Nº27 marzo 2016

CompartiMOSS

REVISTA ESPECIALIZADA EN TECNOLOGÍAS MICROSOFT

Entrevista Enrique Rhenals

SharePoint 2013 Client Side Rendering en detalle

Introducción al PnP Program y Provisioning framework

ENMARCHA: Un Framework Open Source para agilizar el desarrollo en SharePoint

Contenido

Staff

CompartiMOSS es una publicación independiente de distribución libre en forma electrónica. Las opiniones aquí expresadas son de estricto orden personal, cada autor es completamente responsable de su propio contenido.

DIRECCIÓN GENERAL
- Gustavo Velez
- Juan Carlos Gonzalez
- Fabian Imaz
- Alberto Diaz

DISEÑO Y DIAGRAMACIÓN
- Santiago Porras Rodríguez

Contacte con nosotros

revista@compartimoss.com
gustavo@gavd.net
jcgonzalezmartin1978@hotmail.com
fabian@siderys.com.uy
adiazcan@hotmail.com

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

REDES SOCIALES
Facebook:
http://www.facebook.com/group.php?gid=128911147140492
LinkedIn:
http://www.linkedin.com/groups/CompartiMOSS-3776291
Twitter:
@CompartiMOSScom

03

Editorial

La revista sigue creciendo un número más y, por su puesto, les tenemos que dar gracias a ustedes los lectores, que trimestre a trimestre dedican un ratito de su preciado tiempo a leer y/o descargar el tan trabajado número que nuestros autores han creado.

Este número viene con una novedad muy importante: queremos que CompartiMOSS sea la revista referencia de tecnologías Microsoft, no sólo en cuanto a SharePoint y Office 365 se refiere, si no también, de cualquier producto de Microsoft o relacionado con este.

Por eso, los queremos seguir invitando a enviar artículos y a abrir sus cerebros ante la nueva oleada de autores que esperamos recibir con artículos de Azure, Xamarin o .NET. ¿Qué les parece este cambio?

Desde la dirección de la revista estamos ilusionados con el reto y esperamos crecer en el número de autores y lectores para conseguir ser el referente de habla hispana en tecnología Microsoft.

Esperamos que se diviertan tanto leyendo este número, como los autores se divirtieron creando los artículos, y nosotros, el grupo editor, manejando la revista.

La dirección de CompartiMOSS

04 Entrevista Enrique Rhenals

Mi nombre es Enrique Rhenals Bárcenas y nací en 1976 en Cartagena (Colombia) y vivo en Bogotá D.C.

Inicio a laborar en el mundo de la tecnología en 1998 cuando estoy realizando mis estudios de Ingeniería de Sistemas.

Inicio como Desarrollador en .Net y años después seguir en el mundo de las ETL's, Cubos y Reportes empezando con SQL Server 2005 (y siguiendo en sus versiones siguientes).

En el año 2010 inicio mi aventura en el mundo SharePoint, con proyectos varios en diferentes clientes.

Y en estas aventuras es que me encuentro hoy día.

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

Desde niño siempre quise estar en el mundo de la tecnología por lo cual estudié Ingeniería de Sistemas.

¿Cuáles son tus principales actividades tecnológicas hoy en día?

Actualmente soy Arquitecto de Soluciones Microsoft (en SharePoint e Inteligencia de Negocios) de uno de los Partners Microsoft más grandes en servicios tecnológicos de Colombia, dentro de mis actividades actuales están: consultoría en productos Microsoft (SharePoint y SQL Server, enfocado a inteligencia de negocios), entre otras.

Hoy día soy integrante del equipo Core de la comunidad SharePoint de Colombia "SHARECOL" en el cual se realizan eventos a nivel Colombia y Latinoamérica (en compañía de las diversas comunidades de SharePoint en Latinoamérica).

También actualizo con cierto frecuenta mi blog (http://enriquerhenalsb.blogspot.com/) con contenidos de SharePoint que, entre sus diversos artículos, se encuentran tips de cómo resolver diferentes errores que se presentan en SharePoint, también hay videos ilustrativos de como instalar una granja, mejores prácticas, etc.

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

El tiempo que tengo libre lo invierto en leer, hablar con mi familia, ver tv, ir a cine, bailar y escuchar música (mis ritmos favoritos son: Vallenato, Merengue (Wilfrido Vargas, Rikarena, Sergio Vargas, entre otros) y Reggae (Bob Marley, UB40 entre otros).

Me encanta ver películas de ciencia ficción (estilo Star Wars y Star Trek) y las películas de acción y de motivación. Soy gran fan de la Guerra de las Galaxias, creo que me he visto la saga completa muchísimas veces (la VII me la vi 3 veces en cine en menos de dos semanas. Hago anotación que no me gusto la muerte de Han Solo. Si alguien conoce a JJ Abrams … háganselo saber de mi parte).

¿Cuáles son tus hobbies?

Leer en un parque al aire libre los domingos, ir a cine, ver TV.

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

Muy emocionante e interesante… pienso que en los próximos años se hará más normal (como ha pasado en años anteriores) el uso de la tecnología para usos más comunes.

Pienso en el gran avance que tendrá el análisis de volúmenes grandes de información para usos más de la persona común (no del área de tecnología). Ej.: controles médicos a través de dispositivos estilo SmartWatch, entre otros.

También será tendencia el tema de que cada vez más se tendrá más servicios e información en la nube (Cloud).

ENRIQUE RHENALS BÁRCENAS
Arquitecto de Soluciones Microsoft
MVP Office Servers and Services
erhenalsb@hotmail.com
@erhenals
http://enriquerhenalsb.blogspot.com/

05

SharePoint y Azure – Cloud Services

Muchos de los servicios ofrecidos por Azure se pueden utilizar para ampliar y mejorar el funcionamiento de SharePoint, tanto on-premises como en la nube. Uno de los servicios que ofrece Azure es «Cloud Services», que es una solución de Plataforma como Servicio (PaaS) que proporciona toda la infraestructura física más el software de base para ejecutar aplicaciones Web personalizadas. Utilizando Azure Cloud Services solamente es necesario construir (desarrollar) la aplicación, compilarla, empaquetarla y desplegarla al servicio; de todo el resto, crear máquinas virtuales, configurarlas, darles mantenimiento y respaldo, se encarga Azure.

Azure Cloud Services ofrece una máquina virtual para cada servicio que se crea, en la que el usuario es administrador y puede controlar todo su funcionamiento. En contraposición, Azure Web Apps permite crear aplicaciones Web de la misma forma que Cloud Services, pero toda la infraestructura es compartida y el usuario no tiene ni derechos de administrador del servicio ni en la máquina virtual que hospeda el servicio.

Cuando se crea un Azure Cloud Service, el motor de servicios de Azure (llamado el «Fabric Controller») despliega y configura todos los aspectos necesarios para que el servicio funcione inspeccionando el paquete de despliegue. De esta forma, si por alguna razón el servicio deja de funcionar, el Fabric Controller puede en cualquier momento crear un nuevo servicio, instalar el paquete y seguir funcionando con una caída del sistema de solamente algunos minutos. Lo mismo ocurre cuando se necesita incrementar el poder de procesamiento del servicio: si el usuario indica que necesita más CPU o memoria, el Controller se encarga de crear la (o las) máquina virtual necesaria, configurarla, instalar el paquete y configurar el balanceador de carga para que todo funcione apropiadamente, sin que el usuario tenga que intervenir de ninguna manera. Una consecuencia importante de esta arquitectura es que no se debe mantener ningún tipo de dato localmente en un servidor del Cloud Service, es decir, el software creado tiene que ser «stateless». Tampoco se deben realizar configuraciones manualmente en los servidores, todo debe ser hecho por medio de scripts.

Azure Cloud Service permite crear dos tipos de roles: «Web Role» y «Worker Role». La principal diferencia es que el primero dispone de IIS instalado por defecto y el segundo no. El Web Role es un servicio ideal para crear servicios WCF (Windows Communication Foundation) que se pueden utilizar de diferentes maneras con SharePoint. Por medio de un servicio WCF se pueden conectar sistemas externos con SharePoint de tal forma que los datos que contienen se pueden procesar dentro de SharePoint mismo y se pueden hacer interactuar con el Business Connectivity Service de SharePoint, haciéndolos, al mismo tiempo, indexables con el motor de búsqueda.

Creación de un servicio de Azure Cloud Service

Para comenzar a utilizar el servicio es necesario crear primero un Recurso de Trabajo de Cloud Service:

1.– Desde el portal de administración de Azure (https://portal.azure.com/), haga un login con sus credenciales.
2.– Utilice el botón de “+” en la esquina superior izquierda para agregar un nuevo servicio.
3.– Seleccione “Compute” – “Cloud Service” – “Create”.

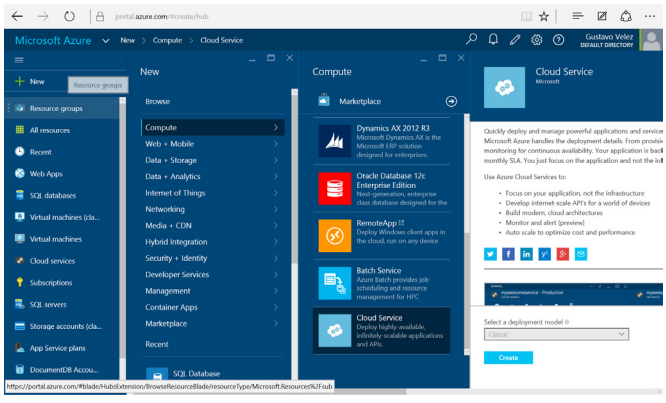


Imagen 1. Creación de un Cloud Service.

4.– En el nuevo panel defina un nombre para el servicio (“ZooCS” en el ejemplo), la suscripción de Azure a utilizar, el grupo de recursos de Azure (o crear uno nuevo) y la localización del centro de datos de Azure. Utilice el botón de “Create”. Después de algunos segundos, el nuevo servicio aparece en la lista de “Cloud Services” del portal.

Creación de un servicio Web WCF (SOAP)

WCF permite crear servicios Web que pueden ser utilizados por múltiples tipos de sistemas. Por defecto, las plantillas de Visual Studio para WCF crean servicios SOAP (Simple Object Access Protocol) que utiliza XML para el intercambio de información estructurada entre los sistemas conectados. WCF SOAP es, también, el protocolo indicado para conectar sistemas externos con el Business Connectivity Service de SharePoint. El siguiente ejemplo crea un servicio Web WCF con Visual Studio 2015 (aunque el procedimiento es igual a con Visual Studio 2013) que no utiliza datos de ningún otro sistema, sino que maneja datos creados y mantenidos en memoria. El objetivo del ejemplo es mostrar cómo crear el servicio; después de creado, el servicio se puede conectar de forma sencilla con cualquier tipo de fuente de datos (Base de Datos, sistema externo, etc.). Note, además, que el código fuente del ejemplo solamente contiene rutinas básicas para atrapar excepciones, por lo que no está listo para ser utilizado en sistemas de producción.

5.– Instale el Azure SDK para Visual Studio 2015 si no lo tiene instalado en VS. Desde el sitio de Microsoft (https://azure.microsoft.com/en-us/downloads/archive-net-downloads/) se puede descargar e instalar.

6.– Inicie Visual Studio 2015 y cree un nuevo proyecto del tipo “Cloud” - “Azure Cloud Service” y asígnele un nombre («ZooWCF» en el ejemplo).

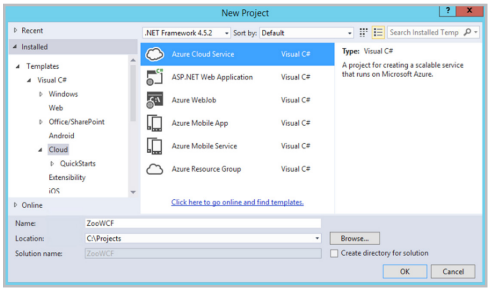


Imagen 2.- Creación del proyecto de Visual Studio.

7.– En la ventana de “New Microsoft Azure Cloud Service” seleccione “WCF Service Web Role” y utilice la flecha hacia la derecha para mover el tipo de servicio al panel de la derecha. Haga clic sobre el nombre en el panel de la derecha y cambie el nombre de “WCFServiceWebRole1” a “ZooWebRole”.”OK”

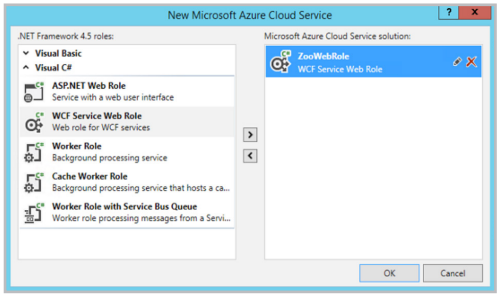


Imagen 3.- Selección del role.

8.– Visual Studio inicia el proyecto y abre el archivo “Service1.svc.cs”. Usando el “Solution Explorer” de Visual Studio, renombre los archivos “IService1.cs” en “IZoo.cs” y “Service1.svc en “Zoo.svc”.

9.– En el archivo “Service1.svc.cs”, renombre la clase “Service1” a “Zoo”. Elimine todos los comentarios agregados por defecto por la plantilla.

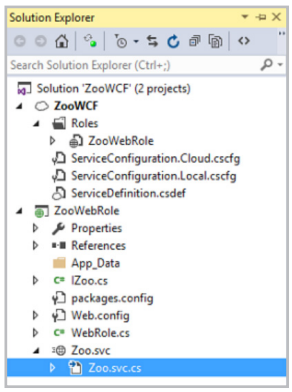


Imagen 4.- El proyecto de VS con los nombres correctos.

10.– Si está utilizando Visual Studio 2015, el proyecto “ZooWebRole” es creado utilizando .NET Framework 4.5.2. En el momento (cambió en enero del 2016 a 4.5.2, pero servicios creados anteriormente siguen funcionando con 4.5.0), Azure utiliza la versión 4.5.0, por lo que es indispensable cambiar el framework en el proyecto de Visual Studio desde su pantalla de propiedades. Otra posibilidad es que después de desplegar el proyecto en Azure, utilizando su máquina virtual se instale el framework 4.5.2 (vea el punto 29 que trata sobre la máquina virtual del servicio).

11.– Abra el archivo «IZoo.cs» y reemplace el código dentro de la función «public class CompositeType» con el siguiente fragmento. Esta es la entidad que se va a utilizar para definir los objetos de tipo «Animal» del Zoológico. Note que las tres variables iniciales utilizadas para las propiedades tienes que tener alcance «public», de otra forma el servicio de SharePoint BDC no podrá utilizar el servicio WCF:

```
[DataContract]
  public class Animal
  {
    Int32 animalID;
    string animalName;
    string animalDanger;

    [DataMember]
    public Int32 AnimalID
    {
      get { return animalID; }
      set { animalID = value; }
    }

    [DataMember]
    public string AnimalName
    {
      get { return animalName; }
      set { animalName = value; }
    }

    [DataMember]
    public string AnimalDanger
    {
      get { return animalDanger; }
      set { animalDanger = value; }
    }
  }
```

En el mismo archivo “IZoo.cs” reemplace el código dentro de la función “public interface IZoo” con el siguiente fragmento. Esta es la interface que se va a utilizar en la definición de los métodos de tipo “Zoo”:

```
[ServiceContract]
  public interface IZoo
  {
    [OperationContract]
    IEnumerable<Animal> ReadList();

    [OperationContract]
    Animal ReadItem(int animalID);

    [OperationContract]
    Animal Create(Animal newAnimal);

    [OperationContract]
    void Update(Animal animal);

    [OperationContract]
    void Delete(int animalID);
  }
```

12.– Abra el archivo “Zoo.svc.cs” y reemplace todo el código dentro de la función “public class Zoo : IZoo” con el siguiente código. Este fragmento crea e inicializa una lista genérica de objetos “myZoo” del tipo “Animal” que se va a utilizar como fuente de datos. Visual Studio indica un error en “IZoo” porque los métodos de la interface todavía no se han definido:

```
static List<Animal> myZoo = new List<Animal>
    {
      new Animal(){ AnimalID = 1, AnimalName = “Cucaracha”,
AnimalDanger = “Poco” },
      new Animal(){ AnimalID = 2, AnimalName = “Pulga”,
AnimalDanger = “Mucho” }
    };
```

13.– En seguida del código del punto 13, incluya la siguiente función que define el método “ReadList” encargado de leer la fuente de datos y devolver un objeto con todos los animales en ella. El método utiliza internamente LINQ para Objetos:

```
public IEnumerable<Animal> ReadList()
    {
      try
      {
        IEnumerable<Animal> animalList =  // Using LINQ
          from oneAnimal in myZoo
          orderby oneAnimal.AnimalID
          select new Animal
          {
            AnimalID = oneAnimal.AnimalID,
            AnimalName = oneAnimal.AnimalName,
            AnimalDanger = oneAnimal.AnimalDanger
          };

        return animalList;
      }
      catch (Exception ex)
      {
        throw new Exception(“There was a problem reading animal
data - “, ex);
      }
    }
```

14.– El siguiente método define como encontrar con LINQ un animal especifico y devolver su valor. Agregue el código después del código del punto 14:

```
public Animal ReadItem(int animalID)
    {
      try
      {
        IEnumerable<Animal> animalList =
          from oneAnimal in myZoo
          where oneAnimal.AnimalID == animalID
          select new Animal
          {
            AnimalID = oneAnimal.AnimalID,
            AnimalName = oneAnimal.AnimalName,
            AnimalDanger = oneAnimal.AnimalDanger
          };

        return animalList.First();
      }
      catch (Exception ex)
      {
        throw new Exception(“There was a problem reading animal
data - “, ex);
      }
    }
```

15.– La creación de un nuevo animal en el zoológico no implica más que agregar un nuevo objeto en la colección. Añada el siguiente fragmento después del código indicado en el punto 15:

```
public Animal Create(Animal newAnimal)
    {
      try
      {
        myZoo.Add(newAnimal);

        return newAnimal;
      }
      catch (Exception ex)
      {
        throw new Exception(“There was a problem creating a new
animal - “, ex);
      }
    }
```

los servicios ofrecidos por Azure se pueden utilizar para ampliar y mejorar el funcionamiento de SharePoint

16.– Para modificar los valores de un animal, es necesario primero encontrar el objeto en la colección y luego aplicarle los cambios. En el método indicado a continuación, se utiliza como ejemplo una expresión Lambda para encontrar el animal, aunque es posible utilizar LINQ como en los otros métodos. Agregue el fragmento después del indicado en el punto 16:

```
    public void Update(Animal animal)
    {
      try
      {
        int animalIndex = myZoo.FindIndex(a => a.AnimalID == animal.
AnimalID);  // Using Lambda Expressions
        if (animalIndex >= 0)
          myZoo[animalIndex] = animal;
      }
      catch (Exception ex)
      {
        throw new Exception(“Problem updating animal with ID = “ +
animal.AnimalID.ToString() + “ - “, ex);
      }
    }
```

17.– Finalmente, para eliminar un animal del zoológi-

co, es necesario encontrarlo primero, y luego sacarlo de la colección (en el ejemplo se indica cómo hacerlo con LINQ y con una expresión Lambda):

```
public void Delete(int animalID)
    {
      try
      {
        IEnumerable<Animal> animalList =
         from oneAnimal in myZoo
         where oneAnimal.AnimalID == animalID
         select oneAnimal;

        myZoo.Remove(animalList.First());

        //myZoo.Remove(myZoo.Find(a => a.AnimalID == animalID));       //
Using Lambda Expressions
      }
      catch (Exception ex)
      {
        throw new Exception("Problem deleting animal with ID = " +
animalID.ToString() + " - ", ex);
      }
    }
```

18.– En principio, el servicio WCF contiene todo el código para definir la entidad, la interface y los métodos CRUD y está listo para funcionar. Ejecute el proyecto en forma de debuggeo. Esto inicia el Emulador de Azure y también IIS Express. Verifique que el proyecto compila y ejecuta correctamente (una instancia de IE debe iniciarse automáticamente con algunos mensajes indicando que el servicio está ejecutando sin problemas). Visual Studio incluye un Emulador de Azure Service Cloud que permite hacer funcionar el servicio WCF localmente sin necesidad de subirlo a Azure. De igual manera, se inicia IIS Express de forma automática para hostear el servicio. Es importante hacer notar que, si el computador en donde se está desarrollando el código es también un servidor de SharePoint, la combinación de Emulador e IIS Express destruye totalmente algunos de los servicios de SharePoint (BCS, Apps entre otros) y no hay forma de volverlos a hacer funcionar fuera de reinstalar totalmente a SharePoint.

19.– Detenga la ejecución del proyecto. Agréguele a la Solución de Visual Studio un nuevo proyecto del tipo "Console Application" llamado "ZooWCFTest"

20.– Después de que el proyecto ha sido creado, seleccione su directorio de "References" y utilizando el menú contextual, seleccione "Add Service Reference". En la ventana de "Add Service Reference" utilice el botón de "Discover". En el panel de "Services" debe aparecer el servicio acabado de crear. Selecciónelo y utilice el botón de "OK" para crear la referencia

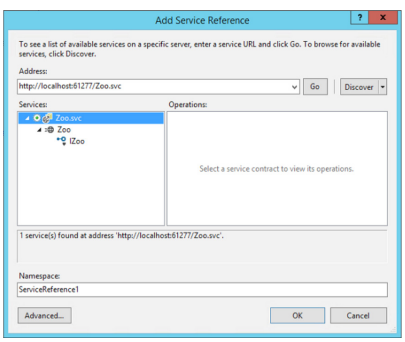


Imagen 5.- Utilizando el servicio WCF en una Aplicación de Consola.

21.– Abra el archivo "Program.cs" de la aplicación de consola, y agregue el siguiente fragmento de código en la rutina "static void Main(string[] args)". En la aplicación de consola se hace una llamada a cada uno de los métodos del servicio WCF para comprobar que todo funciona bien:

```
static void Main(string[] args)
    {
      ZooWCFTest.ServiceReference1.ZooClient myclient = new
ZooWCFTest.ServiceReference1.ZooClient();
      Console.Clear();

      Console.WriteLine("** ReadList test");
      Console.WriteLine("Numero inicial de animales = " + myclient.
ReadList().Count().ToString());

      Console.WriteLine("** ReadItem test");
      ZooWCFTest.ServiceReference1.Animal myResponse =
myclient.ReadItem(1);
      Console.WriteLine("Animal con ID 1 = " + myResponse.
AnimalName + " " + myResponse.AnimalDanger);

      Console.WriteLine("** Delete test");
      myclient.Delete(2);
      Console.WriteLine("Animal con ID 2 eliminado");
      Console.WriteLine("Numero de animales = " + myclient.
ReadList().Count().ToString());

      Console.WriteLine("** Create test");
      ZooWCFTest.ServiceReference1.Animal newAnimal = new
ServiceReference1.Animal();
      newAnimal.AnimalID = 10;
      newAnimal.AnimalName = "Elefante";
      newAnimal.AnimalDanger = "Ninguno";
      ZooWCFTest.ServiceReference1.Animal myAnimal = myclient.
Create(newAnimal);
      Console.WriteLine("Nuevo animal = " + myAnimal.AnimalName
+ " " + myAnimal.AnimalDanger);
      Console.WriteLine("Numero de animales = " + myclient.
ReadList().Count().ToString());

      Console.WriteLine("** Update test");
      ZooWCFTest.ServiceReference1.Animal updateAnimal = new
ServiceReference1.Animal();
      updateAnimal.AnimalID = 1;
      updateAnimal.AnimalName = "Cucaracha";
      updateAnimal.AnimalDanger = "Peligrosisima";
      myclient.Update(updateAnimal);
      myResponse = myclient.ReadItem(1);
      Console.WriteLine("Animal con ID 1 = " + myResponse.
AnimalName + " " + myResponse.AnimalDanger);
    }
```

22.– Compile el programa de prueba. Inicie el servicio WCF de nuevo (F5) para que esté funcionando cuando se hacen las llamadas desde el programa de prueba (si da un error, detenga a IIS Express e inicie el servicio de nuevo). Inicie la aplicación de consola y ejecute el programa. Cada uno de los métodos del servicio WCF debe reaccionar de la forma esperada:

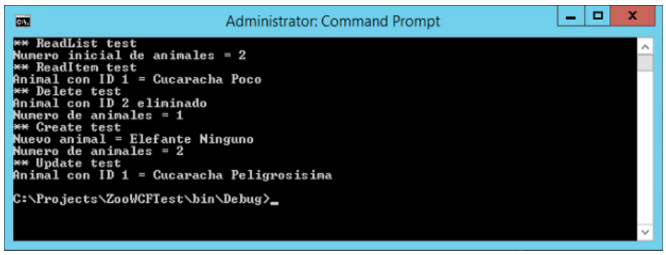


Imagen 6.- Salida de la Aplicación de Consola probando el servicio WCF.

23.– Detenga el servicio WCF. Una vez comprobado que el servicio funciona correctamente en el Emulador local, es necesario subirlo a Azure. En el Solution Explorer de Visual Studio seleccione el proyecto "ZooWCF" y utilizando el menú de contexto selec-

cione “Publish”. En la ventana de “Publish Azure Application” que aparece, introduzca la cuenta y clave de Azure en donde se ha creado el Cloud Service del punto 1

24.– Una vez las credenciales han sido aceptadas, la casilla de “Choose your subscription” permite seleccionar la suscripción utilizada para crear el servicio del punto 1. “Next”:

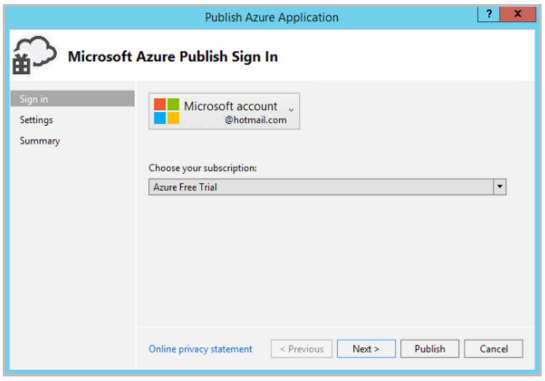


Imagen 7.- Asistente de publicación del servicio WCF.

25.– La siguiente ventana permite seleccionar los detalles de la publicación. Seleccione el servicio creado en el punto 1 en la casilla “Cloud Service”, y el tipo de ambiente que se quiere utilizar (Cloud Service permite publicar un servicio en Producción o en Staging, de tal forma que se pueda testear antes de que los usuarios lo utilicen). De la misma forma, la pestaña “Advanced Settings” permite ajustar la etiqueta del servicio y seleccionar si se desea utilizar “IntelliTrace” y “Remote Debugger”: la primera opción permite ver los logs de utilización del servicio desde Visual Studio divididos en entradas para excepciones, consultas, información del sistema, hilos de Windows y módulos utilizados. Con el debugeo remoto se puede acoplar el Visual Studio local con el servicio en Azure para detectar problemas.

En la primera pestaña (“Common Settings”) se puede seleccionar también “Enable Remote Desktop for all roles”. Un Servicio Cloud en Azure crea una máquina virtual a la que es posible acceder por medio de Remote Desktop. Cuando se selecciona esta opción, es necesario definir las credenciales del usuario de Remote Desktop en la ventana que aparece automáticamente.

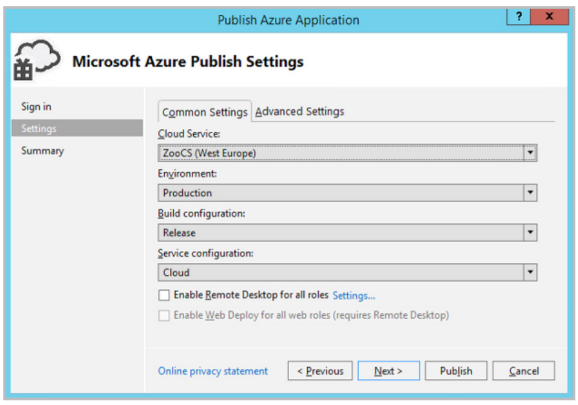


Imagen 8. Asistente de publicación del servicio WCF.

26.– La ultima ventana del asistente presenta un resumen de las opciones; si es necesario, se puede utilizar el botón de “Previous” para modificar cualquiera de ellas. Utilice el botón de “Publish” para publicar el servicio en Azure. Después de algunos segundos, Visual Studio abre la ventana de “Microsoft Azure Activity Log” que permite seguir los pasos del despliegue y revisar su estado:

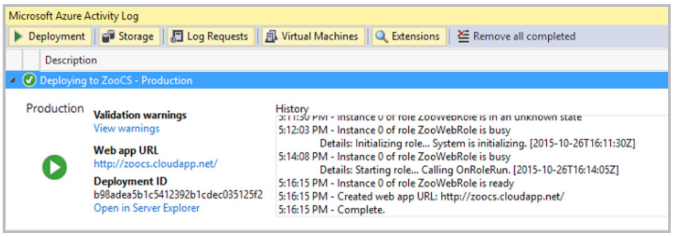


Imagen 9.- Despliegue del servicio WCF monitorizado desde Visual Studio.

Azure Cloud Service permite crear dos tipos de roles: “Web Role” y “Worker Role”

27.– Una vez desplegado en Azure, el servicio se puede testear con la Aplicación de Consola creada en el punto 20 cambiando el URL de la referencia de servicio, o utilizando alguna otra herramienta como “WcfTestClient”. Note que WcfTestClient ya no viene incluida con Visual Studio 2015, pero se puede descargar desde varios sitios en Internet. El URL del servicio es el configurado en el punto 4 cuando se creó el Cloud Service en Azure, seguido por “.cloudapp.net/[NombreClase].svc” (en el ejemplo seria “http://zoocs.cloudapp.net/zoo.svc”). Desde la consola de manejo de Azure se puede encontrar también el URL, junto con todas sus opciones de configuración.

28.– Si se seleccionó la opción de “Enable Remote Desktop for all roles” en el punto 26 de despliegue, se puede interactuar directamente con la máquina virtual del Cloud Service. En Visual Studio abra el “Server Explorer” y expanda la sección de “Azure” - “Cloud Services”. Utilizando el menú contextual sobre la entrada de la instancia aparece una opción para iniciar el Remote Desktop de Windows y conectarse con la máquina virtual. Utilice la clave configurada en el punto 26.

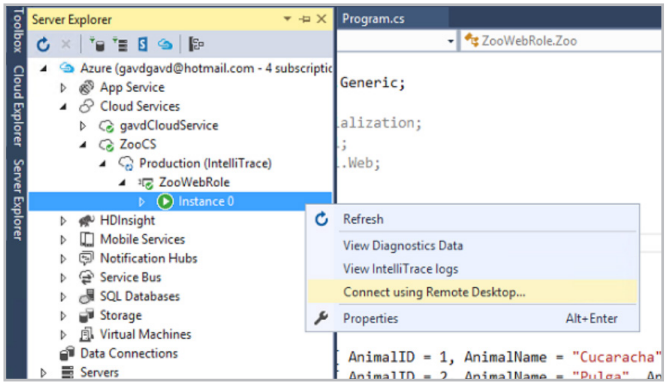


Imagen 10.- Conectar con el Remote Desktop del servicio desde Visual Studio.

29.– El servicio también se puede desplegar desde Azure mismo, sin necesidad de hacerlo desde Visual Studio. Primero compile el paquete de instalación: desde el Solution Explorer de Visual Studio, seleccione el proyecto "ZooWCF" y utilizando su menú contextual seleccione "Package". En la ventana que abre seleccione la configuración que se desea empacar ("Cloud") y la configuración de compilación ("Release"). También puede seleccionar si se desea utilizar el Remote Desktop y/o el debugeo remoto. El botón de "Package" compila todo el proyecto y crea dos archivos "ServiceConfiguration.Cloud.cscfg" y "ZooWCF.cspkg" (los dos son archivos .zip que se pueden descomprimir fácilmente). Desde el panel de control de Azure, seleccione "Cloud Services", seleccione el servicio creado en el punto 1 y en el panel con los detalles del servicio utilice el botón de "Update". En el nuevo panel que abre al lado derecho, hay casillas para subir el archivo del paquete (.cspkg) y el archivo de configuración (.cscfg), lo mismo que para definir el título del despliegue.

Utilización del servicio SOAP en el servicio de BCS de SharePoint

Una vez el servicio WCF está instalado y configurado en Azure Cloud Services, puede ser consumido por el Business Connectivity Service de SharePoint. Esto permite no solo integrar los datos e interactuar con ellos directamente desde SharePoint, sino que también se pueden indexar para poderlos consultar por medio del motor de búsqueda, y utilizarlos como metadatos por medio de la columna de datos externos.

30.– Desde la Administración Central de SharePoint, vaya a "Administración de aplicaciones" - "Administrar aplicaciones de servicio" - "Servicio de almacenamiento seguro". Si el servicio no ha sido configurado, comience por generar una clave para activarlo.

31.– Utilice el botón de "Nueva" (en la cinta). En la nueva página, defina el ID (una cadena cualquiera), nombre para mostrar y correo electrónico; seleccione "Grupo restringido" en la sección "Tipo de aplicación de destino" - "Siguiente".

32.– Para los nombres de campo utilice "Nombre de usuario" y "Contraseña" y en las casillas de "Tipo de campo" seleccione "Nombre de usuario" y "Contraseña". Conserve las otras opciones por defecto. "Siguiente".

33.– En la siguiente página seleccione el (o los) usuario que va a ser administrador y los usuarios que serán miembros del grupo de seguridad. "Aceptar".

34.– Una vez la aplicación de destino ha sido creada aparece en la lista de SharePoint. Selecciónela y utilice el botón de "Establecer". En la ventana que abre introduzca el nombre de usuario de Azure que tiene acceso al servicio WCF y su contraseña (dos veces). De esta forma se está dando acceso a un determinado número de usuarios de SharePoint al servicio de WCF en Azure. Note que desde el portal de Azure, en la sección del Cloud Service creado en el punto 1, es posible definir tanto los usuarios (dentro de Azure) que tienen acceso al servicio, como los roles que se le pueden asignar

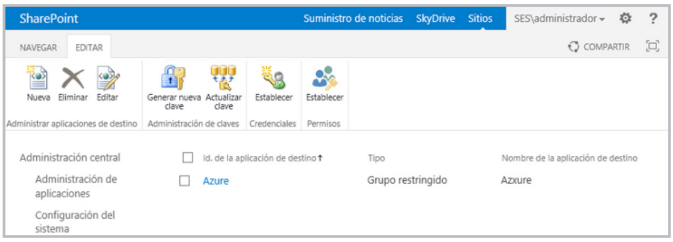


Imagen 11.- Servicio de Almacenamiento Seguro de SharePoint.

35.– Abra el sitio de SharePoint en donde se quiera utilizar el servicio WCF, utilice la pestaña de "Pagina" e inicie SharePoint Designer desde "Editar" - "Editar en SharePoint Designer".

36.– En SharePoint Designer haga clic sobre "Tipos de contenido externo" en el menú vertical al lado izquierdo. Utilice el botón de "Tipo de contenido externo" en la sección de "Nueva" de la cinta.

37.– Asígnele un nombre al Tipo de Contenido ("Zoo") y utilice el vínculo de "Haga clic aquí para detectar orígenes de datos externos…".

38.– En la nueva página utilice el botón de "Agregar conexión". Seleccione "Servicio WCF" como "Tipo de origen de datos" y "Aceptar".

39.– En la casilla de "Dirección URL de metadatos de servicio" introduzca el URL del servicio WCF en Azure (punto 28) seguido por "?wsdl", así que en el ejemplo quedaría "http://zoocs.cloudapp.net/zoo.svc?wsdl". En "Dirección URL de extremo de servicio" utilice el mismo URL pero sin "?wsdl" esta vez ("http://zoocs.cloudapp.net/zoo.svc"). Seleccione la opción "Conectar con identidad personalizada suplantada" en la sección de "Configuración de autenticación del servicio WCF", y en la casilla de "Id. de aplicación de almacenamiento seguro" utilice el valor creado en el punto 32. No modifique las opciones restantes. "Aceptar". SharePoint Designer hace contacto con el servicio en Azure y muestra los métodos del servicio WCF:

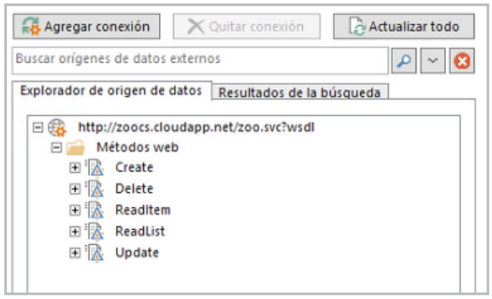


Imagen 12.- Conexión con el servicio WCF desde el BDC.

40.– La conexión con el servicio WCF está asegurada, pero los métodos no están mapeados a los que utiliza el BDC. Seleccione "ReadList" en el panel del BDC y utilizando el menú contextual seleccione "Nueva

operación Leer lista". Utilice el botón de "Siguiente" - "Siguiente" en la ventana que aparece y en la ventana de "Parámetro de devolución" seleccione "AnimalID" en el panel central y "Asignar a identificador" en el panel de "Propiedades" - "Finalizar".

41.– Seleccione "ReadItem" en el panel del BDC y utilizando el menú contextual seleccione "Nueva operación Leer elemento". Utilice el botón de "Siguiente" en la ventana que aparece y en la ventana de "Parámetro de entrada" seleccione "AnimalID" en el panel central y "Asignar a identificador" en el panel de "Propiedades" - "Siguiente". En la ventana de "Parámetro de devolución" seleccione "AnimalID" en el panel central y "Asignar a identificador" en el panel de "Propiedades" - "Finalizar".

42.– Seleccione "Create" en el panel del BDC y utilizando el menú contextual seleccione "Nueva operación Crear". Utilice el botón de "Siguiente" en la ventana que aparece y en la ventana de "Parámetro de entrada" seleccione "AnimalID" en el panel central y "Asignar a identificador" en el panel de "Propiedades" - "Siguiente" - "Finalizar".

43.– Seleccione "Delete" en el panel del BDC y utilizando el menú contextual seleccione "Nueva operación Eliminar". Utilice el botón de "Siguiente" en la ventana que aparece y en la ventana de "Parámetro de entrada" seleccione "AnimalID" en el panel central y "Asignar a identificador" en el panel de "Propiedades" - "Finalizar".

44.– Seleccione "Update" en el panel del BDC y utilizando el menú contextual seleccione "Nueva operación Actualizar". Utilice el botón de "Siguiente" en la ventana que aparece y en la ventana de "Parámetro de entrada" seleccione "AnimalID" en el panel central y "Asignar a identificador" en el panel de "Propiedades" - "Finalizar".

45.– Guarde la configuración del BDC (icono con un disco en la esquina superior izquierda).

46.– Utilice el botón de "Crear lista y formulario" en la cinta. En la nueva ventana que aparece asígnele un nombre a la Lista (campo "Nombre de lista") y "Aceptar".

