
```

Paso: 2 Abrir el formulario de nuevo elemento en el navegador. Editar página y editar parte web.

Paso: 3 En las propiedades del elemento Web, vaya a Misceláneos -> Enlace JS -> pegue la ruta de acceso de su archivo js (por ejemplo, ~ sitecollection / SiteAssets / CSRValidations.js)

Paso: 4 Guardar las páginas y las propiedades del elemento web.

Cambiar el nombre de visualización de la columna en la vista de lista usando CSR

Hay casos en los que necesita cambiar el Nombre para mostrar de la columna en una vista de lista

por ejemplo, el nombre de la columna que se muestra en la vista es "IsApprovalNeeded" y desea que aparezca como "¿Se necesita aprobación?".

Por supuesto, puede cambiar el nombre de visualización de una columna cambiando el título de la columna en la configuración de la lista, pero si desea mantenerlo como está en la configuración de la lista y solo modificarlo en la vista previa de la página, entonces puede hacerlo usando CSR (Client-Side-Rendering).

Aquí está el código ...

```
(function () {

    function preTaskFormRenderer(renderCtx) {
       modifyColumns(renderCtx);
    }

    function modifyColumns(renderCtx)
    {
      var arrayLength= renderCtx.ListSchema.Field.length;
        for (var i=0; i < arrayLength;i++)
        {
           if(renderCtx.ListSchema.Field[i].DisplayName == 'IsApprovalNeeded')
             {
               var newTitle= "Is Approval Needed?";
               var linkTitleField = renderCtx.ListSchema.Field[i];
               linkTitleField.DisplayName = newTitle;
             }
         }
    }

    function registerRenderer()
    {
      var ctxForm = {};
      ctxForm.Templates = {};
      ctxForm.OnPreRender = preTaskFormRenderer;
      SPClientTemplates.TemplateManager.RegisterTemplateOverrides(ctxForm);
    }

    ExecuteOrDelayUntilScriptLoaded(registerRenderer, 'clienttemplates.js');

})();
```

Lea Representación del lado del cliente de SharePoint 2013 en línea: https://riptutorial.com/es/sharepoint/topic/8317/representacion-del-lado-del-cliente-de-sharepoint-2013

Capítulo 6: Servicios de descanso

Observaciones

URL de punto final de servicio REST

La API de acceso de cliente REST se introdujo por primera vez en SharePoint 2010, pero se amplió considerablemente en SharePoint 2013. Se accede a la API REST en SharePoint 2010 a través del servicio web ListData en la URL `/_vti_bin/ListData.svc` . SharePoint 2013 introdujo las URL de punto final de `/_api/lists/` y `/_api/web` , que se comportan de forma ligeramente diferente.

Las URL de puntos finales anteriores deben ir precedidas por `http://server/site` donde el `server` representa el nombre del servidor, y el `site` representa el nombre o la ruta al sitio específico.

URL de ejemplo para ...	SharePoint 2010	SharePoint 2013
Obteniendo una lista:	`/_vti_bin/ListData.svc/ListName`	`/_api/lists('ListGuid')`
Obteniendo un artículo:	`/_vti_bin/ListData.svc/ListName(1)`	`/_api/lists('ListGuid')/items(1)`
Obteniendo una web:	(no equivalente)	`/_api/web`

A pesar de las diferencias en el acceso a las listas y los elementos de la lista, trabajar con esos resultados es muy similar en ambas versiones.

Tenga en cuenta que el servicio `ListData.svc` todavía está disponible en SharePoint 2013 por compatibilidad con versiones anteriores.

Enviando solicitudes de REST

Una solicitud REST se puede enviar a través de un JavaScript nativo XMLHttpRequest o a través de la construcción de envoltura jQuery AJAX.

Sintaxis de XMLHttpRequest

```
var xhr = new XMLHttpRequest();
xhr.open(verb, url, true);
xhr.setRequestHeader("Content-Type","application/json");
xhr.send(data);
```

jQuery AJAX Sintaxis

```
$.ajax({
    method: verb,
    url: url,
    headers: { "Content-Type":"application/json" },
    data: data
});
```

Para obtener más detalles sobre el envío de solicitudes a través de AJAX, consulte la documentación de JavaScript AJAX .

Examples

Trabajando con listas

Obtener artículos de la lista

Este ejemplo muestra cómo recuperar todos los elementos de la lista e iterarlos. Puede utilizar el parámetro `top` para solicitar un cierto número de resultados. También puede usar el parámetro `select` para seleccionar ciertos campos (`$select=id, Title, uri`).

JavaScript

```
function GetListItems(){
    $.ajax({
        url: "../_api/web/lists/getbytitle('List Title')/items?$top=50"
        contentType: "application/json;odata=verbose",
        method: "GET",
        headers: { "accept": "application/json;odata=verbose" },
        success: function (data) {
            $.each(data.d.results, function(index,item){
                //use item to access the individual list item
                console.log(item.Id);
            });
        },
        error: function(error){
            console.log(error);
        }
    });
    }
```

Obtener un elemento de lista individual

JavaScript

```
function GetListItem(){
    $.ajax({
        url: "../_api/web/lists/getbytitle('List Title')/items(1)",
        contentType: "application/json;odata=verbose",
        method: "GET",
        headers: { "accept": "application/json;odata=verbose" },
```

```
        success: function (data) {
            console.log(data.d.Id);
        },
        error: function(error){
            console.log(error);
        }
    });
    }
```

Obtener elementos de lista con columnas de búsqueda

A veces, puede tener una estructura de lista que se parece a esto:

Tabla de listado de animales

Nombre	Tipo	Descripción
Título	Cadena (Texto)	Nombre del animal
Años	Número	Cuantos años tiene el animal
Valor	Moneda	Valor del animal
Tipo	*Búsqueda (tabla de tipos de animales)*	Campo de búsqueda (ofrece un menú desplegable de opciones de la tabla de tipos de animales)

Tabla de tipos de animales

Nombre	Tipo	Descripción
Título	Cadena (Texto)	Nombre de la especie / tipo de animal (ej. Cerdo)
NumLegs	Número	Número de patas sobre el animal.

Probablemente no sea el ejemplo más serio, pero el problema aquí sigue siendo válido. Cuando utiliza la solicitud habitual para recuperar valores de la lista de SharePoint, para el `Type` del animal, solo obtendrá un campo llamado `TypeId` en la respuesta JSON. Para expandir estos elementos en una sola llamada AJAX, se requiere un marcado adicional en los parámetros de la URL.

Este ejemplo se aplica a más que solo columnas de búsqueda también. Cuando utiliza las columnas `People/Groups` , también son esencialmente solo búsquedas, por lo que puede extraer elementos como `Title` , `EMail` y otros fácilmente.

Código de ejemplo

Nota importante : cuando defina los campos que desea recuperar de las columnas de búsqueda, debe prefijar el nombre del campo con el nombre del campo de búsqueda en la tabla original. Por ejemplo, si desea recuperar el atributo `NumLegs` de la columna de búsqueda, debe escribir `Type/NumLegs` .

JavaScript

```
// webUrl: The url of the site (ex. https://www.contoso.com/sites/animals)
// listTitle: The name of the list you want to query
// selectFields: the specific fields you want to get back
// expandFields: the name of the fields that need to be pulled from lookup tables
// callback: the name of the callback function on success
function getItems(webUrl,listTitle,selectFields, expandFields, callback){
    var endpointUrl = webUrl + "/_api/web/lists/getbytitle('" + listTitle + "')/items";
    endpointUrl+= '?$select=' + selectFields.join(",");
    endpointUrl+= '&$expand=' + expandFields.join(",");
    return executeRequest(endpointUrl,'GET', callback);
}

function executeRequest(url,method,callback,headers,payload)
{
    if (typeof headers == 'undefined'){
        headers = {};
    }
    headers["Accept"] = "application/json;odata=verbose";
    if(method == "POST") {
        headers["X-RequestDigest"] = $("#__REQUESTDIGEST").val();
    }

    var ajaxOptions =
    {
    url: url,
    type: method,
    contentType: "application/json;odata=verbose",
    headers: headers,
    success: function (data) { callback(data) }
    };
    if(method == "POST") {
    ajaxOptions.data = JSON.stringify(payload);
    }

    return $.ajax(ajaxOptions);
}

// Setup the ajax request by setting all of the arguments to the getItems function
function getAnimals() {
    var url = "https://www.contoso.com/sites/animals";
    var listTitle = "AnimalListing";

    var selectFields = [
        "Title",
        "Age",
        "Value",
        "Type/Title",
        "Type/NumLegs"
    ];

    var expandFields = [
        "Type/Title",
```

```
        "Type/NumLegs"
    ];

    getItems(url, listTitle, selectFields, expandFields, processAnimals);
}

// Callback function
// data: returns the data given by SharePoint
function processAnimals(data) {
    console.log(data);
    // Process data here
}

