DIWUG

Dutch Information Worker User Group

SharePoint®

eMagazine

Acrobat version
with hyperlinks

Wow!

80 pages filled with expert information available for Acrobat® Reader®, e-Reader and in print (limited qty.)

#13

May 2014

Contents

Colofon

DIWUG SharePoint eMagazine

Nr. 13, May 2014

Publisher:
Stichting Dutch Information Worker User Group (DIWUG)
http://www.diwug.nl

Editors:
Marianne van Wanrooij
marianne@diwug.nl
Mirjam van Olst
mirjam@diwug.nl

Special thanks to:
All authors and sponsors!

Design and layout:
Barth Sluyters
barthsluyters@telfort.nl

13: Our lucky number

Welcome to our lucky number 13 DIWUG SharePoint eMagazine! We have a record number of 10 articles in this edition, so a big "THANK YOU!" to all our authors is in order. You are all amazing!

Of course we'd also like to thank our sponsors, as without them it would just be articles and not a good looking magazine.

It's a busy time for DIWUG, with this new edition of the magazine coming out and SharePoint Saturday Netherlands coming up on May 24th. We are still having a lot of fun though and we keep being pleasantly surprised by how great the SharePoint Community is, both in the Netherlands, but also around the world. The positive reactions to the magazine, the user group events and the SharePoint Saturday filling up within ours and sponsors fighting over sponsor slots for SharePoint Saturday. It all shows how tight our community is and it is what has been keeping us going for the last 10 years.

After SharePoint Saturday Netherlands the next event will be SharePoint Evolution Conference Roadshow. It's like the great SharePoint conferences that Combined Knowledge usually organizes in London, only this time they are taking the show on the road! It's 12 days of conference in 12 different cities across the UK. MVP Eric Shupps described it perfectly in his blog post: http://www.binarywave.com/blogs/eshupps/Lists/Posts/Post.aspx?ID=290.

On November 18 and 19 the SharePoint Connect conference 2014 will be held again in Amsterdam. DIWUG is supporting the conference again and we will be running the Ask the Experts area. The conference is supporting DIWUG by giving our members 25% off the registration fee. Use TF493 to get the DIWUG member discount.

If you want to write for, or sponsor our magazine that has a world-wide audience with around 10.000 downloads and 800 printed copies don't hesitate to contact Marianne or Mirjam.

Mirjam van Olst

Editor DIWUG SharePoint eMagazine

mirjam@diwug.nl

This issue is sponsored by

When using the Adobe® Acrobat® version all sponsor information is hyper linked.

9,1

Of: waarom een 7,44 voor klanttevredenheid ons een beetje tegenvalt.

Als zakenblad Incompany* je bedeelt met een 7,44 voor klanttevredenheid, mag je best tevreden zijn. Zeker als je met een 7,8 voor accuratesse en een 7,64 voor vakkennis ook nog eens eindigt op plaats 7 van de 151 onderzochte bedrijven. Toch hangt bij CGI de vlag niet uit. Onze ambities liggen namelijk hoger. Wie in internationale metingen al meer dan tien jaar op rij een 9,1 of hoger scoort, kan zo'n 7,44 onmogelijk voldoende vinden.

En dus zetten we een tandje bij. Dat zijn we aan onze klanten verplicht, vinden we als vijf na grootste IT-dienstverlener ter wereld. Onze kracht schuilt immers in de driehoek hoogwaardige IT- en consultancy-expertise, diepgaande industriekennis én nauwe lokale partnerships met opdrachtgevers. Een noodzakelijke voorwaarde voor onze end-to-end dienstverlening op het gebied van, in veel gevallen 'missiekritische', bedrijfsprocessen en IT.

Gelukkig zijn we goed op weg. Dat bewijst ons indrukwekkende trackrecord. En het feit dat we 95 procent van onze projecten leveren binnen tijd en budget. Maar goed kan altijd beter. Alle reden dus om ons werk met nóg meer toewijding voort te zetten. Tot onze klanten helemaal tevreden zijn. Dat is ons commitment. Naar onze opdrachtgevers. En naar onszelf.

* "CGI verrast enorm met een eindcijfer dat de zevenenhalf nadert. Overall ranking 2012 was plaats 47. Nu plaats 7 van de 151 organisaties. Opvallend zijn verder de topcijfers voor 'accuratesse' (7,8) en 'vakkennis' (7,64)." Bron: Incompany 100

CGI

Experience the commitment®

Implementing eDiscovery in SharePoint 2013

by Maarten Eekels

More and more organisations face more and more litigations. Whether they are shareholder lawsuits, fraud cases, or competitive investigations, litigation cases have proven to be costly, time consuming, and business disruptive. The new eDiscovery solution of SharePoint 2013 could really help organisations to lower eDiscovery costs, mitigate risks of data tempering or accidental deletion, and minimize business interruption.

