|  |
| --- |
|  |

17/3/2015

|  |
| --- |
|  |
| https://uknowit.uwgb.edu/images/group91/41011/Office365.jpg |
|  |



|  |  |
| --- | --- |
|  | Laboratorio De Aplicaciones SharePoint Provider Hosted |

**Indice**

[1 Introducción 2](#_Toc414189710)

[2 Requisitos 2](#_Toc414189711)

[3 Creación de la aplicación 2](#_Toc414189712)

[4 Aplicación del Chrome de SharePoint 4](#_Toc414189713)

[5 Creación de una Lista 6](#_Toc414189714)

[6 Uso de REST con la Lista 6](#_Toc414189715)

# Introducción

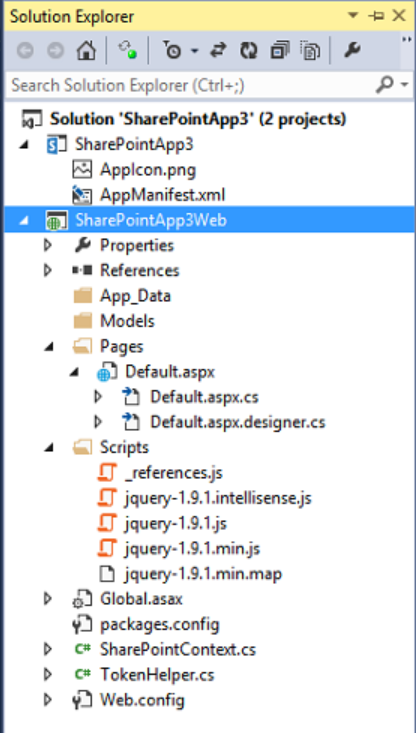
SharePoint 365 permite crear aplicaciones SharePoint Provider Hosted. En este laboratorio se creara una aplicación de este tipo utilizando la plantilla por defecto de Visual Studio.

# Requisitos

* Cuenta activada de Office 365
* SharePoint 365 configurado con una Colección de Sitios de Desarrollo y otra para el Catalogo de Aplicaciones
* Computador con Visual Studio 2013 (cualquier update). Puede ser en el computador local, en un computador virtualizado o un computador en Azure

# Creación de la aplicación

1. Inicie Visual Studio 2013 con la cuenta de administrador del computador
2. Seleccione "New project"
3. En la lista de plantillas de Visual Studio seleccione "Visual C#" 🡪 "Office/SharePoint" 🡪 "Apps" 🡪 "App for SharePoint"
4. Acepte el nombre por defecto ("SharePointApp2")
5. En la ventana de "New app for SharePoint":
   1. Indique en la casilla de "What SharePoint site do you want to use for debugging your app" el URL base de la Colección de Sitios de Desarrollo en SharePoint 365, por ejemplo <http://[nombre].sharepoint.com/sites/devsite/>
   2. Seleccione "Provider-hosted" en "How do you want to host your app for SharePoint"
   3. Seleccione "ASP.NET Web Forms Application" en la ventana de "Specify the web project type"
   4. Utilice "Use Windows Azure Access Control Service (for SharePoint cloud apps)" en la ventana de "Configure authentication settings"
   5. "Finish"
6. Si aparece, en la ventana de "Connect to SharePoint" indique la cuenta ("User ID") y contraseña de Office 365 y seleccione "Keep me signed" 🡪 "Sign in"
7. Revise la organización de la aplicación creada usando el "Solution Explorer" de Visual Studio:



1. Una Aplicación Provider Hosted consiste de dos soluciones:
   1. Una Solución ("SharePointApp2" en el ejemplo) para el software que se va a instalar en SharePoint 365 que incluye el archivo "AppManifest.xml"
   2. Una Solución ("SharePointApp2Web") en donde se va a crear el software propiamente dicho de la Aplicación. En este caso se seleccionó que se quería utilizar software del tipo ASP.NET, por lo que aparece una Solución Web que incluye una página aspx, lo mismo que scripts de JavaScript
   3. El archivo "TokenHelper.cs" contiene todas las rutinas necesarias para la comunicación, autenticación y autorización entre la aplicación Web y SharePoint
2. Inicie el debugeo de la página (F5)
3. Cuando el despliegue termine, autorice ("Trust") la Aplicación en la ventana que aparece
4. Note que la parte correspondiente a la Solución de SharePoint se instala en el sitio de desarrollo de SharePoint 365 y la parte correspondiente a la Solución Web ejecuta localmente en el computador donde se encuentra Visual Studio utilizando IIS Express
5. El texto que aparece en la página aspx (el título de la Aplicación) es generado por la línea 39 del archivo "Default.aspx.cs". Note que la página no tiene el Chrome (la barra azul superior característica de SharePoint)
6. Detenga el debugeo e impida que la línea 39 del archivo "Default.aspx.cs" ejecute poniéndole un comentario