47.– Regrese al sitio de SharePoint (punto 36), vaya a "Contenidos del sitio" y haga clic sobre la nueva Lista creada. SharePoint muestra la Lista con los datos del servicio WCF en Azure. Todas las operaciones CRUD están disponibles desde SharePoint.

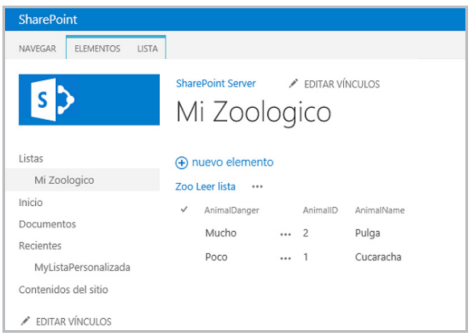


Imagen 13.- El servicio WCF en SharePoint.

Convertir el servicio WCF a REST

Aunque un servicio SOAP se puede utilizar con una multitud de tecnologías, es mucho más fácil de programar y mucho más compatible con otros sistemas si dispone de una interface REST (Representational State Transfer). El servicio SOAP se puede extender para que soporte las dos interfaces, permitiendo que un programador lo utilice en la forma que sea más apropiada.

Para hacer el servicio WCF compatible con REST es necesario crear dos modificaciones en el código fuente: decorar el contrato de la entidad, y modificar el web.config para que el servicio tenga doble punto de entrada.

48.– Abra el archivo "IZoo.cs" del proyecto "ZooWebRole" en Visual Studio. Decore cada uno de los métodos en el contrato de servicio con una propiedad "WebGet" o "WebInvoke" como indica el siguiente fragmento. Cada propiedad indica el método a utilizar ("WebGet" siempre produce un método "GET"), la ruta para acceder al método usando el URL de REST ("UriTemplate") y la forma en la que los resultados serán enviados al cliente ("ResponseFormat"), JSON en el caso del ejemplo.

```
public interface IZoo
 {
    [WebGet(UriTemplate = "/ReadList", ResponseFormat =
WebMessageFormat.Json)]
    [OperationContract]
    IEnumerable<Animal> ReadList();

    [WebGet(UriTemplate = "/ReadItem?AnimalID={animalID}",
ResponseFormat = WebMessageFormat.Json)]
    [OperationContract]
    Animal ReadItem(int animalID);

    [WebInvoke(Method = "POST", UriTemplate = "/Create",
ResponseFormat = WebMessageFormat.Json)]
    [OperationContract]
    Animal Create(Animal newAnimal);

    [WebInvoke(Method = "PUT", UriTemplate = "/Update",
ResponseFormat = WebMessageFormat.Json)]
    [OperationContract]
    void Update(Animal animal);

    [WebInvoke(Method = "DELETE", UriTemplate = "/
Delete?AnimalID={AnimalID}", ResponseFormat =
WebMessageFormat.Json)]
    [OperationContract]
    void Delete(int animalID);
```

49.– Abra el archivo "web.config" del proyecto "ZooWebRole" (asegúrese de que es el archivo web.config, no uno de los sub-archivos "Web.Debug.config" o "Web.Release.config". Reemplace toda la sección "system.serviceModel" con la que se indica a continuación. En esta configuración se indican dos "endpoint": uno para REST y otro para SOAP. El primero indica que la ruta para accederlo será "restservice" (parámetro "address") y el segundo sigue utilizando la ruta por defecto, para no afectar el BDC configurado anteriormente. Note también la diferencia en el "binding" para cada uno de los puntos de entrada.

```
<system.serviceModel>
  <behaviors>
   <serviceBehaviors>
    <behavior name =”servicebehavior”>
     <serviceMetadata httpGetEnabled=”true”/>
     <serviceDebug includeExceptionDetailInFaults=”true”/>
    </behavior>
   </serviceBehaviors>
   <endpointBehaviors>
    <behavior name=”restbehavior”>
     <webHttp />
    </behavior>
   </endpointBehaviors>
  </behaviors>
  <services>
   <service name=”ZooWebRole.Zoo”
behaviorConfiguration=”servicebehavior” >
    <endpoint name=”RESTEndPoint” contract=”ZooWebRole.
IZoo” binding=”webHttpBinding” address=”restservice”
behaviorConfiguration=”restbehavior”/>
    <endpoint name=”SOAPEndPoint” contract=”ZooWebRole.
IZoo” binding=”basicHttpBinding” address=”” />
   </service>
  </services>
  <serviceHostingEnvironment
multipleSiteBindingsEnabled=”true” />
 </system.serviceModel>
```

50.– Agregue el código indicado a continuación al final de la sección “system.webServer” (justo entre las líneas “<directoryBrowse enabled=”true”/>” y “</system.webServer>”. Esta configuración permite reducir los problemas con CORS, como se indicará al final del artículo:

```
  <httpProtocol>
   <customHeaders>
    <add name=”Access-Control-Allow-Origin” value=”*” />
    <add name=”Access-Control-Allow-Headers” value=”Origin,
X-Requested-With, Content-Type, Accept” />
    <add name=”Access-Control-Allow-Methods” value=”GET,
POST, PUT, DELETE, OPTIONS” />
   </customHeaders>
  </httpProtocol>
```

51.– Despliegue la solución de nuevo tal como se indicó en el punto 24. Si la ventana de publicación muestra un icono de error en la suscripción de Azure, utilice el botón de “Previous” dos veces y “Reenter your credentials”; luego de logeado en Azure puede publicar de nuevo. Después de un par de segundo, aparece una ventana preguntando si desea reemplazar el despliegue que ya existe (“Replace”).

52.– Para testear el servicio REST se puede utilizar un navegador yendo al URL “http://[NombreCloudService].cloudapp.net/[NombreServicio].svc/[DireccionEndPoint]/[Metodo]” , para el ejemplo seria “http://zoocs.cloudapp.net/zoo.svc/restservice/ReadList” para utilizar el método ReadList. Los resultados aparecen en formato JSON, tal como se indicó en el contrato:

Imagen 14.- El servicio WCF con su interface REST en acción.

Un navegador solamente da acceso a métodos GET básicos (de hecho, solamente al método ReadList). Para testear todos los métodos se pueden utilizar otro tipo de herramientas que interceptan el tráfico y pueden inyectar parámetros, como por ejemplo Fiddler.

Utilización del servicio REST en una aplicación SharePoint Hosted

Cuando se desea conectar una aplicación SharePoint Hosted con un servicio WCF es indispensable que el servicio tenga una interface REST para poder utilizarlo por medio de JavaScript. El siguiente ejemplo crea una aplicación SharePoint Hosted y la conecta con el servicio WCF REST desarrollado en la sección anterior.

53.– Inicie Visual Studio y cree una nueva solución del tipo “Office/SharePoint” - “Aplicaciones” - “Aplicación para SharePoint”. Asígnele un nombre.

54.– En la ventana de “Nueva aplicación para SharePoint” configure el URL del sitio de desarrollador de SharePoint y seleccione “Hospedado por SharePoint”.

55.– En la siguiente ventana del asistente de creación seleccione “SharePoint2013” (aunque la aplicación puede funcionar, y se programa, de forma exactamente igual que para “SharePoint Online”).

56.– Cuando el proyecto termina de cargar, en el archivo “Default.aspx” reemplace todo el código en la sección bajo el placeholder “PlaceHolderMain” con el código siguiente, que simplemente crea cinco botones que llaman funciones respectivas:

```
  <div>
   <p>
    <input id=”btnReadList” type=”button” value=”Read List”
onclick=”btnReadList_Click()” />
    <input id=”btnReadItem” type=”button” value=”Read
Item” onclick=”btnReadItem_Click()” />
    <input id=”btnCreate” type=”button” value=”Create”
onclick=”btnCreate_Click()” />
    <input id=”btnUpdate” type=”button” value=”Update”
onclick=”btnUpdate_Click()” />
    <input id=”btnDelete” type=”button” value=”Delete”
onclick=”btnDelete_Click()” />
   </p>
  </div>
```

se pueden crear servicios WCF con interfaces SOAP y/o REST que pueden ser consumidos directamente por el BCS de SharePoint

57.– Abra el archivo “App.js” y reemplace todo el código por el siguiente fragmento. Aquí se define una función general de JavaScript llamada “CallService” que recibe como parámetros el método HTML a utilizar, el URL del servicio con el método a utilizar y el tipo de contenido. Funciones individuales para cada uno de los métodos del servicio WCF simplemente

llaman la función general con los parámetros correctos.

```
'use strict';

var baseUrl = "http://[NombreCloudService].cloudapp.net/zoo.svc/Service1.svc/restservice/";

function btnReadList_Click() {
  CallService("GET", baseUrl + "ReadList", "application/xml", "");
}

function btnReadItem_Click() {
  CallService("GET", baseUrl + "ReadItem?AnimalID=1", "application/xml", "");
}

function btnCreate_Click() {
  CallService("POST", baseUrl + "Create", "application/json; charset=utf-8", '{"AnimalID": 4, "AnimalName": "Elefante", "AnimalDanger": "Muy Poco"}');
}

function btnUpdate_Click() {
  CallService("PUT", baseUrl + "Update", "application/json; charset=utf-8", '{"AnimalID": 4, "AnimalName": "Elefantico", "AnimalDanger": "Excesivamente"}');
}

function btnDelete_Click() {
  CallService("DELETE", baseUrl + "Delete?AnimalID=4", "application/xml", "");
}

function CallService(csType, csUrl, csContentType, csData) {
  $.ajax(
  {
    type: csType,
    url: csUrl,
    contentType: csContentType,
    data: csData,
    success: CallbackSuccess,
    error: CallbackError
  });

  function CallbackSuccess(response) { alert("Bingo! --> " + response.text); }
  function CallbackError(err) { alert("Nopes! --> " + err.statusText); }
}
```

58.– Compile y despliegue la aplicación en SharePoint. Si se utiliza IE como navegador, todos los botones darán un error. Esto ocurre porque CORS (Cross-Origin Resource Sharing) impide que el programa que el navegador está ejecutando haga consultas a un servicio en un dominio diferente al de la llamada original. Aunque el código en la configuración del servicio (punto 51) alivia la situación un poco, ningún navegador moderno permite CORS por defecto.
Para hacer que IE acepte CORS, abra sus "Opciones de Internet" y la pestaña de "Seguridad". Utilice el botón de "Nivel personalizado" y seleccione la opción "Habilitar" de "Tener acceso a origen de datos entre dominios". Desde este momento la aplicación funciona sin problemas. Todos los navegadores modernos tienen la posibilidad de configurar CORS (menos Microsoft Edge hasta el momento de escribir este artículo).

Cuando se utiliza una aplicación de SharePoint Provider Hosted las posibilidades de programación son mucho más amplias. Por ejemplo, utilizando ASP.NET en una aplicación SharePoint Provider Hosted, el código de la interface en una página aspx podría ser de la forma siguiente para producir exactamente el mismo efecto que el del ejemplo con JavaScript (y sin problemas de CORS):

```
<body>
  <form id="form1" runat="server">
  <div>
    <asp:Button ID="btnReadList" runat="server" Text="Read List" OnClick="btnReadList_Click" />  
    <asp:Label ID="lblReadList" runat="server" Text="Result"></asp:Label>
  </div><br />
  <div>
    <asp:Button ID="btnReadItem" runat="server" Text="Read Item" OnClick="btnReadItem_Click" />  
    <asp:Label ID="lblReadItem" runat="server" Text="Result"></asp:Label>
  </div><br />
  <div>
    <asp:Button ID="btnCreate" runat="server" Text="Create" OnClick="btnCreate_Click" />  
    <asp:Label ID="lblCreate" runat="server" Text="Result"></asp:Label>
  </div><br />
  <div>
    <asp:Button ID="btnUpdate" runat="server" Text="Update" OnClick="btnUpdate_Click" />  
    <asp:Label ID="lblUpdate" runat="server" Text="Result"></asp:Label>
  </div><br />
  <div>
    <asp:Button ID="btnDelete" runat="server" Text="Delete" OnClick="btnDelete_Click" />  
    <asp:Label ID="lblDelete" runat="server" Text="Result"></asp:Label>
  </div>
  </form>
</body>
```

Y el código behind de la página tal como el del siguiente fragmento:

```
namespace ZooWebApp
{
  public partial class myZooASPNET : System.Web.UI.Page
  {
    string baseUrl = "http://[NombreCloudService].cloudapp.net/zoo.svc/restservice/";

    protected void btnReadList_Click(object sender, EventArgs e)
    {
      string url = baseUrl + "ReadList";

      var syncClient = new WebClient();
      var content = syncClient.DownloadString(url);

      List<Animal> readAnimals = (List<Animal>)JsonConvert.DeserializeObject(content, typeof(List<Animal>));

      lblReadList.Text = "Done";
    }

    protected void btnReadItem_Click(object sender, EventArgs e)
    {
      string url = baseUrl + "ReadItem?AnimalID=1";

      var syncClient = new WebClient();
      var content = syncClient.DownloadString(url);

      Animal readAnimal = (Animal)JsonConvert.DeserializeObject(content, typeof(Animal));

      lblReadItem.Text = "Done";
    }

    protected void btnCreate_Click(object sender, EventArgs e)
    {
      string url = baseUrl + "Create";

      Animal myAnimal = new Animal { AnimalID = 3, AnimalName = "Elefante", AnimalDanger = "Muy poco" };
```

```
        string jsonAnimal = JsonConvert.SerializeObject(myAnimal);

        var syncClient = new WebClient();
        syncClient.Headers[HttpRequestHeader.ContentType] =
"application/json";
        syncClient.Encoding = Encoding.UTF8;
        var content = syncClient.UploadString(url, "POST",
jsonAnimal);

        lblCreate.Text = "Done";
    }

    protected void btnUpdate_Click(object sender, EventArgs e)
    {
        string url = baseUrl + "Update";

        Animal myAnimal = new Animal { AnimalID = 3, AnimalName =
"Elefantico", AnimalDanger = "Muy peligroso" };
        string jsonAnimal = JsonConvert.SerializeObject(myAnimal);

        var syncClient = new WebClient();
        syncClient.Headers[HttpRequestHeader.ContentType] =
"application/json";
        syncClient.Encoding = Encoding.UTF8;
        var content = syncClient.UploadString(url, "PUT", jsonAnimal);

        lblUpdate.Text = "Done";
    }

    protected void btnDelete_Click(object sender, EventArgs e)
    {
        string url = baseUrl + "Delete?AnimalID=3";
        var syncClient = new WebClient();
        var content = syncClient.UploadString(url, "DELETE", "");

        lblDelete.Text = "Done";
    }
  }

  public class Animal
  {
    public Int32 AnimalID { get; set; }
    public string AnimalName { get; set; }
    public string AnimalDanger { get; set; }
  }
}
```

En el código se define una clase con la entidad de la misma forma que está definida en el servicio WCF. Luego, para los dos métodos GET, cada uno de los eventos de los botones llama el servicio utilizando el método "DownloadString" de la clase "WebClient" de System.Net y deserializa la respuesta JSON utilizando el método "DeserializeObject" de JsonConvert. Los métodos PUT, POST y DELETE crean primero un objeto del tipo Animal con los valores apropiados, lo serializa a JSON utilizando el método "SerializeObject" de JsonConvert y lo envía al servicio WCF utilizando el método "UploadString" de la clase "WebClient".

Conclusiones

Dentro de las múltiples maneras de utilizar los servicios de Azure para complementar el funcionamiento de SharePoint, Cloud Services permite crear servicios Web WCF que sirven como la puerta de entrada a cualquier tipo de sistema externo.

Con Azure Cloud Services se pueden crear servicios WCF con interfaces SOAP y/o REST que pueden ser consumidos directamente por el BCS de SharePoint (y su motor de búsqueda), o por Aplicaciones de SharePoint, Provider o SharePoint Hosted. Por medio de la interface REST se pueden crear aplicaciones SharePoint Hosted o Provider Hosted que consuman los datos ofrecidos por el servicio WCF.

CORS puede constituirse en un problema cuando se utiliza JavaScript con la interface REST del servicio, pero este es un problema general de JavaScript y los navegadores modernos, no un problema de SharePoint o Azure Cloud Services.

GUSTAVO VELEZ
Office Severs and Services MVP
gustavo@gavd.net
http://www.gavd.net

15

Personalización de un Flujo de Trabajo de Aprobación con SharePoint Designer 2013

Los flujos de trabajo que ofrece por defecto la plataforma SharePoint, son un mecanismo para que rápidamente las organizaciones implementen procesos semi-automatizados que permitan agilizar algunas tareas como: aprobación, publicación y recolección de firmas, que permiten ahorro de tiempo en su implementación. Pero estos beneficios también traen consigo algunas limitaciones importantes, las cuales se revisarán en éste artículo, con el objetivo de mejorar la funcionalidad ofrecida por defecto en un flujo de aprobación.

Es importante revisar algunos artículos anteriores que se han escrito en esta revista respecto a este tema, los cuales se han enfocado en personalizar flujos de trabajo desde cero con SharePoint Designer 2013:

- ¿Cómo hacer un Flujo de Trabajo de aprobación con SharePoint Designer 2013?
- SharePoint Designer 2013 y Flujos de Trabajo

En este artículo nos enfocaremos en la personalización de un flujo de trabajo de aprobación a partir de la plantilla que ofrece la plataforma por defecto, con el objetivo de mejorar algunas de sus funcionalidades.

Para el desarrollo del ejemplo haremos uso de un sitio de SharePoint Online, SharePoint Designer 2013 e InfoPath 2013.

Limitaciones de un Flujo de Aprobación

A continuación, veremos algunas de las limitaciones presentadas por un flujo de aprobación que está basado en el motor de flujos de trabajo de SharePoint 2010:

- Quizá la primera limitante y la más evidente es que el flujo de trabajo bajo el cual iniciaremos las tareas de personalización corre bajo el motor de flujos de trabajo de SharePoint 2010. Esto es un indicador importante, que nos limita a capacidades que ya conocemos y que seguramente hereda los inconvenientes que tuvimos en 2010. Sería muy importante que Microsoft provea mejoras sustanciales sobre estos flujos y que tengamos acceso a plantillas actualizadas y unificadas.
- Un inconveniente importante heredado del primer punto, es que en el momento de encarar soluciones con cierta complejidad, nos vamos a dar cuenta que los flujos de trabajo que corren con el motor de 2010, no permiten o no cuentan con algunas de las acciones importantes que han sido incluidas en 2013. Esto se convierte en un problema, debido a que estas soluciones con cierto grado de complejidad, comienzan a verse afectadas y generan una mezcla de flujos que pueden ser implementados bajo una plataforma u otra. Por ejemplo, un flujo reusable asociado un tipo de contenido específico solo puede ser implementado utilizando el motor de 2010, esto debido a que no es posible seleccionar el tipo de contenido cuando se escoge la plataforma 2013, tal como se muestra en la Imagen 1.

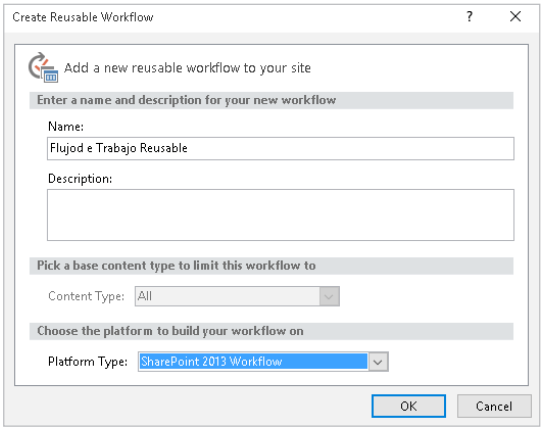


Imagen 1.- El Flujo reusable con la plataforma 2013 no permite seleccionar el Tipo de Contenido.

Son un mecanismo para que rápidamente las organizaciones implementen procesos semi-automatizados

- Esta limitante está relacionada con los dos puntos anteriores, y es que cuando tenemos un flujo de trabajo implementado con el motor de 2013, el cual ha agregado otra funcionalidad bastante importante, y es la acción que permite la invocación de otros flujos dentro del flujo principal, estas acciones no pueden ser agregadas dentro de un App Step. Un App Step es similar a un Impersonation Step, en el sentido que permite ejecutar las acciones que contiene, con permisos elevados. Supongamos que el flujo secundario que se invoca, ha sido diseñado para asignar permisos a un ítem de una lista a unos usuarios determinados. Como ya se explicó es posible que el usuario que está ejecutando el proceso no tenga los niveles de permisos

necesarios para ejecutar las acciones. Por esa razón se hace uso del App Step, pero encontraremos que la acción de invocar un flujo secundario no está permitida dentro de este paso. Esto se puede apreciar en la Imagen 2.

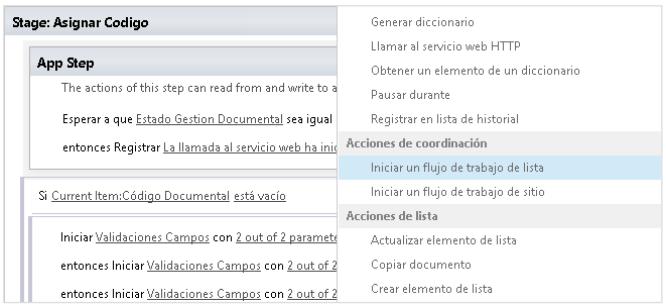


Imagen 2.- Las Acciones de coordinación para iniciar flujos de trabajo externos no están disponibles en el App Step.

- La plantilla de aprobación de los WorkFlows reusables basados en el motor de 2010, solo están basados en 2 acciones básicas: Aprobar o Rechazar. Estas acciones son devueltas a través de un mecanismo conocido como Task Outcomes. A través de SharePoint Designer 2013 pueden ser personalizadas, permitiendo cambiarles el título al botón del formulario asociado, y el orden de los botones. Cuando se está haciendo uso de estas funcionalidades se quisiera tener un mayor control, pero la realidad es que son acciones que encapsulan operaciones basadas casi en un concepto de caja negra. Si se requiere agregar más Task Outcomes, es posible, pero estarán regidas por la naturaleza del mensaje devuelto, es decir podríamos agregar un botón adicional que se llame Revisar, y lo que se hace en el flujo es interceptar ese mensaje por el nombre que se le ha asignado en los Task Outcomes, y finalmente agregar condiciones y acciones bajo ese valor devuelto. En la Imagen 3 se aprecia el nuevo Task Outcomes. Cabe anotar que para poder personalizar el formulario que se asocia con estos botones generados por las Task Outcomes, es necesario tener instalado InfoPath 2013. Como ya se comentó no hay mucho donde se pueda intervenir en la funcionalidad que ya tienen los botones por defecto, y por experiencia, hemos tenido problemas agregando o quitando estos botones, ya que los cambios no se ven reflejados, o el flujo trabaja de manera inconsistente. Por esa razón es bueno que se tengan copias del flujo antes de proceder con cambios de esta naturaleza. También debido a la necesidad de InfoPath 2013, el proceso de cambios y publicación del formulario puede volverse un poco tedioso cuando estamos realizando soluciones con cierto grado de complejidad y donde el tiempo es importante.

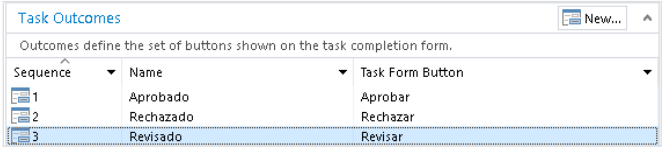


Imagen 3.- Agregar una nuevo Task Outcomes con el nombre Revisado.

Luego que el flujo es publicado se genera el formulario de InfoPath, el cual puede personalizarse a través de InfoPath 2013 al hacer clic sobre el nombre del archivo. En la imagen 4 se puede ver este procedimiento

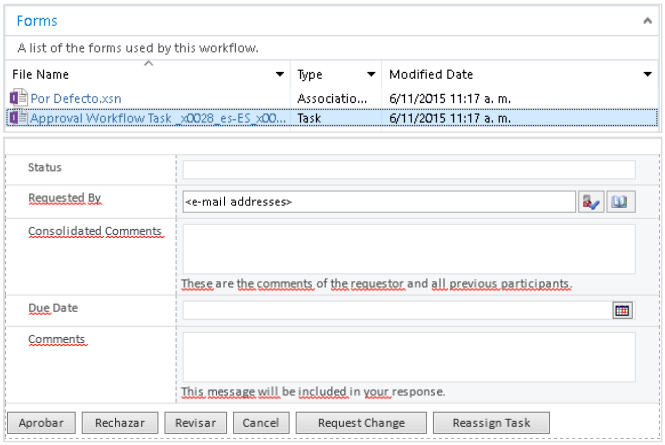


Imagen 4.- Personalizar el Formulario de Aprobación con InfoPath 2013.

En la Imagen 4 se puede observar que se ha agregado un formulario con el nombre Approval Workflow Task … No hay demasiado control en el momento de asignarle un nombre más amigable al archivo, el cual ha sido generado automáticamente, luego de publicar el flujo de trabajo. También se aprecia el botón Revisar que se agregó en la sección de Task Outcomes. Los demás botones presentes, son agregados por defecto por la plantilla del flujo.

- Una limitante notable en el formulario anterior es que los botones están disponibles para todos los participantes del flujo. Eso en la práctica es algo no deseado, porque nos gustaría que solo el Aprobador apruebe o rechace, pero no tenga la posibilidad de Revisar, que esperaríamos sea la tarea de otro actor, y así en el caso contrario. En este momento a través de reglas de InfoPath es donde debemos agregar personalizaciones al formulario y ver el modo de mostrar los botones según corresponda al actor indicado.
- Otro punto importante es el idioma. Como se observa en la imagen anterior algunas etiquetas están en inglés. Supongamos que nuestro cliente quiere todo en español, esto sería un problema, por lo que es necesario desde InfoPath 2013 modificar los textos de estos controles. No parece un problema muy importante, pero es bueno tenerlo en cuenta, porque estos detalles afectan en ocasiones las entregas a nuestros clientes.
- Más que una limitante, hacemos una advertencia. En caso de querer eliminar cualquiera de los botones o los campos presentes en el formulario, se debe tener cuidado porque todos esos elementos vienen fuertemente relacionados con el flujo y su lista de tareas. Así que la recomendación es buscar mecanismos desde InfoPath que permitan ocultar esos elementos en lugar de eliminarlos. Por ejemplo agregar un control de InfoPath que contenga a todos los controles que no queremos mostrar y a través de reglas del control principal ocultarlo.

Pasos iniciales Para crear el Flujo

de Trabajo

Hasta este momento hemos querido mostrar algunas limitaciones importantes, de los elementos esenciales del flujo de trabajo reusable ofrecido por la plataforma a través del motor de 2010. Pero es momento de mostrar un poco más el detalle de los pasos que debemos seguir para poder crear y personalizar un nuevo flujo basado en la plantilla que ya viene por defecto, y buscar la manera de mejorar las limitaciones que se han presentado.

- Lo primero que debemos hacer es abrir el sitio de SharePoint Online en SharePoint Designer 2013. Lo recomendable es realizar estas tareas con permisos de Propietario del Sitio o de Administrador de la Colección de Sitios.
- Cuando el sitio ha cargado en SPD 2013 podremos ver la opción WorkFlows, tal como se aprecia en la Imagen 5.

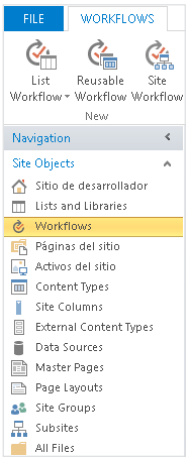


Imagen 5.- La sección WorkFlows agrupa y permite gestionar los WorkFlows creados en el sitio.

- Al hacer clic en la opción WorkFlows se despliega una ventana con los WorkFlows disponibles por defecto y los que se hayan creado de ceros en el sitio. En este caso nos vamos a ubicar en el grupo llamado Globally Reusable WorkFlow.
- En ese grupo vamos a ubicar y seleccionar el flujo llamado Aprobación - SharePoint 2010, como se muestra en la Imagen 6.

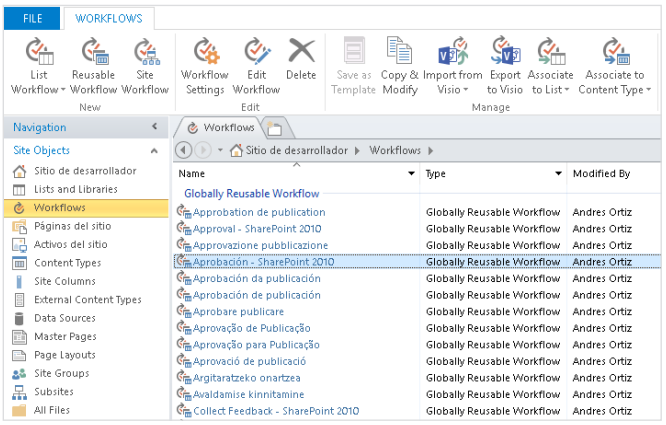


Imagen 6.- WorkFlow de Aprobación bajo la plataforma 2010.

- Podemos observar la opción en la parte superior llamada Copy & Modify. Al hacer clic en ese botón se despliega una ventana que permitirá crear un nuevo flujo de trabajo basado en dicha plantilla.

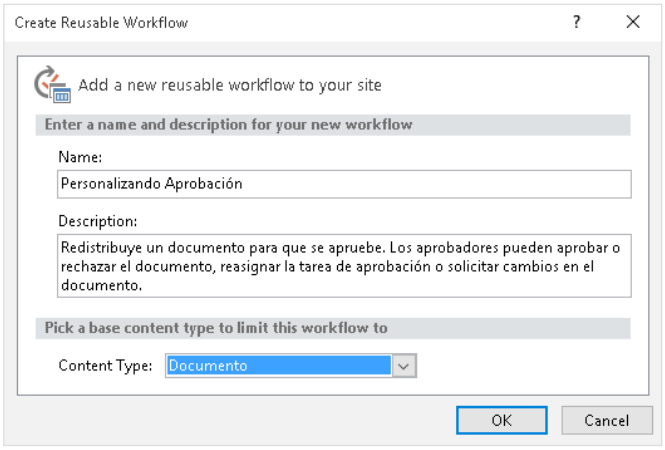


Imagen 7.- Creación de un nuevo WorkFlow reusable.

Agregamos un nombre al nuevo flujo y podemos seleccionar un tipo de contenido para asociarlo al flujo de trabajo. Esta característica basada en el motor de flujos de 2010, es la que permite que estos flujos puedan ser reusados en las listas y/o bibliotecas que hereden del tipo de contenido seleccionado.

- Al hacer clic en el botón OK se creará el nuevo flujo y se abrirá la ventana de edición que nos permite comenzar a realizar los cambios.

  Al ser esta una plantilla basada en el motor de 2010, se puede notar que el diseñador de flujos es el mismo que ya conocíamos de la versión 2010, basado en pasos, y con la opción Impersonation Step, que permite agregar un paso donde se pueden agregar acciones que se ejecuten con permisos elevados.

- En el paso que se ha agregado por defecto, observamos una acción fundamental, la cual será el eje de la personalización que se quiere mostrar en este artículo, y es la que se llama Iniciar el Proceso.

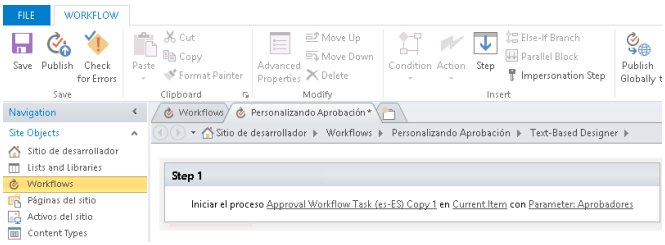


Imagen 8.- Paso creado por defecto del nuevo WorkFlow.

Es importante en este momento hacer clic en el botón Save, para que el flujo se guarde. Si se cierra las ventanas sin guardar, el flujo no quedará registrado en el listado de flujos de la plataforma.

De los WorkFlows reusables basados en el motor de 2010, solo están basados en 2 acciones básicas: Aprobar o Rechazar

- Una de las cosas importantes de este tipo de flujos, es que algunos elementos están parametrizados. Por ejemplo en la Imagen 8 se puede apreciar el parámetro Aprobadores. En ese parámetro se deben definir las cuentas de los usuarios que harán parte del proceso de aprobación. Al hacer clic en ese parámetro, se abre una ventana donde se pueden configurar los participantes, bien sea utilizando el parámetro, seleccionando los usuarios determinados o buscándolos a partir de otros elementos del WorkFlow. También se puede seleccionar si la asignación de tareas se hace uno por uno o de manera paralela, es decir se asignan las tareas a todos los participantes.

Personalización del Flujo de Trabajo

Luego de crear el flujo de trabajo y tener claro los elementos que se han generado por defecto, podemos comenzar con la personalización del flujo. Es importante validar algunos elementos en la página de resumen de opciones del flujo, como se muestra en la Imagen 9.

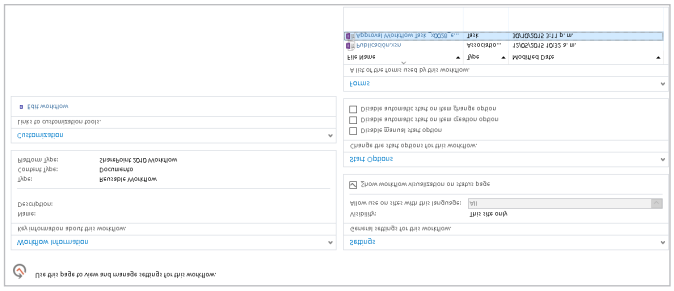


Imagen 9.- Información general del WorkFlow.

- En la Imagen 9 se puede visualizar el nombre y descripción dados al nuevo flujo.
- Se muestran el tipo, tipo de contenido y la plataforma en la cual se ha creado el flujo.
- Las opciones de Inicio ninguna ha sido marcado lo cual significa lo siguiente: El flujo puede ser iniciado manualmente y automáticamente cuando se crea o actualiza un elemento.
- Los formularios de InfoPath creados luego de publicar el flujo
- Un enlace para iniciar la Edición del WorkFlow. En este momento iniciaremos la personalización de la acción Iniciar el proceso.
- Además de los Task Outcomes, se puede ver también varios enlaces que permitirán personalizar las diferentes fases del proceso de aprobación:

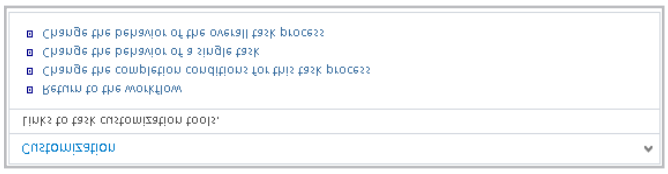


Imagen 10.- Personalización del Proceso de Aprobación.

- Cambiar las condiciones de completitud para el proceso de tarea: En esta sección se encuentran las condiciones y acciones que permiten personalizar el comportamiento del flujo de trabajo cuando entra en su fase de finalización. Y para este ejemplo ha sido una sección personalizada, debido a que por defecto el comportamiento es solo permitir el manejo de 2 estados, aprobado o rechazado. Pero qué sucede cuando hemos agregado nuevos Task Outcomes, lo cual agrega más estados al proceso, como se mostró en la Imagen 4 tenemos el botón Revisar. Si en esta sección no se controla que tenemos un estado más, el WorkFlow simplemente finalizará completamente sin permitir completar todas las tareas asignadas. Por eso esta sección difiere de lo que podrían ver cuando se crea el flujo por primera vez, y con la condición que se ha establecido se garantiza el correcto manejo de más de 2 estados.

Imagen 11.- Personalización para manejar más de 2 estados de aprobación.

- Cambiar el comportamiento de una sola tarea: Cuando hemos entendido completamente un flujo de aprobación de SharePoint 2010, nos damos cuenta que todo está basado en la asignación y terminación de tareas, por parte de cada responsable. Supongamos que lo hemos configurado para que el flujo cuente con 3 participantes de manera serial o uno a la vez. Esto significa que al participante A se le asigna una Tarea 1, y hasta que no la complete el flujo no avanzará y tampoco será asignada la siguiente tarea al Participante B y así sucesivamente, hasta el Participante C y su respectiva tarea. Cuando este último completa su tarea asignada es cuando el flujo finalizará. Cada una de estas tareas por separado es controlada en esta sección, y es aquí donde quizá realizamos la mayor parte de la personalización. En esta sección se cuenta con el mecanismo para personalizar las notificaciones, condiciones y acciones en general que ocurren durante la vida de una tarea. Como se muestra en la Imagen 12, se puede ver que las etapas de la tarea son bastante claras: Antes que la tarea sea asignada, Cuando una tarea está pendiente, Cuando una Tarea Expira, Cuando una Tarea es Eliminada, Cuando una tarea se completa:

Luego que el flujo es publicado se genera el formulario de InfoPath, el cual puede personalizarse a través de InfoPath 2013

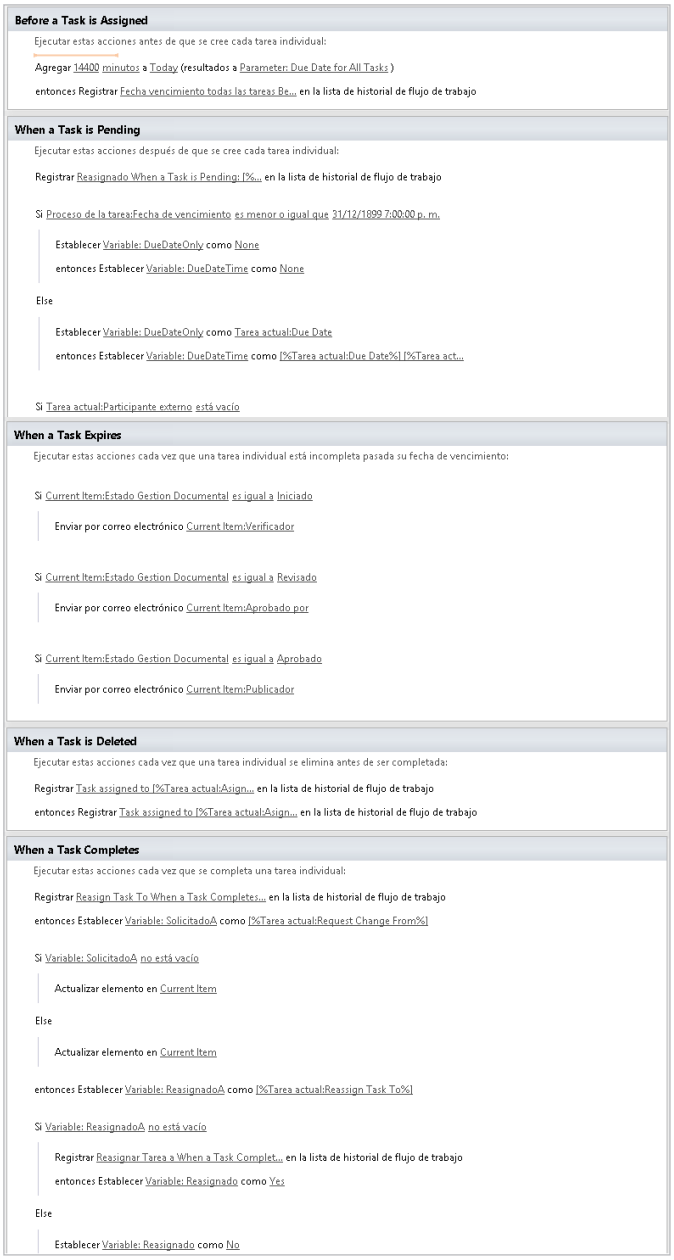


Imagen 12.- Personalización de una tarea específica.