With SharePoint 2013 it is possible to run an eDiscovery case on SharePoint, Exchange, Lync, and File Shares at the same time, from one unified central management console. Sander van den Hoven already did a great article about eDiscovery in SharePoint 2013 from a functional perspective in issue #9 of the DIWUG Magazine. In my article I would like to cover the technical architecture of eDiscovery throughout the Microsoft 2013 product stack, a step-by-step guide on implementing eDiscovery (both in an Online and On-premises environment) and some implications that are worth considering during and after implementation.

Conceptual architecture

Figure 1 shows the conceptual architecture of eDiscovery in SharePoint 2013.

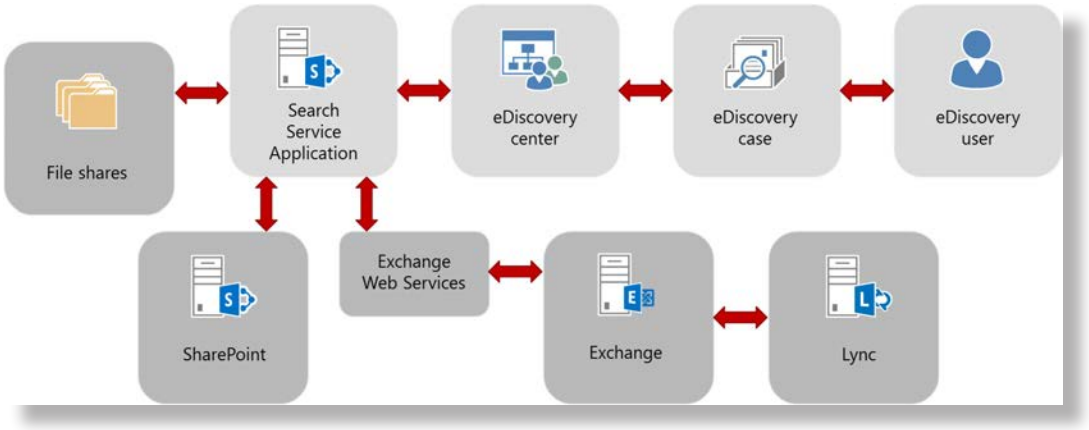

Figure 1: Conceptual architecture.

The eDiscovery user is the person going through an eDiscovery Case Lifecycle. This involves creating a new eDiscovery case site in SharePoint, discovering relevant content based on search queries, optionally placing all found content on hold, refining the content before exporting it, and finally releasing the hold and closing the case. All case sites reside in the eDiscovery center site collection.

The Search Service Application (SSA) is the core of the eDiscovery solution in SharePoint 2013. Of course, when a user types in a query in an eDiscovery case, this query is executed on the search index by the SSA. But the SSA does more. For example it also keeps track of hold and release hold commands in an actions table. At regular intervals (by default 1 hour), the eDiscovery in-place hold timer job is kicked off in a SharePoint farm. The timer job consults the actions table to see if there are any pending actions, executes the actions, and sends back status updates.

This means that when a user decides to put content on hold from an eDiscovery case site, the content is not actually put on hold until the timer job has run!

This might sound as a flaw, but if you consider the “old days”, when you had to crawl through all your content manually, and set aside relevant content on an item-per-item basis if you wanted it to be put on hold, a process which could take days, if not weeks, I guess a delay of up to one hour is not that bad.

With eDiscovery in SharePoint 2013 you can process content from SharePoint (of course!), Exchange, Lync, and File Shares.

For SharePoint and File Shares, eDiscovery uses the SharePoint search index. This means that all SharePoint sites and File Shares you want to include in an eDiscovery case have to be added as Content Sources in your Search Service Application and the Content Access Account you use to crawl the content, must have access to all content in these sources.

For Exchange content, the process works differently. Exchange 2013 uses its own eDiscovery infrastructure to search for content and put content on hold (actually, Exchange 2013 even has its own eDiscovery Center, which you could use if you would only want to process Exchange data, or if you don't have SharePoint 2013 yet available). SharePoint uses the Exchange Web Services (EWS) API to pass on eDiscovery calls to Exchange, and receives back results or responses through EWS as well.