// Start the entire process
getAnimals();
```

Agregar selecciones a un campo de búsqueda multivalor

Este ejemplo asume que su columna de búsqueda se llama `MultiLookupColumnName` y que desea configurar su campo de búsqueda múltiple para buscar los elementos con ID 1 y 2.

Usando jQuery AJAX

2010

```
var listName = "YourListName";
var lookupList = "LookupListName";
var idOfItemToUpdate = 1;
var url = "/server/site/_vti_bin/ListData.svc/"+listName+"("+idOfItemToUpdate+")";
var data = JSON.stringify({
    MultiLookupColumnName:[
        {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(1)"}},
        {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(2)"}}
    ]
});
$.ajax({
    method: 'POST',
    url: url,
    contentType: 'application/json',
    headers: {
      "X-HTTP-Method" : "MERGE",
      "If-Match" : "*"
    },
    data: data
});
```

2013

```
var listGuid = "id-of-list-to-update"; // use list GUID here
var lookupGuid = "id-of-lookup-list"; // use lookup list GUID here
var idOfItemToUpdate = 1;
var url = "/server/site/_api/lists('"+ listGuid + "')/items("+ idOfItemToUpdate + ")";
var data = JSON.stringify({
    MultiLookupColumnName:[
        {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(1)"}},
        {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(2)"}}
    ]
```

```
});
$.ajax({
    method: 'POST',
    url: url,
    contentType: 'application/json',
    headers: {
      "X-HTTP-Method" : "MERGE",
      "If-Match" : "*"
    },
    data: data
});
```

Usando XMLHttpRequest

2010

```
var listName = "YourListName";
var lookupList = "LookupListName";
var idOfItemToUpdate = 1;
var url = "/server/site/_vti_bin/ListData.svc/YourListName("+idOfItemToUpdate+")";
var data = JSON.stringify({
    MultiLookupColumnName:[
       {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(1)"}},
        {__metadata:{uri:"http://yoursiteurl/_vti_bin/ListData.svc/"+lookupList+"(2)"}}
    ]
});
var xhr = new XMLHttpRequest();
xhr.open("POST",url,true);
xhr.setRequestHeader("X-HTTP-Method", "MERGE");
xhr.setRequestHeader("If-Match", "*");
xhr.setRequestHeader("Content-Type","application/json");
xhr.send(data);
```

2013

```
var listGuid = "id-of-list-to-update";
var lookupGuid = "id-of-lookup-list";
var idOfItemToUpdate = 1;
var url = "/server/site/_api/lists('"+ listGuid + "')/items("+ idOfItemToUpdate + ")";
var data = JSON.stringify({
    MultiLookupColumnName:[
        {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(1)"}},
        {__metadata:{uri:"http://yoursiteurl/_api/lists('" + lookupGuid + "')/items(2)"}}
    ]
});
var xhr = new XMLHttpRequest();
xhr.open("POST",url,true);
xhr.setRequestHeader("X-HTTP-Method", "MERGE");
xhr.setRequestHeader("If-Match", "*");
xhr.setRequestHeader("Content-Type","application/json");
xhr.send(data);
```

Elementos de la lista de paginación devueltos de una consulta

Para simular la paginación usando REST puedes hacer lo siguiente:

1. Use el parámetro `$skip=n` para omitir las primeras `n` entradas de acuerdo con el parámetro

`$orderby`

2. Use el parámetro `$top=n` para devolver las entradas top `n` acuerdo con los parámetros `$orderby` y `$skip` .

```
var endpointUrl = "/_api/lists('guid')/items"; // SP2010: "/_vti_bin/ListData.svc/ListName";
$.getJSON(
    endpointUrl + "?$orderby=Id&$top=1000",
    function(data){
        processData(data); // you can do something with the results here
        var count = data.d.results.length;
        getNextBatch(count, processData, onComplete); // fetch next page
    }
);

function getNextBatch(totalSoFar, processResults, onCompleteCallback){
    $.getJSON(
        endpointUrl + "?$orderby=Id&$skip="+totalSoFar+"&$top=1000",
        function(data){
            var count = data.d.results.length;
            if(count > 0){
                processResults(data); // do something with results
                getNextBatch(totalSoFar+count, callback); // fetch next page
            }else{
                onCompleteCallback();
            }
        }
    );
}
```

Recuperar una ID del elemento recién creado en la lista de SharePoint

Este ejemplo muestra cómo recuperar una ID de un elemento recién creado mediante la API REST de SharePoint.

Nota :

listName : esta variable contiene el nombre de tu lista.

newItemBody : este será el cuerpo de su solicitud para agregar un nuevo elemento a la lista.

por ejemplo, var newItemBody = {__metadata: {'type': 'SP.Data.MyListNameItem'}, Título: 'Some value value'};

```
function CreateListItemWithDetails(listName, newItemBody) {

    var item = newItemBody;
    return $.ajax({
        url: _spPageContextInfo.siteAbsoluteUrl + "/_api/web/lists/getbytitle('" + listName +
"')/items",
        type: "POST",
        contentType: "application/json;odata=verbose",
        data: JSON.stringify(item),
        headers: {
            "Accept": "application/json;odata=verbose",
            "X-RequestDigest": $("#__REQUESTDIGEST").val(),
```

```
            "content-Type": "application/json;odata=verbose"
        }

    });
}

CreateListItemWithDetails(listName, newItemBody)
    .then(function(data){
        //success callback
        var NewlyCreatedItemId = data.d.ID;
    }, function(data){
        //failure callback
    });
```

Cómo realizar operaciones CRUD mediante la interfaz REST de SharePoint 2010

Crear

Para realizar una operación Crear a través de REST, debe realizar las siguientes acciones:

Cree una solicitud HTTP utilizando el verbo `POST` . Utilice la URL de servicio de la lista a la que desea agregar una entidad como objetivo para el POST. Establezca el tipo de contenido en `application/json` . Serialice los objetos JSON que representan sus nuevos elementos de lista como una cadena y agregue este valor al ejemplo de JavaScript del cuerpo de la solicitud:

```
function createListItem(webUrl,listName, itemProperties, success, failure) {

    $.ajax({
        url: webUrl + "/_vti_bin/listdata.svc/" + listName,
        type: "POST",
        processData: false,
        contentType: "application/json;odata=verbose",
        data: JSON.stringify(itemProperties),
        headers: {
            "Accept": "application/json;odata=verbose"
        },
        success: function (data) {
            success(data.d);
        },
        error: function (data) {
            failure(data.responseJSON.error);
        }
    });
}
```

Uso

```
var taskProperties = {
    'TaskName': 'Order Approval',
    'AssignedToId': 12
};

createListItem('https://contoso.sharepoint.com/project/','Tasks',taskProperties,function(task){

    console.log('Task' + task.TaskName + ' has been created');
```

```
  },
  function(error){
    console.log(JSON.stringify(error));
  }
);
```

Leer

Para realizar una operación de lectura a través de REST, debe realizar las siguientes acciones:

Crea una solicitud HTTP usando el verbo `GET` . Utilice la URL de servicio del elemento de la lista a la que desea agregar una entidad como objetivo para el GET. Establezca el tipo de contenido en `application/json` . Ejemplo de JavaScript:

```
function getListItemById(webUrl,listName, itemId, success, failure) {
    var url = webUrl + "/_vti_bin/listdata.svc/" + listName + "(" + itemId + ")";
    $.ajax({
        url: url,
        method: "GET",
        headers: { "Accept": "application/json; odata=verbose" },
        success: function (data) {
            success(data.d);
        },
        error: function (data) {
            failure(data.responseJSON.error);
        }
    });
}
```

Uso

```
getListItemById('https://contoso.sharepoint.com/project/','Tasks',2,function(taskItem){
    console.log(taskItem.TaskName);
  },
  function(error){
    console.log(JSON.stringify(error));
  }
);
```

Actualizar

Para actualizar una entidad existente, debe realizar las siguientes acciones:

Cree una solicitud HTTP utilizando el verbo `POST` . Agregue un encabezado `X-HTTP-Method` con un valor de `MERGE` . Use la URL de servicio del elemento de la lista que desea actualizar como objetivo para la POST. Agregue un encabezado `If-Match` con un valor del ETag original de la entidad, o *. Ejemplo de JavaScript:

```
function updateListItem(webUrl,listName,itemId,itemProperties,success, failure)
{
   getListItemById(webUrl,listName,itemId,function(item){