En la sección Before a Task is Assigned, es donde podemos aprovechar para realizar cualquier condición y/o acción, precisamente antes que la tarea sea asignada a un responsable. En el ejemplo se puede visualizar que estamos incrementando en 10 días el periodo de vencimiento de cada tarea, e invocando una acción de registro de un mensaje en el historial del flujo.

En la sección When a Task is pending, igualmente podemos controlar las condiciones y/o acciones que ocurrirán luego de que cada tarea sea asignada a un responsable. Aunque en la imagen 12 no se alcanza a visualizar, en esta sección hemos configurado en su mayoría notificaciones personalizadas. Tal vez uno de los mayores puntos de frustración de los usuarios finales, son los correos que estos flujos envían por defecto, debido a que no son muy claros, podrían estar en inglés, y no reflejan los comportamientos organizacionales. Por esta razón en estas secciones se pueden eliminar esas notificaciones y crear unas propias con mensajes e información que sea relevante para los actores del proceso.

En When a Task completes, igualmente se ejecutan todas las condiciones y/o acciones en el momento que un participante completa la tarea que tenía asignada. Es por eso que es en esta sección donde podemos capturar los mensajes enviados por los botones de Aprobar, Rechazar o Revisar, que habíamos mostrado en la sección de los Task Outcomes, como se muestra en la Imagen 13:

Imagen 13.- Personalización de una tarea completada.

- Cambiar el comportamiento del proceso de tarea completo: A diferencia de la opción anterior, en este caso tenemos acceso a personalizar lo que ocurre con el proceso completo. Por ejemplo podemos agregar acciones o condiciones durante la ejecución del proceso, justo antes que la primera tarea sea asignada. Pero también podremos controlar acciones y condiciones durante la ejecución, es decir en el momento que una tarea cambie o sea eliminada, e incluso cuando sea cancelada.

- Finalmente grabamos el flujo, y lo publicamos para que los cambios se vean reflejados en el tipo de contenido.

Agregar el Flujo de Trabajo

Luego de grabar y publicar el flujo no significa que esté ya está listo para ser utilizado en la biblioteca de documentos. Recordemos que es un flujo de trabajo reusable, es decir que ha sido asociado a un tipo de contenido. En este caso vamos a suponer que nuestra biblioteca ya hereda del tipo de contenido.

- Lo primero que debemos hacer es iniciar sesión en el sitio de SharePoint y navegar hasta la biblioteca donde queremos asociar el flujo de trabajo que hemos publicado.
- En el TAB BIBLIOTECA se encuentra la opción Agregar un flujo de trabajo como se muestra en la Imagen 14.

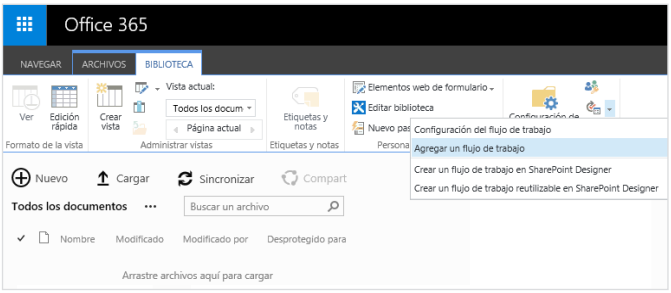


Imagen 14.- Agregar un flujo de trabajo.

- Luego seleccionamos el tipo de contenido, por ejemplo Documento, y debería listarse nuestro flujo de trabajo que hemos publicado previamente como en la Imagen 15.

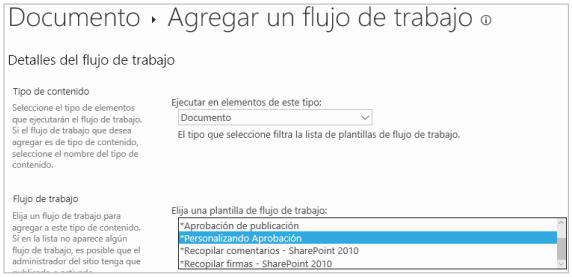


Imagen 15.- Seleccionar el flujo de trabajo.

- Le damos un nombre, asociamos sus listas de tareas e historial, y asignamos las opciones de inicio y hacemos clic en el botón Siguiente.
- En el siguiente formulario se solicitaran diferentes valores, que por defecto toma a través de parámetros que ya tiene preestablecidos. Debemos recordar que estamos implementando un flujo personalizado a partir de la plantilla y tal vez algunos de estos valores ya no serán necesarios, así que podrimos hacer clic en el botón Save y con eso finalizaremos la asociación del flujo con la biblioteca.

Conclusión

Para concluir este artículo podemos decir que le hace falta bastante a la plataforma SharePoint para convertir sus flujos de trabajo por defecto en verdaderos elementos de automatización de procesos. Pero se vienen haciendo mejoras e incorporando funcionalidades que han estado presentes antes en herramientas de terceros. Y muy seguramente hemos y lograremos generar soluciones a nuestros clientes con lo que tenemos a mano, pero como hemos intentado mostrar en este artículo, hay que realizar diferentes personalizaciones y modificaciones, para que entreguemos soluciones realmente útiles y funcionales a nuestros usuarios, así que a medida que pase el tiempo esperamos de igual manera por parte de Microsoft, mejoras sustanciales en el motor de flujos y en las herramientas que nos permiten construir soluciones para tener a la mano más funcionalidades que estén aterrizadas a lo que realmente esperan los clientes, o que se puedan extender o mejorar.

ANDRÉS ORTÍZ
jaortizgonzalez@hotmail.com
@jandresortiz

21

Configurando Azure Directory Services para Office365

El uso de Azure AD Connect con Office 365 proporciona los siguientes beneficios:

- Los usuarios pueden firmar con una identidad común tanto en la nube y local. No necesita recordar múltiples contraseñas o cuentas y administradores no tienen que preocuparse acerca de la sobrecarga adicional que pueden traer múltiples cuentas.
- Una sola herramienta y experiencia guiada para conectar tus directorios locales con Azure Active Directory. Una vez instalado el asistente implementa y configura todos los componentes necesarios para conseguir su integración de directorio y funcionamiento incluyendo servicios de sincronización, sincronización de contraseña o AD FS y requisitos previos como el módulo de AD PowerShell de Azure.

Por qué usar Azure Connect de AD

La integración de un Directorio Activo local con Azure AD facilita que los usuarios sean más productivos proporcionando una identidad común para acceder a los recursos de la nube y local. Con esta integración los usuarios y las organizaciones pueden aprovecharse de las siguientes ventajas:

- Proporcionar a los usuarios con una identidad híbrida a través de locales o servicios en la nube aprovechando Windows Server Active Directory y luego conectar a Active Directory de Azure.
- Los administradores pueden proporcionar acceso condicional basado en recursos de aplicación e identidad de usuario, ubicación de red y autenticación de múltiples factores.
- Los usuarios pueden aprovechar su identidad común a través de cuentas en Azure para Office 365, Intune, aplicaciones SaaS y aplicaciones de terceros.
- Los desarrolladores pueden crear aplicaciones que aprovechan el modelo de identidad común, integración de aplicaciones en Active Directory local o Azure para aplicaciones basadas en cloud.

Los usuarios pueden firmar con una identidad común tanto en la nube y local

Azure Connect facilita esta integración y simplifica la gestión de su local y la nube infraestructura de identidad.

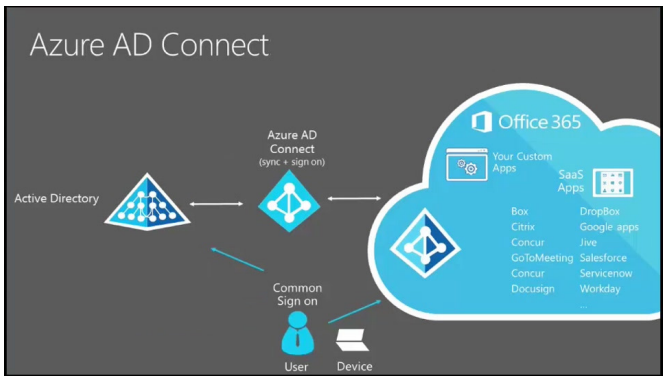


Imagen 1.- Esquema de funcionamiento de Azure AD Connect.

A continuación, se detalla el proceso a seguir para sincronizar un Directorio Activo de Azure AD con Office 365:

1.– Procedemos a ingresar al portal y nos ubicamos en el AD Azure –> pestaña Integración de Directorios.

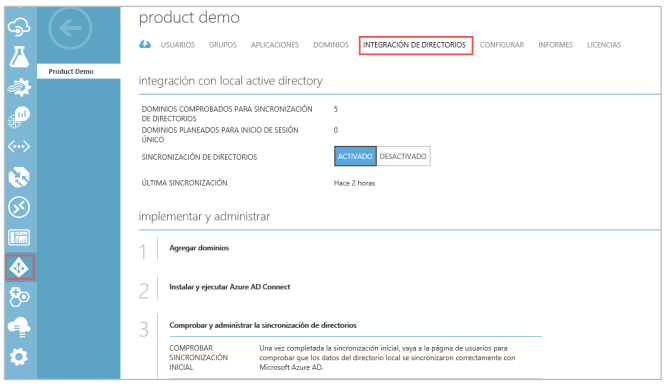


Imagen 2.- Pestaña INTEGRACIÓN DE DIRECTORIOS en Azure AD.

2.– Antes de iniciar el procedimiento de instalación tenga en cuenta lo siguiente:

Es importante verificar por medio del monitor de sincronización si las replicaciones de nuestro Directorio Activo son exitosas

Se debe ingresar con una cuenta que este en AD en mi caso el usuario es Administrador y que además está en el grupo de Enterprise Admins y pertenezca al grupo de administradores del equipo local.

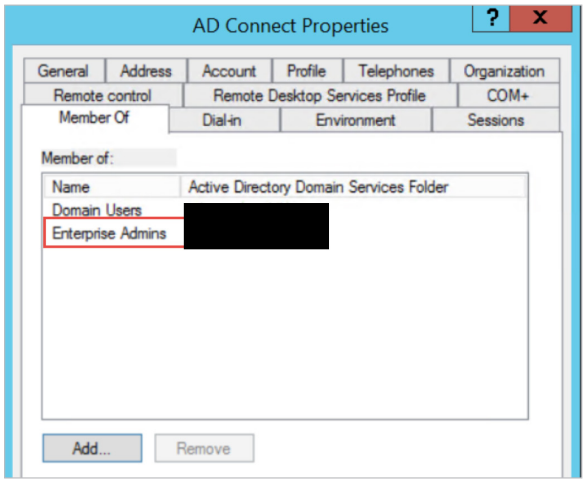


Imagen 3.- Propiedades de la cuenta de usuarios en Active Directory.

3.– Descargar el Azure Connect AD

Para empezar a usar Azure Connect AD puede descargar la versión más reciente utilizando el siguiente: Descargar AD de Azure Connect

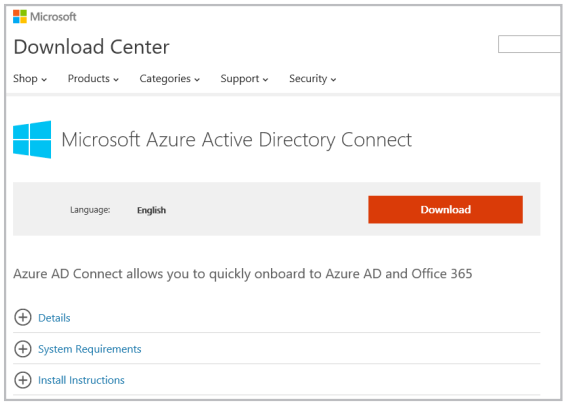


Imagen 4.- Pagina de descarga de Microsoft Azure Directory Connect.

Una vez completado todos los pasos anteriores entramos en la fase de instalación y configuración del servicio Azure AD Connect, para ello aplicamos el siguiente procedimiento:

4.– En el servicio de sincronización de Azure AD Connect realizaremos la configuración rápida en nuestro ambiente teniendo en cuenta las opciones que se realizarán por defecto.

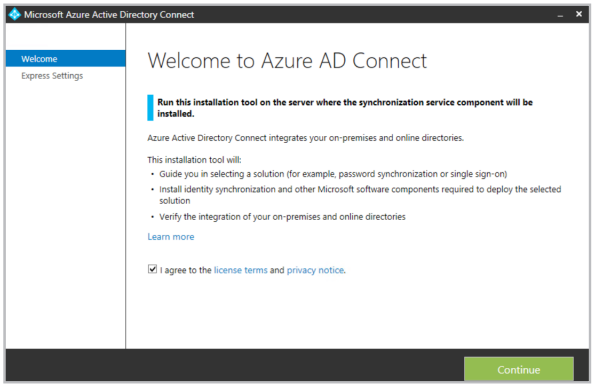


Imagen 5.- Asistente para la configuración de Azure AD Connect.

5.– A continuación, se muestra la pantalla de Express Settings. Hacemos clic en el botón "Use express settings" que nos permite hacer una sincronización rápida.

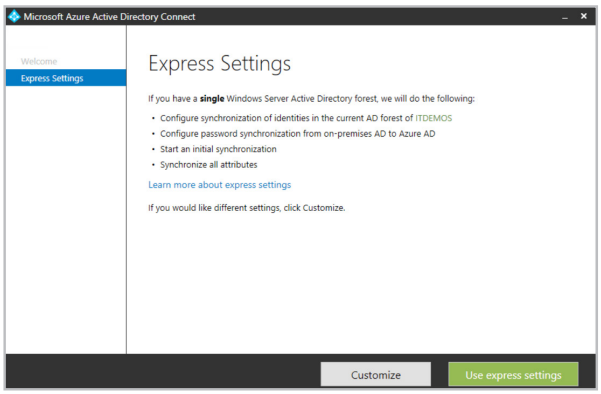


Imagen 6.- Asistente para la configuración de Azure AD Connect.

6.– La siguiente pantalla, "Express Settings" realiza todo el proceso de instalación y configuración que se necesita para poner en marcha Azure AD Connect.

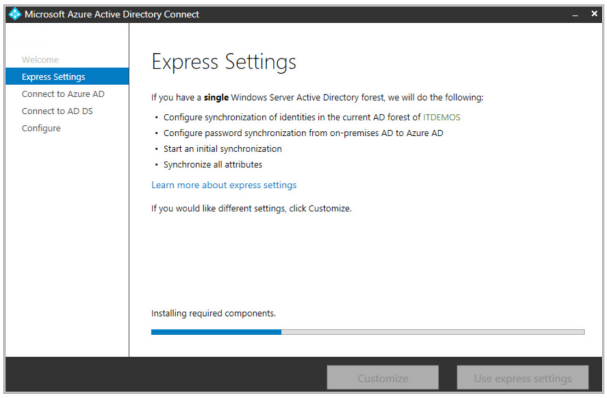


Imagen 7.- Azure AD Coonect – Configuración Express.

7.– Deberemos proporcionar el nombre de usuario de Azure AD que es un miembro de la función de administrador Global y un usuario de Active Directory que está en el grupo de Enterprise Admins.

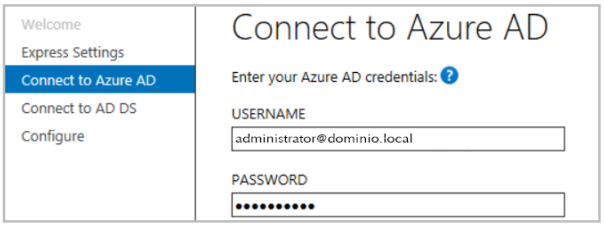


Imagen 8.- Azure AD Connect: Configuración de la cuenta de replicación.

8.– Se debe utilizar la cuenta que pertenece a Domain Admin. Recuerde que en un dominio de Windows los domain admin por defecto son administradores locales en los servidores miembros y estaciones de trabajo del dominio.

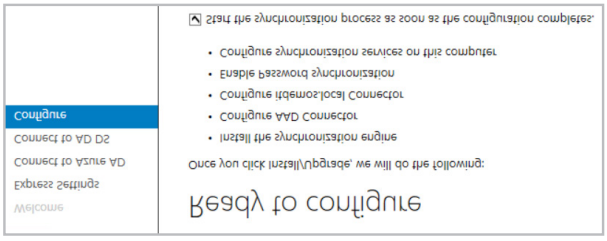


Imagen 9.- Asistente para la configuración de Azure AD Connect.

9.– Finalmente se muestra el resultado de la configuración, que mostrará un mensaje de que todo ha ido bien si la configuración se ha realizado de forma correcta.

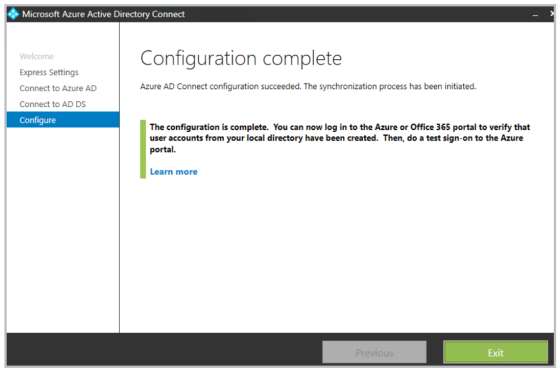


Imagen 10.- Asistente para la configuración de Azure AD Connect, Finalizado con Éxito.

Nota: Al momento de finalizar el asistente ya ha replicado las cuentas según se lo hemos indicado (via express), ahora nos toca verificar si la réplica ha sido exitosa mediante el monitor de replicaciones.

Verificando la configuración:

Para verificar la configuración, seguiremos los siguientes pasos:

1.– Abrimos el servicio de sincronización la herramienta (miisclient.exe) se encuentra en el siguiente directorio: C:\Program Files\Microsoft Azure AD Sync\UIShell\miisclient.exe

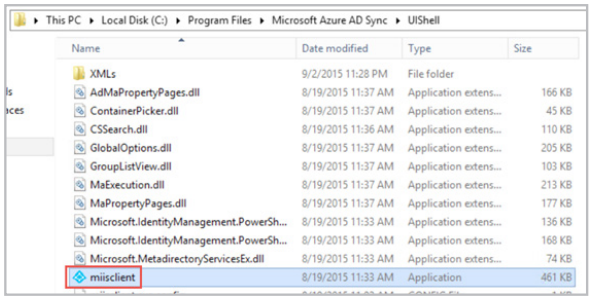


Imagen 11.- Directorio de ficheros de Azure AD Conect,

Nota: Es importante ejecutar el fichero siempre con permisos elevados en el caso que el UAC este activo.

2.– Procedemos abrir y veremos la última sincronización completada.

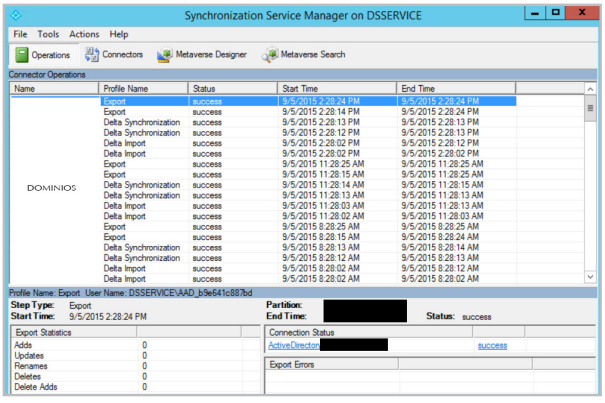


Imagen 12.- Monitor de Sincronización de Azure AD Connect.

3.– Vemos las cuentas que están sincronizadas con Office 365 y el Azure AD.

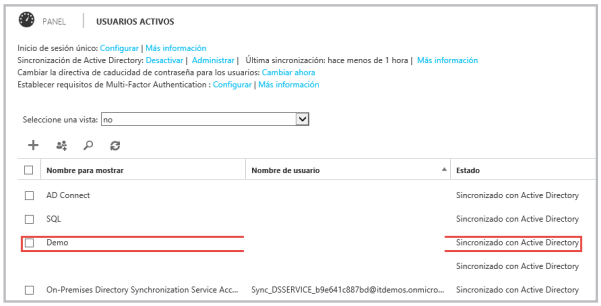


Imagen 13.- Consola de administración de usuarios de Office 365.

Es muy importante verificar que la cuenta con que se esté haciendo la replicación sea Domain Admin

Conclusiones

Es importante verificar por medio del monitor de sincronización si las replicaciones de nuestro Directorio Activo son exitosas constantemente, sobre todo si nuestra empresa ha tenido movimientos de objetos a nivel de directorio (Usuarios/Grupos). Particularmente recomiendo instalar el Azure AD Connect en el controlador de dominio, esto ayudara que las consultas sean más rápidas y nos aseguramos que tenemos todos los privilegios necesarios para hacer dicha replicación, basándonos en teoría básica de Windows Server hago esta recomendación porque solo los administradores pueden iniciar sesión local dentro de un controlador de dominio.

Finalmente, es muy importante verificar que la cuenta con que se esté haciendo la replicación sea Domain Admin, porque solo los miembros de este grupo tienen privilegios para extraer los SIDs de los usuarios, SIDs que copiara Azure desde nuestro directorio activo para lograr hacer el Single SingOn y la replicación del esquema del directorio.

DARWIN CASTRO
MCSE Windows Server 2012 *Charter
darwin.castro@ancadia.com
@ancadiasystems
http://www.ancadia.com

24

Introducción al PnP Program y Provisioning framework

Desde hace cerca ya de dos años, un grupo de Patterns and Practices de Microsoft, liderado por Vesa Juvonen (http://blogs.msdn.com/b/vesku), inicio el programa Office 365 Dev PnP, con la idea principal de ayudar a los clientes a transformar y realizar la transición del modelo Soluciones Full Trust de SharePoint, al modelo de AddIns, tratando de establecer unas guías y buenas prácticas para hacer ese paso.

Componentes del programa PnP

El Office 365 Dev PnP se liberó en GitHub, y poco tiempo después se dividió en distintos repositorios. Estos son algunos de los componentes principales:

- PnP-Sites-Core:
  - https://github.com/OfficeDev/PnP-Sites-Core Es el componente principal del PnP. Se trata de una .dll con diferentes clases y métodos que nos harán la vida más fácil al trabajar con el modelo cliente de SharePoint (CSOM). Además, incluye el framework de Provisioning del que hablaremos en breve.
- PnP-Provisioning-Schema:
  - https://github.com/OfficeDev/PnP-Provisioning-Schema Este proyecto contiene el Schema XML que es utilizado por el framework de provisioning. Este XML nos recordara bastante al XML utilizado en las "antiguas" soluciones SharePoint, y desde él se pueden definir la mayoría de los componentes de una solución de SharePoint: Fields, ContentTypes, Lists, Files, Pages, RegionalSettings, etc.
- PnP-PowerShell:
  - https://github.com/OfficeDev/PnP-PowerShell Power-Shell CommandLets con más utilidades.
- PnP:
  - https://github.com/OfficeDev/PnP Este fue el repositorio original, que posteriormente fue dividido en varios repositorios, entre los que se encuentran los anteriores. Actualmente este repositorio contiene varios ejemplos de uso del PnP y otras buenas prácticas.

Realizar la transición del modelo Soluciones Full Trust de SharePoint, al modelo de AddIns

Ejemplos interesantes

El siguiente listado contiene algunos de los ejemplos más interesantes que podemos encontrar en el PnP, si bien es muy recomendable revisar todos los ejemplos.

- Branding.ApplyBranding: https://github.com/OfficeDev/PnP/tree/master/Samples/Branding.ApplyBranding Muestra buenas practicas a la hora de aplicar branding y otras características de publishing.
- Core.CreateContentTypes: https://github.com/OfficeDev/PnP/tree/master/Samples/Core.CreateContentTypes Permite crear Tipos de Contenido usando el PnP Core.
- Core.EmbedJavaScript: https://github.com/OfficeDev/PnP/tree/master/Samples/Core.EmbedJavaScript Buena práctica a la hora de inyectar JavaScript y actualizar la Interfaz de Usuario.
- Core.MMSSync: https://github.com/OfficeDev/PnP/tree/master/Samples/Core.MMSSync Sincronización de Términos del Almacén de Metadatos a través de múltiples TermStores.
- Provisioning.Cloud.Async.Webjob: https://github.com/OfficeDev/PnP/tree/master/Samples/Provisioning.Cloud.Async.WebJob Creación de Colecciones de Sitios desde un WebJob en Azure.
- UserProfile.Manipulation.CSOM.Console: https://github.com/OfficeDev/PnP/tree/master/Samples/UserProfile.Manipulation.CSOM.Console Actualización de los Perfiles de Usuario mediante el Modelo de Objetos en Cliente (CSOM).

Extensiones y utilidades con CSOM

Algunas de las clases y métodos que extienden CSOM:

- ClientContextExtensions.ExecuteQueryRetry Similar al método ExecuteQuery(), pero gestiona posibles errores de conexión y aplica varios reintentos (10, por defecto, con un retardo de 500 ms).
- FieldAndContentTypeExtensions: Clase con varias extensiones para trabajar con Campos y Tipos de contenido.
- ListExtensions: Extensiones para trabajar con Listas (crear Listas, Vistas, obtener listas, etc.).
- TaxonomyExtensions: Extensiones para trabajar con Campos de tipo Metadato Administrados (crear campos de tipo Metadato, crear Términos, Crear Grupos, etc).

Provisioning Framework

Probablemente sea la estrella del PnP. Este framework pretende sustituir el antiguo modelo de despliegue de soluciones SharePoint basadas en soluciones WSP y hacerlo basado en un modelo remoto, usando CSOM, y un esquema XML propio para la definición de Columnas, Listas, Paginas, etc.

Como hemos mencionado anteriormente, el esquema XML está definido en el proyecto PnP-Provisioning-Schema y estos son los nodos principales:

```
<pnp:Provisioning
      xmlns:pnp="http://schemas.dev.office.com/PnP/2015/12/Provi
   <pnp:Preferences />
   <pnp:Localizations />
   <pnp:Templates />
   <pnp:Sequence />
   <pnp:ImportSequence />
</pnp:Provisioning>
```

Y la definición de una Plantilla se compone de:

```
<pnp:ProvisioningTemplate
      xmlns:pnp="http://schemas.dev.office.com/PnP/2015/12/Provisi
      ID="xsd:ID"
      Version="xsd:decimal"
      ImagePreviewUrl="xsd:string"
      DisplayName="xsd:string"
      Description="xsd:string">
   <pnp:Properties />
   <pnp:SitePolicy />
   <pnp:WebSettings />
   <pnp:RegionalSettings />
   <pnp:SupportedUILanguages />
   <pnp:AuditSettings />
   <pnp:PropertyBagEntries />
   <pnp:Security />
   <pnp:SiteFields />
   <pnp:ContentTypes />
   <pnp:Lists />
   <pnp:Features />
   <pnp:CustomActions />
   <pnp:Files />
   <pnp:Pages />
   <pnp:TermGroups />
   <pnp:ComposedLook />
   <pnp:Workflows />
   <pnp:SearchSettings />
   <pnp:Publishing />
   <pnp:AddIns />
   <pnp:Providers />
</pnp:ProvisioningTemplate>
```

La definición completa del esquema la podéis encontrar en:

https://github.com/OfficeDev/PnP-Provisioning-Schema/blob/master/ProvisioningSchema-2015-12.md

Nota: Es importante saber que no todos los nodos del Esquema son posteriormente soportados por el framework de provisioning. Por ejemplo, el Nodo Sequence permite definir SiteCollections (Colecciones de Sitios) y Sites (Sitios) dentro de él, por lo que sería lógico pensar que el framework permite crear Colecciones de Sitio y Sitios, sin embargo, por desgracia, esto es algo que (todavía) no está soportado por el framework.

Un sencillo, pero completo ejemplo de XML lo tenemos en:

https://github.com/OfficeDev/PnP-Provisioning-Schema/blob/master/Samples/ProvisioningSchema-2015-12-ReferenceSample-01.xml

Y el siguiente código C# provisionaría el XML sobre un sitio existente:

```
XMLFileSystemTemplateProvider provider =
        new XMLFileSystemTemplateProvider(@"c:\temp\
pnpprovisioningdemo", "");
string templateName = "template.xml";

ctx.Web.ApplyProvisioningTemplate(template);
```

Lo bueno es que el framework también permite crear el XML desde un sitio ya existente (no se exporta absolutamente todo, por ejemplo, no se exportan todas las paginas existentes). Para exportar un sitio y guardar el XML generado, podemos utilizar el siguiente código:

```
XMLFileSystemTemplateProvider provider =
new   XMLFileSystemTemplateProvider(@"c:\temp\
pnpprovisioningdemo", "");
string templateName = "template.xml";
provider.SaveAs(template, templateName);
```

También podemos exportar un sitio como Template, o aplicar una plantilla XML desde PowerShell, con los comandos:

- Apply-SPOProvisioningTemplate
  - https://github.com/OfficeDev/PnP-PowerShell/blob/c1cacc5394908c996fd543f887463702cf63a9d9/Documentation/ApplySPOProvisioningTemplate.md
- Get-SPOProvisioningTemplate
  - https://github.com/OfficeDev/PnP-PowerShell/blob/c1cacc5394908c996fd543f887463702cf63a9d9/Documentation/GetSPOProvisioningTemplate.md

Seguimiento del programa PnP

Además de los proyectos de GitHub, el equipo de PnP sigue liberando continuamente material muy interesante sobre el desarrollo para Office 365:

- Yammer Network:
  - https://www.yammer.com/itpronetwork. Esta es posiblemente la mejor manera de seguir informado sobre el PnP. Además, podrás dejar tus dudas y comentarios y ser ayudado por el mismo equipo del PnP.
- Canal PnP en Channel 9:
  - https://channel9.msdn.com/blogs/OfficeDevPnP Videos con demostraciones y ejemplos de uso del PnP.
- Blog del PnP: http://dev.office.com/blogs
- Community call mensual del PnP:
  - https://github.com/OfficeDev/PnP/wiki/Community-call Apunta esta cita en tu calendario. Cada mes el equipo de PnP hace un meeting donde informa del estado del programa, el roadmap, muestra ejemplos, etc. Un verdadero lujo poder reunirse con el equipo del PnP y recibir las actualizaciones de mano del mismísimo Vesa.

LUIS MAÑEZ
SharePoint / Cloud Solutions Architect en ClearPeople LTD
@luismanez
http://geeks.ms/lmanez/

27

SharePoint 2013 Client Side Rendering en detalle

Del mismo modo que en la versión 2010 de SharePoint utilizábamos XLST y CSS para hacer cambios en el diseño de las intranets construidas en SharePoint. Ahora con el éxito del JavaScript y gracias a que cada vez son más potentes los dispositivos cliente y navegadores, cambia un poco el panorama. Por esto es que Microsoft para su versión 2013 se inventó el client-side rendering (en adelante CSR) de cara a diseñar un mecanismo para poder personalizar el estilo y diseño de las listas y bibliotecas de SharePoint usando un lenguaje algo más agradable que XSLT.

Introducción

CSR es un Framework que permite personalizar el diseño en las vistas y formularios que trae SharePoint de forma predeterminada. Digamos que este Framework aplica tanto a SharePoint 2013 como a Office 365 (al menos por ahora, febrero 2016).

Básicamente, lo que hace CSR es transformar un objeto de JavaScript en HTML y después insertarlo en el DOM.

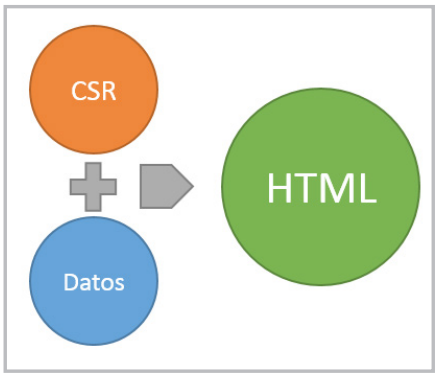


Imagen 1.- Proceso de transformación de CSR.

En la versión 2010 de SharePoint utilizábamos XLST y CSS para hacer cambios en el diseño de las intranets construidas en SharePoint

Además, esta transformación la realiza por etapas o ámbitos según la parte de la vista o el formulario que queramos cambiar. En la figura de abajo vemos los distintos ámbitos que podemos definir a la hora de configurar una plantilla para CSR:

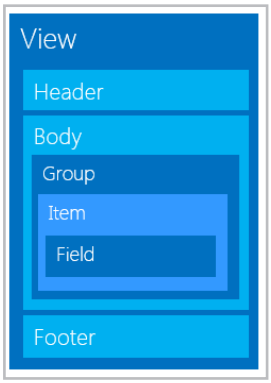


Imagen 2.- Ámbitos para configurar una plantilla para CSR.

Para los que ya hemos trasteado con CSR y JSLink es muy importante saber que estos dos conceptos son diferentes cosas:

- CSR: es un Framework de JavaScript que nos expone a los desarrolladores una serie de métodos que podemos usar para modificar el comportamiento grafico de las vistas y formularios de SharePoint.
- JSLink: es un modo de asociar un fichero de JavaScript a un objeto de SharePoint.

Ambos pueden convivir juntos, pero también el uno sin el otro. Por ejemplo, podemos asociar un JS a un XSLTListViewWebPart de SharePoint y que este fichero de JS no tenga nada de CSR y seguirá ejecutándose el código que hay dentro cuando se carga la página donde este ese webpart. Y por el contrario también podemos usar CSR directamente asociando un fichero JS a la master page usando una Custom Action, y el CSR se ejecutaría y sobrescribiría el comportamiento nativo de la vista que estemos reescribiendo.

A continuación, en este artículo veremos cómo es esta interfaz que nos proporciona Microsoft para poder sobrescribir las plantillas y registrar plantillas nuestras personalizadas.

Relación entre CSR y clienttemplates.js

En SharePoint 2013 hay un montón de ficheros de JavaScript creados con el objetivo de mejorar la experiencia de usuario. Por ejemplo, el fichero clienttemplates.js es donde está el núcleo del Framework CSR, es decir, todos los objetos, “clases” y funciones de JS que definen toda la lógi-

ca y que es cargado en las páginas de SharePoint y Office 365 de forma predeterminada. Véase en la imagen como y en qué orden se carga usando, en este caso, la plantilla de Team Site:

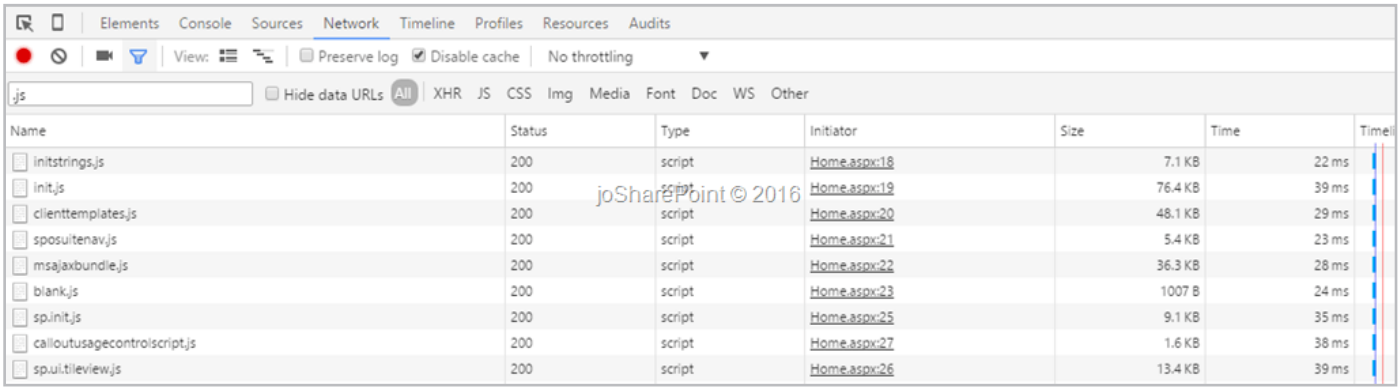


Imagen 3.- Librería JavaScript que se cargan en un sitio de Grupo.

Lo que también es interesante es que después de clienttemplates.js se cargan otros ficheros, donde se utilizar la función RegisterTemplateOverrides, que fue previamente creada en clienttemplates.js y cuya finalidad es permitir definir una plantilla y registrarla, veremos de qué forma más adelante.

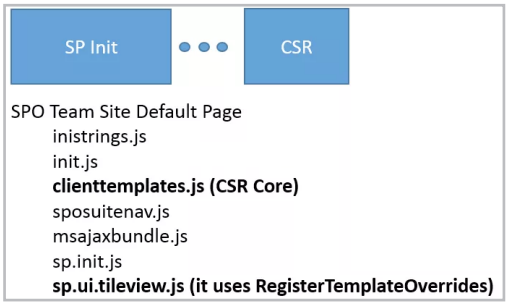


Imagen 4.- Otros archivos JavaScript que se cargan después de clienttemplates.js.

Husmeando dentro de clienttemplates.js

El fichero clienttemplates.js se encuentra, de forma predeterminada, en la ruta: C:\Program Files\Common Files\microsoft shared\Web Server Extensions\15\TEMPLATE\LAYOUTS

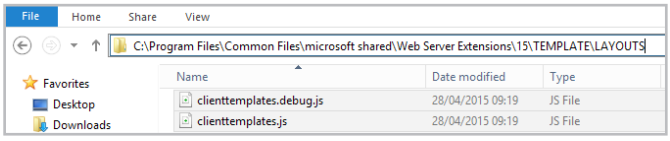


Imagen 5.- Ruta de clienttemplates.js.