And then there is Lync. With Lync 2013 it is possible to store conversation histories in Lync (which actually stores it in the LcsLog database in SQL Server) or in Exchange. If you want eDiscovery to work on Lync content, you need to configure Lync to store the conversation histories in Exchange. Lync then uses Exchange Web Services to send archived information to Exchange.

Compatibility

As explained before, it is possible to search for content, to put content on hold, and to export content. For the entire 2013 suite of products from Microsoft (SharePoint 2013, Exchange 2013, and Lync 2013) all these options are indeed available. For older versions of these products, or for File Shares, this is not the case. For example, it is possible to crawl a SharePoint 2010 farm and add the results to the search index. Therefore it is possible to identify relevant content in an SP2010 farm by querying it from an eDiscovery case. But it is not possible to put this content on hold or include it in the export. You need to manually download the content and set it aside. Another example: Exchange 2010 and Lync 2010 are not supported at all.

Table 1 shows an overview of what is possible and what not for different content sources and product versions.

Source	Search		In-Place Hold		Export	
	On-Prem	O365	On-Prem	O365	On-Prem	O365
SharePoint 2013	Yes	Yes	Yes	Yes	Yes	Yes
SharePoint 2010	Yes	n/a	No	n/a	Yes	n/a
SharePoint 2007	Yes	n/a	No	n/a	Yes	n/a
Exchange 2013	Yes	Yes	Yes	Yes	Yes	Yes
Exchange 2010	No	n/a	No	n/a	No	n/a
Lync 2013 (archived in Exchange 2013)	Yes	Yes	Yes	Yes	Yes	Yes
Lync 2010	No	n/a	No	n/a	No	n/a
File shares	Yes	No	No	No	Yes	No

Table 1: eDiscovery compatibility.

We would be very glad to receive your article on SharePoint 2013 for our 14th eMagazine.

Configuration of eDiscovery

In order to make eDiscovery work in SharePoint 2013 (and Exchange / Lync 2013), the following steps need to be performed:

1. Install Exchange Web Services API
2. Configure communication between servers
3. Create eDiscovery Center site collection
4. Grant permissions
5. Configure Search

1. Install Exchange Web Services API

The Exchange Web Services API has to be installed on every SharePoint server in your farm. First, download the EWS API from http://www.microsoft.com/en-us/download/details.aspx?id=35371.

Next, use Windows PowerShell (run as administrator) to run the command in Listing 1 (you need to run it using PowerShell, because some files need to be placed in the Global Assembly Cache, by just clicking and installing it, required files will not be placed in the GAC):

```
msiexec /i EwsManagedApi.msi addlocal= “ExchangeWebServicesApi _ Feature,
ExchangeWebServicesApi _ Gac”
```

Listing 1: Installing the Exchange Web Services API on the SharePoint servers.

2. Configure communication between servers

Next we need to configure the server to server trust between SharePoint and Exchange. Let’s start with the SharePoint side.

First we need to make sure the Exchange SSL certificate you use is trusted on your SharePoint machines. For 3rd party certificates this is mostly the case, but especially if you use self-signed certificates, this could be a point of attention. You can find out if the SSL certificate of your Exchange environment is trusted by browsing to Outlook Web App in Internet Explorer. If you receive a certificate warning, do the following:

- Accept to trust the certificate by clicking Continue to website.
- Click Certificate Error info in Internet Explorer next to the Address bar, and then click View Certificates.
- Select Install Certificate and then select Place all certificates in the following store.
- Select the checkbox to show physical stores.
- Install the certificate to Trusted Root Certification Authorities > Local Computer.

Again, do this on every SharePoint server in your farm.

The same procedure is applicable if your Exchange servers don’t trust the SSL certificate you use for your SharePoint web application where you are going to create the eDiscovery Center site collection.

If you decide not to use an SSL certificate for your SharePoint web application, you need to configure your Security Token Service to allow http traffic instead of https. Note that it is a best practice to run your SharePoint web application on HTTPS. If you don’t want to do this you need to run the commands shown in Listing 2 on your SharePoint environment.

```
$sts = Get-SPSecurityTokenServiceConfig
$sts.AllowMetadataOverHttp = $true
$sts.AllowOAuthOverHttp = $true
$sts.Update()
```

Listing 2: Allowing OAuth over HTTP if you are not using HTTPS.

Now we are ready to create the trust using the command in Listing 3.

```
New-SPTrustedSecurityTokenIssuer –MetadataEndpoint “https://<ExchangeURL>/
autodiscover/metadata/json/1” –Name “<ExchangeFriendlyName>”
```

Listing 3: Making the SharePoint farm trust the Exchange farm.