      $.ajax({
         type: 'POST',
```

```
        url: item.__metadata.uri,
        contentType: 'application/json',
        processData: false,
        headers: {
               "Accept": "application/json;odata=verbose",
               "X-HTTP-Method": "MERGE",
               "If-Match": item.__metadata.etag
        },
        data: Sys.Serialization.JavaScriptSerializer.serialize(itemProperties),
        success: function (data) {
               success(data);
        },
        error: function (data) {
               failure(data);
        }
      });

   },
   function(error){
       failure(error);
   });

}
```

Uso

```
var taskProperties = {
    'TaskName': 'Approval',
    'AssignedToId': 12
};


updateListItem('https://contoso.sharepoint.com/project/','Tasks',2,taskProperties,function(item){

    console.log('Task has been updated');
  },
  function(error){
    console.log(JSON.stringify(error));
  }
);
```

Borrar

Para eliminar una entidad, debe realizar las siguientes acciones:

Cree una solicitud HTTP utilizando el verbo `POST` . Agregue un encabezado `X-HTTP-Method` con un valor de `DELETE` . Utilice la URL de servicio del elemento de la lista que desea actualizar como objetivo para la POST. Agregue un encabezado `If-Match` con un valor de la etiqueta original de la entidad. Ejemplo de JavaScript:

```
function deleteListItem(webUrl, listName, itemId, success, failure) {
    getListItemById(webUrl,listName,itemId,function(item){
        $.ajax({
            url: item.__metadata.uri,
            type: "POST",
            headers: {
                "Accept": "application/json;odata=verbose",
```

```
                "X-Http-Method": "DELETE",
                "If-Match": item.__metadata.etag
            },
            success: function (data) {
                success();
            },
            error: function (data) {
                failure(data.responseJSON.error);
            }
        });
    },
   function (error) {
       failure(error);
   });
}
```

Uso

```
deleteListItem('https://contoso.sharepoint.com/project/','Tasks',3,function(){
    console.log('Task has been deleted');
  },
  function(error){
    console.log(JSON.stringify(error));
  }
);
```

Lea Servicios de descanso en línea: https://riptutorial.com/es/sharepoint/topic/3045/servicios-de-descanso

Capítulo 7: Trabajar con cuadros de diálogo modales con JavaScript

Sintaxis

- var options = SP.UI. $ create_DialogOptions ();
- var modalDialog = SP.UI.ModalDialog.showModalDialog (opciones);

Parámetros

opciones de propiedad	Descripción
título	Una cadena que contiene el título del diálogo.
url	Una cadena que contiene la URL de la página que aparece en el cuadro de diálogo. Se debe especificar **url** o **html** . **url** tiene prioridad sobre **html** .
html	Un elemento HTML para mostrar dentro del diálogo.
X	El desplazamiento x del diálogo como un valor entero.
y	El desplazamiento de y del diálogo como un valor entero.
anchura	El ancho del diálogo como un valor entero. Si no se especifica y el **tamaño automático** es **falso,** el ancho se establece en 768 px
altura	La altura del diálogo como un valor entero. Si no se especifica y el **tamaño automático** es **falso,** la altura se establece en 576px
permitir maximizar	Un valor booleano que especifica si se debe mostrar el botón **Maximizar** .
showMaximized	Un valor booleano que especifica si el cuadro de diálogo se abre al máximo.
showCerrar	Un valor booleano que especifica si el botón **Cerrar** aparece en el cuadro de diálogo.
tamaño automático	Un valor booleano que especifica si la plataforma de diálogo controla el tamaño del diálogo automáticamente.
dialogReturnValueCallback	Un puntero a función que especifica la función de devolución

opciones de propiedad	Descripción
	de devolución de llamada. La función toma dos parámetros: un *dialogResult* de tipo SP.UI.DialogResult Enumeration, y un objeto *returnValue* que contiene los datos devueltos por el diálogo.
args	Un objeto que contiene datos que se pasan al diálogo.

Observaciones

El espacio de nombres `SP.UI.ModalDialog` se introdujo en el Modelo de objetos de JavaScript con SharePoint 2010, y está disponible en versiones posteriores de SharePoint 2013, Office365 y 2016.

Materiales de referencia adicionales:

- Referencia de MSDN para SP.UI.ModalDialog.showModalDialog (opciones)
- Referencia de MSDN para la enumeración SP.UI.DialogResult

Examples

Realizar una acción cuando un cuadro de diálogo está cerrado

```
SP.SOD.executeOrDelayUntilScriptLoaded(showDialog,"sp.js");

function showDialog(){
    var options = SP.UI.$create_DialogOptions();
    options.url = "/mySite/lists/myList/NewForm.aspx";
    options.dialogReturnValueCallback = myCallBackFunction;
    SP.UI.ModalDialog.showModalDialog(options);
    function myCallBackFunction(result,data){
        switch(result){
            case SP.UI.DialogResult.invalid:
                alert("The dialog result was invalid");
                break;
            case SP.UI.DialogResult.cancel:
                alert("You clicked cancel or close");
                break;
            case SP.UI.DialogResult.OK:
                alert("You clicked OK, creating an item in the list.");
                break;
        }
    }
}
```

Mostrar una página existente en un diálogo

```
SP.SOD.executeOrDelayUntilScriptLoaded(showDialog,"sp.js");

function showDialog(){
```

```
    SP.UI.ModalDialog.showModalDialog(
        { url: "/org/it/web/wik/Lists/ExampleCode/DispForm.aspx?ID=6" }
    );
}
```

Mostrar un diálogo personalizado

```
SP.SOD.executeOrDelayUntilScriptLoaded(showDialog,"sp.js");

function showDialog(){
    var dialogOptions = SP.UI.$create_DialogOptions();
    dialogOptions.title = "Your Title Here!";
    var dummyElement = document.createElement("div");
    dummyElement.style.textAlign = "center";
    dummyElement.appendChild(document.createElement("br"));
    dummyElement.appendChild(document.createTextNode("Some beautifully crafted text."));
    dummyElement.appendChild(document.createElement("br"));
    dialogOptions.html = dummyElement;
    SP.UI.ModalDialog.showModalDialog(dialogOptions);
}
```

Lea Trabajar con cuadros de diálogo modales con JavaScript en línea: https://riptutorial.com/es/sharepoint/topic/6868/trabajar-con-cuadros-de-dialogo-modales-con-javascript

Capítulo 8: Trabajar con JavaScript Client Object Model (JSOM)

Observaciones

Fondo

El modelo de objetos de JavaScript se introdujo en SharePoint 2010. Expone en el lado del cliente muchos de los objetos a los que anteriormente solo se podía acceder a través del código del lado del servidor o a través de servicios web dedicados.

Incrustar JavaScript en las páginas de SharePoint

En SharePoint 2013 puede poner su JavaScript en un elemento web de Script Editor.

En SharePoint 2010, puede usar la propiedad "enlace de contenido" de un elemento web del Editor de contenido para vincularlo a un archivo HTML que contiene su script incrustado.

Referencia de objeto

Los constructores, los métodos y las propiedades de todos los objetos encontrados en el espacio de nombres del `SP` se documentan en la referencia del modelo de objeto del cliente de SharePoint 2013 aquí .

La referencia del modelo de objeto cliente de SharePoint 2010 JavaScript está disponible aquí .

Patrón de programación asíncrono de JSOM

Cuando se usa el modelo de objeto cliente de JavaScript, el código generalmente toma el siguiente patrón:

1. Obtener un objeto `ClientContext` .
2. Utilice el objeto `ClientContext` para recuperar objetos que representan entidades en el modelo de objetos de SharePoint, como listas, carpetas, vistas.
3. Poner en cola las instrucciones a realizar contra los objetos. Estas instrucciones no se transmiten al servidor todavía.
4. Utilice la función de `load` para indicar al `ClientContext` qué información desea recibir del servidor.
5. Invoque la función `ClientContext` objeto `executeQueryAsync` para enviar las instrucciones en cola al servidor, pasando dos funciones de devolución de llamada para que se ejecuten en caso de éxito o fracaso.
6. En la función de devolución de llamada, trabaje con los resultados devueltos por el servidor.

Alternativas

Las alternativas del lado del cliente al JSOM incluyen servicios web de SharePoint, puntos finales

REST y el modelo de objeto de cliente .NET .

Examples

Obtención de tipos de contenido de la biblioteca utilizando el nombre de la biblioteca

```
function getContentTypes(site_url,name_of_the_library){
    var ctx = new SP.ClientContext(site_url);
    var web = ctx.get_web();
    list = web.get_lists().getByTitle(name_of_the_library);