Como somos muy curiosos, veamos a ver que está ocurriendo dentro de este JS. Siendo sincero, hay muchas partes de este código que no entiendo y que no he tenido tiempo de investigar, pero veamos un poco las partes más importantes. Estas son las primeras líneas de código:

En SharePoint 2013 hay un montón de ficheros de JavaScript creados con el objetivo de mejorar la experiencia de usuario

```
function $_global_clienttemplates() {
    SPClientRenderer = {
    SPClientRenderer.IsDebugMode = function(renderCtx) {
    SPClientRenderer.Render = function(node, renderCtx) {
    SPClientRenderer.RenderReplace = function(node, renderCtx) {
    SPClientRenderer._ExecuteRenderCallbacks = function(renderCtx, callbackType) {
    SPClientRenderer._ExecuteRenderCallbacksWorker = function(renderCtx, callbackType, templateExecContext) {
    SPClientRenderer.RenderCore = function(renderCtx) {
    SPClientRenderer.ParseTemplateString = function(templateStr, c) {
    SPClientRenderer.ParseTemplateStringWorker = function(templateStr, c) {
    SPClientRenderer.ReplaceUrlTokens = function(tokenUrl) {
    clientHierarchyManagers = {};
    ClientHierarchyManager = function(wpq, bRtl) {
    if (typeof window.ClientPivotControl == "undefined") {
    SPClientTemplates = {};
    SPClientTemplates.FileSystemObjectType = {
    SPClientTemplates.ChoiceFormatType = {
    SPClientTemplates.ClientControlMode = {
    SPClientTemplates.RichTextMode = {
    SPClientTemplates.UrlFormatType = {
    SPClientTemplates.DateTimeDisplayFormat = {
    SPClientTemplates.DateTimeCalendarType = {
    SPClientTemplates.UserSelectionMode = {
    SPClientTemplates.PresenceIndicatorSize = {
    SPClientTemplates.TemplateManager = {};
    SPClientTemplates.TemplateManager._TemplateOverrides = {};
    SPClientTemplates.TemplateManager._TemplateOverrides.View = {};
    SPClientTemplates.TemplateManager._TemplateOverrides.Header = {};
    SPClientTemplates.TemplateManager._TemplateOverrides.Body = {};
    SPClientTemplates.TemplateManager._TemplateOverrides.Footer = {};
    SPClientTemplates.TemplateManager._TemplateOverrides.Group = {};
    SPClientTemplates.TemplateManager._TemplateOverrides.Item = {};
    SPClientTemplates.TemplateManager._TemplateOverrides.Fields = {};
    SPClientTemplates.TemplateManager._TemplateOverrides.OnPreRender = {};
    SPClientTemplates.TemplateManager._TemplateOverrides.OnPostRender = {};
    SPClientTemplates.TemplateManager._RegisterDefaultTemplates = function(renderCtx) {
    SPClientTemplates.TemplateManager.RegisterTemplateOverrides = function(renderCtx) {
    SPClientTemplates.TemplateManager._RegisterTemplatesInternal = function(renderCtx, registeredOverrides) {
    SPClientTemplates.TemplateManager.GetTemplates = function(renderCtx) {
```

- $_global_clienttemplates: es el contenedor, la función principal que contiene todos los objetos y funciones que necesita CSR. Esta función es ejecutada al final del fichero clienttemplates.js de forma que se carguen todas las variables iniciales y funciones en el motor de ejecución de JS.
- SPClientRenderer: "Clase" de JavaScript cuyo objetivo es definir todas las funciones "core" usadas en el framework de renderizado. Las funciones más importantes son:
  - SPClientRenderer.ReplaceUrlTokens (tokenUrl): reemplaza los tokens típicos como ~sitecollection por las URLs correctas en cada entorno.
  - SPClientRenderer.Render (node, renderCtx): Función Render que divide la ejecución en tres partes invocando a las funciones OnPreRender, RenderCore and OnPostRender.

```
SPClientRenderer.Render = function(node, renderCtx) {
    if (node == null || renderCtx == null)
        return;
    SPClientRenderer._ExecuteRenderCallbacks(renderCtx, 'OnPreRender');
    var result = SPClientRenderer.RenderCore(renderCtx);

    if (renderCtx.Errors != null && renderCtx.Errors.length > 0) {
    if (result != null && result != '') {
    SPClientRenderer._ExecuteRenderCallbacks(renderCtx, 'OnPostRender');
};
```

  - SPClientRenderer.RenderCore (renderCtx): esta función define una plantilla "Template", para cada ámbito de CSR (View, Header, Body, Footer, Groups, Items and Fields). La ventaja de esto es

- que podemos decidir que parte de los formularios o vistas cambiar y podemos ahorrarnos bastantes líneas de código sobrescribiendo la vista completa, si, por ejemplo, lo único que queremos cambiar es un color de fondo, o una imagen de un campo. Si echamos un vistazo dentro del código de la función, vemos como es la función ResolveTemplate (véase la figura) la que internamente escogerá entre usar una plantilla por defecto, o una nuestra personalizada si la hay, y luego veremos en qué objeto de JS se basa para tomar esta decisión.

```
SPClientRenderer.RenderCore = function(renderCtx) {
    if (renderCtx == null)
        return '';
    renderCtx.RenderView = RenderView;
    renderCtx.RenderHeader = RenderHeader;
    renderCtx.RenderBody = RenderBody;
    renderCtx.RenderFooter = RenderFooter;
    renderCtx.RenderGroups = RenderGroups;
    renderCtx.RenderItems = RenderItems;
    renderCtx.RenderFields = RenderFields;
    renderCtx.RenderFieldByName = RenderFieldByName;
    return RenderView(renderCtx);
    function RenderView(rCtx) {
    function RenderHeader(rCtx) {
    function RenderBody(rCtx) {
    function RenderFooter(rCtx) {
    function ResolveTemplate(rCtx, component, level) {
    function DoSingleTemplateRender(inCtx, tplTag) {
    function RenderGroups(inCtx) {
    function RenderItems(inCtx) {
    function RenderFields(inCtx) {
    function RenderFieldByName(inCtx, fName) {
    function ExecuteFieldRender(inCtx, fld) {
    function GetGroupsKey(c) {
    function GetItemsKey(c) {
};
```

- SPClientTemplates: De la misma forma que el objeto SPClientRenderer contiene todas las funciones relativas al “render”, esta clase contiene las variables y funciones que implementan el sub-framework de plantillas “Templates”. El cual será usado para gestionar todos los ámbitos explicados anteriormente dentro de una misma vista común. Nótese que este objeto contiene la función SPClientTemplates.TemplateManager.RegisterTemplateOverrides que nos permitirá definir nuestras plantillas personalizadas para cada ámbito.

Analizando la función RegisterTemplateOverrides

La función RegisterTemplateOverrides se basa en el objeto interno TemplateManager._TemplateOverrides para mantener y guardar el estado de todas las personalizaciones de plantillas que se han hecho hasta el momento, por lo que lo primero que se hace es obtener ese objeto y luego pasárselo a la función _RegisterTemplatesInternal, como se puede observar en el código:

```
SPClientTemplates.TemplateManager.RegisterTemplateOverrides = function (renderCtx) {
        if (!renderCtx || !renderCtx.Templates && !renderCtx.OnPreRender && !renderCtx.OnPostRender)
            return;
        var tempStruct = SPClientTemplates.TemplateManager._TemplateOverrides;

        SPClientTemplates.TemplateManager._RegisterTemplatesInternal(renderCtx, tempStruct);
    };
```

También vemos que se le pasa como parámetro un objeto llamado “renderCtx”. Es un objeto de tipo RenderContext y está disponible en tiempo de ejecución para proveer algunas variables y mantener el estado del framework completo y que se pueda usar de forma consistente creando varios templates desde distintos ficheros JavaScript.

RenderContext Object

Este objeto es usado en casi todas las funciones de clienttemplates.js. Es un objeto dinámico compuesto de propiedades dinámicas. Propiedades que para cada función pueden ser distintas. En el caso concreto de la función RegisterTemplateOverrides las propiedades más importantes de este objeto son:

```
var renderContext = {
 BaseViewID: /* ID en Schema.xml. Ver más  en este post de Andrey */,
 ListTemplateType: /* List Template ID. Lista completa de template IDs
*/,
 ViewStyle: /* View Style: Boxed, no labels (12), Boxed(13),... */,
 OnPreRender: /* function or array of functions */,
 Templates: {
  View: /* function or string */,
  Body: /* function or string */,
  Header: /* function or string */,
  Footer: /* function or string */,
  Group: /* function or string */,
  Item: /* function or string */,
  Fields: {
   'Field1 Internal Name': {
     View: /* function or string */,
     EditForm: /* function or string */,
     DisplayForm: /* function or string */,
     NewForm: /* function or string */
   },
   'Field2 Internal Name': {
     View: /* function or string */,
     EditForm: /* function or string */,
     DisplayForm: /* function or string */,
     NewForm: /* function or string */
   },
  },
 },
 OnPostRender: /* function or array of functions */
};
```

Es muy importante echarle un ojo despacio a estas propiedades para tener e incluso guardarse esta plantilla en favoritos de cara a poder reusarla luego cada vez que tengamos que definir una nueva plantilla de CSR.

Hasta ahora hemos echado un vistazo al código fuente que forma el Framework de los Templates, pero como realmente se conoce y aprende es depurando la aplicación y viendo qué valor van obteniendo cada una de las propiedades y variables con respecto al momento de ejecución.

A continuación, veremos un ejemplo en el que depuramos “hacemos debug” de esta función usando el debugger de Chrome (F12). En el ejemplo aplicaremos una personalización básica usando CSR a nivel de Item que aplique solamente a las listas Custom (100):

```
SP.SOD.executeFunc("clienttemplates.js",
"SPClientTemplates", function() {
 SPClientTemplates.TemplateManager.
RegisterTemplateOverrides({
 Templates: {
  Item: function(ctx) {
   return String.format("Hi world");
  }
 },
 ListTemplateType: 100
});
})
```

- que podemos decidir que parte de los formularios o vistas cambiar y podemos ahorrarnos bastantes líneas de código sobrescribiendo la vista completa, si, por ejemplo, lo único que queremos cambiar es un color de fondo, o una imagen de un campo. Si echamos un vistazo dentro del código de la función, vemos como es la función ResolveTemplate (véase la figura) la que internamente escogerá entre usar una plantilla por defecto, o una nuestra personalizada si la hay, y luego veremos en qué objeto de JS se basa para tomar esta decisión.

```
SPClientRenderer.RenderCore = function(renderCtx) {
    if (renderCtx == null)
        return '';
    renderCtx.RenderView = RenderView;
    renderCtx.RenderHeader = RenderHeader;
    renderCtx.RenderBody = RenderBody;
    renderCtx.RenderFooter = RenderFooter;
    renderCtx.RenderGroups = RenderGroups;
    renderCtx.RenderItems = RenderItems;
    renderCtx.RenderFields = RenderFields;
    renderCtx.RenderFieldByName = RenderFieldByName;
    return RenderView(renderCtx);
    function RenderView(rCtx) {
    function RenderHeader(rCtx) {
    function RenderBody(rCtx) {
    function RenderFooter(rCtx) {
    function ResolveTemplate(rCtx, component, level) {
    function DoSingleTemplateRender(inCtx, tplTag) {
    function RenderGroups(inCtx) {
    function RenderItems(inCtx) {
    function RenderFields(inCtx) {
    function RenderFieldByName(inCtx, fName) {
    function ExecuteFieldRender(inCtx, fld) {
    function GetGroupsKey(c) {
    function GetItemsKey(c) {
};
```

- SPClientTemplates: De la misma forma que el objeto SPClientRenderer contiene todas las funciones relativas al "render", esta clase contiene las variables y funciones que implementan el sub-framework de plantillas "Templates". El cual será usado para gestionar todos los ámbitos explicados anteriormente dentro de una misma vista común. Nótese que este objeto contiene la función SPClientTemplates.TemplateManager. RegisterTemplateOverrides que nos permitirá definir nuestras plantillas personalizadas para cada ámbito.

Analizando la función Register-TemplateOverrides

La función RegisterTemplateOverrides se basa en el objeto interno TemplateManager._TemplateOverrides para mantener y guardar el estado de todas las personalizaciones de plantillas que se han hecho hasta el momento, por lo que lo primero que se hace es obtener ese objeto y luego pasárselo a la función _RegisterTemplatesInternal, como se puede observar en el código:

```
SPClientTemplates.TemplateManager.RegisterTemplateOverrides = function (renderCtx) {
        if (!renderCtx || !renderCtx.Templates && !renderCtx.OnPreRender && !renderCtx.OnPostRender)
            return;
        var tempStruct = SPClientTemplates.TemplateManager._TemplateOverrides;

        SPClientTemplates.TemplateManager._RegisterTemplatesInternal(renderCtx, tempStruct);
    };
```

También vemos que se le pasa como parámetro un objeto llamado "renderCtx". Es un objeto de tipo RenderContext y está disponible en tiempo de ejecución para proveer algunas variables y mantener el estado del framework completo y que se pueda usar de forma consistente creando varios templates desde distintos ficheros JavaScript.

RenderContext Object

Este objeto es usado en casi todas las funciones de clienttemplates.js. Es un objeto dinámico compuesto de propiedades dinámicas. Propiedades que para cada función pueden ser distintas. En el caso concreto de la función RegisterTemplateOverrides las propiedades más importantes de este objeto son:

```
var renderContext = {
 BaseViewID: /* ID en Schema.xml. Ver más  en este post de Andrey */,
 ListTemplateType: /* List Template ID. Lista completa de template IDs
*/,
 ViewStyle: /* View Style: Boxed, no labels (12), Boxed(13),... */,
 OnPreRender: /* function or array of functions */,
 Templates: {
  View: /* function or string */,
  Body: /* function or string */,
  Header: /* function or string */,
  Footer: /* function or string */,
  Group: /* function or string */,
  Item: /* function or string */,
  Fields: {
   'Field1 Internal Name': {
     View: /* function or string */,
     EditForm: /* function or string */,
     DisplayForm: /* function or string */,
     NewForm: /* function or string */
   },
   'Field2 Internal Name': {
     View: /* function or string */,
     EditForm: /* function or string */,
     DisplayForm: /* function or string */,
     NewForm: /* function or string */
   },
  },
 },
 OnPostRender: /* function or array of functions */
};
```

Es muy importante echarle un ojo despacio a estas propiedades para tener e incluso guardarse esta plantilla en favoritos de cara a poder reusarla luego cada vez que tengamos que definir una nueva plantilla de CSR.

Hasta ahora hemos echado un vistazo al código fuente que forma el Framework de los Templates, pero como realmente se conoce y aprende es depurando la aplicación y viendo qué valor van obteniendo cada una de las propiedades y variables con respecto al momento de ejecución.

A continuación, veremos un ejemplo en el que depuramos "hacemos debug" de esta función usando el debugger de Chrome (F12). En el ejemplo aplicaremos una personalización básica usando CSR a nivel de Item que aplique solamente a las listas Custom (100):

```
SP.SOD.executeFunc("clienttemplates.js",
"SPClientTemplates", function() {
 SPClientTemplates.TemplateManager.
RegisterTemplateOverrides({
 Templates: {
  Item: function(ctx) {
   return String.format("Hi world");
  }
 },
 ListTemplateType: 100
});
})
```

Depurando la función RegisterTemplateOverrides

A veces, lo más difícil cuando se quiere depurar un fichero de JavaScript en SharePoint (o cualquier web) es encontrar la línea donde poner el punto de interrupción. En este caso es doblemente difícil, porque el fichero clienttemplates.js de primeras no aparece en la pestaña Sources y de segundas el fichero está en su versión reducida.

Lo más difícil cuando se quiere depurar un fichero de JavaScript en SharePoint (o cualquier web) es encontrar la línea donde poner el punto de interrupción

Veamos como he conseguido poner ese punto de interrupción:

1.– Abrimos el sitio de SharePoint y pulsamos F12 (en mi caso usando Chrome).
2.– Vamos a la sección Network.

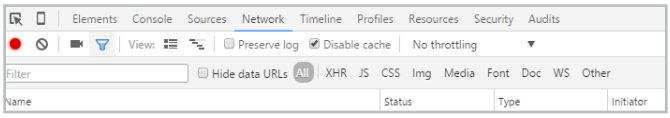


Imagen 6.- Depuración de la función RegisterTemplateOverrides.

3.– Usamos el textbox para filtrar por “clienttemplate”.

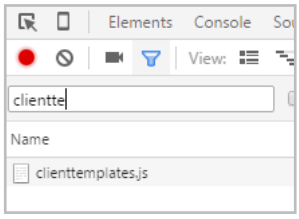


Imagen 7.- Uso del textbox para filtrar.

4.– Botón derecho en el fichero clienttemplates.js y clicamos “Open in Sources panel”.

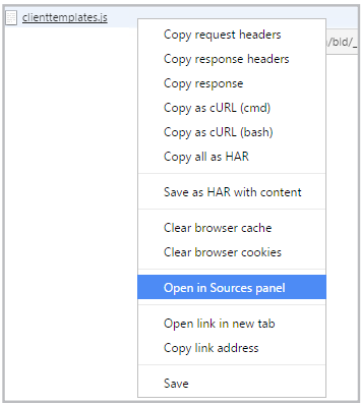
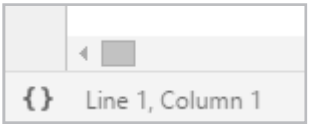


Imagen 8.- Opción “Open in sources panel”.

5.– Ahora veremos el fichero clienttemplates.js en una única línea, por lo que usamos el botón “Pretty Print” para obtener el código formateado.



6.– Una vez tenemos el código formateado, clicamos Control + F para buscar sobre él.

Imagen 9.- Código formateado.

7.– Buscamos el nombre de la función y le asignamos el punto de interrupción.

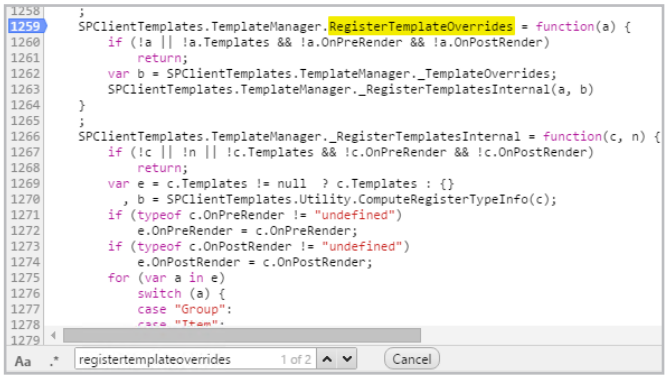


Imagen 10.- Asignación de punto de ruptura.

8.– Feliz depuración!

Veamos el ejemplo ahora realizado con un SharePoint OnPrem y con la depuración activada para que saque los ficheros .debug en lugar de los ficheros reducidos. Lo primero que comprueba la función RegisterTemplateOverrides es si le hemos pasado por parámetro el objeto renderContext con alguna personalización sobre algunos de los Templates, OnPreRender u OnPostRender. Si no es así, se sale de la función sin hacer nada, como es lógico.

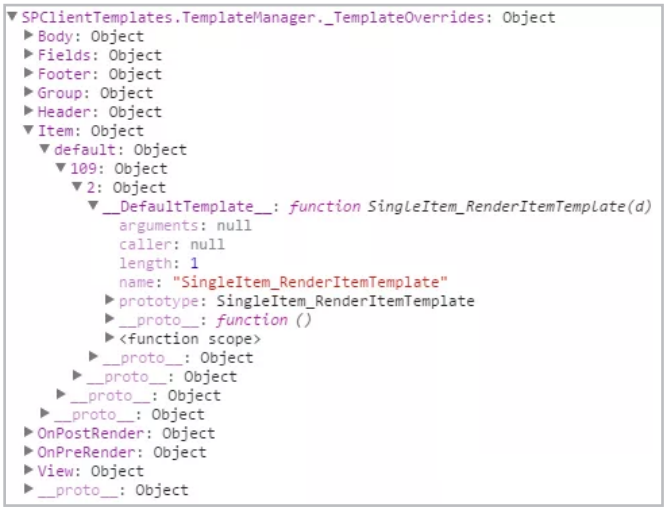
```
SPClientTemplates.TemplateManager.RegisterTemplateOverrides = function(renderCtx) {  renderCtx =
    if (!renderCtx || !renderCtx.Templates && !renderCtx.OnPreRender && !renderCtx.OnPostRender)
        return;
    var tempStruct = SPClientTemplates.TemplateManager._TemplateOverrides;

    SPClientTemplates.TemplateManager._RegisterTemplatesInternal(renderCtx, tempStruct);
};
```

Si por el contrario hemos configurado algún Template que aplicar, lo primero que se hace es obtener las Templates que ya han sido registradas hasta el momento usando la línea de código:

var tempStruct = SPClientTemplates.TemplateManager._ TemplateOverrides;

La estructura del objeto mediante el cual se mantiene el estado de todas las personalizaciones o Templates es la siguiente:



Vemos como para cada ámbito o Template se almacena una sub-estructura de datos que contiene la siguiente forma:

Template Scope (Body, Fields, Footer, Group, Header, Item y View)
ViewStyle (default en este caso)
ListTemplateType (109 in this case)
BaseViewID (2 in this case)
Nombre de la función

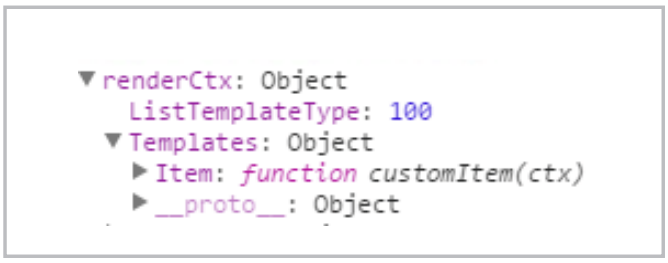
Para ver más información sobre:

- ViewStyle: http://www.codeproject.com/Articles/620110/SharePoint-Client-Side-Rendering-List-Views
- ListTemplateType: http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.listtemplatetype
- BaseViewID: http://sharepoint.stackexchange.com/questions/28349/how-can-i-use-normal-paging-on-a-custom-view/28538#28538

Después se invoca a la función _RegisterTemplatesInternal que hace un merge entre las Templates aplicadas hasta el momento, obtenidas en el objeto anteriormente mostrado y las Templates pasadas como parámetro.

SPClientTemplates.TemplateManager._RegisterTemplatesInternal = function(renderCtx, registeredOverrides) { … }

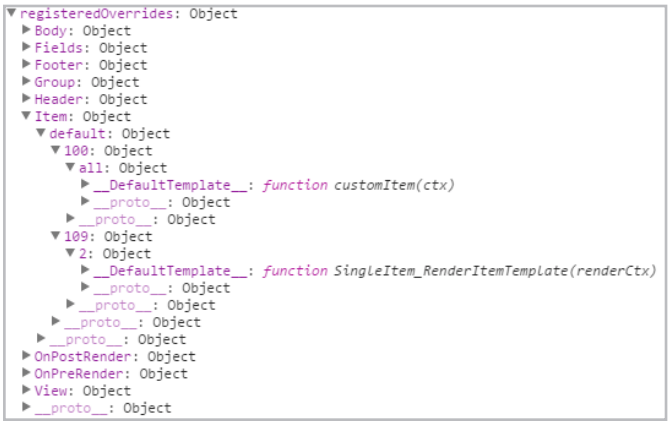
Template pasada como parámetro: renderCtx:



RenderContext Object es un objeto dinámico compuesto de propiedades dinámicas

Templates aplicadas hasta el momento: registeredOverrides:

Y después de hacer el merge:



Nótese como en la versión final del objeto que mantiene el estado "registeredOverrides" tenemos aplicadas dos personalizaciones distintas para Item, una para las listas de tipo 109 y otra para listas de tipo 100 (Custom List), y concretamente como en el código del ejemplo no especificamos BaseViewID, se asigna para todas usando el nombre "all".

Conclusiones

El objetivo de este artículo es conocer a fondo la tecnología Client-side rendering y JSLink de cara a poder saber qué tipo de personalizaciones podemos hacer y hasta donde podemos llegar. Me gustaría remarcar y compartir algunos aspectos descubiertos durante esta investigación:

- Client-side Rendering / CSR es un framework de "renderizado" y JSLink es un mecanismo para asociar un fichero JavaScript a un objeto de SharePoint.
- Podemos usar CSR sin JSLink, simplemente haciendo una llamada a RegisterTemplateOverrides después de que se ejecute el fichero clienttemplates.js. Por ejemplo, desde un Script Editor.
- Con CSR es posible implementar diferentes personalizaciones para la misma vista y lista en páginas de Sha-

rePoint distintas.

- No se puede usar CSR para aplicar dos personalizaciones diferentes para la misma vista y lista dentro de la misma página de SharePoint. Por ejemplo, tenemos una custom list y queremos aplicar diferentes presentaciones de ella en la misma página usando la misma vista. Esto no sería posible porque como hemos visto en el artículo, solamente se almacena una función JS por ListTemplateType y BaseViewID. Hay diferentes formas para hacerle workaround a esto, pero no estoy seguro que estén 100 soportadas.
- Debemos tener en cuenta que cuando se usan múltiples instancias de un List View Web Part en la misma página y se aplica CSR a una de las instancias, el diseño del resto de los List View Web Part puede también cambiar porque internamente SharePoint utiliza el mismo framework. (https://msdn.microsoft.com/en-us/magazine/dn745867.aspx)
- No lo he probado, pero parece posible usar diferentes funciones para la misma vista y lista si se cambia el ViewStyle. Esto podría darnos alguna flexibilidad en algunos escenarios.

Referencias

- Artículo de Andrei (en inglés) http://www.codeproject.com/Articles/620110/SharePoint-Client-Side-Rendering-List-Views
- Artículo original en inglés (http://josharepoint.com/2016/01/14/sharepoint-2013-client-side-rendering-register-templates-overrides-in-detail)

JOSÉ QUINTO
@jquintozamora
http://www.josharepoint.com

34

¿Es Project Server / Online realmente multiidioma?

En un proyecto para un cliente que tenía sedes en varios países se planteó la posibilidad de hacer que el sitio de PWA fuese multiidioma y no desplegarlo todo sólo en inglés. Según Microsoft, Project Server (y, por lo tanto, Project Online) soporta el multiidioma: se pueden instalar los Language Packs correspondientes tanto para SharePoint como para Project Server y, de esta forma, la interfaz se adaptará al idioma por defecto del usuario:

Sitio de pwa en español:

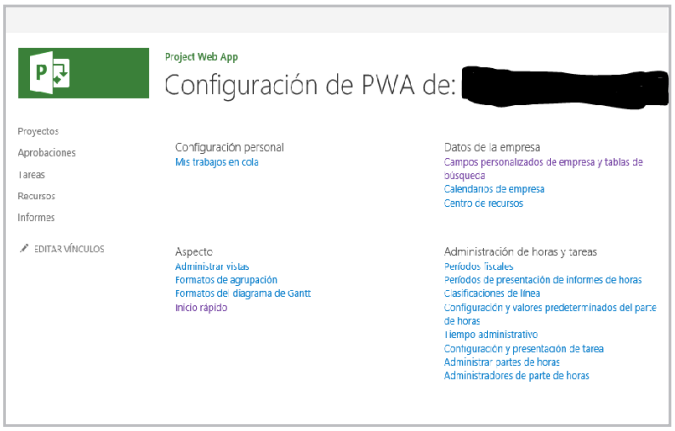


Imagen 1.- Sitio de PWA en castellano.

El mismo sitio, cambiando el idioma por defecto del usuario a inglés:

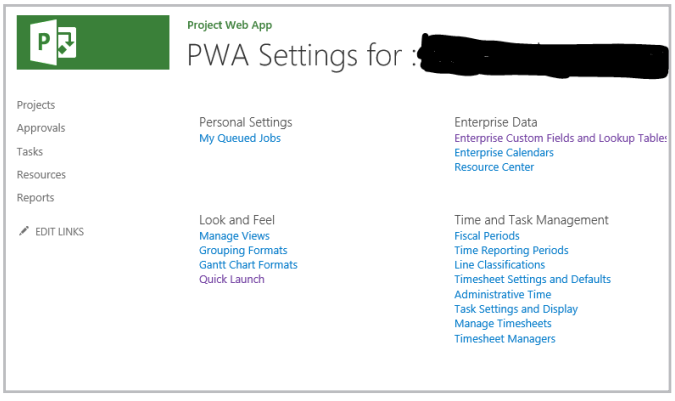


Imagen 2.- Sitio de PWA con cambio de idioma aplicado.

Hasta aquí, aparentemente todo bien. Sin embargo, ¿qué pasa si creo un campo personalizado nuevo? Pues que el nombre del campo no puede crearse en varios idiomas. Si, por ejemplo, creamos el campo "Mi campo", en la interfaz en español:

Imagen 3.- Campo personalizado.

Cuando cambiemos a la interfaz en inglés seguirá llamándose "Mi campo":

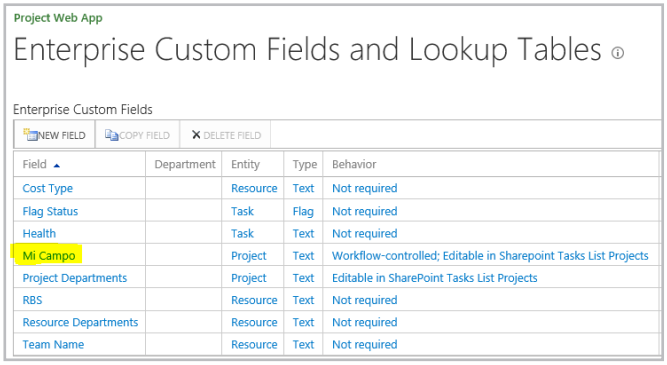


Imagen 4.- Campo personalizado tras aplicar el cambio de idioma.

Hasta aquí podría parecerse al comportamiento de SharePoint ya que, cuando creamos un campo nuevo para una lista en un idioma, hasta que no se cambia de interfaz y se modifica su título adaptándolo al nuevo lenguaje no se guarda para el mismo. Sin embargo, esto no funciona así para Project Server. Si modificamos el nombre del campo para el idioma inglés, se queda en inglés también en español:

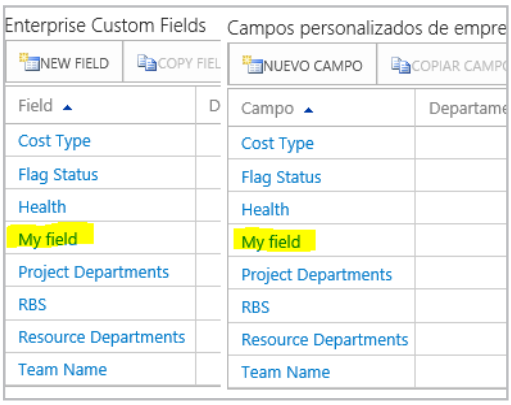


Imagen 5.- Efecto de cambio del campo personalizado en inglés.

Está claro que las bases de datos de Project Server no permiten mantener la información de los nombres de los campos en más de un idioma. Bueno, hay una excepción: las tablas de búsqueda personalizadas (Enterprise Lookup Tables). Curiosamente, este componente sí permite la creación de sus elementos en múltiples idiomas, pero sólo mediante desarrollo utilizando el API de la PSI, no a través de la interfaz de usuario:

Lookup tables can be localized through the PSI. Neither Project Web Access nor Project Professional includes a way to localize lookup tables. Project Web Access always shows lookup table values using the highest preference Internet Explorer language available. If no lookup table language is found that matches an Internet Explorer language, then Project Web Access uses the primary lookup table language.

Incluso, para acabar de salir de dudas, podemos consultar la información nuestros proyectos mediante el API REST, que permite el multiidioma. Para ello, simplemente tenemos que realizar la siguiente llamada:

http://Mi_Instancia_PWA/_api/ProjectData/[es-ES]/Proyectos

Y, si buscamos en el xml nuestro campo, aparecerá tal y como ha quedado modificado al inglés:

<d:Myfield m:null="true" /></m:properties></content></entry>

Esta limitación afecta a todos los elementos de la interfaz de usuario que contengan campos personalizados: PDP's, vistas, Inicio rápido (Quick Launch), etc.

Sin embargo, ¿se podría hacer algo para aparentar, al menos, el multiidioma? La respuesta es sí: duplicarlo todo por tantos idiomas cómo sea necesario. Y podemos duplicarlo de dos formas: usando una sola instancia de PWA o usando tantas como idiomas se requieran.

Multiidioma con una sola instancia de PWA

Como hemos comentado antes, habría que generar los campos personalizados por todos los idiomas como sea necesario soportar, lo que significa que cuánto más complejo sea el sistema a implementar más complejo será su mantenimiento. Además, también habría que multiplicar el resto de elementos de Project Server / Online por el número de idiomas como, por ejemplo, las PDP's y, con ellas los tipos de proyectos y flujos, para que cada tipo de proyecto muestre las páginas con los campos en el idioma correspondiente. Y, si hay algún workflow, entonces habrá que generarlo tantas veces como tipologías de proyectos a las que esté asociado, ya que los campos que se usarán para cada idioma serán diferentes, así como los correos.

Otro tema son las vistas: no sólo habría que generarlas tantas veces como lenguajes sino que, además, habría que controlar su visibilidad para que sólo se muestren a los usuarios del país que corresponda. Esto nos obligaría a tener que utilizar el modelo de seguridad clásico de Project Server, ya que es la única forma de controlar la seguridad de las vistas tal y como necesitamos. Y, por último, habría que implementar tantos grupos de seguridad como grupos de usuarios con un idioma común.

Está claro que las bases de datos de Project Server no permiten mantener la información de los nombres de los campos en más de un idioma

En resumen, un suplicio. Funcionaría, pero sería una pesadilla tanto su implementación como su mantenimiento.

Multiidioma con varias instancias de PWA

La otra opción es generar tantas instancias de PWA como idiomas sean necesarios. Así, aunque se debería replicar todo igualmente, al menos tendríamos la información organizada de una forma lógica. En una instalación On Premise se podrían generar tantas colecciones de sitios de PWA como idiomas sean necesarios. En Project Online, oficialmente solo se permiten siete instancias de PWA, pero puede ser más que suficiente.

Otra ventaja es que, si se usan básicamente los grupos de seguridad estándar de Project, no será necesario replicarlos en cada uno de los sitios.

La mayor dificultad de esta solución estriba en la visibilidad de la cartera de proyectos de forma global (ver el estado de los proyectos de todos los países a la vez), pero esto se podría solventar con informes a medida para aglutinar la información necesaria de todas las instancias de PWA.

Conclusiones

Como se ha podido comprobar, Project Server / Online no está pensado para el multiidioma, lo que lleva a que la mayoría de implementaciones en entornos internacionales se hagan en inglés. Quizás la integración de las bases de datos de Project en la base de datos de contenido de SharePoint sea un primer paso para mejorar este aspecto. La verdad es que no he encontrado software de terceros que permita hacer multiidioma la interfaz de usuario. A mí se me ha ocurrido un "workaround" basado en JavaScript y una lista de SharePoint de traducciones que podría dar una solución a esta carencia del producto.

JOSÉ RAFAEL GARCÍA
josex1975@gmail.com
Blog: www.projectservernotes.com | Twitter: jrgarcia1975

36

Integración de Powershell local con Powershell de Azure

PowerShell puede ser utilizado tanto de forma local en nuestro equipo haciendo uso de las funciones disponibles, como de forma remota para configurar y administrar servicios OnPremises o en la nube. En este artículo hablaremos de cómo integrar PowerShell local con PowerShell de Azure para poder realizar de manera más ágil nuestras operaciones en Azure.

Pasos a seguir para la integración

Para integrar nuestro PowerShell local con PowerShell en Azure hay que seguir los pasos que se indican a continuación:

- Llegamos a la dirección: http://azure.microsoft.com/en-us/downloads/, en la cual encontramos diferentes herramientas que podemos utilizar para interactuar con Azure. Elegimos la opción: Azure command-line interface, Windows Install (Download .exe)

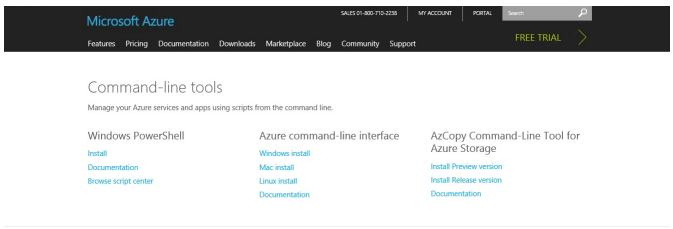


Imagen 1.- Microsoft Azure Download.

- Procedemos a la instalación del mismo presionando el botón “Instalar”

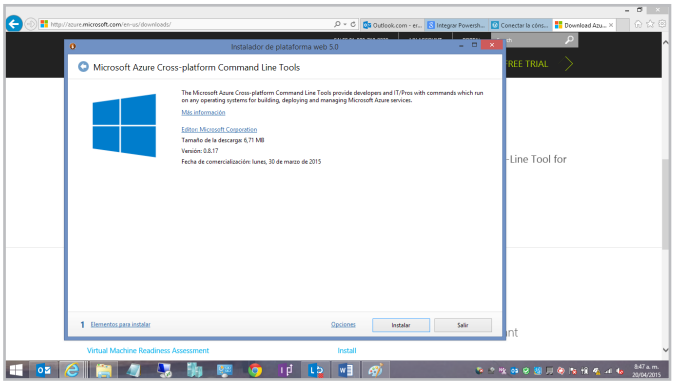


Imagen 2.- Instalación de Azure command-line interface.

- En requisitos, tomamos la opción Windows Azure PowerShell y aceptar. Se iniciará el proceso de descarga e instalación de los componentes para PowerShell de Azure.

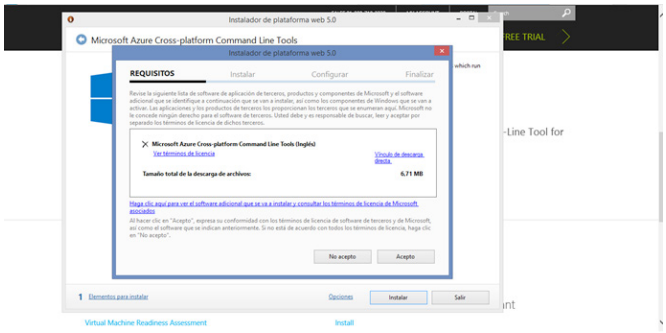


Imagen 3.- Instalación de Azure command-line interface.

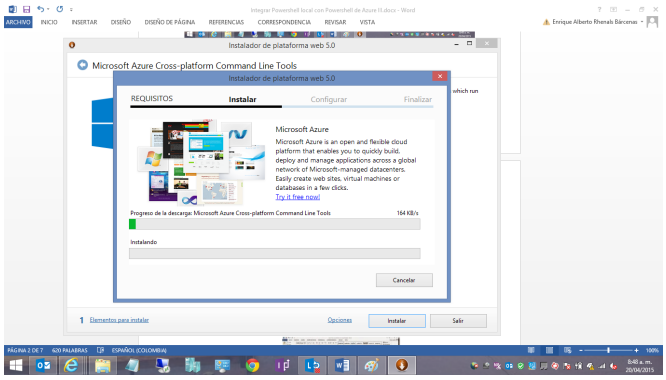


Imagen 4.- Instalación de Azure command-line interface.