And finally we need to grant the Exchange service principal full control permissions to the SharePoint site subscription as shown in Listing 4.

```
$exchange=Get-SPTrustedSecurityTokenIssuer
$app=Get-SPAppPrincipal -Site http://<SharePointURL> -NameIdentifier
$exchange.NameId
$site=Get-SPSite http://<SharePointURL>
Set-SPAppPrincipalPermission -AppPrincipal $app -Site $site.RootWeb -Scope
sitesubscription -Right fullcontrol –EnableAppOnlyPolicy
```

Listing 4: Granting the Exchange service principal full control to the SharePoint site subscription.

On the Exchange side we only need to run one script, which is a default script provided by Exchange 2013 to configure partner applications. Before you run this script in the Exchange Management Shell, make sure the user you are logged in with, is member of the Organization Management role group in Exchange. Next run the script from Listing 5.

```
cd c:’Program Files’\Microsoft\’Exchange Server’\V15\Scripts
.\Configure-EnterprisePartnerApplication.ps1 -AuthMetadataUrl
https://<SharePointURL>/ _ layouts/15/metadata/json/1 -ApplicationType
SharePoint
```

Listing 5: Making the SharePoint farm a partner application of the Exchange farm.

Now we are ready to create the eDiscovery Center site collection.

3. Create eDiscovery Center site collection

The previous two steps are required for on-premises installations only. From here the steps are applicable to Office 365 environments as well.

In the Central Administration site of your SharePoint on-premises farm, or SharePoint admin center if you are in a SharePoint Online environment, create a new site collection, and choose the eDiscovery Center site collection template. Make sure the user you are logged in with, is member of the Farm Administrators group in SharePoint (on-premises) or is a SharePoint Online admin (Office 365).

4. Grant permissions

The eDiscovery user managing a particular eDiscovery case is only able to identify relevant content to which this user has access. This means that in order to identify all relevant content, the user must have access to all possibly relevant content.

First let’s have a look at making SharePoint content discoverable. In on-premises farms the preferred method is to grant eDiscovery users read access to all content in a web application by using a web application user policy. You probably want to use a security group for this, because if you add/remove individual users, a full search crawl is triggered. If you use a security group, you can add/remove users to and from this group as you wish, without the need to perform a full crawl.

The second method (and unfortunately the only method in SharePoint Online) is to add all eDiscovery users as a site collection administrator for every site collection you want to be discoverable.

In Exchange you have to add all eDiscovery users to the Discovery Management role group. This allows those users to use In-Place eDiscovery to search all Exchange 2013 mailboxes and access all email content in user mailboxes. By default, this permission isn’t assigned to any user, including members of the Organization Management role group.

And then there is File Shares. You have to make sure that all eDiscovery users have access to all relevant content on File Shares you want to be discoverable.

Finally there is the search crawl log. It could be very relevant to an eDiscovery case to export a list of content items that could not be indexed. These could be encrypted files or files of a file type SharePoint Search doesn't know how to handle. Since these items could be relevant, the SharePoint eDiscovery export provides the option to include this list, based on information in the search crawl log. But this does require the eDiscovery user to have access to this log. For on-premises scenarios, use the PowerShell commands from Listing 6 in the SharePoint Management Shell to grant users access to the log.

```
$ssa = Get-SPEnterpriseSearchServiceApplication
Set-SPEnterpriseSearchCrawlLogReadPermission -SearchApplication $ssa
-UserNames "user1;user2"
```

Listing 6: Grant users access to the Search crawl log.

In SharePoint Online there is a Crawl Log Permissions option in the search administration section of the SharePoint admin center.

5. Configure Search

The last thing we need to do is to configure search to include all content sources we want to be discoverable in our eDiscovery process.

In an on-premises farm you can add external SharePoint sources and/or File Shares as additional content sources in the Central Administration site (make sure your default content access account has read permissions to those content sources).

In both an on-premises environment and in Office 365 you have add a search result source for Exchange. This can either be done from the Central Administration site (on-premises) / SharePoint admin center (SharePoint Online), or from the eDiscovery Center site collection. The latter is the preferred location to add the Exchange result source, because then the result source is only available in the eDiscovery site collection, instead of the entire farm. Just make sure you do NOT create it from an eDiscovery Case site, that won't work.

- Type in a name and description for the result source
- Choose Exchange as protocol
- Using Autodiscover will work for Office 365 and most on-premises environments. If this doesn't work, manually type in your Exchange Web Services URL (typically https://<ExchangeURL>/EWS/Exchange.asmx)
- Leave the Query Transform settings to {searchTerms}

Now your eDiscovery environment is all set. You should be able to create eDiscovery cases, discover all relevant content (and optionally place the content on hold), and export the content.