    // You can include any property of the SP.ContentType object (sp.js), for this example we
are just getting the name
    ctx.load(list,'ContentTypes.Include(Name)');
    ctx.executeQueryAsync(onQuerySucceeded, onQueryFailed);
}

function onQuerySucceeded(sender, args) {
    // var list is the one that we used in function "getContentTypes"
    var contentTypesEnumerator = (list.get_contentTypes()).getEnumerator();

    while (contentTypesEnumerator.moveNext()) {
        var contentType = contentTypesEnumerator.get_current();
        alert(contentType.get_name());
    }
}

function onQueryFailed(sender, args) {
    alert('Request failed. ' + args.get_message() + '\n' + args.get_stackTrace());
}
```

Eliminar un elemento en una lista

```
SP.SOD.executeOrDelayUntilScriptLoaded( function(){ deleteItem(1); }, "sp.js");

function deleteItem(id){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var item = list.getItemById(id);
    item.deleteObject();
    clientContext.executeQueryAsync(function(){
        alert("Item #"+id+" deleted successfully!");
    },function(sender,args){alert(args.get_message());});
}
```

Creación de elementos o carpetas

Creación de elementos de lista

```
SP.SOD.executeOrDelayUntilScriptLoaded(createItem,"sp.js");
```

```
function createItem(){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var newItem = list.addItem();
    newItem.set_item("Title","Example Title");
    newItem.update();
    clientContext.load(newItem); // only needed to retrieve info from newly created item
    clientContext.executeQueryAsync(function(){
        var itemId = newItem.get_item("ID");
        alert("Item #"+itemId+" Created Successfully!");
    },function(sender,args){
        alert(args.get_message());
    });
}
```

El ejemplo anterior demuestra que un elemento de lista se crea realizando lo siguiente:

1. Llame al método `addItem` de un objeto de lista para obtener un objeto de elemento
2. Llame al método `set_item` en el objeto de elemento de lista resultante para establecer cada valor de campo como desee
3. Llame al método de `update` en el objeto de elemento de lista para indicar que los cambios se deben confirmar
4. Llame al método `executeQueryAsync` del objeto de contexto de cliente para ejecutar las instrucciones en cola

Tenga en cuenta que **no** necesita pasar el nuevo objeto de elemento al método de `load` del contexto del cliente para crear el elemento. Ese paso solo es necesario si desea recuperar cualquiera de los valores de campo del elemento desde el servidor.

Creación de carpetas

Crear una carpeta es similar a agregar un elemento a una lista. La diferencia es que primero se debe crear un objeto `ListItemCreationInformation` y establecer su propiedad `underlyingObjectType` en `SP.FileSystemObjectType.folder` , y su propiedad `leafName` en el nombre deseado de la nueva carpeta.

El objeto se pasa luego como un parámetro en el método `addItem` en la biblioteca para crear la carpeta.

```
// ...
var itemCreateInfo = new SP.ListItemCreationInformation();
itemCreateInfo.set_underlyingObjectType(SP.FileSystemObjectType.folder);
itemCreateInfo.set_leafName(folderName);
var newItem = list.addItem(itemCreateInfo);
// ...
```

Para confirmar el cambio, invocar el `executeQueryAsync` método de la `ClientContext` objeto a través del cual se accede a la biblioteca.

El ejemplo completo a continuación crea una carpeta con un nombre basado en la marca de tiempo actual, y luego abre esa carpeta en un diálogo modal.

```
SP.SOD.executeOrDelayUntilScriptLoaded(createFolder,"sp.js");

function createFolder(){
    var now = new Date();
    var timeStamp = now.getYear() + "-" + (now.getMonth()+1) + "-" + now.getDate()
        + "T" + now.getHours()+"_"+now.getMinutes()+"
"+now.getSeconds()+"_"+now.getMilliseconds();
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("Library Title");
    var itemCreateInfo = new SP.ListItemCreationInformation();
    itemCreateInfo.set_underlyingObjectType(SP.FileSystemObjectType.folder);
    itemCreateInfo.set_leafName(timeStamp);
    var newItem = list.addItem(itemCreateInfo);
    newItem.update();
    clientContext.load(newItem);
    var rootFolder = list.get_rootFolder(); // Note: use a list's root folder to determine its
server relative URL
    clientContext.load(rootFolder);
    clientContext.executeQueryAsync(function(){
        var itemId = newItem.get_item("ID");
        var name = newItem.get_item("FileLeafRef");
        SP.UI.ModalDialog.showModalDialog(
            {
                title: "Folder \""+name+"\" (#"+itemId+") Created Successfully!",
                url: rootFolder.get_serverRelativeUrl() + "/" + name
            }
        );
    },function(sender,args){alert(args.get_message());});
}
```

Obtener información actual del usuario

```
SP.SOD.executeOrDelayUntilScriptLoaded(showUserInfo,"sp.js");

function showUserInfo(){
    var clientContext = new SP.ClientContext();
    var user = clientContext.get_web().get_currentUser();
    clientContext.load(user);
    clientContext.executeQueryAsync(function(){
            var details = "ID: "+user.get_id()+"\n"+
                "Title: "+user.get_title()+"\n"+
                "Login: "+user.get_loginName()+"\n"+
                "Email: "+user.get_email();
            alert(details);
        },function(sender,args){alert(args.get_message());})
}
```

Obtener un elemento de lista por ID

```
SP.SOD.executeOrDelayUntilScriptLoaded(myFunction,"sp.js");

function myFunction(){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var item = list.getItemById(1); // get item with ID == 1
    clientContext.load(item);
    clientContext.executeQueryAsync(
```

```
        function(){ // onSuccess
            var title = item.get_item("Title");
            alert(title);
        },
        function(sender,args){ // onError
            alert(args.get_message());
        }
    );
}
```

Obtener elementos de lista por consulta CAML

Ejemplo básico

Usar la `set_viewXml` método del objeto SP.CamlQuery para especificar una consulta CAML para recuperar elementos.

```
SP.SOD.executeOrDelayUntilScriptLoaded(showListItems,"core.js");

function showListItems(){
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml(
        "<View><Query>" +
            "<Where>" +
                "<Eq><FieldRef Name=\"Title\"/><Value Type=\"Text\">Value</Value></Eq>" +
            "</Where>" +
            "<OrderBy><FieldRef Name=\"Modified\" Ascending=\"FALSE\"/></OrderBy>" +
        "</Query>"+
        //"<RowLimit>5000</RowLimit>" +
        "</View>");
    var items = list.getItems(camlQuery);
    clientContext.load(items);
    clientContext.executeQueryAsync(function(){
        var itemArray = [];
        var itemEnumerator = items.getEnumerator();
        while(itemEnumerator.moveNext()){
            var item = itemEnumerator.get_current();
            var id = item.get_item("ID");
            var title = item.get_item("Title");
            itemArray.push(id + ": " + title);
        }
        alert("ID: Title\n"+itemArray.join("\n"));
    },function(sender,args){alert(args.get_message());});
}
```

Paginar los resultados de una consulta CAML

Puede aprovechar el elemento `RowLimit` en una consulta CAML para recuperar solo un subconjunto de resultados con cada consulta.

Utilice el método `get_listItemCollectionPosition` de una colección de elementos de lista para

recuperar la posición actual, luego use ese valor como el parámetro en el método `set_listItemCollectionPosition` un objeto `set_listItemCollectionPosition` para recuperar el siguiente lote de resultados.

```
SP.SOD.executeOrDelayUntilScriptLoaded(showListItems,"sp.js");

function showListItems(){
    var itemArray = [];
    var clientContext = new SP.ClientContext();
    var list = clientContext.get_web().get_lists().getByTitle("List Title");
    var viewXml =
        "<View><Query>" +
            "<OrderBy><FieldRef Name=\"Modified\" Ascending=\"FALSE\"/></OrderBy>" +
        "</Query>"+
          "<RowLimit>1</RowLimit>" +
        "</View>";
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml(viewXml);
    var items = list.getItems(camlQuery);
    clientContext.load(items);
    clientContext.executeQueryAsync(loadResults,showError);