PowerShell puede ser utilizado tanto de forma local en nuestro equipo haciendo uso de las funciones disponibles, como de forma remota

- Y aquí continuamos con la finalización de la instalación inicial.

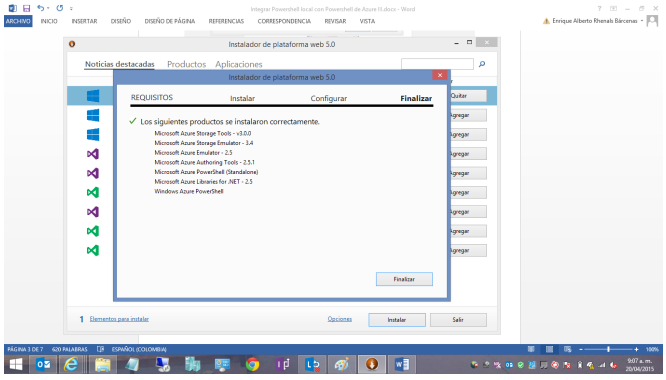


Imagen 5.- Herramienta instalada.

- Luego de tener instalada la herramienta procedemos a ir al panel de Windows y elegimos Azure PowerShell y tomamos la opción “ejecutar como administrador”.

Imagen 6.- Acceso a Azure Powershell en nuestro equipo.

- Estando en la opción de Azure PowerShell ejecutamos la sentencia Set-ExecutionPolicy RemoteSigned que nos sirve para dar permisos de ejecución de scripts PowerShell en nuestro ambiente.

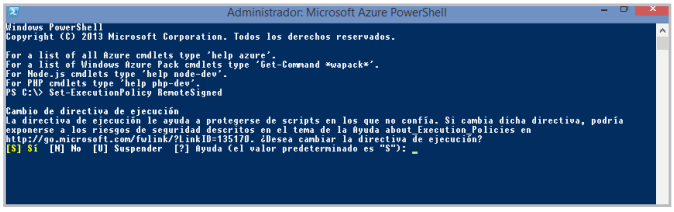


Imagen 7.- Ejecución sentencia en Azure Powershell.

- Luego de realizar lo anterior procedemos a integrar las herramientas de Azure al ambiente normal de Power-Shell ejecutamos lo siguiente:

```
Import-Module “C:\Program Files (x86)\Microsoft SDKs\
Windows Azure\PowerShell\Azure\Azure.psd1”
```

Allí elegimos la opción “Certificate”.

- Posteriormente nos aparece la opción de loguearnos en Azure, para lo cual entramos con el usuario y contraseña que tenemos asignada para realizar esta operación.

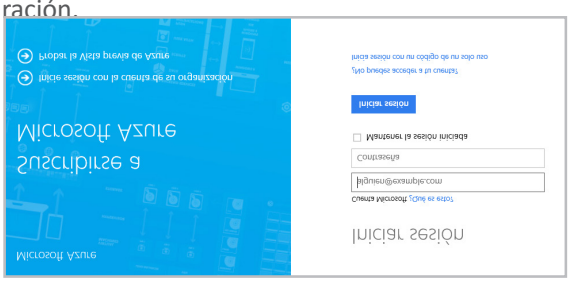


Imagen 8.- Login en Azure.

- Posteriormente procedemos a salvar el archivo que utilizaremos para la suscripción.

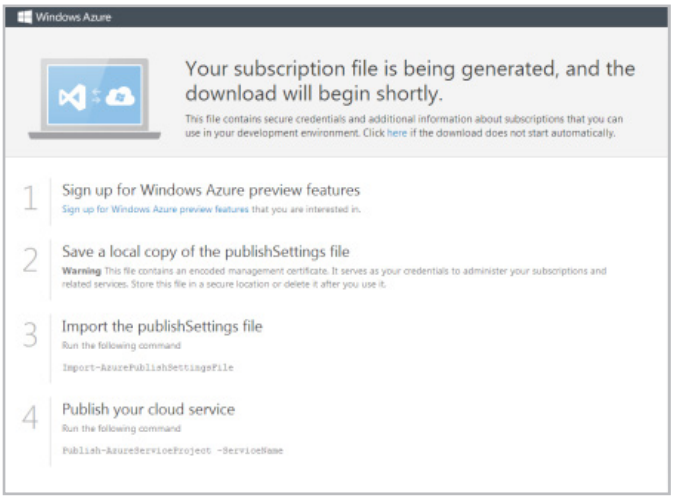


Imagen 11.- Guardar el archivo correspondiente a la suscripción de Azure.

- Si queremos establecer una conexión entre nuestra cuenta en la nube de Azure y nuestro equipo debemos instalar un certificado en nuestro PC proveniente de Azure.

```
Import-AzurePublishSettingsFile    -PublishSettingsFile
D:\V2\credentials.publishsettings
```

Donde D:\V2\ es la ubicación del archivo

- Realizamos la verificación de nuestra suscripción del comando PowerShell a nuestro pc con la sentencia: Get-AzureSubscription .
- Con la siguiente sentencia verificamos nombre de suscripción que tenemos por PowerShell:

```
Get-AzureSubscription
```

Conclusiones

En este artículo vimos como configurar nuestro pc local para poder conectarnos a Windows Azure desde el Shell de sistema, y así poder administrar desde línea de comandos cualquier aspecto de nuestra infraestructura en la nube

ENRIQUE RHENALS BÁRCENAS
MVP Office Servers and Services
MCT
erhenalsb@hotmail.com
@enriquerhenals
http://enriquerhenalsb.blogspot.com

Mentoring

Comparti*MOSS*

Un servicio experto alrededor de su SharePoint

CompartiMOSS le puede ayudar a través de su programa de Mentoring!

Contacte con nosotros y le enviaremos los planes de mentoring que tenemos disponibles para SharePoint.

38

StratusForms – ¿Adios InfoPath?

Microsoft no nos ha dado ninguna alternativa a InfoPath y por tanto nosotros seguimos buscando soluciones para poder realizar formularios de la forma más eficiente. Entre las diferentes alternativas que van apareciendo destaca StratusForms, presentando una versión gratuita y otra de pago que además de las características básicas de la versión gratuita nos permitiría enlazar con otros sistemas como SAP.

Todos nosotros hemos utilizado JavaScript para crear formularios customizando la entrada de datos en nuestras listas, lo que nos ofrece StratusForms es olvidarnos de la parte tediosa de la realización de formularios, en concreto del código de inserción y actualización de datos, con lo cual nosotros sólo nos tenemos que ocupar de la parte de UX.

Para probar el funcionamiento de StratusForms vamos a crear dos listas, Autores y Artículos y vamos a cambiar el formulario de edición por defecto de la lista de autores por otro formulario que nos permitirá editar los datos del autor y los datos de sus artículos. A continuación, se indican las columnas de cada una de las listas:

LISTA AUTORES	
Columna	Tipo
Autor	Single line of text
StratusFormsData	Multiple lines of text – Plain text

LISTA	
Columna	Tipo
Artículo	Single line of text
Descripcion	Single line of text
Autores	Lookup – apunta a la tabla Autor campo ID
StratusFormsRowID	Single line of text

StratusForms utiliza el campo StratusFormsData para guardar los cambios en la lista padre, incluyendo este campo ya podemos utilizar StratusForms en una lista sin más.

Para poder enlazar la lista hija utilizamos el campo Autores que es un lookup a la lista Autores, StratusForms nos obliga a que el nombre del campo lookup sea el mismo que la lista padre.

Si además queremos que la lista padre tenga varios elementos de la lista hija tenemos que añadir otro campo a la lista hija, StratusFormsRowId.

Estos campos que utiliza StratusForms no hace falta que los incluyamos en el formulario, los utiliza StratusForms de forma interna.

Como hemos dicho nuestra intención es poder cambiar el formulario por defecto para edición de la lista Autores por un nuevo formulario que permita además la edición de elementos de la lista dependiente Artículos y todo esto con pocas líneas de código, es decir vamos a crear un formulario para editar Autores y añadir artículos a este autor.

Nota: para crear el formulario podemos utilizar cualquier editor HTML y subirlo a nuestro SharePoint mediante SharePoint Designer, yo lo he subido a la biblioteca Site Assets.

A continuación, pongo todo el código necesario para crear un formulario de este tipo:

```
<script type=»text/javascript» src=»//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js»></script>
<script type=»text/javascript» src=»//ajax.googleapis.com/ajax/libs/jqueryui/1.10.0/jquery-ui.min.js»></script>
<script type=»text/javascript» src=»//cdnjs.cloudflare.com/ajax/libs/jquery.SPServices/0.7.2/jquery.SPServices-0.7.2.min.js»></script>
<script type=”text/javascript” src=”//www.stratusforms.com/scripts/stratus-forms-1.1.js”></script>
<script type=”text/javascript” src=”//www.stratusforms.com/scripts/stratus-forms-data-SPServices-1.1.js?rev=1”></script>
<div id=’SFForm’>
<fieldset>
<!---- HTML FOR Autor---------->
<label for=”Autor”>Nombre del autor:</label> <input id=”Autor” listFieldName=”Autor” class=”inputClass” type=”text” />
<br>
<!-------------------------------------------------->
<!---- HTML FOR Artículos---------->
<h2>Artículos:</h2>
<br>
<table>
<tr id=”repeatingRow” data-StratusFormsRepeatable=”Y” data-StratusFormsChildList=”Articulos”>
        <td>
        <div>
                <label class=”thisLabel” for=”Articulo”>Título del artículo:</label> <input id=”Articulo” listFieldName=”Articulo” type=”text” /><br>
                <label for=”Description”>Descripción del artículo:</label> <input id=”Description” listFieldName=”Description” type=”text” />
                </div>
                <br>
        </td>
</tr>
```

```
<tr>
        <td><a onclick="$0.
StratusFormsRepeat('repeatingRow')">Haga clic aquí para
crear otro artículo.</a></td>
</tr>
</table>
<!------------------------------------------------->
<div>
<input type='button' value='Submit/Update Form"
onclick='SubmitForm();'>
</div>
</fieldset>
</div>
<script>
ExecuteOrDelayUntilScriptLoaded(Init,"SP.js");
 function Init(){
//Initialization function. Tells StratusForms which Query String
Variable
//has the ID of the form, and the name of the list to read data
from
$("#SFForm").StratusFormsInitialize({
  queryStringVar: "ID",
  listName: "Autores"
});
}
function SubmitForm()
{
        //When the form is submitted store it to the specified
list
        //also pasas in the x and y offset of error messages for
elements
        //this allows you to change their location in reference
to the form field
        $("#SFForm").StratusFormsSubmit({
          listName: "Autores",
          completefunc: function(id) {
            alert("Save was successful. ID = " + id);
            window.location = window.location.pathname +
"?ID=" + id;
          }
        });
}
</script>
```

El código como veis es muy simple, pero vamos a desgranarlo para comprender el funcionamiento:

- Lo primero cargamos las librerías de jQuery desde el CDN y luego las librerías de StratusForms, en esta ocasión las cargamos de stratusforms.com pero podríamos tenerlas en local junto con el HTML de nuestro formulario.

Vamos a la parte donde definimos el formulario de StratusForms y los campos de la lista Autores

```
<div id='SFForm'>
<fieldset>
<!---- HTML FOR Autor---------------->
<label for="Autor">Nombre del autor:</label>
<input id="Autor" listFieldName="Autor" class="inputClass"
type="text" />
```

- Como vemos necesitamos una etiqueta DIV con un id que posteriormente vamos a utilizar para poner en marcha el formulario, en nuestro caso SFForm.
- Si nos fijamos en el input para el campo Autor vemos que tenemos que marcarlo con la etiqueta listFieldName para indicarle a StratusForms el nombre interno del campo en la lista. También tenemos otra etiqueta type="text" para indicarle a StratusForms como lidiar con este campo, en este artículo no vamos a entrar en detalles, pero StratusForms nos permite crear campos de tipo PeoplePicker, tipo Date, etc. y nos permite también rellenar campos de forma automática a partir de otras listas, pero como he dicho eso lo dejamos para otro artículo para no liarnos demasiado.

Ahora vamos a ver como insertamos la lista hija, para ello simplemente hay que utilizar una etiqueta HTML, por ejemplo, DIV o TR y marcarlo con el atributo data-StratusFormsChildList donde pondremos el nombre de la lista hija:

```
<tr id="repeatingRow" data-StratusFormsRepeatable="Y" data-
StratusFormsChildList="Articulos">
```

Si queremos poder tener varios ítems de la lista hija tenemos que marcar el id que engloba a la lista hija con la etiqueta data-StratusFormsRepeatable="Y", para finalizar el apartado de la lista Artículos ponemos una etiqueta href para poder crear nuevos elementos:

```
<a href="#" onclick="$0.
StratusFormsRepeat('repeatingRow');">Add Row</a>
```

Cada vez que hagamos clic en el enlace se creará un nuevo elemento en la lista hija relacionado con la tabla padre y nos aparecerá en el formulario para poder editarlo.

Y ya sólo nos queda explicar el script que aparece al final de la página y que tiene dos funciones, la primera inicializa el formulario StratusFormsInitialize y la segunda se encarga de grabar los cambios en las dos listas.

Una de las cosas que me gusta de StratusForms es que podemos seguir utilizando HTML y JavaScript tal como lo hacíamos antes, sin cambiar nuestra forma de trabajar, pero nos ahorra bastante código.

Para poder probar el formulario antes de insertarlo en nuestro formulario de edición por defecto de la tabla Autores podemos crear una nueva página en la biblioteca Site Pages y añadirle un WebPart de tipo Editor de Contenido y modificar su propiedad Content Link para que apunte a nuestro archivo, en mi caso ../SiteAssets/stFormAutores.html

Ahora vamos a insertar nuestro nuevo formulario en la lista Autores, para ello nos vamos a editar la configuración de la lista Autores -> Form web parts –> Default Edit form. Insertamos un nuevo web part de tipo Content Editor y la propiedad Content link apuntando a ../../SiteAssets/stFormAutores.html

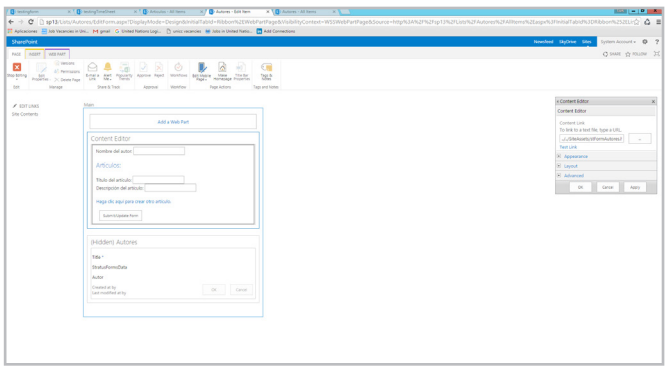


Imagen 1.- WebPart de tipo Content Editor añadido.

Solo nos queda ocultar el Web Part original para ello editamos el WebPart original y marcamos en la pestaña Layout la opción Hidden.

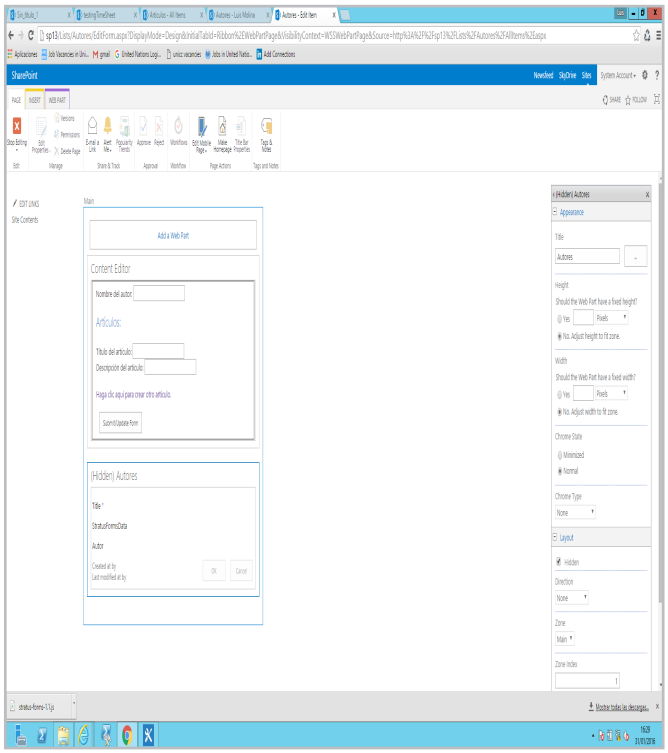


Imagen 2.- Ocultando el Web Part original del formulario.

Bien, ya tenemos nuestro nuevo formulario y podemos proceder a evaluar si todo funciona correctamente.

Imagen 3.- Prueba del formulario creado con StratusForms.

Por supuesto esta librería está empezando y todavía tiene que evolucionar, pero le veo bastante futuro, todo lo que sea ayudarnos en nuestro trabajo bienvenido sea. Podéis ampliar más información en http://www.stratusforms.com/.

LUIS MOLINA
SharePoint developer
micuentadecasa@gmail.com
http://www.devaim.com

41

Particularidades de la instalación de una Granja de varios servidores de SharePoint 2016

El proceso de instalación de SharePoint Server 2016 viene determinado por el tipo de Granja:

- Granja de servidor único, es decir, todos las Aplicaciones de Servicio y Componentes necesarios se instalan en un único servidor. Este tipo de Granja está pensado para el despliegue rápido de entornos de desarrollo o de pruebas.
- Granja de varios servidores, es decir, los distintos roles que pueden formar parte de una Granja (Frontal Web, Búsquedas, Aplicaciones, Caché y Personalizado) se configuran en varios servidores de acuerdo a la topología de Granja elegida.

La principal novedad en cuanto a tipos de Granja de SharePoint server 2016 la aporta el concepto de MinRole que aplica a un escenario de configuración y despliegue de una Granja de varios servidores. La característica de minRole de SharePoint Server 2016 permite a los administradores especificar el rol concreto que un servidor va a desempeñar en una Topología de Granja de acuerdo a los siguientes Roles disponibles:

- Frontal Web, los servidores con este Rol están optimizados para proporcionar baja latencia de manera que hospedan Aplicaciones de Servicio, Servicios y Componentes que se encargar de dar respuesta a las peticiones de usuario relativas a servidores Frontales como por ejemplo el acceso a un Sitio de la Granja, lanzar una Consulta de Búsqueda, etc.
- Aplicación, en este caso los Servidores con este Rol se han optimizado para proporcionar alto rendimiento por lo que hospedan Aplicaciones de Servicio, Servicios y Componentes que sirven peticiones de backend como por ejemplo las peticiones de rastreo del buscador.
- Caché Distribuida, hospedan Aplicaciones de Servicios, Servicios y Componentes que se requieren para la Caché Distribuida en Servidores de Caché Distribuida.
- Búsqueda, Servidores que hospedan Aplicaciones de Servicios, Servicios y Componentes que se requieren para realizar búsquedas en Servidores de Búsqueda.
- Personalizado, es decir, Servidores que hospedan las Aplicaciones de Servicio, Servicios y Componentes personalizados que están fuera de la topología de MinRole. Un ejemplo de Aplicación de Servicio que se hospedaría en un servidor con este Rol es la de Reporting Services.

Como se verá a continuación, el Rol de un servidor se especifica en el momento en el que se crea una nueva Granja o bien se une un nuevo servidor a una Granja existente. SharePoint se encarga de configurar de forma automática los servicios de cada servidor de la Granja de acuerdo al Rol asignado. Para más información sobre MinRole: https://technet.microsoft.com/EN-US/library/mt346114(v=office.16).aspx.

SharePoint Server 2016, frente a versiones previas de la plataforma, únicamente permite realizar instalaciones de tipo Granja de Servidores

Particularidades en la instalación de una Granja de varios Servidores

A continuación, se detallan las particularidades en la instalación de una Granja de varios Servidores de SharePoint 2016 (versión Release Candidate – RC) bajo la característica de minRole asumiendo las siguientes premisas de partida:

- Existe un Directorio Activo creado y pre-configurado al que previamente se han unido cada uno de los Servidores de la Granja.
- Existe un Servidor en el que se ha instalado y configurado una versión de SQL Server soportada por SharePoint Server 2016 (SQL Server 2014 o bien SQL Server 2016 CTP 3.3).
- Se han creado las cuentas de instalación y configuración necesarias para instalar y configurar SharePoint Server 2016. A modo de ejemplo, recomiendo seguir el siguiente artículo de Vlad Catrinescu como referencia de cuentas requeridas: https://absolute-sharepoint.com/2013/01/sharepoint-2013-service-accounts-best-practices-explained.html

Con estas premisas de partida, el proceso de instalación de cada uno de los Roles de la Granja es el siguiente:

- Descargar SharePoint Server 2016 Beta2 y el parche relativo a la RC de los siguientes enlaces:
  - SharePoint Server 2016 Beta 2:
  - https://www.microsoft.com/en-us/download/de-

tails.aspx?id=49961

- SharePoint Server 2016 RC:
- https://www.microsoft.com/en-us/download/details.aspx?id=50737

  Adicionalmente, para la RC necesitaremos seguir las instrucciones de instalación específicas y descargar e instalar (dependiendo del idioma base para la Beta 2) el paquete o paquetes de idiomas necesarios:

- Instrucciones de instalación de la RC:
  - https://technet.microsoft.com/library/mt674910(v=office.16).aspx
  - Paquetes de idioma para la RC:
  - https://www.microsoft.com/en-us/download/details.aspx?id=50736
- Instalación y configuración de los pre-requisitos de software necesarios para SharePoint 2016. Durante el proceso de instalación y configuración de los pre-requisitos, el servidor se re-iniciará 1-2 veces.

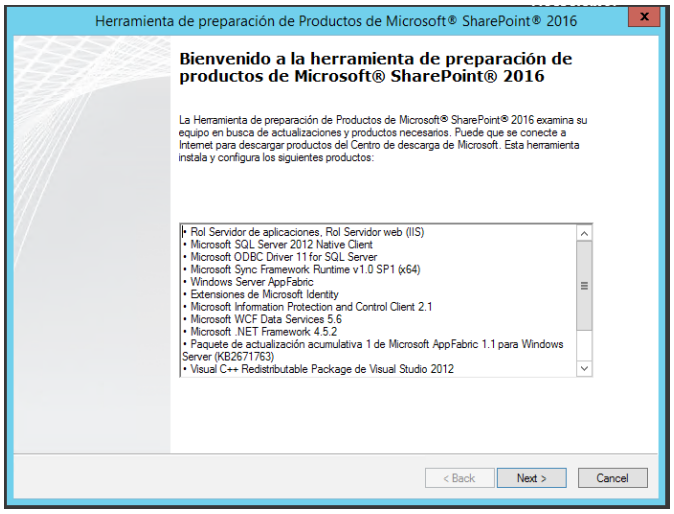


Imagen 1.- Pre-requisitos software para SharePoint 2016.

- Con los pre-requisitos instalado, podemos proceder a instalar en cada uno de los servidores de la Granja los bits de SharePoint 2016. Para la versión RC, lo que haremos es instalar SharePoint 2016 Beta2 en cada servidor y posteriormente aplicar el parche correspondiente a la RC. Como primer paso del proceso de instalación tendremos que especificar la correspondiente clave de producto de manera que se pueda iniciar la instalación de los bits en el servidor:

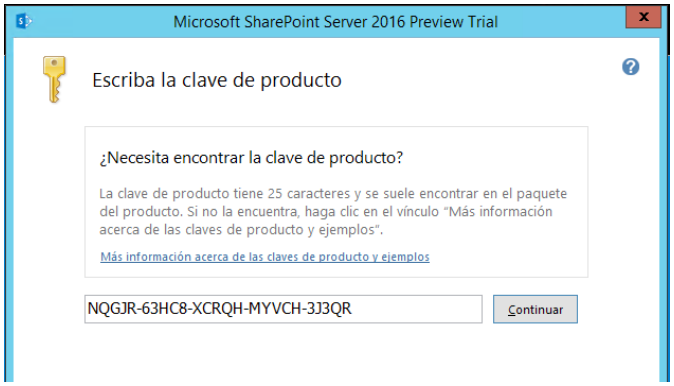


Imagen 2.- Especificando la clave de producto para SharePoint Server 2016.

- Como en versiones previas, el proceso de instalación de los bits de SharePoint Server 2016 en el primer servidor de la granja concluye mostrando la pantalla "Ejecutar el Asistente para configuración" con la opción "Ejecutar el Asistente para configuración de Productos de SharePoint en este momento" seleccionada. Si se hace clic en el botón "Cerrar", se iniciará el asistente de configuración.

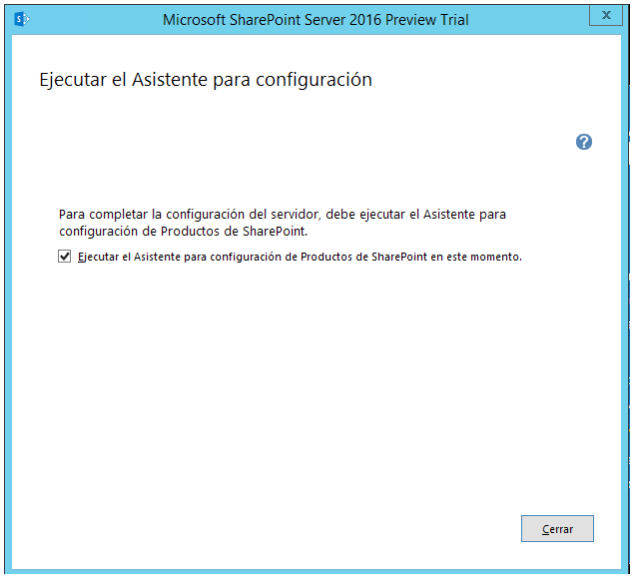


Imagen 3.- Ventana "Ejecutar el Asistente para configuración".

- En la primera pantalla del asistente, presione "Siguiente" de forma que se muestre el correspondiente mensaje de advertencia que informa sobre los servicios se van a reiniciar o restablecer durante la configuración. A continuación, la ventana "Conectar a un conjunto de servidores" nos permite indicar si el servidor actual va a formar parte de una Granja de servidores existente o bien se va a crear una nueva Granja. En este caso, como se está instalando el primer servidor de la Granja, seleccione la segunda opción y haga clic en "Siguiente".
- La siguiente pantalla permite especificar los parámetros relativos a la BD de Configuración que se requiere en la Granja:

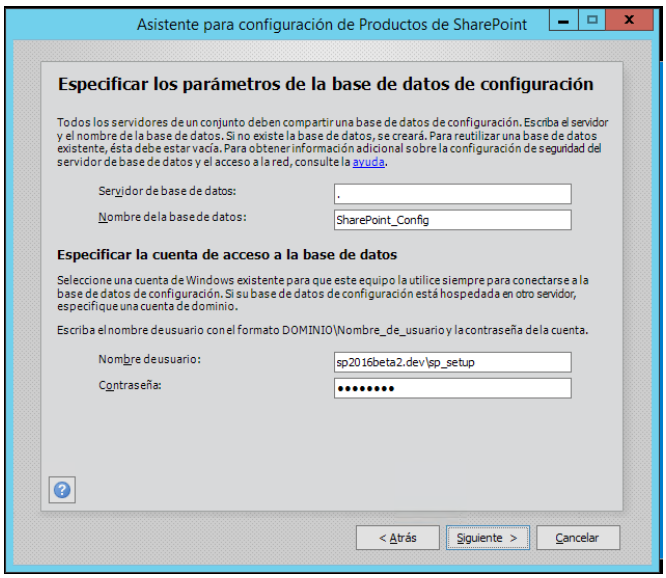


Imagen 4.- Parámetros de la BD de configuración.

- A continuación, especifique la frase de contraseña. Es importante recordar / guardar la misma puesto que será necesaria para poder añadir a la Granja el resto de Roles.
- En la pantalla especificar el Rol del servidor es dónde empiezan las particularidades de SharePoint Server 2016 a la hora de desplegar y configurar una Granja de varios servidores: es necesario elegir el Rol en la Granja del Servidor que se está configurando. Como este servidor es el primero que se configura en la Granja, es recomendable elegir el Rol de Aplicación lo que implica, entre otros, que este servidor va a hospedar la Administración Central de SharePoint 2016 y todos las Aplicaciones de Servicio de la plataforma que se requieran a excepción de la Aplicación de Servicio de Búsquedas.

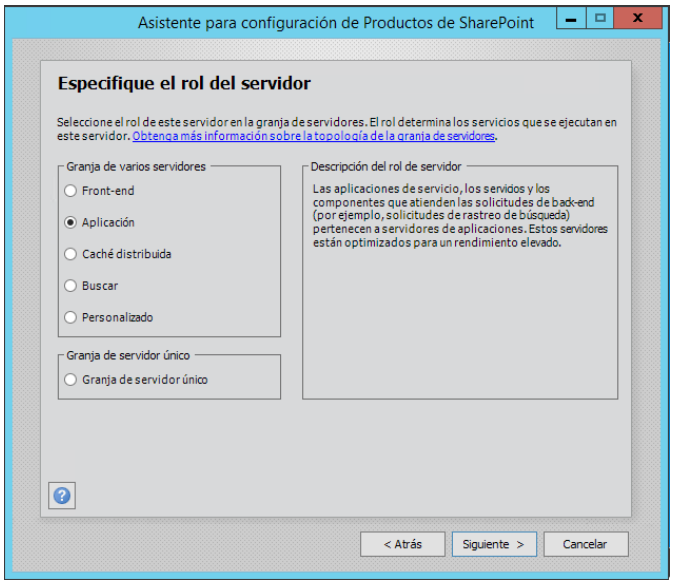


Imagen 5.- Selección del Rol de Aplicación para el servidor.

- Los dos últimos pasos del asistente, previos al inicio del asistente de configuración, permiten por un lado configurar el puerto para la Administración Central de SharePoint 2016 y el método de autenticación (opción NTLM por defecto) y visualizar el resumen de las parametrizaciones que se van a realizar. A partir de aquí, se inicia el proceso de configuración que, si todo va bien, concluye lanzando la Administración Central para iniciar el Asistente de configuración de las Aplicaciones de Servicio que hospedará el servidor en la Granja.

La principal novedad en cuanto a tipos de Granja de SharePoint server 2016 la aporta el concepto de MinRole

Una vez que se ha configurado el primer servidor de la Granja con el Role de Aplicación, el proceso para agregar el resto de servidores y el resto de Roles consiste en seguir los siguientes pasos que se repiten por cada nuevo Role a añadir y configurar:

- En la ventana "Conectar a un conjunto de servidores" indicaremos que el servidor actual va a formar parte de una Granja de servidores existente.
- A continuación, la ventana "Especificar los parámetros de la base de datos de configuración" permite indicar el nombre del Servidor de BD y validar la misma mediante el botón "Recuperar nombres de bases de datos" que permite cargar el nombre de la BD de Configuración.

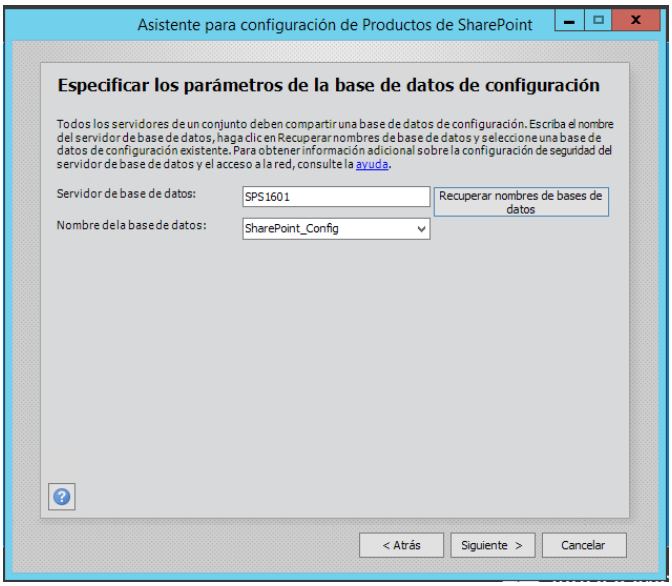


Imagen 6.- Ventana "Especificar los parámetros de la base de datos de configuración" en el segundo y siguientes servidores a configurar en la Granja.

Nota: Puede ocurrir que al intentar recuperar la BD de configuración del Servidor de BD se muestre un error indicando que no es posible realizar la comunicación con el servidor. Este problema puede ser debido a que en el servidor de BD no se ha configurado el Firewall de Windows para abrir los puertos requeridos tal y como se detalla en el siguiente artículo de Microsoft TechNet: https://msdn.microsoft.com/en-us/library/cc646023.aspx

Los distintos roles que pueden formar parte de una Granja (Frontal Web, Búsquedas, Aplicaciones, Caché y Personalizado)

- En la ventana "Especificar configuración de seguridad del conjunto de servidores" especifique la frase de contraseña configurada al instalar el Rol de "Aplicación" y haga clic en "Siguiente" de forma que se muestre la pantalla "Especifique el rol del servidor". Seleccione en este caso "Buscar" y presione "Siguiente".
- La pantalla la pantalla "Finalizando el asistente para configuración de Productos SharePoint" muestra el resumen de las configuraciones que se van a realizar en el servidor y además dispone de un botón "Configuración avanzada" que permite indicar si el servidor que se está configurando va a hospedar o no la Administración Central de SharePoint 2016 (la opción por defecto es No).

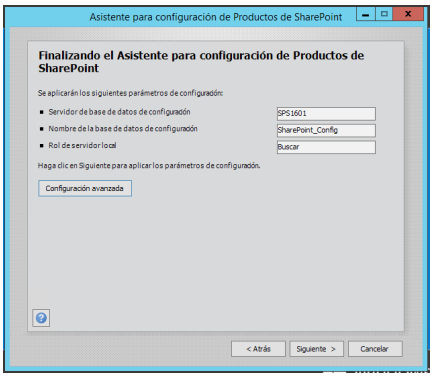


Imagen 7.- Ventana "Finalizando el Asistente para configuración de Productos SharePoint".

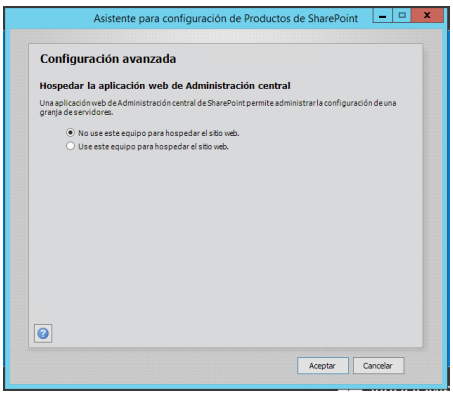


Imagen 8.- Ventana de Configuración avanzada que se abre desde "Finalizando el Asistente para configuración de Productos SharePoint".

- De vuelta a la pantalla de "Finalizando el asistente para configuración de Productos SharePoint", haga clic en "Siguiente" para que se inicie la configuración de SharePoint en el servidor actual.

Una vez se ha configurado el segundo servidor de la Granja, el proceso a seguir para el resto de servidores es idéntico al que se ha detallado para el Rol de "Buscar".

Conclusiones

SharePoint Server 2016 únicamente permite realizar instalación y configuración en modo Granja de Servidores que a su vez incluye dos tipos de instalación: Granja de un único servidor y Granja de varios servidores. En el primer caso, todas las Aplicaciones de Servicio y Componentes necesarios se instalan en un mismo servidor. En el segundo, los distintos roles que pueden formar parte de una Granja (Frontal Web, Búsquedas, Aplicaciones, Caché y Personalizado) de SharePoint se configuran en varios servidores de acuerdo a la topología de Granja elegida

JUAN CARLOS GONZÁLEZ MARTÍN
Office Servers and Services MVP
Cloud & Productivity Advisor en MVP CLUSTER
jcgonzalezmartin1978@hotmail.com
@jcgm1978 | https://jcgonzalezmartin.wordpress.com/

45

Conociendo a MinRole de SharePoint 2016

Una de las primeras novedades de SharePoint 2016 con la que nos tendremos que enfrentar es el nuevo concepto de la estrategia MinRole, que estará presente desde que comencemos a configurar nuestra primera granja de servidores.

Durante los últimos años ofreciendo el servicio de Office 365, Microsoft ha identificado muchas restricciones o problemas de gestión a nivel de infraestructura cuando ha querido diseñar grandes granjas de SharePoint que pudieran ofrecer distintos servicios a millones de usuarios. En numerosas ocasiones, la tradicional división entre servidores frontal web y servidores de aplicación ha sido insuficiente, con multitud de servicios duplicados entre estos dos tipos de servidor, así como falta de optimización de recursos hardware y problemas de configuración que esta organización supone.

los grandes cambios a nivel de infraestructura que incorpora SharePoint Server 2016 es la nueva organización de servidores por roles

Es por ello que SharePoint 2016 implementa el nuevo concepto de MinRole, que permite asociar roles concretos a los distintos servidores de la granja y así optimizar al máximo el rendimiento y conseguir un diseño, administración y escalado más cómodo. Las principales mejoras de MinRole son:

- Se evitan problemas de configuración. Anteriormente, la posibilidad de distribuir los distintos servicios en los distintos servidores, independientemente del rol, provocaba multitud de problemas técnicos en las granjas de SharePoint.
- Se consigue mejoras notables en el rendimiento y fiabilidad del servicio. Gracias a la experiencia conseguida por Microsoft manejando el servicio de Office 365 durante los últimos años se ha conseguido que la distribución de servidores por roles optimice al máximo la gestión de recursos como memoria, E/S de disco y latencia de red.
- Se logra un escalado más sencillo y la minimización del tiempo de downtime cuando se realizan actualizaciones en la granja.

Cuando se configure un servidor para entrar en la granja, el asistente de configuración de SharePoint solicitará el rol que llevará dicho servidor (Imagen 1).

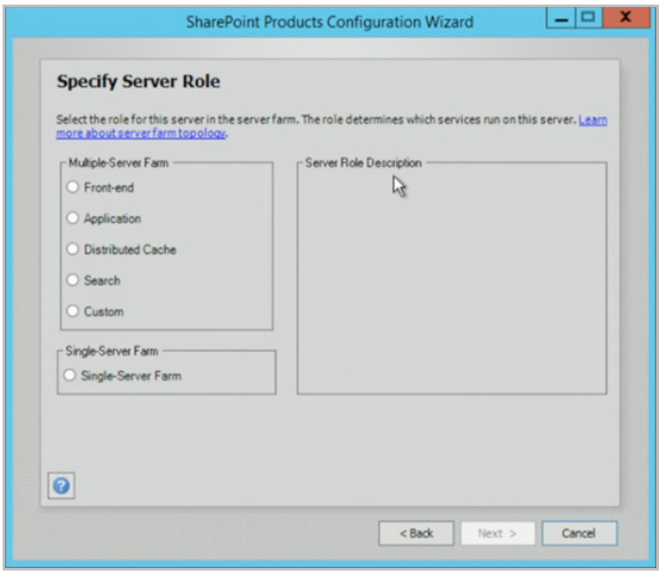


Imagen 1.- El asistente de configuración de la granja solicita el rol de cada servidor.