# Aplicación del Chrome de SharePoint

1. Abra el archivo "Default.aspx" y entre los tags "<form id="form1" runat="server">" y "<div>" agregue el siguiente código que crea un lugar para mostrar el Chrome

<div id="chrome\_ctrl\_placeholder"></div>

1. Cree una nueva carpeta en la Solución "SharePointApp2Web" llamada "Images" y copie dentro de la carpeta el archivo "Suges.jpg" que se encuentra junto con este documento
2. Abra la carpeta de "Scripts" en la Solución "SharePointApp2Web" y agréguele un archivo del tipo JavaScript llamado "App.js"
3. Agregue el siguiente código en el archivo "App.js". Las rutinas de JavaScript incluirán el Chrome de SharePoint tal y como lo utiliza el sitio de SharePoint 365

"use strict";

var ChromeControl = function () {

var init = function () {

var hostWebUrl = queryString("SPHostUrl");

$.getScript(hostWebUrl + "/\_layouts/15/SP.UI.Controls.js", render);

},

render = function () {

var options = {

"appIconUrl": "../Images/AppIcon.png",

"appTitle": "Mi Aplicacion SharePoint Provider Hosted",

"settingsLinks": [

{

"linkUrl": "../Pages/Default.aspx?" + document.URL.split("?")[1],

"displayName": "Aplicacion SharePoint Provider Hosted"

}

]

};

var nav = new SP.UI.Controls.Navigation(

"chrome\_ctrl\_placeholder",

options

);

nav.setVisible(true);

},

queryString = function (p) {

var params =

document.URL.split("?")[1].split("&");

var strParams = "";

for (var i = 0; i < params.length; i = i + 1) {

var singleParam = params[i].split("=");

if (singleParam[0] == p)

return decodeURIComponent(singleParam[1]);

}

}

return {

init: init,

}

}();

(function () {

"use strict";

jQuery(function () {

ChromeControl.init();

});

}());

1. Ejecute la Aplicación y compruebe que el Chrome es visible en la pagina

# Creación de una Lista

1. En el "Solution Explorer" de Visual Studio seleccione la Solución "SharePoint2" y agréguele una carpeta llamada "Listas"
2. Dentro de la carpeta agregue un nuevo item del tipo "List". Conserve el nombre por defecto ("List1")
3. Ejecute la Aplicación
4. Después de que la página de la Aplicación ha cargado completamente, vaya al sitio de desarrollo de SharePoint 365 y cambie el URL a "https://[nombre]-[identificador].sharepoint.com/sites/[nombre devsite]/SharePointApp2/Lists/List1"
5. La página de la Lista debe aparecer. Note que la Lista ha sido creada en el sitio de desarrollo de SharePoint 365, no en la parte de la Aplicación que está ejecutando localmente (que es una aplicación Web normal, y por lo tanto no tiene nada que ver con SharePoint)
6. Detenga la Aplicación

# Uso de REST con la Lista

Para agregar código que lea los elementos de la Lista desde la Aplicación, usando REST:

1. Abra el archivo "Default.aspx" y entre los tags "<div>" y "</div>" agregue el siguiente codigo:

<ul id="elementosLista"></ul>

1. Abra el archivo "App.js" y agréguele el siguiente código al final de la pagina

(function () {

"use strict";

jQuery(function () {

//Dar Host y App web URLS

var appWebUrl = "";

var spHostUrl = "";

var args = window.location.search.substring(1).split("&");

for (var i = 0; i < args.length; i++) {

var n = args[i].split("=");

if (n[0] == "SPHostUrl")

spHostUrl = decodeURIComponent(n[1]);

}

for (var i = 0; i < args.length; i++) {

var n = args[i].split("=");

if (n[0] == "SPAppWebUrl")

appWebUrl = decodeURIComponent(n[1]);

}

//Cargar Librerias

var scriptbase = spHostUrl + "/\_layouts/15/";

jQuery.getScript(scriptbase + "SP.RequestExecutor.js", function (data) {

//Llamar Host Web con REST

var executor = new SP.RequestExecutor(appWebUrl);

executor.executeAsync({

url: appWebUrl + "/\_api/web/lists/getbytitle('List1')/items",

method: "GET",

headers: { "accept": "application/json;odata=verbose" },

success: function (data) {

var results = JSON.parse(data.body).d.results;

for (var i = 0; i < results.length; i++) {

$("#elementosLista").append("<li>" + results[i].Title + "</li>");

}

},

error: function () {

alert("Error!");

}

});

});

});

}());

1. Inicie la Aplicación en modo de debugeo
2. Espere a que la pagina cargue por completo y luego vaya al sitio de desarrollo de SharePoint 365, a la página indicada en el punto 4 de la sección 5
3. Agregue unos cuantos elementos a la Lista
4. Regrese a la página de la Aplicación y refrésquela. Compruebe que los títulos de los elementos creados en la Lista aparecen en la pagina