    function loadResults(){
        var resultsFound = false;
        var itemEnumerator = items.getEnumerator();
        while(itemEnumerator.moveNext()){
            var item = itemEnumerator.get_current();
            var id = item.get_item("ID");
            var title = item.get_item("Title");
            itemArray.push(id + ": " + title);
        }
        var pos = items.get_listItemCollectionPosition();// <- get position
        if(pos !== null){ // <-- position is null when no more results are returned
            if(confirm("Results so far: \nID: Title\n"+itemArray.join("\n"))){
                camlQuery = new SP.CamlQuery();
                camlQuery.set_listItemCollectionPosition(pos);// <- set position for next
batch
                camlQuery.set_viewXml(viewXml);
                items = list.getItems(camlQuery);
                clientContext.load(items);
                clientContext.executeQueryAsync(loadResults,showError);
            }
        }else{
            alert("Total Results: \nID: Title\n"+itemArray.join("\n")); // <- display when no
more results
        }
    }
    function showError(sender,args){
        alert(args.get_message());
    }
}
```

Lea Trabajar con JavaScript Client Object Model (JSOM) en línea: https://riptutorial.com/es/sharepoint/topic/1316/trabajar-con-javascript-client-object-model--jsom-

Capítulo 9: Trabajar con Managed Client Side Model Model (CSOM)

Observaciones

- La mayoría de los ejemplos son de MSDN .
- Para crear una aplicación cliente administrada .NET que use el modelo de objeto cliente, debe establecer referencias a dos DLL de biblioteca de cliente: Microsoft.SharePoint.Client.dll y Microsoft.SharePoint.Client.Runtime.dll. Puede encontrarlo en la carpeta% Archivos de programa% \ Archivos comunes \ Microsoft Shared \ web server extensions \ 16 \ ISAPI o en su servidor de SharePoint.
- o Instale el paquete NuGet de Microsoft.SharePointOnline.CSOM, que funcionará "en prem" así como en SP O365.
- La mayoría de las propiedades son propiedades de valor y antes de acceder a ellas debe llamar explícitamente a clientContext.Load () y clientContext.ExecuteQuery (). Más información aquí: Call Load y ExecuteQuery antes de acceder a las propiedades de Value

Examples

Hola mundo (obteniendo titulo del sitio)

Todas las versiones de SharePoint se basan en Sitios (SPSite (SSOM) o Sitio (CSOM)) y Webs (SPWeb (SSOM) o Web (CSOM)). Un sitio no se representa en la interfaz de usuario, aunque contiene metadatos y características que se aplican a sus hijos. Una web es el bloque de construcción básico que representa una IU para el usuario que accede al sitio. Todos los sitios tienen una raíz web que contiene información y / o metadatos como bibliotecas de documentos. Este ejemplo muestra una llamada básica para obtener la web ubicada en el servidor `MyServer` en los `sites` ruta virtual.

```
using System;
using Microsoft.SharePoint.Client;

namespace Microsoft.SDK.SharePointServices.Samples
{
    class RetrieveWebsite
    {
        static void Main()
        {
            // This is the URL of the target web we are interested in.
            string siteUrl = "http://MyServer/sites/MySiteCollection";
            // The client context is allows us to queue up requests for the server
            // Note that the context can only ask questions about the site it is created for
            using (ClientContext clientContext = new ClientContext(siteUrl))
            {
                // To make it easier to read the code, pull the target web
                // context off of the client context and store in a variable
                Web oWebsite = clientContext.Web;
```

```
                // Tell the client context we want to request information about the
                // Web from the server
                clientContext.Load(oWebsite);
                // After we are done creating the batch of information we need from the sever,
                // request the data from SharePoint
                clientContext.ExecuteQuery();
                // Print the results of the query
                Console.WriteLine("Title: {0} Description: {1}", oWebsite.Title,
oWebsite.Description);
            }
        }
    }
}
```

Web. Recuperando las propiedades de un sitio web

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
clientContext.Load(oWebsite);
clientContext.ExecuteQuery();
Console.WriteLine("Title: {0} Description: {1}", oWebsite.Title, oWebsite.Description);
```

Web. Recuperar solo las propiedades especificadas de un sitio web

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
clientContext.Load(
                oWebsite,
                website => website.Title,
                website => website.Created);
clientContext.ExecuteQuery();
Console.WriteLine("Title: {0} Created: {1}", oWebsite.Title, oWebsite.Created);
```

Web. Actualización del título y descripción de un sitio web.

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = context.Web;
oWebsite.Title = "Updated Web Site";
oWebsite.Description = "This is an updated Web site.";
oWebsite.Update();
clientContext.ExecuteQuery();
```

Web. Creando un sitio web

```
string siteUrl = "http://MyServer/sites/MySiteCollection";
string blogDescription = "A new blog Web site.";
int blogLanguage = 1033;
string blogTitle = "Blog Web Site";
string blogUrl = "blogwebsite";
bool blogPermissions = false;
string webTemplate = "BLOG#0";

ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
```

```
WebCreationInformation webCreateInfo = new WebCreationInformation();
webCreateInfo.Description = blogDescription;
webCreateInfo.Language = blogLanguage;
webCreateInfo.Title = blogTitle;
webCreateInfo.Url = blogUrl;
webCreateInfo.UseSamePermissionsAsParentSite = blogPermissions;
webCreateInfo.WebTemplate = webTemplate;

Web oNewWebsite = oWebsite.Webs.Add(webCreateInfo);

clientContext.Load(
    oNewWebsite,
    website => website.ServerRelativeUrl,
    website => website.Created);

clientContext.ExecuteQuery();

Console.WriteLine("Server-relative Url: {0} Created: {1}", oNewWebsite.ServerRelativeUrl,
oNewWebsite.Created);
```

Lista. Recuperar todas las propiedades de todas las listas en un sitio web

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

clientContext.Load(collList);

clientContext.ExecuteQuery();

foreach (List oList in collList)
{
    Console.WriteLine("Title: {0} Created: {1}", oList.Title, oList.Created.ToString());
}
```

Lista. Recuperando solo propiedades especificadas de listas

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

clientContext.Load(
    collList,
    lists => lists.Include(
        list => list.Title,
        list => list.Id));

clientContext.ExecuteQuery();

foreach (List oList in collList)
{
    Console.WriteLine("Title: {0} ID: {1}", oList.Title, oList.Id.ToString("D"));
}
```

Lista. Almacenar listas recuperadas en una colección

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

IEnumerable<List> resultCollection = clientContext.LoadQuery(
    collList.Include(
        list=>list.Title,
        list=>list.Id));

clientContext.ExecuteQuery();

foreach (List oList in resultCollection)
{
    Console.WriteLine("Title: {0} ID: {1}", oList.Title, oList.Id.ToString("D"));
}
```

Lista. Recuperar campos de lista de un sitio web

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;
ListCollection collList = oWebsite.Lists;

IEnumerable<SP.List> listInfo = clientContext.LoadQuery(
    collList.Include(
        list => list.Title,
        list => list.Fields.Include(
            field => field.Title,
            field => field.InternalName)));

 clientContext.ExecuteQuery();

foreach (SP.List oList in listInfo)
{
    FieldCollection collField = oList.Fields;

    foreach (SP.Field oField in collField)
    {
        Regex regEx = new Regex("name", RegexOptions.IgnoreCase);

        if (regEx.IsMatch(oField.InternalName))
        {
            Console.WriteLine("List: {0} \n\t Field Title: {1} \n\t Field Internal Name: {2}",

                oList.Title, oField.Title, oField.InternalName);
        }
    }
}
```

Lista. Creando y actualizando una lista

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;

ListCreationInformation listCreationInfo = new ListCreationInformation();
listCreationInfo.Title = "My Announcements List";
listCreationInfo.TemplateType = (int)ListTemplateType.Announcements;
```

```
List oList = oWebsite.Lists.Add(listCreationInfo);

clientContext.ExecuteQuery();
```

Lista. Añadiendo un campo a una lista

```
ClientContext clientContext = new ClientContext(siteUrl);

SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");

SP.Field oField = oList.Fields.AddFieldAsXml("<Field DisplayName='MyField' Type='Number' />",

    true, AddFieldOptions.DefaultValue);

SP.FieldNumber fieldNumber = clientContext.CastTo<FieldNumber>(oField);
fieldNumber.MaximumValue = 100;
fieldNumber.MinimumValue = 35;

fieldNumber.Update();

clientContext.ExecuteQuery();
```

Lista. Borrando una lista

```
ClientContext clientContext = new ClientContext(siteUrl);
Web oWebsite = clientContext.Web;

List oList = oWebsite.Lists.GetByTitle("My Announcements List");

oList.DeleteObject();

clientContext.ExecuteQuery();
```

Ít. Recuperar elementos de una lista

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");

CamlQuery camlQuery = new CamlQuery();
camlQuery.ViewXml = "<View><Query><Where><Geq><FieldRef Name='ID'/>" +
    "<Value Type='Number'>10</Value></Geq></Where></Query><RowLimit>100</RowLimit></View>";
ListItemCollection collListItem = oList.GetItems(camlQuery);

clientContext.Load(collListItem);

clientContext.ExecuteQuery();

foreach (ListItem oListItem in collListItem)
{
    Console.WriteLine("ID: {0} \nTitle: {1} \nBody: {2}", oListItem.Id, oListItem["Title"],
oListItem["Body"]);
}
```

Ít. Recuperar elementos (utilizando el método Include)