Los roles que se pueden configurar son:

- Front-end: frontal web que servirá las aplicaciones web de SharePoint. Únicamente aceptará peticiones web de los usuarios. No hospedará servicios.
- Application: contendrá distintos servicios y aplicaciones de servicio de SharePoint que requieran peticiones al backend (excepto caché distribuida y búsquedas).
- Distributed Cache: hospedará el servicio de cache distribuida.
- Search: hospedará el servicio de búsquedas (administración, index, rastreo, consulta).
- Custom (o Special Load): destinado a servicios, aplicaciones de servicio y componentes personalizados que no se integran con la estructura MinRole.
  - El uso de este rol es principalmente por retro compatibilidad. Es posible diseñar una granja tradicional usando servidores con el rol “custom”, ya que nos permitirá configurar aplicaciones, servicios y aplicaciones de servicio a nuestro antojo.
- Single Server Farm: solo para desarrollo y pruebas. Instalación de todos los componentes en una única máquina.

El uso de MinRole permite no sólo diseñar granjas con todos los servicios (frontal web, aplicaciones de servicio, caché y búsqueda), sino que facilita en gran medida la creación de granjas dedicadas a un único servicio (por ejemplo, granja de búsquedas, granja con varios servicios compartidos, granjas de metadatos…).

La posibilidad de contar con granjas dedicadas es muy necesaria principalmente en entornos SharePoint muy grandes, en los que determinados servicios requieren un uso intensivo de recursos hardware. Estos servicios se pueden ejecutar en una granja central para minimizar la sobrecarga de la administración y escalar horizontalmente de forma fácil y eficiente según crecen las necesidades. En estos casos MinRole reluce porque permite aprovechar al máximo los recursos de las granjas dedicadas al rol en concreto que queremos.

El diseñar una granja con una estrategia basada en MinRole precisa disponer obligatoriamente de determinados roles. La siguiente tabla muestra los roles obligatorios según el tipo de granja que se desee diseñar (granja para contenido, granja de servicios compartidos o granja de servicio de búsqueda):

SERVIDOR	¿REQUERIDO PARA GRANJA DE CONTENIDO?	¿REQUERIDO PARA GRANJA DE SERVICIOS COMPARTIDOS?	¿REQUERIDO PARA GRANJA DE BÚSQUEDA?
Servidor frontal web	Sí	Sí	No
Servidor de aplicación	Sí	Sí	No
Caché distribuida	Sí	Sí	No
Búsqueda	Sí, si se implementan las búsquedas	Sí, si se implementan las búsquedas	Sí
Custom	Opcional	Opcional	Opcional

El nuevo modelo de MinRole proporciona nuevas funcionalidades en la Administración Central de SharePoint 2016 que permiten gestionar los distintos servidores y servicios.

La página de servidores en la granja (Imagen 2) muestra dos nuevas columnas para cada servidor:

- Role: indica el rol que se le ha asignado a cada servidor de la granja.
- In Compliance: detalla si el servidor está cumpliendo únicamente con su rol. Si no es así, aparecerá un botón que permite repararlo para hacer que cumpla con la configuración que se espera de su rol.

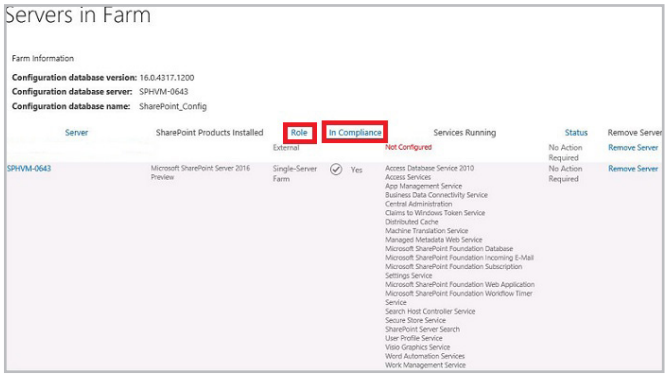


Imagen 2.- Página de servidores en la granja con las mejoras de MinRole.

Aparece una nueva página llamada servicios en la granja (Imagen 3) que muestra el estado de cada servicio de la granja. En esta página, aparecen tres columnas para cada servicio:

- Auto Provision: muestra si el servicio está arrancado en el servidor correcto. Si el valor es Yes, las instancias para este servicio están correctamente arrancadas en los servidores que le corresponden de la granja. Si el valor es No, hay instancias para este servicio que están paradas en los servidores que le corresponden.
- Action: dependiendo del tipo de servicio y si está habilitado mostrará una acción posible:
  - Manage Service Application: indica que el servicio está asociado a una aplicación de servicio y que se debe arrancar o apagar desde la aplicación de servicio. El enlace lleva a la app de servicio.
  - Disable Auto Provision: desactiva el servicio en la granja.
  - Enable Auto Provision: activa el servicio en la granja.
- In Compliance: detalla si el servicio está en cumpliendo su función y que está habilitado en los servidores correctos. Si hay un problema, aparece un botón de reparación.

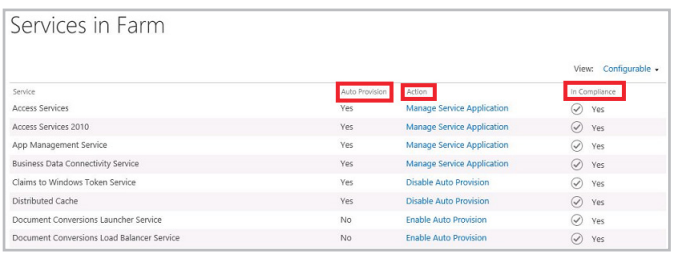


Imagen 3.- Nueva página de servicios en la granja para MinRole.

La página de servicios en el servidor (Imagen 4) muestra una nueva columna y cambia el funcionamiento de otra:

- In Compliance: detalla si el servicio está arrancado en el servidor y si es el servidor en el que se corresponde. En el caso de estar parado, o pertenecer a otro rol, aparece un botón de reparación.
- Action: se cambia su funcionamiento. Al tener un servidor dedicado a un rol, sus servicios deben estar en todo momento arrancados, con lo que no tiene sentido parar un servicio. Ahora sólo permite reiniciar el servicio. Si se necesitara parar o arrancar un servicio por algo, hay que hacerlo en la página de servicios en la granja mediante el "Enable Auto Provision" o el "Disable Auto Provision" mencionados anteriormente.

Imagen 4.- Página de servicios en el servidor con las mejoras de MinRole.

Si se deseara cambiar el rol de un servidor, se puede completar esta tarea desde la nueva página de conversión de roles (Imagen 5):

Imagen 5.- Nueva página de conversión de roles.

Además, aparece una nueva regla en el Health Analyzer de SharePoint que comprueba si la configuración de rol del servidor es correcta:

Imagen 6.- Nueva regla en el Health Analyzer para MinRole.

También se dispone de nuevos comandos PowerShell para administrar la configuración de roles y servicios en los roles:

NOMBRE	DESCRIPCIÓN	EJEMPLO DE SINTAXIS
Get-SPService	El cmdlet Get-SPService obtiene un servicio en la granja.	Get-SPService -Identity "Microsoft SharePoint Foundation Sandboxed Code Service"
Start-SPService	El cmdlet Start-SPService habilita un servicio en la granja. Las instancias de servicio para este servicio se inician en los servidores correspondientes en la granja.	Start-SPService -Identity "Microsoft SharePoint Foundation Sandboxed Code Service"
Stop-SPService	El cmdlet Stop-SPService deshabilita un servicio en la granja. Las instancias de servicio para este servicio se detienen en los servidores correspondientes en la granja.	Stop-SPService -Identity "Microsoft SharePoint Foundation Sandboxed Code Service"

Es importante conocer en profundidad estos nuevos conceptos, ya que van a ser clave de aquí en adelante para cualquier nueva instalación de SharePoint Server 2016. Recomiendo familiarizarse cuanto antes con ellos y hacer pruebas de creación de granjas en laboratorio. Para estas pruebas, os recomiendo el uso de los scripts AutoSPInstaller, que ya soportan el uso de MinRole.

MIGUEL TABERA PACHECO
SharePoint Lead en Plain Concepts
MVP de Office Servers and Services
miguel.tabera@outlook.com
@migueltabera
http://docs.com/migueltabera

48

Usando Azure Search en soluciones de búsqueda para Office 365

Azure ofrece una amplia gama de servicios para los desarrolladores, entre los que se encuentra Azure Search. Este servicio, salido recientemente de su versión preview, permite incorporar una solución completa de búsqueda a las aplicaciones. En este artículo, se va a realizar un overview de este servicio, para adquirir unas nociones básicas de los pasos que hay que seguir para utilizarlo, pero, además, combinándolo con otra de las herramientas que recientemente Microsoft ha puesto a disposición de los desarrolladores, la API Microsoft Graph. En el artículo se tratará:

- Configuración y creación de índices en el servicio Azure Search.
- Configuración de un webjob para usar la API Microsoft Graph y acceder al contenido de Office 365.
- Indexación de contenido de Office 365 en Azure Search.

El servicio Azure Search

El servicio Azure Search permite a través de una API REST o una SDK de .NET implementar las búsquedas sin gestionar la infraestructura o llegar a ser un experto en búsquedas.

Lo primero que se debe hacer, es aprovisionar el servicio en la subscripción de Azure. Para ello, en el nuevo portal, hay que acceder a Nuevo->Datos y almacenamiento->Búsqueda de Azure. A continuación, se deben indicar los datos básicos de configuración del servicio: el nombre, grupo de recursos al que pertenece, la ubicación y el nivel de precios.

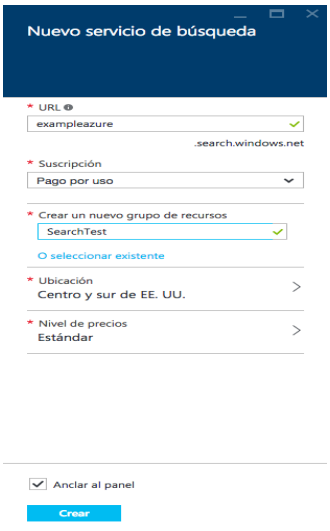


Imagen 1.- Parámetros de configuración del servicio Azure Search.

Una vez configurados los parámetros, se pulsa sobre Crear y el servicio se aprovisionará.

Creando un índice en Azure Search

Una vez aprovisionado el servicio de Azure Search se deben crear los índices en el mismo. Un índice es el medio principal para organizar y buscar documentos en el servicio de búsqueda de Azure, similar a cómo se organizan los registros en una tabla de una base de datos. Cada índice tiene una colección de documentos que cumplen el esquema de índice (nombres de campo, tipos de datos y propiedades), pero los índices también especifican construcciones adicionales (proveedores de sugerencias, perfiles de puntuación y configuración de CORS) que definen otros comportamientos de la búsqueda.

Por tanto, es necesario, como paso previo a subir documentos al servicio de búsqueda, definir un índice y su estructura, que no es más que una estructura de tablas que acepta datos y consultas. Los índices en el servicio de búsqueda de Azure, se pueden crear de tres formas, o desde el propio portal de Azure o por medio de código usando una SDK de .NET o una API REST.

Para crear un índice, en este caso, se usará una aplicación de consola en Visual Studio y la SDK de .NET. Por tanto, será necesario añadir las referencias a la SDK. Esto, se consigue ejecutando en la consola de Nuget el siguiente comando:

```
Install-Package Microsoft.Azure.Search -Pre
```

La creación de un índice por medio de la SDK de Azure Search consta de cuatro pasos:

1.– Conectarse al servicio: por medio de la url y la api-key que se obtiene al aprovisionar el servicio.
2.– Comprobar si el índice ya existía previamente y en ese caso, borrarlo.
3.– Crear la estructura del índice
4.– Añadir el índice a nuestra instancia de Azure Search

Este servicio proporciona a las aplicaciones una experiencia de búsqueda

El código de la aplicación de consola que permite hacer esto, puede ser como el que el siguiente:

```
class Program
{
static void Main(string[] args)
{
            //step one
            SearchServiceClient serviceClient =
GetSearchServiceClient();
            //step two
            DeleteIndexIfExists(serviceClient, "indexName");
            //step three and four
            CreateIndexUsingSDK(serviceClient, "indexName");
}

private static SearchServiceClient GetSearchServiceClient()
{
string serviceName = ConfigurationManager.
AppSettings["SearchServiceName"];
string apiKey = ConfigurationManager.AppSettings["ApiKey"];

SearchServiceClient serviceClient = new
SearchServiceClient(serviceName, new
SearchCredentials(apiKey));

return serviceClient;
}

private static void DeleteIndexIfExists(SearchServiceClient
serviceClient, string indexName)
{
if (serviceClient.Indexes.Exists(indexName))
serviceClient.Indexes.Delete(indexName);
}

private static void
CreateIndexUsingNetSDK(SearchServiceClient serviceClient,
string indexName)
{
            //Index Schema Definition
var index = new Index()
{
Name = indexName,
Fields = new[]
{
new Field("ContactId", DataType.String) {IsKey = true,
IsRetrievable = true, IsFilterable = true},
new Field("Name",DataType.String) { IsRetrievable = true,
IsSearchable=true, IsFilterable = true},
new Field("Job", DataType.String) { IsRetrievable = true,
IsSearchable = true, IsFilterable = true},
new Field("Department", DataType.String) { IsRetrievable = true,
IsSearchable = true, IsFilterable = true},
new Field("Email", DataType.String) { IsRetrievable = true,
IsSearchable = true, IsFilterable = true}
}
 };

//Adding Index in Azure Search
serviceClient.Indexes.Create(index);
}
}
```

Ejecutando esta aplicación de consola, se creará el índice en Azure Search. Si posteriormente, se accede al portal de Azure, al servicio de búsqueda que se había aprovisionado, se observará como se ha añadido un nuevo índice en el mismo.

Indexando contenido de Office 365 en Azure Search

Una vez creado el índice, el siguiente paso es añadir contenido al mismo. Para la indexación de contenido de Office 365, que es el objetivo de este artículo, se creará una aplicación de consola, que se ejecutará como un WebJob de Azure y que usará la nueva API Microsoft Graph y de nuevo la SDK de .NET para acceder al servicio e indexar información de Office 365.

El flujo habitual de autenticación para usar la API Microsoft Graph en las aplicaciones se basa en el protocolo OAuth2. Este protocolo, parte de un token que hace una delegación de una serie de permisos específicos para un usuario concreto. Para obtener dicho token, es necesario que el usuario se identifique en Office 365 al menos una vez, por lo que tiene que introducir sus credenciales en una pantalla de autenticación que se muestre.

Sin embargo, en determinadas aplicaciones este flujo de autenticación no es posible. Por ejemplo, aplicaciones que ejecutan tareas en background y en las que no hay un usuario que interaccione como un WebJob de Azure. Para este tipo de aplicaciones, el protocolo OAuth2 proporcionar un flujo adicional de autenticación. Este flujo es el que se va a utilizar en la aplicación de consola que se va a desarrollar en esta ocasión. Este flujo de autenticación está actualmente soportado para obtener acceso a las API de contactos, mail y calendario. Sin embargo, aún no está disponible para acceder, por ejemplo, a APIs como la de OneDrive.

A continuación, se va a indicar el proceso de configuración del WebJob para que tenga acceso al contenido de Office 365 y pueda indexar dicha información.

1.– Crear el certificado

Para comenzar, se tiene que crear un certificado. Al tratarse de un entorno de pruebas, se puede utilizar un certificado auto-firmado, para lo que se usará la aplicación makecert.exe de la siguiente forma (sustituyendo los valores Tenant y AppName por los valores correspondientes):

```
makecert -r -pe -n "CN=Tenant AppName Cert" -b 1½5/2015 -e
1½5/2017 -ss my -len 2048
```

A continuación, se exportará dicho certificado en formato PFX y CER. Para ello primero se abre la aplicación Microsoft Management Console (mmc.exe). Una vez aquí, se accede a Archivo->Agregar o quitar complemento y ahí se añade la opción certificados.

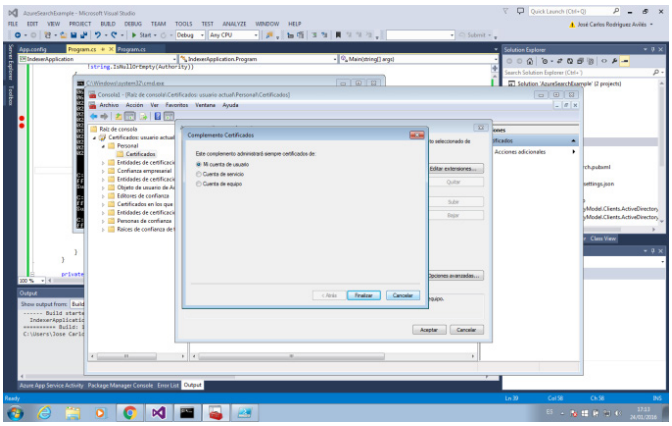


Imagen 2.- Aplicación Microsoft Management Console. Configuración del complemento de certificado.

Tras esto, se busca el certificado que se acaba de crear y se hace clic sobre el botón derecho del mismo seleccionando Todas las tareas->exportar.

Habrá disponibles dos opciones de exportación, "Exportar la clave privada" y "No exportar la clave privada", para obtener los dos formatos del certificado, será necesario ejecutar la exportación para cada una de las opciones y completar, en ambos casos, el asistente.

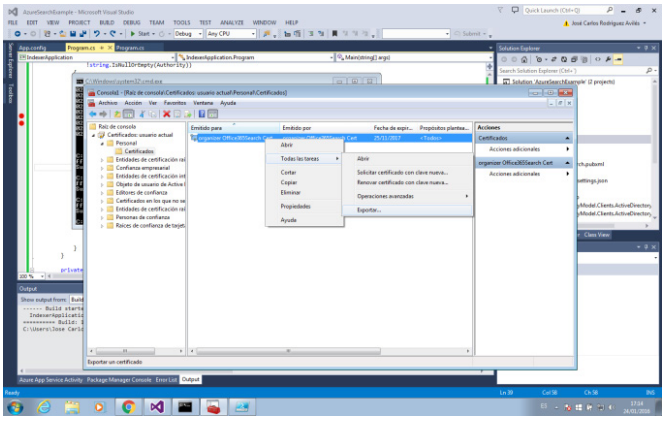


Imagen 3.- Exportación de los certificados en formato CER y PFX.

2.– Crear y configurar los permisos de la aplicación en AAD

El siguiente paso es acceder a Azure Active Directory, registrar aquí una nueva aplicación y conceder los permisos oportunos a la misma. Para ello, en la pestaña "aplicaciones" se hace click en "Agregar", tras lo cual, se deberán completar los pasos de configuración de la aplicación.

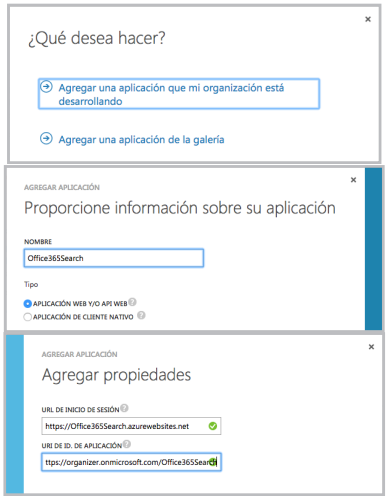


Imagen 4.- Registros de una aplicación en Azure Active Directory.

Una vez creada la aplicación, en la sección "configuración", se debe acceder a "permisos para otras aplicaciones" y una vez ahí "Agregar aplicación". En la ventana que aparece, habrá que añadir permisos para Microsoft Graph.

autenticación para usar la API Microsoft Graph en las aplicaciones se basa en el protocolo OAuth2

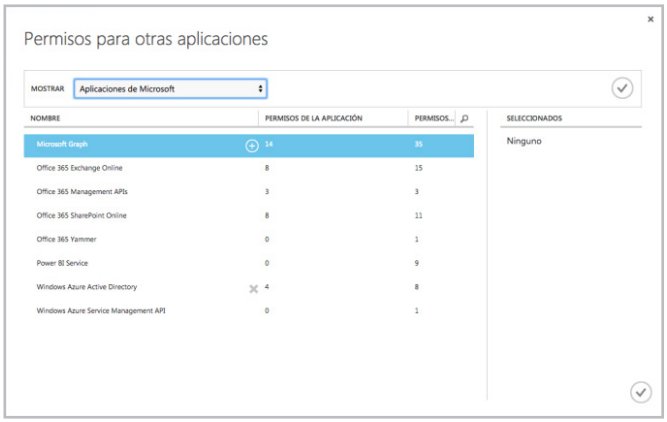


Imagen 5.- Agregar permisos para Microsoft Graph en una aplicación de AAD.

Tras añadirlos, en el desplegable "Permisos de la aplicación", ya que se van a conceder los permisos directamente a la aplicación, hay que seleccionar a qué aspectos de Microsoft Graph se desea conceder acceso.

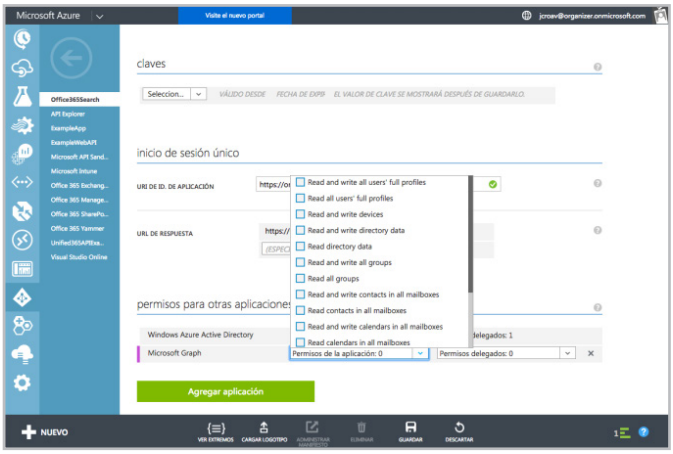


Imagen 6.- Concesión de permisos a la aplicación registrada.

Para terminar de configurar los permisos de la aplicación, habrá que pulsar el botón "Guardar".

3.– Cargar el certificado en AAD

En la aplicación de Azure Active Directory que se acaba de crear, hay que configurar el certificado que se generó en el paso 1. Esto no se puede hacer a través de una interfaz gráfica, por lo que habrá que hacerlo modificando el manifiesto de la aplicación. Para ello, lo primero que se debe hacer, es recuperar las claves del certificado CER que se generó usando el siguiente script de PowerShell

```
$certPath = Read-Host "Enter certificate path (.cer)"
$cert = New-Object System.Security.Cryptography.
X509Certificates.X509Certificate2 $cert.Import($certPath)
$rawCert = $cert.GetRawCertData()
$base64Cert = [System.Convert]::ToBase64String($rawCert)
$rawCertHash = $cert.GetCertHash()
$base64CertHash = [System.
Convert]::ToBase64String($rawCertHash)
$KeyId = [System.Guid]::NewGuid().ToString()

Write-Host "base64Cert:" $base64Cert
Write-Host "base64CertHash:" $base64CertHash
Write-Host "KeyId:" $KeyId
```

Una vez obtenidas las claves, se descarga el manifiesto y sustituimos

"keyCredentials": []

por

```
"keyCredentials": [
{ "customKeyIdentifier": "base64CertHash",
  "keyId": "KeyId",
  "type": "AsymmetricX509Cert",
  "usage": "Verify",
  "value": "base64Cert"
}]
```

Sustituyendo los valores "base64CertHash", "KeyId" y "base64Cert", por la cadena correspondiente obtenida de la ejecución del script powershell anterior.

4.– Cargar el certificado en un sitio web de Azure

El WebJob se va a desplegar en un sitio web de Azure, al que se deberá agregar el certificado que se ha obtenido en el paso 1. Por tanto, antes de nada, se deberá crear un sitio web de Azure, haciendo click, en el antiguo portal de Azure, en Nuevo->Proceso->Aplicación web->Creación rápida.

Una vez aprovisionado, lo primero será promocionar el sitio web creado a un sitio de nivel Básico en la pestaña ESCALAR. A continuación, en la pestaña CONFIGURAR, se hace clic en la opción cargar certificado y se selecciona el certificado con extensión PFX.

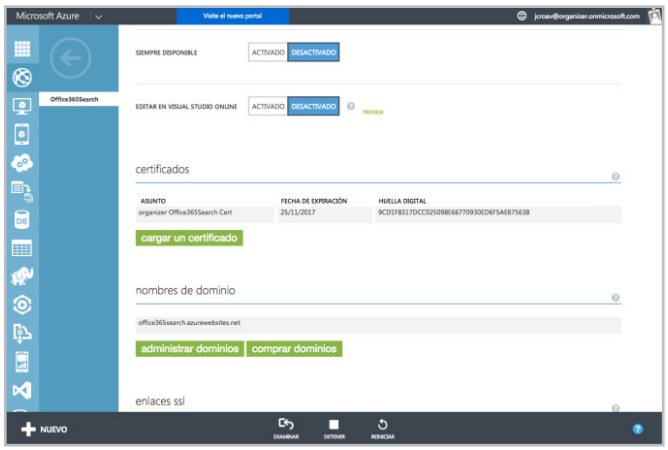


Imagen 7. Resultado de agregar el certificado digital en un sitio web de Azure

Una vez cargado, se obtiene la huella digital de ese certificado que se añadirá como clave-valor en la sección "configuración de la aplicación".

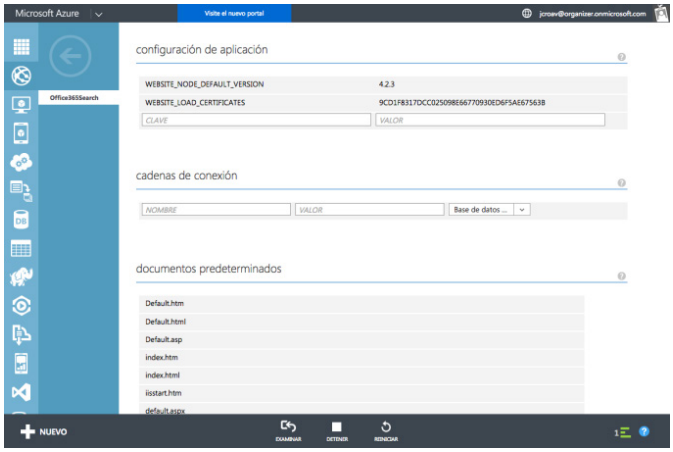


Imagen 8.- Configuración del sitio web de Azure con el certificado.

Una vez realizada toda la configuración en el portal de Azure, se puede proceder a codificar la aplicación de consola que se desplegará como WebJob en Azure y que se encargará de la indexación de contenidos, en este caso, de Office 365 por medio de la API Microsoft Graph.

Los pasos que hay que seguir para llevar a cabo esta tarea son los siguientes:

1.– Obtener el token de autenticación que concede los permisos de acceso para la aplicación de consola.
2.– Hacer la llamada a Microsoft Graph para obtener los contactos.
3.– Crear el array de contactos.
4.– Añadir la información a Azure Search.

El código de la aplicación de consola se puede ver a continuación:

```
class Program
{
private static readonly string GraphUrl = ConfigurationManager.
AppSettings["GraphUrl"];
private static readonly string ClientId = ConfigurationManager.
AppSettings["ClientId"];
private static readonly string Authority = ConfigurationManager.
AppSettings["Authority"];
private static readonly string Thumbprint =
ConfigurationManager.AppSettings["Thumbprint"];

static void Main(string[] args)
{
                // STEP ONE
// Retrieve the certificate
var certificate = GetCertificate();
// Get an access token
var token = GetAccessToken(certificate);

//STEP TWO
// Fetch the contact
var client = new RestClient(GraphUrl);
var request = new RestRequest("/v1.0/users/<userid>/contacts",
Method.GET);          request.AddHeader("Authorization",
"Bearer " + token.Result);          request.AddHeader("Content-
Type", "application/json");          request.AddHeader("Accept",
"application/json");

var response = client.Execute(request);
var content = response.Content;

//STEP THREE
//deserialize json object
dynamic contacts = JObject.Parse(content);

List<Contacts> contactsToPopulate = new List<Contacts>();
int count = 1;

foreach (var contact in contacts.value)
{
Contacts contactToAdd = new Contacts
{
ContactId = count.ToString(),
Name = contact.displayName,
Department = contact.department,
Job = contact.jobTitle,
Email = contact.emailAddresses[0].address          };

contactsToPopulate.Add(contactToAdd);
count++;
}

//STEP FOUR
//Populate the content
SearchIndexClient indexClient = GetSearchIndexClient();
var batch = IndexBatch.Upload(contactsToPopulate.ToArray());
indexClient.Documents.Index(batch);
```

```
Thread.Sleep(3000);
}
private static X509Certificate2 GetCertificate()
{
X509Certificate2 certificate = null;
var certStore = new X509Store(StoreName.My, StoreLocation.
CurrentUser);
certStore.Open(OpenFlags.ReadOnly);
var certCollection = certStore.Certificates.Find(X509FindType.
FindByThumbprint, Thumbprint, false);
// Get the first cert with the thumbprint
if (certCollection.Count > 0)
{
certificate = certCollection[0];
}
certStore.Close();
return certificate;
}
private static async Task<string>
GetAccessToken(X509Certificate2 certificate)
{
var authenticationContext = new
AuthenticationContext(Authority, false);
var cac = new ClientAssertionCertificate(ClientId, certificate);
var authenticationResult = await authenticationContext.
AcquireTokenAsync(GraphUrl, cac);

return authenticationResult.AccessToken;
}
private static SearchIndexClient GetSearchIndexClient()
{
string serviceName = ConfigurationManager.
AppSettings[“SearchServiceName”];
string apiKey = ConfigurationManager.AppSettings[“ApiKey”];
SearchIndexClient indexClient = new
SearchIndexClient(serviceName, “contacts”, new
SearchCredentials(apiKey));

return indexClient;
}
}
```

Este ejemplo, está preparado para ejecutar una vez a modo de prueba. Si se quisiera llevar a un entorno en el que el WebJob se ejecutara de forma recursiva, al código del mismo habría que añadir la lógica correspondiente para, o bien eliminar el contenido indexado e indexarlo todo de nuevo o añadir solo los nuevos elementos.

El paso final es desplegar esta aplicación de consola como WebJob. Para ello, sobre la solución hacemos click en botón derecho y seleccionamos la opción Publish as Azure WebJobs

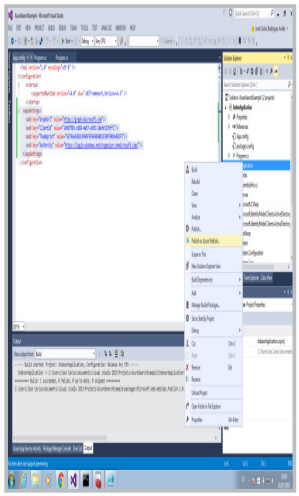


Imagen 9.- Opción de publicación de una Aplicación de consola como WebJob de Azure.

A continuación, se completan los distintos pasos necesarios para la publicación del WebJob: indicar la planificación, seleccionar la aplicación web donde se quiere desplegar y publicar la solución:

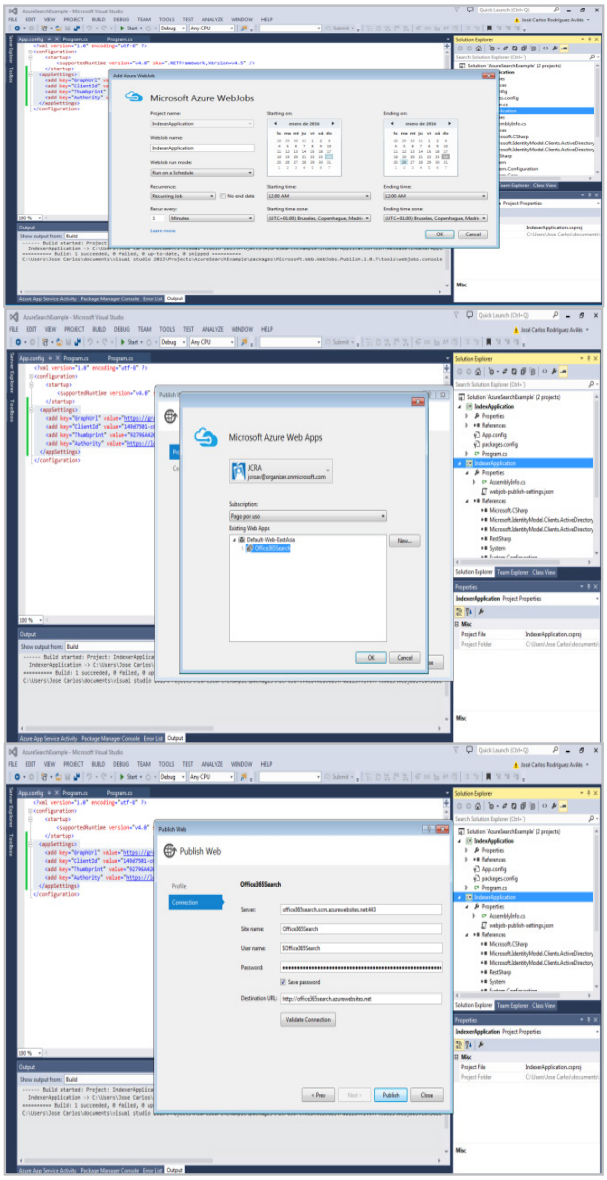


Imagen 10.- Pasos de despliegue de una solución de Visual Studio como webjob en Azure.

Si se prueba el webjob que se acaba de desplegar y tras la ejecución, se accede a la instancia del servicio Azure Search que se ha utilizado para el ejemplo, se puede observar que en el índice correspondiente se ha indexado el contenido.

Buscando contenido a través de Azure Search

Una vez creado el índice e indexado el contenido, se puede incorporar a una aplicación final la experiencia de búsqueda que proporciona, o bien la SDK de .NET, o la API REST de Azure Search. Se podrán incorporar fácilmente paneles de refinamiento, ordenación, sugerencias, etc. En definitiva, una completa experiencia de búsqueda para el proyecto.

Conclusiones

Las búsquedas se convierten en un elemento fundamental

en muchos proyectos informáticos, y a la vez, de las partes más complejas de desarrollar. El servicio Azure Search es una solución muy útil para incorporar la búsqueda en dichos proyectos de una forma eficaz y con un resultado muy completo. Además, como se ha podido comprobar en el artículo, se pueden combinar con la API Microsoft Graph para poder acceder al contenido de Office 365 e indexar el mismo en la solución de búsqueda. En el artículo de hoy, se ha presentado toda la fase de backend del ciclo de vida de una solución de búsqueda, en próximos artículos, se podrá ver como usar este servicio para desarrollar una aplicación basada en búsquedas que proporciona contenido de Office 365 y SharePoint Online.

Bibliografía

- https://elblogdelprogramador.wordpress.com/2016/01/18/primeros-pasos-con-azure-search-los-indices/
- https://elblogdelprogramador.wordpress.com/2016/01/24/autenticacion-app-only-para-usar-la-api-microsoft-graph-en-webjobs-de-azure/
- https://elblogdelprogramador.wordpress.com/2016/02/02/indexando-contenido-en-el-servicio-azure-search/
- https://azure.microsoft.com/en-us/documentation/articles/search-what-is-azure-search/
- http://www.eliostruyf.com/building-daemon-or-service-app-with-the-microsoft-graph-api/
- https://azure.microsoft.com/en-us/documentation/articles/search-howto-dotnet-sdk/
- http://graph.microsoft.io/en-us/docs

JOSÉ CARLOS RODRÍGUEZ AVILÉS
Desarrollador de Soluciones de SharePoint y colaborador del grupo de usuarios MadPoint
@jcroav
http://elblogdelprogramador.wordpress.com

54

¿InfoPath se muere?

Por todos es ya conocido que InfoPath tiene sus días contados. Con sus amantes y detractores, hay quien no ve el momento de que InfoPath desaparezca definitivamente, mientras que para otros es la peor de las noticias. Sea como fuere, InfoPath tenía su espacio dentro del ecosistema SharePoint y Office y para muchos supone un hueco que habrá que cubrir a corto-medio plazo. La pregunta es, por tanto, cómo reemplazarlo. Aquí os dejo una breve reflexión sobre por dónde empezar a buscar esa pieza de recambio.

Desde el principio: ¿Qué es realmente un formulario?

Antes de nada, establezcamos las bases y definamos bien qué es un formulario y para qué lo utilizamos. Un formulario o form puede significar varias cosas, dependiendo a quién se le pregunte. A día de hoy, según mi experiencia, el concepto formulario tiene cuatro tipos de uso.

- Documento Estructurado: Un formulario puede ser entendido como un tipo de documento en el que la información mostrada es información almacenada en listas de SharePoint. Digamos que sería una manera elegante de mostrar los datos de una lista, más orientada a formato documento.
- Data UI o Interfaz gráfica de acceso a datos: Nuestros datos pueden estar contenidos en una base de datos, una lista SharePoint o cualquier otro repositorio. Un formulario nos permite acceder a cada fila/tabla de una base de datos o a cada elemento de una lista y mostrarlo de una manera mucho más atractiva para el usuario; más fácil de entender y de editar.
- Interfaz de Solución: En general, toda aplicación, sea código personalizado, un workflow o un entorno CRM, necesitara mostrar datos al usuario y recibir datos del de alguna manera. El formulario, en esta casuística, será la interfaz gráfica de estos escenarios.
- Aplicación Standalone: Algunos formularios pueden ser un documento y una solución completa en sí. En estos casos, el formulario almacena datos, recoge/envía otros datos, los transforma, hace cálculos con ellos y finalmente muestra el resultado. Todo ello puede estar contenido en el mismo formulario.

Entendido, entonces ¿InfoPath para qué servía?

Con todo esto en mente, analicemos cómo se ha venido utilizando InfoPath en todos estos años:

InfoPath fue originalmente concebido como una herramienta de escritorio para crear documentos estructurados. En aquel tiempo (la era Office 2003), se apostaba por un modelo de trabajo basado en herramientas cliente. En 2007, InfoPath podía ser usado para crear formularios para workflows (interfaces de solución) y, en 2010, InfoPath añadió soporte para conectar datos de otras listas SharePoint en los formularios.

En resumen, durante su ciclo de vida InfoPath ha ido "evolucionando" (si se puede considerar evolución), tratando de albergar todos los casos que veíamos anteriormente.

Pero, ¿realmente InfoPath servía para todo eso?

Aunque ya entramos en el terreno de valoraciones personales, intentemos ser objetivos: ¿Podemos decir que ha conseguido cumplir su función en todos estos casos? Veamos:

- Utilizar InfoPath como interfaz de solución en teoría no parece mala idea, pero en realidad no hay nada en InfoPath Designer para controlar el resto de la solución, como puede ser el contexto de un workflow. Además, los recursos de la solución se tienen que desarrollar de forma independiente, de manera que, si se hace un cambio en los datos, en la lógica del workflow, etc. nos tenemos que asegurar de rediseñar el formulario acorde a estos cambios. Si, por desgracia, además se requiere código, cambiar estos formularios se convierte en una pesadilla.
- Por esta razón, mucha gente que utilizaba InfoPath decidió que la mejor manera de evitar el problema anterior era llevar toda la lógica de la solución en el formulario y utilizar InfoPath para crear aplicaciones Standalone. En mi opinión, esto es un error. Y grave. Durante años nos han enseñado que el diseño de toda aplicación debe dividirse en tres capas: presentación, lógica y datos. La tecnología ha cambiado, pero creo que este modelo de diseño sigue siendo el adecuado. ¿Alguien de verdad considera que tener en el propio formulario toda la lógica de control y el acceso a datos es una buena idea? Espero que no. En cualquier caso, a los que estén en duda, les invito a intentar entender

un formulario InfoPath con vistas, reglas de validación y código embebido; buena suerte y luego hablamos.

- En cuanto a solución Data UI o interfaz de acceso a datos, tengo que decir que InfoPath no me parece tan mala solución, hace su trabajo si tenemos la posibilidad de usar los InfoPath Form Services, lo que requiere la licencia SharePoint Enterprise. Eso sí, espero que no sea este el único motivo de adquirirla.
- Por último, para la generación de Documentos Estructurados, en realidad InfoPath es una gran herramienta. Es para lo que originalmente fue creado, eso sí, antes de que todos los documentos office pasasen a ser documentos XML en 2007.

Conclusión

¿Entonces, con todo esto en contexto, como reemplazamos InfoPath? Teniendo todo lo anterior en cuenta parece claro que lo que necesitamos no es un recambio o alternativa sino tres:

- Si lo que necesitamos es una herramienta para diseñar formularios de apps en SharePoint mi recomendación es decantarse por algún producto vinculado a soluciones de workflow. Obviamente yo tengo mi favorito, pero este no es el ámbito. En general, todas las compañías dedicadas a soluciones de workflow proporcionan soluciones para la creación y edición de formularios. Si nuestras apps no tienen que ver con workflow entonces Visual Studio o Access son herramientas a considerar, por ejemplo.
- Si lo que en realidad necesitamos es crear Documentos Estructurados basta decir que cualquier documento Word o Excel es ya un documento estructurado. Incluso los PDFs.
- Por último, para tener una solución de Data UI o interfaz de acceso a datos las opciones son múltiples, el mismo Visual Studio nos puede servir.

Sí, lo sé, he mencionado cuatro casos de uso y alternativa solo para tres. Creo que la mejor alternativa para los formularios como solución standalone es directamente no planteárselos, ganaremos todos en salud.

GONZALO MARCOS ANSOAIN
Nintex Technical Evangelist - EMEA

56

ENMARCHA: Un Framework Open Source para agilizar el desarrollo en SharePoint

Este mes de Febrero, en ENCAMINA hemos liberado como proyecto Open Source, bajo licencia MIT, Enmarcha, un Framework, Componentes o una serie de librerías para ayudar a los desarrolladores de SharePoint/Office 365 a agilizar los desarrollos y sobre todo estandarizar la forma en la que se desarrolla, de tal forma que desarrollar sobre SharePoint sea exactamente igual que en otras plataformas.

Un poco de historia

ENMARCHA nace internamente en ENCAMINA hace aproximadamente tres años, el motivo es que intentábamos aprovechar todo el conocimiento que se había ido aprendiendo por el equipo a lo largo de muchos proyectos, y que fuera el punto de partida para estandarizar la forma en la que desarrollábamos sobre nuestro servidor favorito.

Uno de los problemas que teníamos es que teníamos implementadas muchas utilidades, pero el equipo no las conocía, lo que provocaba que, si alguien tenía que hacer una utilidad, en muchas ocasiones se volvía a implementar. El volver a hacer este código, además del tiempo de desarrollo, se le unía los problemas sobre cómo implementarlo y sus posibles errores. Otro de los problemas que teníamos era la forma en la que se accedía a las listas, muchas veces teníamos problemas en saber cómo se debe de acceder a las columnas dependiendo del tipo de Columna o como se utiliza la paginación sobre las listas.

Junto con estos inconvenientes, se unió que queríamos desarrollar en SharePoint de una forma similar a como se desarrollaría en cualquier otra plataforma, lo que nos aportaría muchas cosas como reutilización de componentes, buenas prácticas y mejorar la gestión del ciclo de vida de los desarrollos. Con estas premisas creamos nuestra primera versión de ENMARCHA que ha ido evolucionando, proyecto a proyecto, hasta ver lo que actualmente está en el repositorio de GitHub.

Este proyecto no es solamente solo sobre la versión On-Premises de SharePoint 2013, sino que también estará disponible para Office 365, PowerShell y SharePoint 2016, en un futuro próximo.

Como utilizarlo

Para poder hacer uso de ENMARCHA lo podemos hacer bien a través del gestor de paquetes Nuget o utilizar el generador de proyecto Yeoman.

Desde Nuget, el gestor de paquetes de Visual Studio, podemos buscarlo mediante la interfaz de usuario o bien desde el Package Manager Console son la siguiente instrucción:

```
Install-Package Enmarcha.SharePoint
```

Este proceso lo que hace es agregarnos dos librerías:

- Enmarcha.SharePoint.Abstract: En esta librería están todas las Interfaces que hay dentro del FrameWork.
- Enmarcha.SharePoint: Aquí está la implementación de las interfaces para SharePoint OnPremises 2013.

Desde Yeoman, lo que nos descargaremos es un proyecto de SharePoint en el que ya tenemos estas librerías instaladas y listas para utilizarla en nuestro desarrollo. Para poder utilizar esta plantilla debemos de instalar NodeJS. Una vez instalado NodeJS, deberemos de instalar Yeoman con la siguiente instrucción:

```
Npm install -g yo
```

Una vez ya tenemos instalado, tendremos que instalar el generador de ENMARCHA dentro con el siguiente comando:

```
npm i generator-enmarcha
```

Si ahora ejecutamos: yo enmarcha se nos muestra una pantalla donde se nos pregunta el nombre de nuestro proyecto:

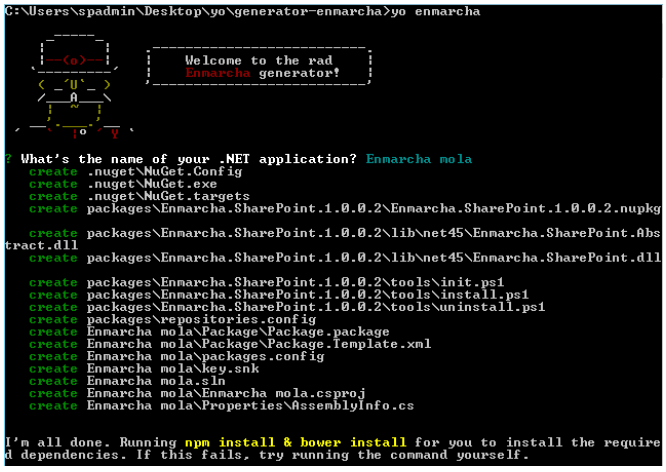


Imagen 1.- Uso de ENMARCHA desde Yeoman.

Un ejemplo de un WebPart haciendo uso de ENMARCHA

En este ejemplo vamos a implementar una WebPart con en el que vamos a mostrar las imágenes en formato de Galería de Imágenes de una forma similar a la siguiente:

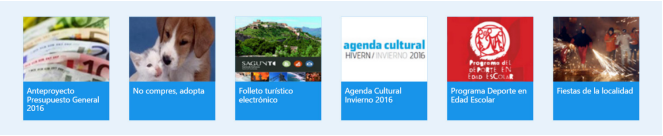


Imagen 2.- Ejemplo de WebPart haciendo uso de ENMARCHA.

Enmarcha, un Framework, Componentes o una serie de librerías para ayudar a los desarrolladores de SharePoint/Office 365 a agilizar los desarrollos y sobre todo estandarizar la forma en la que se desarrolla

Para realizar este ejemplo por un lado vamos a tener una lista donde vamos a tener los campos necesarios para que se muestre. De tal forma que es posible que en un futuro necesitemos utilizar la búsqueda de SharePoint vamos a hacer buenas practicas dentro de la gestión documental y nos crearemos un Tipo de contenido basado en las columnas de sitio que va a tener esta lista y posteriormente este tipo de Contenido se le agregará a la lista. Por último, realizaremos un WebPart Visual en el que realizaremos una consulta a esta lista y la mostraremos según el diseño indicado arriba.

Requisitos previos:

- Visual Studio 2013
- SharePoint On Premises 2013

Manos a la obra:

En primer lugar, se crea una solución en Visual Studio, que está compuesta por tres proyectos:

- Compartimoss.Enmarcha.Model => En esta proyecto se agrega la definición de las listas que se utilizan en el desarrollo.
- Compartimoss.Enmarcha.Service => Este proyecto corresponde con la capa de servicios donde se incluye la lógica de negocio (bussines logic).
- Compartimoss.Enmarcha.SP => Es un proyecto de SharePoint donde se definen varias Features para desplegar los diversos artefactos necesarios.

Una vez creada la solución, añadiremos a los tres proyectos el Nuget de ENMARCHA de la misma forma que hemos comentado al inicio del artículo.

Ahora en la Solución CompartiMOSS.Enmarcha.Modelo se crea una Clase que la llamaremos ImageGalery.cs con la siguiente definición:

```
public class ImageGallery
  {
    [Enmarcha(AddPrefeix = false, Create = false, Type =
TypeField.Text)]
    public string ID { get; set; }
    [Enmarcha(AddPrefeix = false, Create = false, Type =
TypeField.Text)]
    public string Title { get; set; }
    [Enmarcha(AddPrefeix = false, Create = true, Type =
TypeField.Boolean, DisplayName = "Visible in ImageGallery")]
    public bool Visible { get; set; }
    [Enmarcha(AddPrefeix = false, Create = true, Type =
TypeField.Boolean, DisplayName = "Open Link in new Page")]
    public bool OpenWindows { get; set; }
    [Enmarcha(AddPrefeix = false, Create = true, Type =
TypeField.Url, DisplayName = "Url of image")]
    public UrlField Image { get; set; }
    [Enmarcha(AddPrefeix = false, Create = true, Type =
TypeField.Url, DisplayName = "Url of new")]
    public UrlField UrlNew { get; set; }
    [Enmarcha(AddPrefeix = false, Create = true, Type =
TypeField.Text, DisplayName = "Descripcion")]
    public string Description { get; set; }
    public string NewUrl   {  get { return UrlNew.Url;} }
    public string PictureUrl  { get { return Image.Url; } }
    public string TargetBlank  {
  get { return (OpenWindows == true) ? " target='_blank'" :
string.Empty; }
    }
  }
```

Dentro de esta clase se han agregado varios atributos, estos son propios de ENMARCHA. Se utilizan para "mapear" nuestras clases en C# con el artefacto en SharePoint correspondiente. El significado de estos atributos es el siguiente:

- AddPrefix-> Le añada un prefijo cuando crea el campo de forma que se evita que coincida con algún campo ya declarado.
- Create -> Indica si esta propiedad hay que crearla o no.
- Type -> Tipo de SharePoint con el que representa esta propiedad.

Una vez creada la clase, agregaremos una Feature dentro de la solución de CompartiMOSS.Enmarcha.SharePoint y utilizando ENMARCHA crearemos:

- un Tipo de Contenido partiendo de la clase Image Gallery.
- una Lista en SharePoint a la que se le añade ese tipo de contenido.

Para ello dentro de la Feature se añade el siguiente código que se ejecuta en el momento que se active dicha característica:

```
public override void
FeatureActivated(SPFeatureReceiverProperties properties)
    {
      var site = properties.Feature.Parent as SPSite;
      var web = site.RootWeb;
      ILog log = new LogManager().GetLogger(new StackTrace().
GetFrame(0)); ;
      var columnSiteCollection = web.CreateColumnSite("Image
Galery", typeof(ImageGallery));
      web.CreateContentType(Constants.ContentType.
ImageGallery, "Enmarcha ContentType", "Elemento",
columnSiteCollection);
      web.CreateList(Constants.List.ImageGallery, "Lista de la
galeria de imagenes", TypeList.GenericList, true);
      var list = web.Lists.TryGetList(Constants.List.ImageGallery);
      if (list != null)
      {
        list.AddContentTypeLibrary("Image Galery");
      }
}
```

De dicho código, comentar por un lado que hacemos uso de una utilidad de Log, incluida dentro de ENMARCHA. Este log lo que hace es guardar los errores de nuestros desarrollos dentro del propio Log de SharePoint.

Por otro lado, hay cuatro métodos extensores que son propios de ENMARCHA, cuya finalidad es extender el funcionamiento de la API de SharePoint y facilitarnos nuestros desarrollos, estos cuatro métodos son los siguientes:

- CreateColumnSite: Que dado una Clase de C# nos crea las columnas de sitio correspondiente, para la creación hará uso de los atributos que previamente hemos definido.
- CreateContenType: Un método que crea un Tipo de Contenido, y en el que le pasaremos una lista de Strings donde están las columnas de sitio que va a tener dicho tipo de contenido. En caso de que queramos pasar un GUID a dicho tipo de contenido también tiene la opción.
- CreateList: Crea una lista de SharePoint sobre el objeto SPWeb en el que estamos ubicados, a esta lista le pasamos como valores: nombre, descripción, tipo de Lista e indicamos si queremos romper la herencia de los permisos o no.
- AddContentTypeLibrary: Le añadimos un tipo de contenido a nuestra lista.

Una vez ya está desplegada la infraestructura, vamos a crear el WebPart. En primer lugar, nos crearemos el "Service" que va a consultar los elementos de la lista. El motivo es claro, tener separada la lógica de negocio de la interfaz y no producir un acoplamiento entre ambas capas. Por lo que dentro de nuestra solución CompartiMOSS.Enmarcha.Service, por un lado se crea una Interfaz del Servicio con la siguiente definición:

```
public interface IImageGaleryService
  {
     IList<ImageGallery> GetNews();
  }
y posteriormente desarrollamos dicha interfaz de la siguiente forma:
public  class ImageGaleryService:IImageGaleryService
  {
     public SPList List { get; set; }
     public ILog Logger { get; set; }
     public int Items { get; set; }
     public ImageGaleryService(SPList list,int items)
     {
        this.List = list;
        this.Items = items;
        this.Logger = new LogManager().GetLogger(new StackTrace().
GetFrame(0));
  }
     public IList<ImageGallery> GetNews()
     {
        try
        {
           var repositorySharePoint = new
SharePointRepository<ImageGallery>(this.List.ParentWeb, this.
Logger,   this.List.Title, this.Items);
           var imageCollection = repositorySharePoint.GetAll();
           return imageCollection.ToList();
        }
        catch (Exception exception)
        {
           this.Logger.Error(string.Concat("Error GetNews",exception.
Message));
           return null;
        }
     }
  }
```

Del código anterior, cabe mencionar el objeto SharePointRepository. Este objeto se le pasa como parámetro de que clase va a devolver los datos, en nuestro caso ImageGallery, y en el constructor por un lado le tenemos que indicar el objeto SPWeb, el Log que se va a utilizar (está adaptado para que cada desarrollador pueda utilizar el Log que desee conveniente), el nombre de la lista a la que se va a realizar la consulta, y el número de elementos que vamos a devolver como máximo en cada petición.

En el objeto SharePointRepository, dispone de opciones para realizar un CRUD, utilizando los métodos Insert, Save o Delete y también se pueden hacer consultas mediante el uso de sintaxis CamlQuery, LINQ o bien devolver todos los elementos. En este caso, se consultan todos los elementos.

Una vez se ha creado la lógica de negocio, lo que vamos a realizar llamar a este servicio desde el Code-Behind del WebPart:

```
try
        {
           var listSharePoint = SPContext.Current.Web.Lists.
TryGetList(Constants.List.ImageGallery);
           var imageGaleryService = new
ImageGaleryService(listSharePoint, 5);
           var imageGaleryCollection = imageGaleryService.GetNews();
           listViewImageGalery.DataSource = imageGaleryCollection;
           listViewImageGalery.DataBind();
        }
        catch (Exception exception)
        {
           Logger.Error(string.Concat("Error Concat
LoadData",exception.Message));
        }
```

Ester ejemplo se puede descargar desde el Repositorio GitHub

Colaborar con ENMARCHA

Como he dicho en principio ENMARCHA es un proyecto Open Source y está abierto a la colaboración por parte de todo el mundo que así lo desee. Puedes participar en el proyecto con:

- Enviando bugs y solicitando nuevas funcionalidades
- Revisar cambios en el código fuente

Para nosotros en ENCAMINA es fundamental el uso de este Framework en nuestros proyectos, y, por lo tanto, seguiremos evolucionándolo y corrigiendo posibles errores que se puedan producir. Liberarlo como proyecto Open Source forma parte de nuestro ser y de nuestras aportaciones a la comunidad y a nuestros clientes. Desde aquí, os invitamos a probar y usar el Framework en vuestros desarrollos y, en la medida que os sea posible, aportar vuestro granito de arena al mismo, a través de nuestro repositorio en Github.

ADRIÁN DIAZ CERVERA
Software Architect Lead at ENCAMINA
MVP Office Servers and Services
http://blogs.encamina.com/desarrollandosobresharepoint
http://geeks.ms/blogs/adiazcervera
adiaz@encamina.com @AdrianDiaz81

59

Conectores para Grupos de Office 365

Primero de todo empezaremos explicando que son los conectores para Grupos de Office 365. Un conector es una característica que nos permite enviar información en tiempo real a nuestros grupos de office 365, desde otros servicios o plataformas, desde un sistema externo o un sistema propio.

Ejemplo:

Conectar Trello con office 365 groups y recibir notificaciones cuando un board es modificado.

Imagen 1.- Notificaciones en Trello.

Introducción a Grupos de Office 365

Para acceder a esta funcionalidad:

- Simplemente entrando a nuestro correo de office 365, veremos un apartado que pone Grupos.

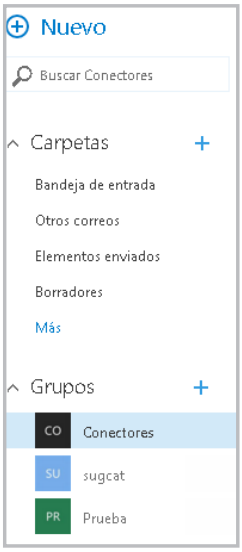


Imagen 2.- Acceso a Grupos desde OWA.

- En este menú podremos ver los grupos que hemos creado o en los que somos administradores. Una vez creado el grupo, nos permite agregar personas de nuestra compañía y tener un sitio donde compartir correos, calendario, archivos, bloc de notas, etc...

Imagen 2.- Opciones disponibles en un Grupo de Office 365

Cómo habilitar Conectores de Grupos de Office 365

Esta característica actualmente está en preview, por lo tanto, aún no es visible para todos los usuarios.

Si queremos habilitar esta característica debemos agregar a la url de nuestra página la siguiente cadena: &EnableConnectorDevPreview=true

En mi caso particular, me he creado un grupo llamado conectores y la URL de acceso al conector sería así:

https://outlook.office.com/owa/#path=/group/conectores@sug.cat/mail&EnableConnectorDevPreview=true

Una vez hemos entrado con esta URL, nos aparecerá una opción extra en el menú del Grupo: Conectores

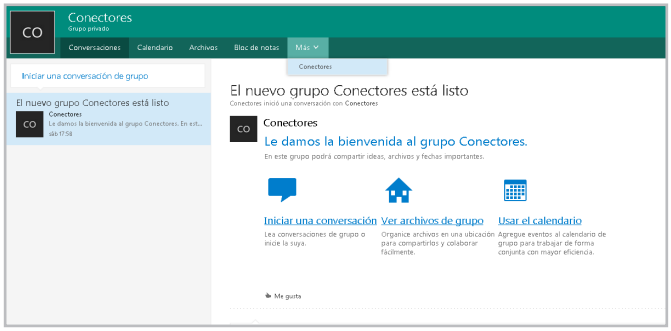


Imagen 3.- Opción Conectores en el Grupo.

Un conector es una característica que nos permite enviar información en tiempo real a nuestros grupos de office 365, desde otros servicios o plataformas, desde un sistema externo o un sistema propio.

Conectores Existentes

Una vez habilitados los conectores para nuestro grupo de office 365, tenemos dos opciones de trabajo con conectores: usar conectores por defecto o usar conectores customizados. Si pulsamos en la opción conectores nos aparecerá un listado de conectores ya existentes.

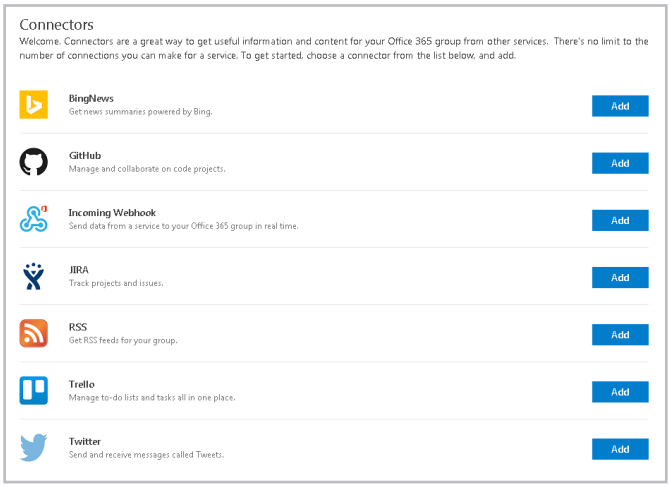


Imagen 4.- Conectores por defecto.

Ejemplo: Uso del Conector de Github

En nuestro ejemplo vamos a usar Github, para que nos notifique cuando haya actividad en nuestros proyectos:

- Pulsamos en Add para este conector de manera que aparece la pantalla de login en GitHub.

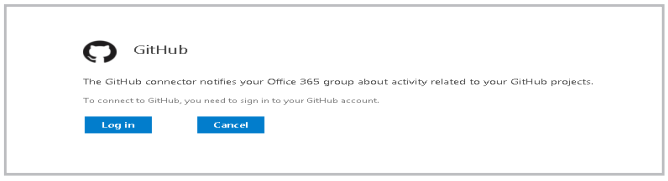


Imagen 5.- Pantalla de Login en Github.

- Pulsamos en login y una vez conectados nos aparece la ventana para dar permisos a la aplicación.

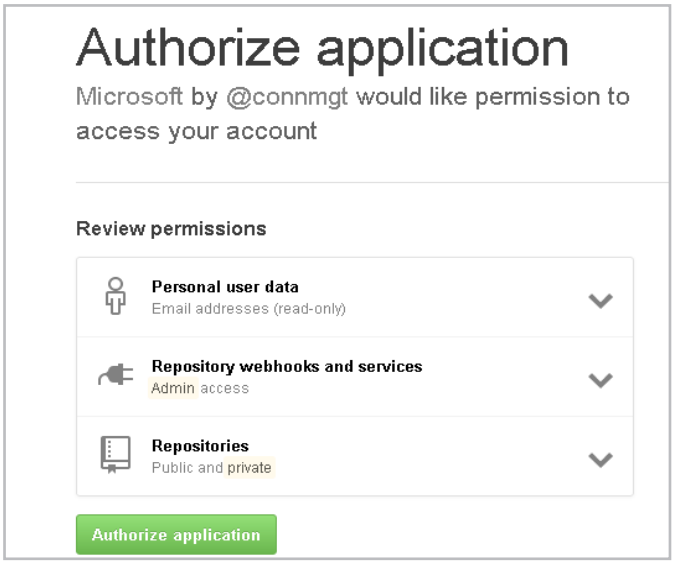


Imagen 6.- Autorización de Github

- Una vez autorizado, se genera una Url única (webhook) para esta aplicación y este grupo en office 365 que será donde hará las llamadas para añadir información. A continuación, nos pide de qué tipo de acciones queremos ser notificados, que cuenta y que repositorio de código.

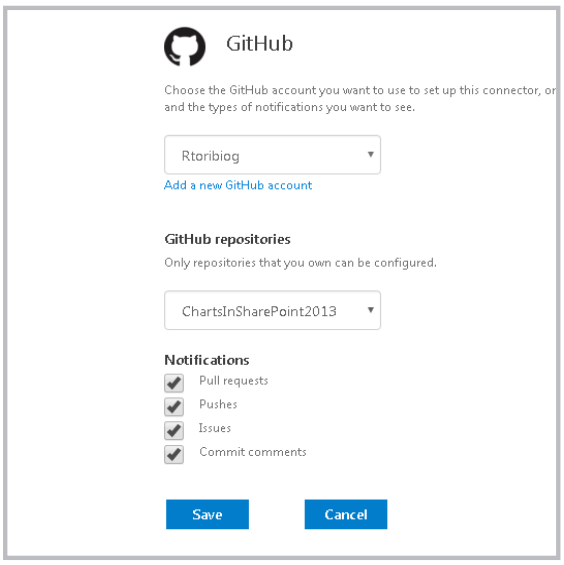


Imagen 7.- Tipo de Notificación que se pueden configurar.

- Una vez hecho esto, en el menú de conectores, nos aparecerá que el conector se ha configurado correctamente.

Imagen 8.- Conectores configurado de forma correcta.

- Por supuesto, el conector añadido se puede modificar, visualizar, eliminar.
- Si vamos a nuestro GitHub, en la sección Applications, veremos como también está configurado el conector.

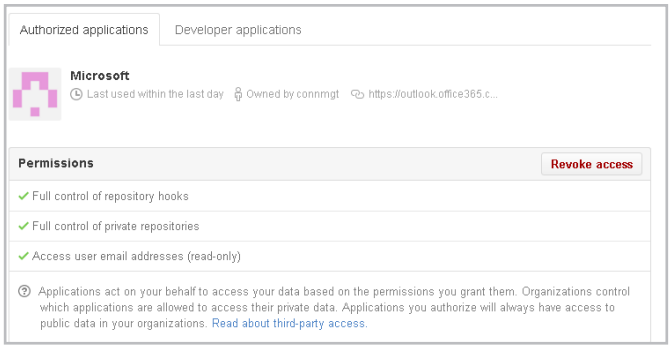


Imagen 9.- Aplicaciones autorizadas en GitHub.

Una vez habilitados los conectores para nuestro grupo de office 365, tenemos dos opciones de trabajo con conectores: usar conectores por defecto o usar conectores customizados.

- Si volvemos a nuestra bandeja de entrada, nos aparecerán dos nuevas notificaciones conforme se ha creado correctamente nuestro conector con GitHub.

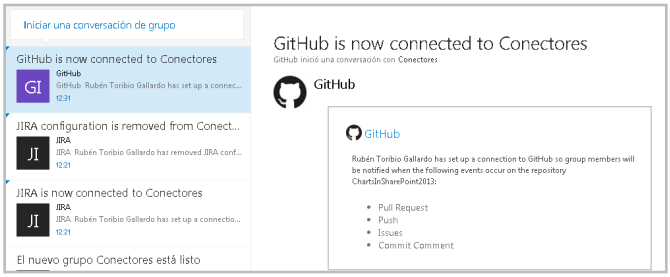


Imagen 10.- Notificación relativa al conector configurado.

y añadimos los correspondientes comentarios. A continuación, en el Grupo de Office 365 verificamos que aparecen las notificaciones de la actividad realizada.

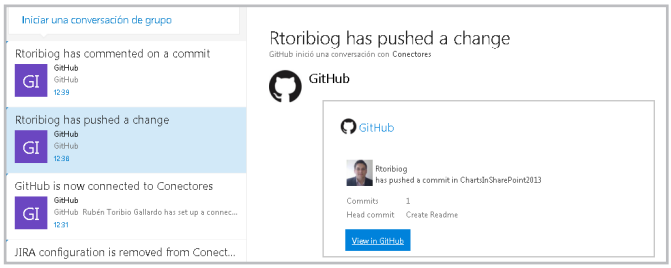


Imagen 11.- Notificaciones de Github en el Grupo.

Conector de GitHub en funcionamiento

Una vez que tenemos el conector configurado, cuando hagamos algún cambio sobre el repositorio nos notificará a nuestro Grupo de Office 365:

- Hacemos un commit de un nuevo fichero en GitHub

Conectores Customizados

A continuación, describiremos como crear un conector customizado, para lo que tenemos varias opciones. Una de ellas es la manual:

- De nuevo en la sección Conectores, pulsamos el botón Add de la opción Incoming webhook.

Imagen 12.- Añadiendo un Incoming Webhook .

- Esto nos va a permitir configurar la conexión y poder enviar información a nuestro Grupo de Office 365 desde nuestra aplicación customizada.
- Seleccionamos un nombre, una imagen y lo creamos.

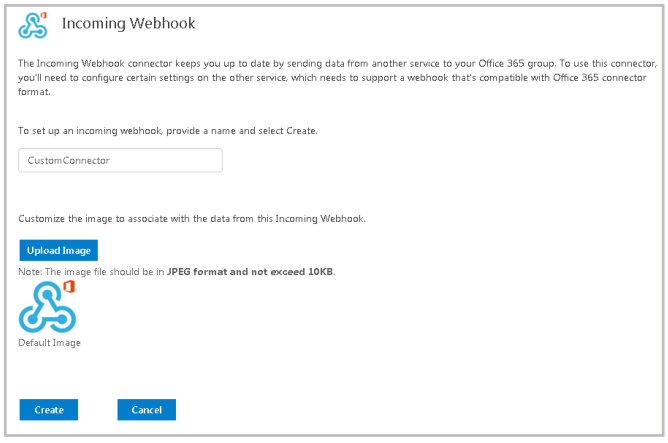


Imagen 13.- Configuración del Incoming Webhook a crear

- Este nos va a generar una Url como la siguiente:

https://outlook.office365.com/webhook/38884380-1870-42d0-9140-9f3889c4814b@d48595fc-ac3b-4f2c-9cd3-b502041ddf18/IncomingWebhook/

- Y con esta Url, qué es única por Grupo y aplicación, ya podremos empezar a enviar mensajes.

Imagen 14.- Webhook generado.

- Y aparecerá en la actividad del Grupo como configurado.

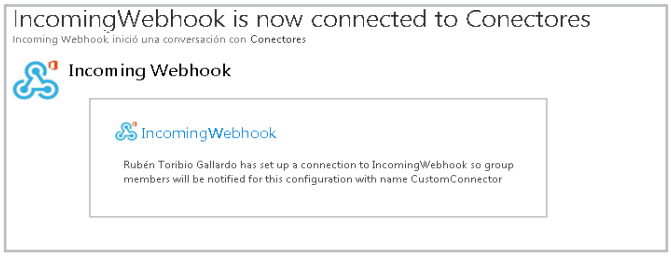


Imagen 15.- Incoming Webhook configurado.

- Ahora sólo tenemos que hacer un POST Request hacia esta dirección con el texto de nuestra card, y nos llegará el mensaje a nuestro grupo. No es necesario autenticación, ya que la url generada es única.

Ejemplo:

- El JSON de ejemplo para una card sería el siguiente:

```
{
 "summary": "New Comment by Rubén on Conectores\"",
 "title": null,
 "text": null,
 "themeColor": "#3479BF",
 "sections": [
  {
   "title": null,
   "text": null,
   "markdown": true,
   "facts": [
    {
     "name": "Added By",
     "value": "Rubén"
    },
    {
     "name": "Date",
```

```
      "value": "15-12-2015"
     },
     {
      "name": "Priority",
      "value": "Medium"
     },
     {
      "name": "State",
      "value": "Active"
     }
    ],
    "images": null,
    "activityTitle": "Rubén commented",
    "activitySubtitle": "on \"Conectores\"",
    "activityText": "We should prioritize this effort.",
    "activityImage": "https://cdn0.iconfinder.com/data/icons/
PRACTIKA/256/user.png"
   }
  ],
  "potentialAction": [
   {
    "@context": "http://schema.org",
    "@type": "ViewAction",
    "name": "View in Sandbox",
    "target": [
     "http://microsoft.com"
    ]
   }
  ]
}
```

Pruebas de Conectores Custom

Para seguir experimentado con Conectores Custom contamos con la página:

http://connectplayground.azurewebsites.net/

Donde nos permite conectarnos a través de una aplicación hospedada en Azure por Microsoft, enviar mensajes de ejemplo a nuestros Grupos de Office 365. Sólo necesitamos conectarnos, nos generará un webhook Url, y ya podremos enviar un mensaje.

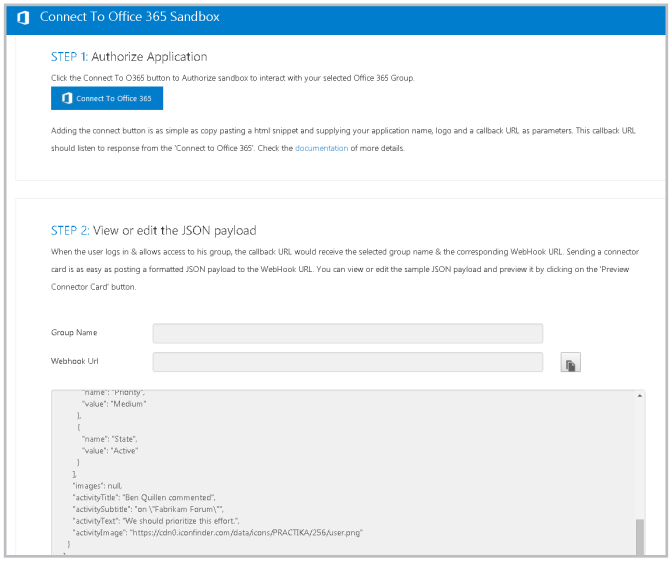


Imagen 16.- Office 365 connect Sandbox.

Office 365 connector card

Las cards es la manera en la que se visualizan los mensajes enviados por un servicio por defecto o nuestro servicio customizado. Es la manera de enviar un mensaje de manera genérica para todas las aplicaciones y con la misma interfaz gráfica. Y hay de diferentes tipos:

- Mensaje.
- Mensaje con Título.
- Mensaje con Link.
- Mensaje con Acción.

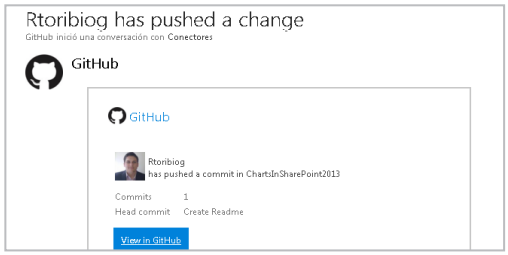


Imagen 17.- Ejemplo de Card.

Los parámetros necesarios para llenar la información de una card son los siguientes:

- Connector Card: Es el contenedor de la card

Attribute	Type	Description
summary	SimpleString*	A string used for summarizing card content. This will be shown as the message subject. This is required if the text parameter isn't populated.
text	String	The main text of the card. This will be rendered below the sender information and optional title, and above any sections or actions present.
title	SimpleString*	A title for the Connector message. Shown at the top of the message.
themeColor	Color Hex Value ex: #737373	Accent color used for branding or indicating status in the card.
sections	Section[]	Contains a list of sections to display in the card
entities	Schema.org/Object[]	This property will not be used to generate the displayed Card, but other experiences may consume the Schema.org content included here.
potentialAction	Schema.org/ViewAction[]	This array of Schema.org/ViewAction objects will power the action links found below the title.

Imagen 18.- Connector Card.

Una vez habilitados los conectores para nuestro grupo de office 365, tenemos dos opciones de trabajo con conectores: usar conectores por defecto o usar conectores customizados.

- Section: Es el contenido dentro de la card.

Attribute	Type	Description
title	String	Title of the section
activityTitle	String	Title of the event or action. Often this will be the name of the "actor".
activitySubtitle	String	A subtitle describing the event or action. Often this will be a summary of the action.
activityImage	URL to image	An image representing the action. Often this is an avatar of the "actor" of the activity.
activityText	String	A full description of the action.
facts	Fact[]	An array of facts, displayed as key-value pairs.
images	Image[]	An array of images that will be displayed at the bottom of the section.
text	String	Optional text that will appear after activity.
markdown	bool	Set this to false to disable markdown parsing on this section's content. Markdown parsing is enabled by default.

Imagen 19.- Section.

Y una vez hagamos la llamada estas serán las diferentes respuestas.

Response Codes

The following HTTP response codes are returned to the sender in the given scenarios.

Response Code	Description	Details
200	Ok	A well-formed request is sent to an existing webhook. The request contains a valid payload, and has a valid corresponding webhook configuration.
400	Bad Request	An incorrectly-formed request is sent to a webhook that exists. The payload could contain non-parseable JSON, incorrect JSON values (e.g. expected a String, got an array), incorrect content-type, etc..
404	Not Found	A request is sent to a webhook that does not exist.
413	Payload Too Large	A request is sent to a webhook that is too large in size for processing.
429	Too Many Requests	Client is sending too many requests and Office 365 is throttling the requests to a webhook.

Imagen 20.- Respuestas.

Office 365 Connect Button

A continuación, vamos a describir la otra alternativa para poder configurar un conector customizado, a través del botón de conexión. En nuestra aplicación crearemos un botón que se conectará a los Grupos de Office 365 vía API y nos dejará seleccionar los grupos existentes, una vez seleccionado este nos devolverá la Url del webhook para poder hacer las llamadas con las Cards. La composición es la siguiente:

```
<a href="https://outlook.office.com/connectors/ConnectToO365?state=xxx&app_ name=xxx&app_logo_url=xxx&callback_url=xxx">
 <img src="https://connecttoo365.blob.core.windows.net/images/ConnectToO365Button.png" alt="Connect to Office 365"></img>
</a>
```

Con los siguientes parámetros.

Parameter	Details
state	You can use the state parameter to save your application state. If you supply a value for state it is returned back to the specified callback_url when the application returns. This is an optional parameter.
app_name	Provide your application name. This name will show up in the authorization popup and in the connector list page that your users would see. The app_name is a mandatory parameter and can range from 1 to 100 characters.
app_logo_url	Provide the URL to your application logo. Ensure that the link is not behind an authentication wall and is publicly reachable. Use a logo of size under 10KB (preferably 256x256px) and type JPEG, PNG, GIF, TIFF, BPM, X-ICON or SVG+XML. The app_logo_url is a mandatory parameter.
callback_url	The callback URL should be a valid HTTPS URL without any query parameters. When the application returns successfully, the state passed, name and webhook URL of the selected group are returned as query parameters to the callback_url. If the application encounters a failure the state passed and the error code are returned to the callback_url.

Imagen 21.- Parámetros del Webhook.

Y este es el resultado.