Este ejemplo muestra cómo recuperar elementos del servidor y obtener propiedades más detalladas de cada elemento de la lista. De forma predeterminada, el servidor solo devolverá la cantidad mínima de datos para representar el objeto. Es responsabilidad de la persona que llama solicitar información adicional del servidor.

```
ClientContext clientContext = new ClientContext(siteUrl);
List oList = clientContext.Web.Lists.GetByTitle("Announcements");

CamlQuery camlQuery = new CamlQuery();
camlQuery.ViewXml = "<View><RowLimit>100</RowLimit></View>";

ListItemCollection collListItem = oList.GetItems(camlQuery);

// The first line of this request indicates the list item collection to load from the server
// The second line uses a lambda to request that from the server
// also include additional properties in the response
// The third though fifth lines are the properties being requested from the server
clientContext.Load(collListItem,
     items => items.Include(
        item => item.Id,
        item => item.DisplayName,
        item => item.HasUniqueRoleAssignments));

clientContext.ExecuteQuery();

foreach (ListItem oListItem in collListItem)
{
    Console.WriteLine("ID: {0} \nDisplay name: {1} \nUnique role assignments: {2}",
        oListItem.Id, oListItem.DisplayName, oListItem.HasUniqueRoleAssignments);
}
```

Ít. Recuperar campos específicos de un número específico de elementos

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");

CamlQuery camlQuery = new CamlQuery();
ListItemCollection collListItem = oList.GetItems(camlQuery);

clientContext.Load(
    collListItem,
    items => items.Take(5).Include(
    item => item["Title"],
    item => item["Body"]));

clientContext.ExecuteQuery();

foreach (ListItem oListItem in collListItem)
{
    Console.WriteLine("Title: {0} \nBody: {1}\n", oListItem["Title"], oListItem["Body"]);
}
```

Ít. Recuperar elementos de todas las listas en un sitio web

```
ClientContext clientContext = new ClientContext(siteUrl);
ListCollection collList = clientContext.Web.Lists;
```

```
clientContext.Load(
    collList,
    lists => lists.Where(
        list => list.Hidden == false).Include(
        list => list.Title,
        list => list.Items.Take(10)));

clientContext.ExecuteQuery();

foreach (SP.List oList in clientContext.Web.Lists)
{
    string listTitle = oList.Title;
    int itemCount = oList.Items.Count;

    Console.WriteLine("List {0} returned with {1} items", listTitle, itemCount);
}
```

Ít. Recuperar elementos utilizando la posición de colección de elementos de lista

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");

ListItemCollectionPosition itemPosition = null;

while (true)
{
    CamlQuery camlQuery = new CamlQuery();

    camlQuery.ListItemCollectionPosition = itemPosition;

    camlQuery.ViewXml = "<View><ViewFields><FieldRef Name='ID'/>" +
        "<FieldRef Name='Title'/><FieldRef Name='Body'/>" +
        "</ViewFields><RowLimit>5</RowLimit></View>";

    ListItemCollection collListItem = oList.GetItems(camlQuery);

    clientContext.Load(collListItem);

    clientContext.ExecuteQuery();

    itemPosition = collListItem.ListItemCollectionPosition;

    foreach (ListItem oListItem in collListItem)
    {
        Console.WriteLine("Title: {0}: \nBody: {1}", oListItem["Title"], oListItem["Body"]);
    }

    if (itemPosition == null)
    {
        break;
    }

    Console.WriteLine("\n" + itemPosition.PagingInfo + "\n");
}
```

Ít. Crear un elemento de lista

Al crear un nuevo elemento de la lista, sus campos se pueden configurar utilizando una sintaxis similar a las matrices de cadenas. Tenga en cuenta que estos campos no se crean sobre la marcha y están definidos por el esquema de la lista. Estos campos (o columnas) deben existir en el servidor, de lo contrario la creación fallará. Todos los elementos de la lista tendrán el campo Título. Algunas listas pueden tener campos obligatorios que deben completarse antes de que el elemento se publique en la lista.

En este ejemplo, la lista está utilizando la plantilla de Anuncios. Además del campo de título, la lista incluye el campo Cuerpo que mostrará el contenido del anuncio en la lista.

```
ClientContext clientContext = new ClientContext(siteUrl);
List oList = clientContext.Web.Lists.GetByTitle("Announcements");

ListItemCreationInformation itemCreateInfo = new ListItemCreationInformation();
ListItem oListItem = oList.AddItem(itemCreateInfo);
oListItem["Title"] = "My New Item!";
oListItem["Body"] = "Hello World!";

oListItem.Update();

clientContext.ExecuteQuery();
```

Ít. Actualización de un elemento de la lista

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");
ListItem oListItem = oList.Items.GetById(3);

oListItem["Title"] = "My Updated Title.";

oListItem.Update();

clientContext.ExecuteQuery();
```

Ít. Eliminar un elemento de la lista

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("Announcements");
ListItem oListItem = oList.GetItemById(2);

oListItem.DeleteObject();

clientContext.ExecuteQuery();
```

Los grupos Recuperar todos los usuarios de un grupo de SharePoint

```
ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
GroupCollection collGroup = clientContext.Web.SiteGroups;
Group oGroup = collGroup.GetById(7);
UserCollection collUser = oGroup.Users;

clientContext.Load(collUser);
```

```
clientContext.ExecuteQuery();

foreach (User oUser in collUser)
{
    Console.WriteLine("User: {0}  ID: {1} Email: {2} Login Name: {3}",
        oUser.Title, oUser.Id, oUser.Email, oUser.LoginName);
}
```

Los grupos Recuperando propiedades específicas de los usuarios

```
ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
GroupCollection collGroup = clientContext.Web.SiteGroups;
Group oGroup = collGroup.GetById(7);
UserCollection collUser = oGroup.Users;

clientContext.Load(collUser,
    users => users.Include(
        user => user.Title,
        user => user.LoginName,
        user => user.Email));

clientContext.ExecuteQuery();

foreach (User oUser in collUser)
{
    Console.WriteLine("User: {0} Login name: {1} Email: {2}",
        oUser.Title, oUser.LoginName, oUser.Email);
}
```

Los grupos Recuperar todos los usuarios en todos los grupos de una colección de sitios

```
ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
GroupCollection collGroup = clientContext.Web.SiteGroups;

clientContext.Load(collGroup);

clientContext.Load(collGroup,
    groups => groups.Include(
        group => group.Users));

clientContext.ExecuteQuery();

foreach (Group oGroup in collGroup)
{
    UserCollection collUser = oGroup.Users;

    foreach (User oUser in collUser)
    {
        Console.WriteLine("Group ID: {0} Group Title: {1} User: {2} Login Name: {3}",
            oGroup.Id, oGroup.Title, oUser.Title, oUser.LoginName);
    }
}
```

Los grupos Agregar un usuario a un grupo de SharePoint

```
ClientContext clientContext = new ClientContext("http://MyServer/sites/MySiteCollection ");
GroupCollection collGroup = clientContext.Web.SiteGroups;
Group oGroup = collGroup.GetById(6);

UserCreationInformation userCreationInfo = new UserCreationInformation();
userCreationInfo.Email = "alias@somewhere.com";
userCreationInfo.LoginName = @"DOMAIN\alias";
userCreationInfo.Title = "John";

User oUser = oGroup.Users.Add(userCreationInfo);

clientContext.ExecuteQuery();
```

Roles Crear una definición de rol

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");

Web oWebsite = clientContext.Web;

BasePermissions permissions = new BasePermissions();
permissions.Set(PermissionKind.CreateAlerts);
permissions.Set(PermissionKind.ManageAlerts);

RoleDefinitionCreationInformation roleCreationInfo = new RoleDefinitionCreationInformation();

roleCreationInfo.BasePermissions = permissions;
roleCreationInfo.Description = "A new role with create and manage alerts permission";
roleCreationInfo.Name = "Create and Manage Alerts";
roleCreationInfo.Order = 4;

RoleDefinition oRoleDefinition = oWebsite.RoleDefinitions.Add(roleCreationInfo);

clientContext.ExecuteQuery();

Console.WriteLine("{0} role created.", oRoleDefinition.Name);
```

Roles Asignar un usuario a un rol en un sitio web

```
ClientContext oClientContext = new
ClientContext("http://MyServer/sites/MySiteCollection/MyWebSite");
Web oWebsite = clientContext.Web;

Principal oUser = oWebsite.SiteUsers.GetByLoginName(@"DOMAIN\alias");

RoleDefinition oRoleDefinition = oWebsite.RoleDefinitions.GetByName("Create and Manage
Alerts");
RoleDefinitionBindingCollection collRoleDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);
collRoleDefinitionBinding.Add(oRoleDefinition);

RoleAssignment oRoleAssignment = oWebsite.RoleAssignments.Add(oUser,
collRoleDefinitionBinding);

clientContext.Load(oUser,
    user => user.Title);

clientContext.Load(oRoleDefinition,
```

```
    role => role.Name);

clientContext.ExecuteQuery();

Console.WriteLine("{0} added with {1} role.", oUser.Title, oRoleDefinition.Name);
```

Roles Crear un grupo de SharePoint y agregar el grupo a un rol

```
ClientContext oClientContext = new
ClientContext("http://MyServer/sites/MySiteCollection/MyWebSite");
Web oWebsite = clientContext.Web;

GroupCreationInformation groupCreationInfo = new GroupCreationInformation();
groupCreationInfo.Title = "My New Group";
groupCreationInfo.Description = "Description of new group.";
Group oGroup = oWebsite.SiteGroups.Add(groupCreationInfo);

RoleDefinitionBindingCollection collRoleDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);

RoleDefinition oRoleDefinition = oWebsite.RoleDefinitions.GetByType(RoleType.Contributor);

collRoleDefinitionBinding.Add(oRoleDefinition);

oWebsite.RoleAssignments.Add(oGroup, collRoleDefinitionBinding);

clientContext.Load(oGroup,
    group => group.Title);

clientContext.Load(oRoleDefinition,
    role => role.Name);

clientContext.ExecuteQuery();

Console.WriteLine("{0} created and assigned {1} role.", oGroup.Title, oRoleDefinition.Name);
}
```

Permisos. Rompiendo la herencia de seguridad de una lista.

```
string siteUrl = "http://MyServer/sites/MySiteCollection";
ClientContext oContext = new ClientContext(siteUrl);
SP.List oList = oContext.Web.Lists.GetByTitle("Announcements");

oList.BreakRoleInheritance(true, false);

oContext.ExecuteQuery();
```

Permisos. Rompiendo la herencia de seguridad de un documento y agregando un usuario como lector

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("MyList");

int itemId = 3;
ListItem oListItem = oList.Items.GetById(itemId);
```

```
oListItem.BreakRoleInheritance(false);

User oUser = clientContext.Web.SiteUsers.GetByLoginName(@"DOMAIN\alias");

RoleDefinitionBindingCollection collRoleDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);

collRoleDefinitionBinding.Add(clientContext.Web.RoleDefinitions.GetByType(RoleType.Reader));

oListItem.RoleAssignments.Add(oUser, collRoleDefinitionBinding);

clientContext.ExecuteQuery();
```

Permisos. Rompiendo la herencia de seguridad de un documento y cambiando los permisos de un usuario

```
ClientContext clientContext = new ClientContext(siteUrl);
SP.List oList = clientContext.Web.Lists.GetByTitle("MyList");

int itemId = 2;
ListItem oListItem = oList.Items.GetById(itemId);

oListItem.BreakRoleInheritance(true);

User oUser = clientContext.Web.SiteUsers.GetByLoginName(@"DOMAIN\alias");
oListItem.RoleAssignments.GetByPrincipal(oUser).DeleteObject();

RoleDefinitionBindingCollection collRollDefinitionBinding = new
RoleDefinitionBindingCollection(clientContext);

collRollDefinitionBinding.Add(clientContext.Web.RoleDefinitions.GetByType(RoleType.Reader));

oListItem.RoleAssignments.Add(oUser, collRollDefinitionBinding);

clientContext.ExecuteQuery();
```

Acción personalizada. Agregar una acción personalizada de usuario para los elementos de la lista

```
string urlWebsite = "http://MyServer/sites/MySiteCollection";
ClientContext clientContext = new ClientContext(urlWebsite);
Web oWebsite = clientContext.Web;

List oList = oWebsite.Lists.GetByTitle("My List");
UserCustomActionCollection collUserCustomAction = oList.UserCustomActions;

UserCustomAction oUserCustomAction = collUserCustomAction.Add();
oUserCustomAction.Location = "EditControlBlock";
oUserCustomAction.Sequence = 100;
oUserCustomAction.Title = "My First User Custom Action";
oUserCustomAction.Url = urlWebsite + @"/_layouts/MyPage.aspx";
oUserCustomAction.Update();

clientContext.Load(oList,
    list => list.UserCustomActions);
```

```
clientContext.ExecuteQuery();
```

Acción personalizada. Modificar una acción personalizada del usuario

```
string urlWebsite = "http://MyServer/sites/SiteCollection";
ClientContext clientContext = new ClientContext(urlWebsite);
Web oWebsite = clientContext.Web;

List oList = oWebsite.Lists.GetByTitle("My List");
UserCustomActionCollection collUserCustomAction = oList.UserCustomActions;

clientContext.Load(collUserCustomAction,
    userCustomActions => userCustomActions.Include(
        userCustomAction => userCustomAction.Title));

clientContext.ExecuteQuery();

foreach (UserCustomAction oUserCustomAction in collUserCustomAction)
{
    if (oUserCustomAction.Title == "My First User Custom Action")
    {
        oUserCustomAction.ImageUrl = "http://MyServer/_layouts/images/MyIcon.png";
        oUserCustomAction.Update();

        clientContext.ExecuteQuery();
    }
}
```

Acción personalizada. Agregar una acción personalizada del usuario a las acciones del sitio de un sitio web

```
string urlWebsite = "http://MyServer/sites/MySiteCollection";
ClientContext clientContext = new ClientContext(urlWebsite);

Web oWebsite = clientContext.Web;
UserCustomActionCollection collUserCustomAction = oWebsite.UserCustomActions;

UserCustomAction oUserCustomAction = collUserCustomAction.Add();

oUserCustomAction.Location = "Microsoft.SharePoint.StandardMenu";
oUserCustomAction.Group = "SiteActions";
oUserCustomAction.Sequence = 101;
oUserCustomAction.Title = "Website User Custom Action";
oUserCustomAction.Description = "This description appears on the Site Actions menu.";
oUserCustomAction.Url = urlWebsite + @"/_layouts/MyPage.aspx";

oUserCustomAction.Update();

clientContext.Load(oWebsite,
    webSite => webSite.UserCustomActions);

clientContext.ExecuteQuery();
```

Parte web Actualización del título de un elemento web

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
File oFile = oClientContext.Web.GetFileByServerRelativeUrl("Default.aspx");
LimitedWebPartManager limitedWebPartManager =
oFile.GetLimitedWebPartManager(PersonalizationScope.Shared);

oClientContext.Load(limitedWebPartManager.WebParts,
    wps => wps.Include(
    wp => wp.WebPart.Title));

oClientContext.ExecuteQuery();

if (limitedWebPartManager.WebParts.Count == 0)
{
    throw new Exception("No Web Parts on this page.");
}

WebPartDefinition oWebPartDefinition = limitedWebPartManager.WebParts[1];
WebPart oWebPart = oWebPartDefinition.WebPart;
oWebPart.Title = "My New Web Part Title";

oWebPartDefinition.SaveWebPartChanges();

oClientContext.ExecuteQuery();
```

Parte web Agregar un elemento web a una página

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
File oFile = oClientContext.Web.GetFileByServerRelativeUrl("Default.aspx");
LimitedWebPartManager limitedWebPartManager =
oFile.GetLimitedWebPartManager(PersonalizationScope.Shared);

string xmlWebPart = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
    "<WebPart xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"" +
    " xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"" +
    " xmlns=\"http://schemas.microsoft.com/WebPart/v2\">" +
    "<Title>My Web Part</Title><FrameType>Default</FrameType>" +
    "<Description>Use for formatted text, tables, and images.</Description>" +
    "<IsIncluded>true</IsIncluded><ZoneID></ZoneID><PartOrder>0</PartOrder>" +
    "<FrameState>Normal</FrameState><Height /><Width /><AllowRemove>true</AllowRemove>" +
    "<AllowZoneChange>true</AllowZoneChange><AllowMinimize>true</AllowMinimize>" +
    "<AllowConnect>true</AllowConnect><AllowEdit>true</AllowEdit>" +
    "<AllowHide>true</AllowHide><IsVisible>true</IsVisible><DetailLink /><HelpLink />" +
    "<HelpMode>Modeless</HelpMode><Dir>Default</Dir><PartImageSmall />" +
    "<MissingAssembly>Cannot import this Web Part.</MissingAssembly>" +
    "<PartImageLarge>/_layouts/images/mscontl.gif</PartImageLarge><IsIncludedFilter />" +
    "<Assembly>Microsoft.SharePoint, Version=13.0.0.0, Culture=neutral, " +
    "PublicKeyToken=94de0004b6e3fcc5</Assembly>" +
    "<TypeName>Microsoft.SharePoint.WebPartPages.ContentEditorWebPart</TypeName>" +
    "<ContentLink xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\" />" +
    "<Content xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\">" +
    "<![CDATA[This is a first paragraph!<DIV> </DIV>And this is a second
paragraph.]]></Content>" +
    "<PartStorage xmlns=\"http://schemas.microsoft.com/WebPart/v2/ContentEditor\"
/></WebPart>";

WebPartDefinition oWebPartDefinition = limitedWebPartManager.ImportWebPart(xmlWebPart);

limitedWebPartManager.AddWebPart(oWebPartDefinition.WebPart, "Left", 1);
```

```
oClientContext.ExecuteQuery();
```