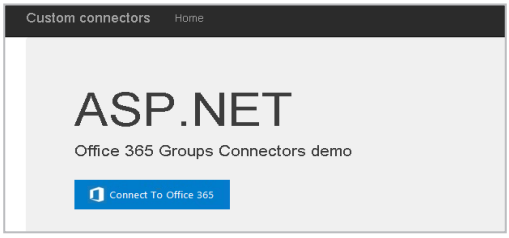


Imagen 22.- Resultado de la URL.

Ejemplo Creando Conector Custom.

Para finalizar el artículo, vamos a crear un Conector personalizado en Visual Studio:

- Primero de todo creamos una aplicación ASP MVC en Visual Studio.

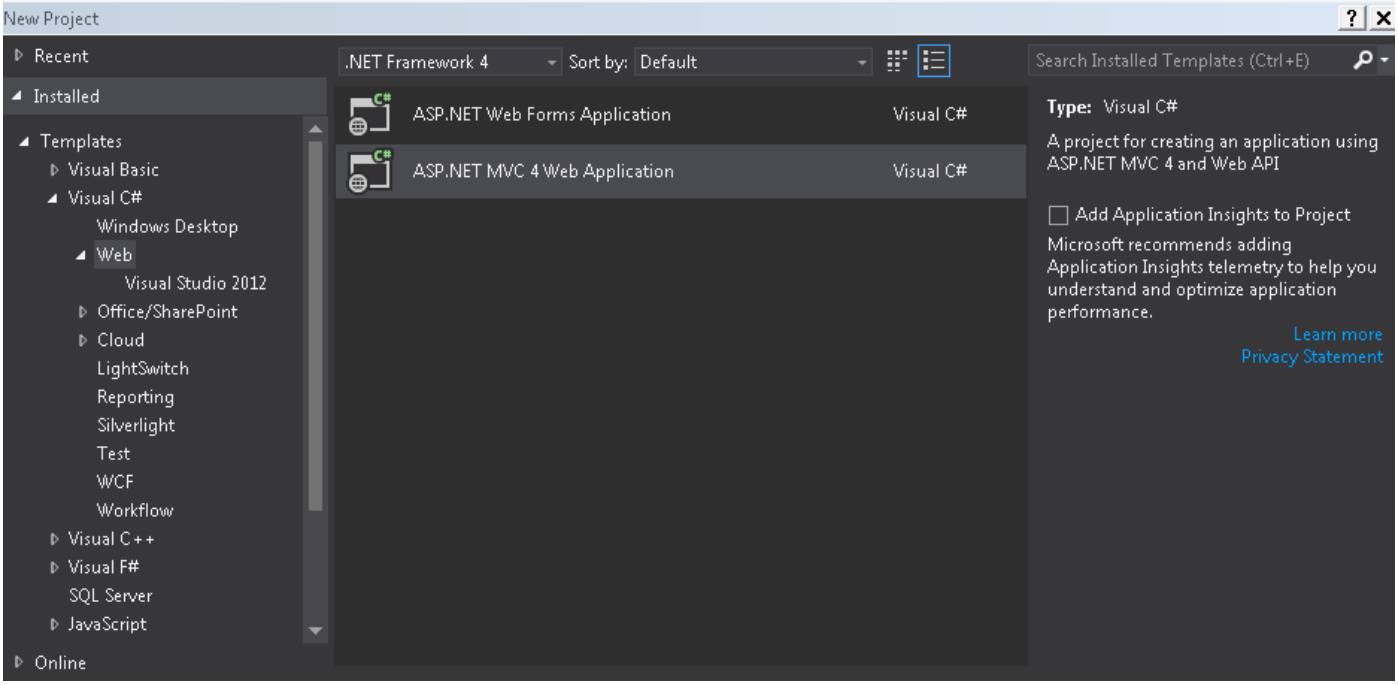


Imagen 23.- Selección del proyecto de tipo ASP.NET MVC en Visual Studio.

- Habilitamos el SSL a nuestro proyecto a través de la ventana de propiedades del IDE.

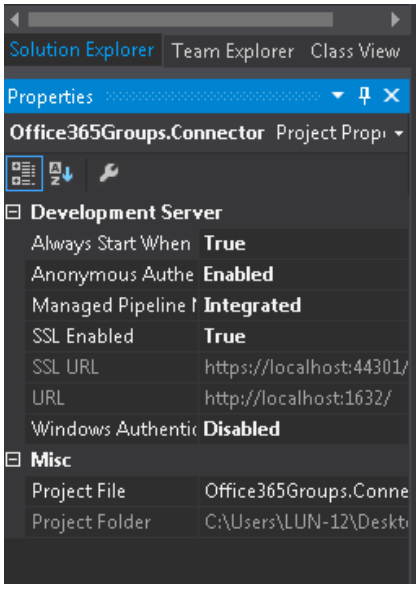


Imagen 25.- Habilitando SSL en el proyecto.

- Agregamos nuestro botón Connect to Office 365 Groups en nuestra vista Home.

```
<div>
    <h1>ASP.NET</h1>
    <p class="lead">Office 365 Groups Connectors demo</p>

    <a href="https://outlook.office.com/connectors/ConnectToO365?state=myAppsState&app_name=APP_Conectores&app_logo_url=http://pccircle.com/image/cache/data/Br
        <img src="https://connecttoo365.blob.core.windows.net/images/ConnectToO365Button.png" alt="Connect to Office 365"></img>
    </a>
</div>
```

Imagen 26.- Nuevo botón Connect to Office 365 Groups.

- Agregamos en la ruta start una vista y un controlador, donde haremos la llamada con el webhookurl

```
<h2>Post a conversation to Office 365 Group: @Request.QueryString["group_name"]</h2>

<div class="linkContainer">
    @Html.ActionLink("Post", "PostMessage", "Start", new { webHookUrl = Request.QueryString["webhook_url"] }, new { role = "button", @class = "btn btn-primary" })
</div>
```

Imagen 27.- Vista y controlador añadido.

```
public ActionResult PostMessage(string webHookUrl)
{
    var client = new RestClient(webHookUrl);

    var request = new RestRequest(Method.POST);
    request.AddHeader("Content-Type", "application/json");

    request.RequestFormat = DataFormat.Json;
    request.AddBody(new
    {
        title = " APP Conectores",
        text = "Mensaje Enviado desde mi Aplicacion Custom",
        themeColor = "DB4C3F"
    });

    var response = client.Execute(request);

    var content = response.Content;

    if (response.StatusCode == System.Net.HttpStatusCode.OK)
    {
        return View("Success");
    }
    else
    {
        return View();
    }
}
```

- Arrancamos nuestra aplicación y nos aparecerá la siguiente pantalla y le damos a Connect

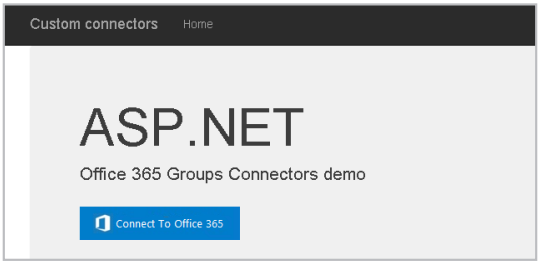


Imagen 28.- Pantalla de inicio de la aplicación.

- Al hacer clic en el botón “Connect to Office 365”, aparecerá la pantalla de autorización, seleccionamos el Grupo y autorizamos.

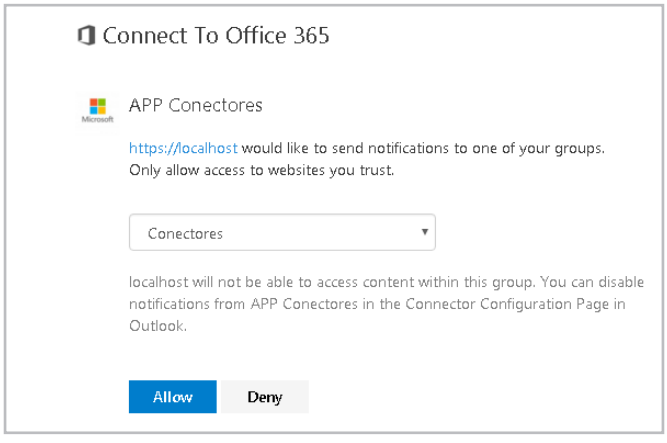


Imagen 29.- Habilitando SSL en el proyecto.

- Una vez autorizada, nos enviará a la vista start, donde tenemos el botón de enviar mensaje.

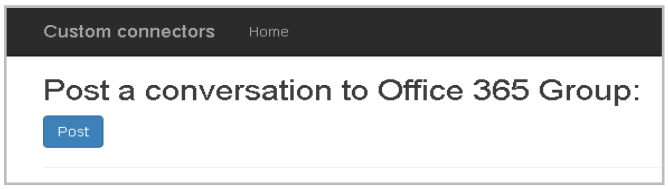


Imagen 30.- Vista start.

- Pulsamos en enviar y si funciona correctamente nos saldrá el siguiente mensaje

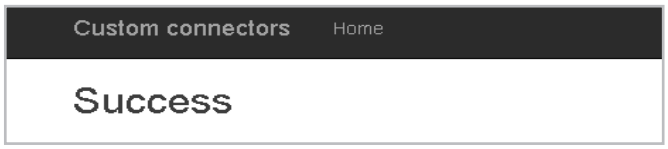


Imagen 31.- Mensaje obtenido.

- Si vamos a nuestro Grupo de Office 365 veremos el mensaje.

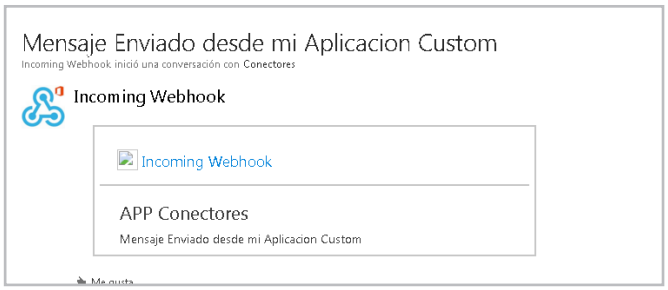


Imagen 32.- Mensaje en el Grupo

Conclusiones:

A través de los ejemplos mostrados hemos podido ver cómo crear conectores para Grupos de Office 365, tanto con aplicaciones por defecto como aplicaciones customizadas y como personalizar estos mensajes.

RUBÉN TORIBIO
SharePoint Architect en Beezy
@rtoribiog

67

¿QUES SON LAS POWERAPPS DE MICROSOFT? – CREANDO NUESTRA PRIMERA APLICACIÓN

Si accedemos a la página de Microsoft y buscamos la definición de PowerApp nos encontraremos algo como "Aplicaciones para conectar, crear y compartir aplicaciones que impulsen su negocio". Analizando la definición debemos empezar por el final, ya que, después de trabajar un poco con esta nueva herramienta, he podido observar que un usuario con conocimientos en office puede hacer uso de las PowerApps y crear una aplicación de forma totalmente autónoma sin necesidad de tener conocimientos en programación por lo que podemos decir desde ya que las PowerApps están pensadas para conseguir una autonomía de los empleados de negocio respecto a los departamentos de IT en el proceso de creación de este tipo de aplicaciones.

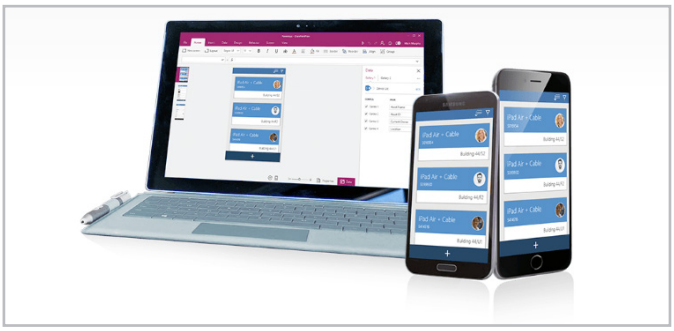


Imagen 1. Aplicaciones PowerApps

Como podéis ver he definido PowerApps como una "herramienta" y muy ligada al paquete Office; y en verdad es eso, una herramienta que permite crear aplicaciones de negocio tanto para móviles como para tablets que aporta una serie de conectores con nuestros servicios en la nube (Office 365, Dynamics CRM, OneDrive…), como con nuestros datos (SharePoint, SQL Server, Oracle, SAP, etc).

Imagen 2. PowerApps conectando tu negocio.

Como veremos más adelante en el ejemplo, el usuario puede desarrollar las aplicaciones de una forma muy parecida a como podemos crear una presentación en PowerPoint, y conectar sus datos haciendo uso de las conexiones a servicios o a datos. El tercer punto fuerte de las PowerApps es, bajo mi punto de vista, el concepto "Compartir" que incluye la herramienta. Podemos compartir nuestras aplicaciones móviles como a día de hoy compartimos un archivo en OneDrive. De forma sencilla podemos dar acceso a la aplicación a usuarios de nuestra organización para que puedan empezar a hacer uso de la plataforma.

Por último, agregar que PowerApps no solo es creación de aplicaciones, sino que también permite la creación de flujos de trabajo que posteriormente podremos utilizar en nuestras aplicaciones.

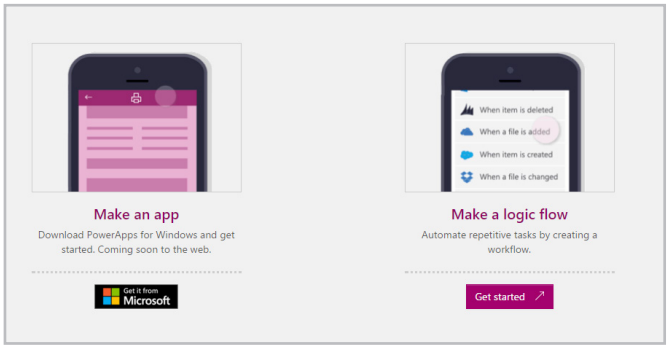


Imagen 3. Creando una aplicación

¿Cómo empezamos con PowerApps?

Para empezar a trabajar con PowerApps debemos adquirir la aplicación que encontraremos en la tienda de Windows 10. Una vez descargada necesitaremos una licencia para poder hacer uso de ella. Existe la posibilidad de solicitar una licencia de prueba directamente a Microsoft https://powerapps.microsoft.com/es-es/pricing/ , y ahí podremos ver que hay varias versiones del producto.

Imagen 4.- Versiones de PowerApps.

Como vemos existe una versión totalmente gratis un poco

limitada en cuanto al número de conectores que podemos utilizar, pero que nos puede servir para iniciarnos en la tecnología. La versión estándar sí es una versión de pago, que viene incluida con la licencia de O365 y aporta un uso ilimitado de conexiones a datos y mayor capacidad que la versión gratuita.

Y la versión Enterprise (de la cual hablaremos en siguientes artículos), es una versión aún por llegar ya que está en beta para empresas y desarrolladores, que incluye una visión más programática de la aplicación ya que permite desarrollar APIS personalizadas y almacenarlas en Azure para conectar nuestros datos con las apps y que no vienen en el catálogo de conectores estándar.

Creando nuestra primera aplicación

Una vez tenemos nuestra aplicación PowerApp, instalada y descargada podemos empezar a crear nuestra aplicación de prueba.

- Lo primero que se nos va a solicitar es una cuenta de usuario con licencia como hablamos en el punto anterior.
- Una vez logados en la plataforma se necesita configurar una conexión donde alojar nuestras aplicaciones (OneDrive, Dropbox, SharePoint Online, GoogleDrive…). Después de elegir donde guardaremos nuestra aplicación seleccionamos “Nuevo”, y veremos que tenemos varias opciones de crear la aplicación:
  - Desde una plantilla predefinida. Se nos aporta una variedad de plantillas base con las que nos podemos ahorrar trabajo y solamente con adaptarla podemos conseguir el objetivo que buscamos.
  - Desde una conexión de datos. Podemos crear una aplicación en base a un modelo de datos, desde una hoja Excel a una base de datos SQL en Azure. Desde este modelo la aplicación nos implementa las pantallas necesarias para conseguir un CRUD, al más puro estilo LightSwitch.
  - En blanco. Total libertad para implementar lo que necesitemos en una aplicación desde cero.

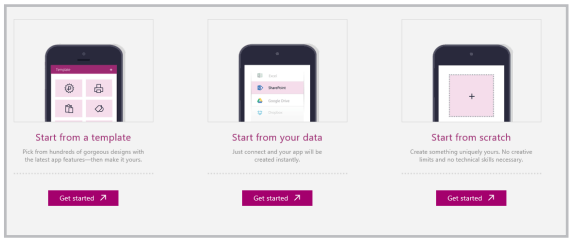


Imagen 5. Plantillas para crear PowerApp.

- En nuestro caso para este primer ejemplo vamos a elegir una plantilla para ahorrarnos trabajo, y en concreto una plantilla de Contactos para una aplicación móvil. Esta plantilla genera una aplicación básica que permite gestionar una lista de contactos almacenados en una hoja de Excel llamada “data.xlsx”, con una pestaña Contacts. Esta hoja Excel se guarda junto a nuestra aplicación PowerApps (OneDrive en este caso).
- Todo el poder de PowerApps reside en la ribbon superior desde la cual podemos utilizar todas las herramientas existentes.

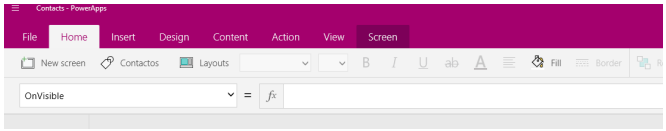


Imagen 6. Barra de herramientas, modela tu aplicación.

- Como explicar cada uno de los controles que existen podía darnos para un manual extenso, vamos a centrarnos en modificar un poco la aplicación para poder adaptarla a nuestras necesidades. Si revisamos de arriba abajo la aplicación que nos genera la plantilla de contactos no es más que un CRUD de nuestra base de datos Contacts (Excel), tal y como vemos en la apartado Content/Data Source en la ribbon superior.
- Vamos a añadir un nuevo campo a nuestra hoja Excel que será el país de nacimiento del usuario y que gestionaremos desde las pantallas de la aplicación. Para ello vamos a guardar nuestra aplicación en la Conexión que configuramos al inicio del artículo, en mi caso he seleccionado una conexión a mi OneDrive personal, con el nombre de MyContact.
- Al guardar la aplicación se genera una carpeta PowerApps/Templates/Contacts + idPowerApps/, y dentro encontramos tanto las imágenes que consume la app como la hoja Excel que mantiene la información de los contactos.
- Vamos a añadir en esa hoja Excel una columna llamada Country para poder guardar el país de nacimiento del usuario.

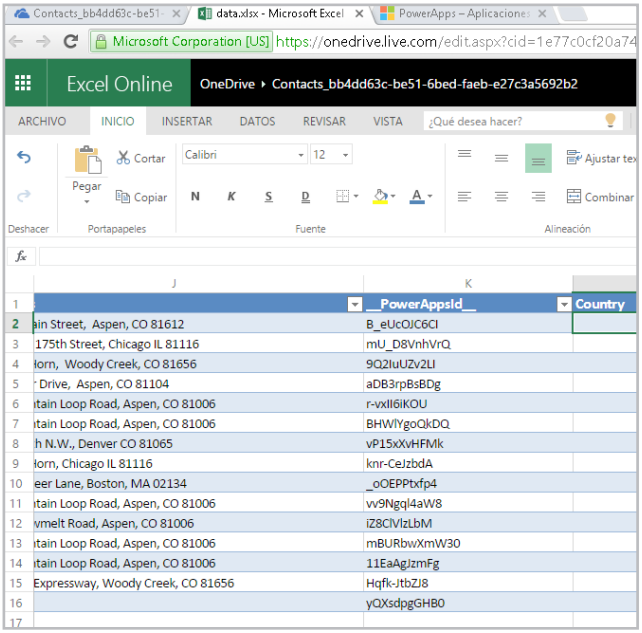


Imagen 7. Almacena tus datos en Excel

- Una vez actualizada la Excel, vamos a modificar las pantallas de la aplicación móvil para poder controlar este nuevo campo. En el margen izquierdo de nuestro editor de PowerApps, encontramos todas las pantallas de la aplicación.

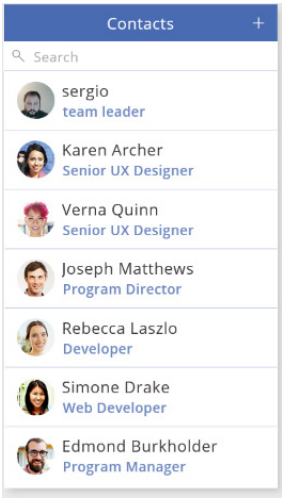


Imagen 8.- Primera pantalla de la Aplicación

- La primera pantalla no es más que un buscador de usuarios por lo que no tenemos que modificar nada en esta, aunque un poco más adelante vamos a hacer una nueva versión de ella ya que es bastante interesante. Nos vamos a centrar en la segunda y tercera pantalla que son la vista de detalle y de edición del contacto.
- Añadiremos un label y un campo de texto para nuestro campo. Para ello en la ribbon seleccionamos insert, y posteriormente le daremos el formato necesario con la pestaña "Home" en la ribbon.
- Además, añadiremos de forma similar un textBox pero en este caso para mapear el campo Country que añadimos en nuestra Excel de Contactos.
- Para conseguir esto nos vamos a fijar en la ribbon de propiedades del campo. Teniendo seleccionado el campo en pantalla, seleccionamos la propiedad "Text" y la configuramos tal cual vemos en la imagen.

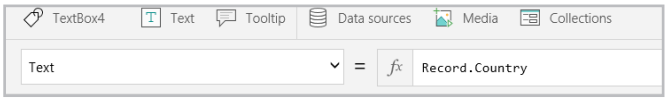


Imagen 9. Propiedades de un control

he definido PowerApps como una "herramienta" y muy ligada al paquete Office; y en verdad es eso, una herramienta que permite crear aplicaciones de negocio tanto para móviles como para tablets

- Como vemos tenemos un objeto "Record" que contiene el metadato country del contacto que estamos visualizando, cuando analicemos la pantalla buscadora de contactos veremos cómo le pasamos el metadato del contacto a las distintas pantallas.
- Este proceso lo repetiremos para la vista de edición (aunque en este caso el segundo campo será un control del tipo text input) de forma que nuestras pantallas queden de la siguiente forma:

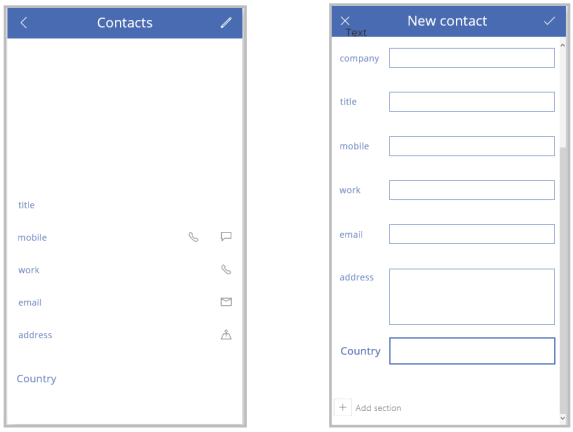


Imagen 10. Vista detalle de un contacto y Vista edición de un contacto.

- Es posible que la conexión Contacts que teníamos configurada como origen de datos no refresque el dato "Country" que hemos añadido. Para evitar problemas, lo mejor es que en el apartado Content/Data Sources borremos la antigua conexión a nuestra excel data. xslx, y volvamos a añadirla llamando a la nueva conexión Contacts consiguiendo así refrescar la conexión.
- Además de actualizar la conexión debemos configurar el proceso de Update que tenemos en el formulario de edición. Si seleccionamos el botón del formulario y seleccionamos en la ribbon OnSelect, podemos ver la fórmula de actualización o creación del modelo.

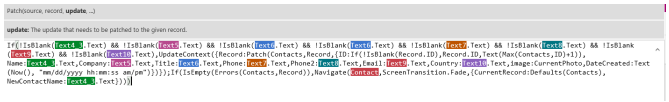


Imagen 11. Fórmulas para eventos.

- Deberemos modificar la formula y dejarla de forma similar a la imagen anterior, agregando nuestro campo Country de la forma "Country:ID_TEXTBOX". El identificador del campo de texto lo encontramos seleccionando el control, y en la pestaña "Home" encontraremos algo similar a esto Text5.
- Dado que ya hemos añadido nuestro nuevo campo a nuestro origen de datos y a las dos pantallas que lo consumen, vamos a probar que todo está correcto, lanzando la aplicación y grabando un nuevo usuario.

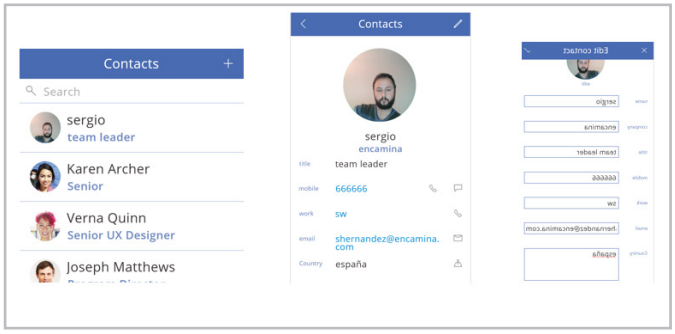


Imagen 12. Vistas de la app MyContacts.

- Si accedemos a la Excel de datos, podemos comprobar que el nuevo campo country se refresca de forma correcta.
- Una vez hemos conseguido añadir un nuevo campo a nuestra aplicación, vamos a replicar la pantalla del

buscador, que posiblemente sea la más interesante. Necesitamos lo primero añadir una nueva pantalla a nuestra aplicación, para ello sobre una pantalla ya existente seleccionamos “New Screen”.

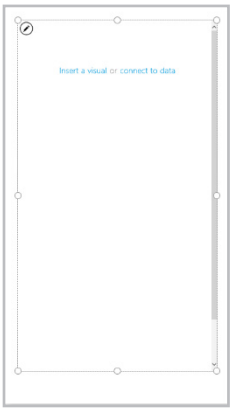


Imagen 14. Custom Gallery en PowerApps.

- Sobre la nueva pantalla vamos a insertar una nueva “Custom Gallery” con orientación vertical, y después editando la galería podemos ir añadiendo controles. En este caso vamos a insertar una imagen y un campo de texto.
- Al seleccionar en nuestra galería veremos una propiedad llamada “Items”, vamos a inicializarla a nuestro valor de Conexión, en nuestro caso Contacts.

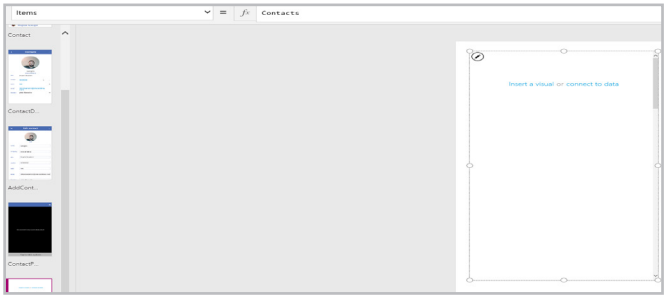


Imagen 15. Mapeando nuestro datasource.

- Al añadir un control del tipo Imagen, necesitamos mapear nuestro campo con el campo Imagen de nuestro origen de datos, para ello inicializamos el control de la forma “thisItem = Field”, o para nuestro caso “ThisItem = Image”.

Imagen 16. Mapeando campos/controles.

- Siguiendo este proceso podemos además de añadir la imagen, el campo Nombre y el Title quedando la pantalla como que viene por defecto.
- Solo nos quedaría una cosa para simular la pantalla original, y es la navegación a la vista de detalle de las fichas del usuario.
- Para ello, vamos a añadir un control del tipo Shape, y desde la pestaña Home de la ribbon le damos estilos quitándole el relleno, dándole un color al borde y ajustando en pantalla para que contenga a la imagen y a los campos de texto que hemos añadido.

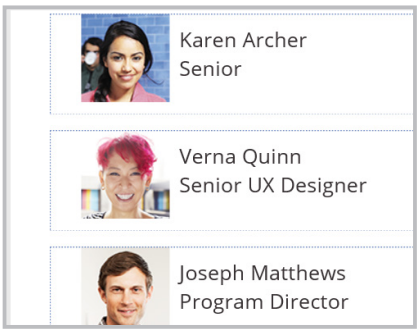


Imagen 17. Custom gallery con estilos.

- Como vemos en la imagen, ya tenemos algo muy similar a la pantalla del buscador, para poder añadir la navegación en el evento OnSelect del control Shape insertamos la formula Navigate (ContactDetail,ScreenTransition. Fade,{Record:ThisItem}).
- Si analizamos la formula, lo que hace es redirigir a la vista “ContacDetail”, con una transición “Fade”, y lo más importante mandar por contexto el ítem seleccionado, así desde la vista de detalle o en la de edición podemos tener acceso al objeto Contacto que necesitamos (Record).
- Con todo esto podemos decir que sabemos crear una aplicación desde una plantilla, gestionar las conexiones a datos (en este caso desde una Excel, pero no es muy distinto gestionar una conexión SQL contra Azure) y hemos empezado a manejar la navegación y los eventos en nuestros controles.

Compartir Nuestra Aplicación

Siguiendo un poco el hilo de nuestra introducción, hemos visto que PowerApps nos permite, sin necesidad de desarrollar código a medida, conectar contra nuestros orígenes de datos nuestra aplicación; hemos podido hacer una aplicación móvil con poco esfuerzo, y nos queda ver como compartir nuestra aplicación de Contactos.

Si accedemos a nuestra aplicación de escritorio PowerApps, lo primero que nos encontramos es el listado de nuestras aplicaciones.

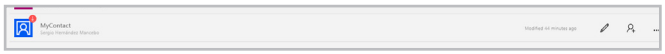


Imagen 18. Listado de nuestras apps.

Para nuestro ejemplo he guardado la aplicación bajo el nombre MyContact, en la conexión al OneDrive que configuramos al inicio. Si seleccionamos  podemos ver los permisos de esta aplicación, y nos permite compartir con nuestra organización (en mi caso los contactos de Office 365) la aplicación de forma directa.

El tercer punto fuerte de las PowerApps es, bajo mi punto de vista, el concepto “Compartir” que incluye la herramienta.

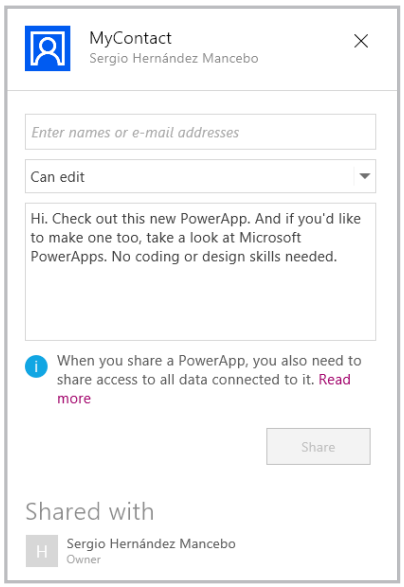


Figura 19. Compartiendo una PowerApp .

Basta con insertar el mail del usuario y asignarle el nivel de permisos para que, tanto desde la aplicación PowerApps de W10 o desde la aplicación cliente de iOS, pueda utilizar nuestra aplicación (por ahora no tenemos versión Android, aunque está en fase de desarrollo tal y como indica Microsoft).

Conclusiones: ¿Hasta donde podemos llegar con PowerAPP?

Posiblemente hacer un juicio o valoración de hasta dónde va a llegar este modelo de aplicación, sea cuanto menos temerario con el poco recorrido que tiene a día de hoy. Pero si podemos empezar a hacernos una ligera idea de cómo vamos a tener que usar esta tecnología. Después de este primer repaso puedo asegurar a todo aquel que no haya tenido el gusto de crear una PowerApp que el esfuerzo invertido respecto al resultado obtenido es más que bueno: es bastante agradecido hacer una aplicación con esta herramienta. Para los poco enamorados de la maquetación, HTML o XAML para hacer sus aplicaciones móviles, con PowerApps pueden encontrar un aliado.

Pero todo esto sin volvernos locos, en mi humilde opinión PowerApps viene a complementar a otras formas de hacer aplicaciones móviles, para casos en los que usuarios de negocio no quieran tener dependencia total con equipos de desarrollo para modificaciones en la app, o por ejemplo para aplicaciones que necesiten tener un coste muy ajustado.

Centrándonos un poco más en la herramienta, y sin entrar en la parte de PowerApps Enterprise que va aportar mucho más a la implementación; esta versión nos aporta un potente editor de formularios, que bien se puede quedar corto en algunos casos como puede ser el control de la navegación, gestión de las conexiones o eventos de nuestros controles. Por corto me refiero a que al final acabamos requiriendo de nociones importantes en desarrollo o infraestructuras, como hemos podido ver con las fórmulas para actualizar nuestro modelo Contacts en la vista de edición, por este motivo se me hace difícil ver a un usuario de negocio manejando este tipo de eventos.

Por el resto queda mucho por andar en cuanto a conexiones a servicios o datos, ya que a nivel de usabilidad / rendimiento está por ver el resultado, pero el concepto de poder conectar nuestros entornos en una app móvil sin mucho esfuerzo cuanto menos es ilusionante. Pero en mi modesta opinión, el mundo más interesante por descubrir va a ser el de la generación de formularios desde un modelo de datos, que sin duda trataremos en el próximo artículo, y de conseguir funcionar mejor que sus predecesores va a ser un ahorro en tiempo y coste muy importante.

¡En líneas generales una herramienta más que interesante y recomendable su uso a la espera de lo que nos vaya regalando con su evolución!

SERGIO HERNÁNDEZ MANCEBO
Team Leader en ENCAMINA
shernandez@encamina.com

72

Nosotros

Alberto Diaz

Alberto Díaz es SharePoint Team Lead en Encamina, liderando el desarrollo de software con tecnología Microsoft. Para la comunidad, ha fundado TenerifeDev (www.tenerifedev.com) con otros colaboradores, un grupo de usuarios de .NET en Tenerife, y coordinador de SUGES (Grupo de Usuarios de SharePoint de España, www.suges.es) y colaborador con otras comunidades de usuarios. Microsoft MVP de SharePoint Server desde el año 2011 y asiduo conferenciante en webcast y conferencias de tecnología de habla hispana.

Sitio Web: http://blogs.encamina.com/negocios-sharepoint/
Email: adiazcan@hotmail.com
Blogs: http://geeks.ms/blogs/adiazmartin
Twitter: @adiazcan

Fabián Imaz

Fabián Imaz, MVP de SharePoint Server trabaja en el mundo del desarrollo de software desde hace más de 10 años, teniendo la suerte de trabajar en distintas arquitecturas y tecnologías Microsoft. Pertenece a la firma Siderys, http://www.siderys.com empresa de desarrollo de Software especializada en SharePoint 2007/2010/2013 y en desarrollo de soluciones inteligentes. Desde los comienzos Fabián ha trabajado en distintitas comunidades donde organiza y promueve eventos locales para la difusión de tecnología dentro de los miembros de las mismas. Es director de la carrera SharePoint 2010 y SharePoint 2013 en Microsoft Virtual Academy, http://www.mslatam.com/latam/technet/mva2/ Home.aspx y cuenta con un sitio en CodePlex con varios desarrollos http://siderys.codeplex.com.

Sitio Web: http://www.siderysbsn.com
Email: fabiani@siderys.com.uy
Blogs: http://blog.siderys.com
Twitter: @fabianimaz

Gustavo Velez

Gustavo Velez es Ingeniero Mecánico y Electrónico; trabaja en el diseño e implementación de sistemas de IT basados en tecnologías de Microsoft, especialmente SharePoint, para Avanade (http://www. avanade.com), una compañía multinacional de IT. Propietario del sitio especializado en información sobre SharePoint en español http://www. gavd.net y autor de seis libros sobre SharePoint y sus tecnologías.

Sitio Web: http://www.gavd.net
Email: gustavo@gavd.net
Blogs: http://geeks.ms/blogs/gvelez/

Juan Carlos González Martín

Juan Carlos González Martín. Ingeniero de Telecomunicaciones por la Universidad de Valladolid y Diplomado en Ciencias Empresariales por la Universidad Oberta de Catalunya (UOC). Cuenta con más de 11 años de experiencia en tecnologías y plataformas de Microsoft diversas (SQL Server, Visual Studio, .NET Framework, etc.), aunque su trabajo diario gira en torno a las plataformas SharePoint & Office 365. Juan Carlos es MVP de Office 365 desde 2015 (anteriormente fue reconocido por Microsoft como MVP de SharePoint Server desde 2008 hasta 2015), coordinador del grupo de usuarios .NET de Cantabria (Nuberos.Net) y co-fundador del Grupo de Usuarios de SharePoint de España (SUGES, www. suges.es), del Grupo de Usuarios de Cloud Computing de España (CLOUDES) y de la Comunidad de Office 365. Desde el año 2011 participa junto con Gustavo Vélez y Fabián Imaz en la dirección de la revista CompartiMOSS (www.compartimoss.com). Hasta la fecha, ha publicado 6 libros sobre SharePoint & Office 365, así como varios artículos en castellano y en inglés sobre ambas plataformas. Actualmente es Cloud & Productivity Advisor en MVP CLUSTER.

Email: jcgonzalezmartin1978@hotmail.com
Blogs: http://geeks.ms/blogs/jcgonzalez &
http://jcgonzalezmartin.wordpress.com/

¿Desea colaborar con CompartiMOSS?

La subsistencia del magazine depende de los aportes en contenido de todos. Por ser una revista dedicada a información sobre SharePoint en español, todo el contenido deberá ser directamente relacionado con Microsoft SharePoint y escrito en castellano. No hay limitaciones sobre el tipo de articulo o contenido, lo mismo que sobre el tipo de versión.
Si desea publicar algo, por favor, utilice uno de los siguientes formatos:

- Artículos de fondo: tratan sobre un tema en profundidad. Normalmente entre 2000 y 3000 palabras y alrededor de 4 o 5 figuras. El tema puede ser puramente técnico, tanto de programación como sobre infraestructura, o sobre implementación o utilización.

- Artículos cortos: Máximo 1000 palabras y 1 o 2 figuras. Describen rápidamente una aplicación especial de SharePoint, o explica algún punto poco conocido o tratado. Experiencias de aplicación de SharePoint en empresas o instituciones puede ser un tipo de artículo ideal en esta categoría.

- Ideas, tips y trucos: Algunos cientos de palabras máximo. Experiencias sobre la utilización de SharePoint, problemas encontrados y como solucionarlos, ideas y trucos de utilización, etc.

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

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

Envíe sus proposiciones, artículos, ideas y comentarios a la siguiente dirección:
revista@compartimoss.com adiazcan@hotmail.com
fabiani@siderys.com.uy jcgonzalezmartin1978@hotmail.com
gustavo@gavd.net


Comparti
MOSS