Parte web Eliminar un elemento web de una página

```
ClientContext oClientContext = new ClientContext("http://MyServer/sites/MySiteCollection");
File oFile =
oClientContext.Web.GetFileByServerRelativeUrl("/sites/MySiteCollection/SitePages/Home.aspx ");
LimitedWebPartManager limitedWebPartManager =
oFile.GetLimitedWebPartManager(PersonalizationScope.Shared);

oClientContext.Load(limitedWebPartManager.WebParts);

oClientContext.ExecuteQuery();

if (limitedWebPartManager.WebParts.Count == 0)
{
    throw new Exception("No Web Parts to delete.");
}

WebPartDefinition webPartDefinition = limitedWebPartManager.WebParts[0];

webPartDefinition.DeleteWebPart();

oClientContext.ExecuteQuery();
```

Contexto. Usando un caché de credenciales para la ejecución elevada de código

Si bien el código del lado del servidor puede ejecutarse con privilegios elevados, no existe un método equivalente para elevar los privilegios en el código del lado del cliente (por razones de seguridad obvias). Como alternativa, puede especificar credenciales para emular el acceso de un usuario específico o cuenta de servicio.

Para especificar las credenciales, `ClientContext` un objeto **`CredentialCache`** , luego asígnele a la propiedad `Credentials` su objeto `ClientContext` .

El siguiente ejemplo emula la cuenta del grupo de aplicaciones y supone un entorno de SharePoint 2013 local con NTLM.

```
using System.Net;
using Microsoft.SharePoint.Client;

using (ClientContext ctx = new ClientContext("https://onpremises.local/sites/demo/"))
{
    // need the web object
    ctx.Load(ctx.Web);
    ctx.ExecuteQuery();

    // here the default network credentials relate to the identity of the account
    // running the App Pool of your web application.
    CredentialCache credCache = new CredentialCache();
    cc.Add(new Uri(ctx.Web.Url), "NTLM", CredentialCache.DefaultNetworkCredentials);

    ctx.Credentials = credCache;
```

```
    ctx.AuthenticationMode = ClientAuthentication.Default;
    ctx.ExecuteQuery();

    // do stuff as elevated app pool account
}
```

Tenga en cuenta que otorgar privilegios elevados a la cuenta del grupo de aplicaciones en SharePoint es contrario a las mejores prácticas, pero que en su lugar se podrían usar las credenciales de red relevantes.

Lea Trabajar con Managed Client Side Model Model (CSOM) en línea: https://riptutorial.com/es/sharepoint/topic/2679/trabajar-con-managed-client-side-model-model--csom-

Capítulo 10: Trabajar con Managed Server Side Object Model (plena confianza)

Observaciones

Jerarquía conceptual

En la jerarquía conceptual de SharePoint, las **colecciones de sitios** contienen **sitios** , que a su vez contienen **listas** . Una colección de sitios (`SPSite`) no tiene una IU explícita, pero siempre contiene un sitio de nivel raíz (accesible a través de la propiedad `RootWeb`) y posiblemente subsitios adicionales bajo ese sitio raíz. Un sitio o web (`SPWeb`) tiene una IU y contiene listas / bibliotecas de documentos (`SPList`), páginas con elementos web y elementos / documentos (`SPListItem`).

Advertencias del lado del servidor

- Para crear una aplicación que use el modelo de objetos del lado del servidor de SharePoint, en su proyecto de Visual Studio, debe agregar una referencia al conjunto Microsoft.SharePoint que se enumera en Ensamblajes de marco.
- Las aplicaciones que utilizan el Modelo de objetos del lado del servidor (plena confianza) solo pueden ejecutarse en un servidor Windows que hospeda SharePoint.
- No puede conectarse a un servidor de SharePoint que no sea en el que se está ejecutando la aplicación.

Examples

Hola Mundo (obteniendo el título del sitio)

2013

SharePoint 2013 y las versiones más nuevas son solo de 64 bits, por lo que el ensamblaje / programa también debe compilarse para un procesador de 64 bits.

Inmediatamente después de crear su proyecto, es necesario cambiar el **objetivo** de la **Plataforma** de **Cualquier CPU** a **x64, de lo** contrario se producirá un error.

```
using System;
using Microsoft.SharePoint;

namespace StackOverflow
{
    class Samples
    {
```

```
        static void Main()
        {
            using (SPSite site = new SPSite("http://server/sites/siteCollection"))
            using (SPWeb web = site.OpenWeb())
            {
                Console.WriteLine("Title: {0} Description: {1}", web.Title, web.Description);
            }
        }
    }
}
```

Recorriendo todo el conjunto de SharePoint

Usando PowerShell ejecutado desde un servidor web de SharePoint:

```
$wacoll = get-spwebapplication
foreach($wa in $wacoll){
    if($wa.IsAdministrationWebApplication -eq $false){
        foreach($site in $wa.Sites){
            foreach($web in $site.AllWebs){
                # your code here
                $web.Dispose()
            }
            $site.Dispose()
        }
    }
}
```

Recuperar los elementos de la lista

```
using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    SPList list = web.Lists["Some list"];

    // It is always better and faster to query list items with GetItems method with
    // empty SPQuery object than to use Items property
    SPListItemCollection items = list.GetItems(new SPQuery());
    foreach (SPListItem item in items)
    {
        // Do some operation with item
    }
}
```

Recuperar elementos utilizando la paginación

```
using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    SPList list = web.Lists["Some list"];
    SPQuery query = new SPQuery()
    {
        RowLimit = 100
    };
```

```
    do
    {
        SPListItemCollection items = list.GetItems(query);
        foreach (SPListItem item in items)
        {
            // Do some operation with item
        }

        // Assign current position to SPQuery object
        query.ListItemCollectionPosition = items.ListItemCollectionPosition;
    } while (query.ListItemCollectionPosition != null);
}
```

Obtener la lista por url

```
using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    string listUrl = string.Format("{0}{1}", web.ServerRelativeUrl, "Lists/SomeList");
    SPList list = web.GetList(listUrl);
}
```

Crear un elemento de lista

Al crear un nuevo elemento de la lista, sus campos se pueden configurar utilizando una sintaxis similar a las matrices de cadenas. Tenga en cuenta que estos campos no se crean sobre la marcha y están definidos por el esquema de la lista. Estos campos (o columnas) deben existir en el servidor, de lo contrario la creación fallará. Todos los elementos de la lista tendrán el campo Título. Algunas listas pueden tener campos obligatorios que deben completarse antes de que el elemento se publique en la lista.

En este ejemplo, la lista está utilizando la plantilla de Anuncios. Además del campo de título, la lista incluye el campo Cuerpo que mostrará el contenido del anuncio en la lista.

```
using (SPSite site = new SPSite("http://server/sites/siteCollection"))
using (SPWeb web = site.OpenWeb())
{
    SPList list = web.Lists["Announcements"];

    SPListItem item = list.AddItem();
    item[SPBuiltInFieldId.Title] = "My new item";
    item[SPBuiltInFieldId.Body] = "Hello World!";
    item.Update();
}
```

Lea Trabajar con Managed Server Side Object Model (plena confianza) en línea: https://riptutorial.com/es/sharepoint/topic/7543/trabajar-con-managed-server-side-object-model--plena-confianza-

Creditos

S. No	Capítulos	Contributors
1	Empezando con sharepoint	Community, Marco, Ryan Gregg, Thriggle, Tom Resing, Zach Koehne
2	Aplicación de SharePoint	Sunil sahu
3	Creación de una aplicación alojada por el proveedor	vinayak hegde
4	Principales lanzamientos	jjr2527, MikhailSP
5	Representación del lado del cliente de SharePoint 2013	Rohit Waghela, Yayati
6	Servicios de descanso	Aaron, Brock Davis, ocelotsloth, R4mbi, Rohit Waghela, Thriggle
7	Trabajar con cuadros de diálogo modales con JavaScript	Thriggle
8	Trabajar con JavaScript Client Object Model (JSOM)	Thriggle, yngrdyn
9	Trabajar con Managed Client Side Model Model (CSOM)	InvoiceGuy, Lukáš Nešpor, MikhailSP, RamenChef, Thriggle, Zach Koehne
10	Trabajar con Managed Server Side Object Model (plena confianza)	Lukáš Nešpor, Thriggle