Considerations

There are some considerations you might want to take into account when you are going to deploy and use SharePoint 2013 as your eDiscovery solution in your organization.

Number of eDiscovery Centers

SharePoint 2013 supports some hybrid scenarios, like for search and for business connectivity services, but eDiscovery is not one of them. That means that you will need separate eDiscovery Centers for your on-premises and your online environment.

Also, in on-premises situations, you have to create a separate eDiscovery Center for each Search Service Application you have running (and want to use for eDiscovery).

Finally, if you have multiple Exchange forests, you are going to need a separate eDiscovery Center for each forest.

Permissions

eDiscovery users need to have access to all content you want to be discoverable for eDiscovery cases. Some organizations might find that troubling. On the other hand, we have lived in situations for years and years where IT administrators have had domain admin rights and access to all content on every file share or e-mail box. Now we have a legal team, or auditors, who have the same.

Of course it is also possible to assign relevant permissions to security groups (best practice anyway) and only add an individual to those groups when this user is going to work on an eDiscovery case.

Finally

I hope you found this article valuable. eDiscovery in SharePoint 2013 is a very powerful tool to lower eDiscovery costs for many organizations, and relatively easy to implement. Happy discovering!

Manage the search schema in Office 365

Within Office 365 you can manage the search schema and create new Managed Properties to use in your search query. Out of the box you can only create new text and yes/no fields. Luckily there are a set predefined properties that you can use for Integer, Decimal, Date and Time and Doubles. They all start with the Refinable prefix so you can search for them and by adding the mapping you need and providing an alias you can use them as normal Managed Properties. The only downside is that it can take up to 24 hours before the mapping take effect. Besides that it is a great way to provide you with the tools to write queries that are tailored for your content.

More info: http://www.sharepointappie.nl/office-365-search-schema/

Albert-Jan Schot

Migrating Term Sets

by Octavie van Haaften

With SharePoint 2010 Service Applications were introduced. One of the service applications is the Managed Metadata Service Application. It contains a Term Store for managing terms. In libraries and lists you can use columns with data type Managed Metadata to populate that field with values from a Term Set. At some point in time you may have to migrate your content along with the Term Sets to another SharePoint environment. This could be from Development to Test to Acceptance to Production, or maybe from a SharePoint 2010 farm to a SharePoint 2013 farm. This article focuses on 2 common scenarios: migrating the Managed Metadata service application database and migrating specific Term Sets.

It's all about ID's

First of all, let's take a look what we are dealing with. Many, many things in SharePoint have ID's. Most of them are beautiful GUIDs and they are often displayed differently: with or without the curly brackets, with or without dashes, etc. The Term Store is no exception. Figures 1, 2 and 3 show several ID's.

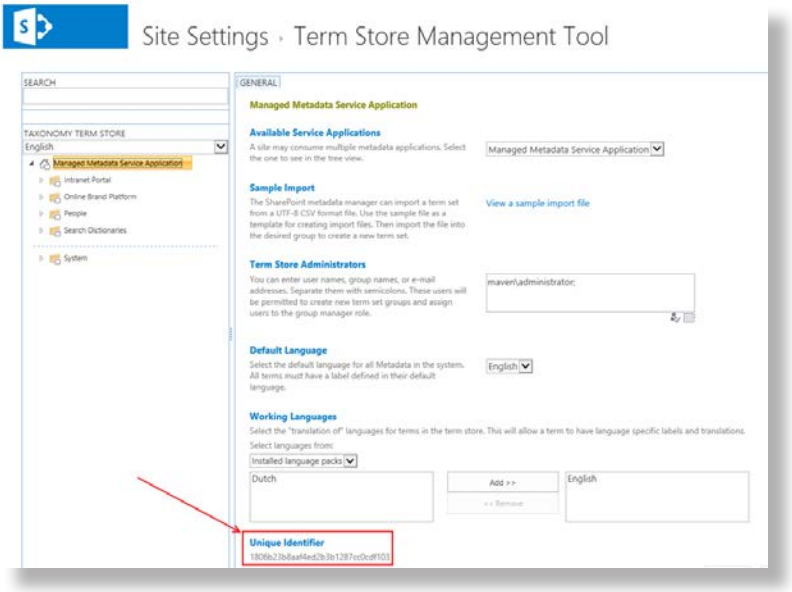

Figure 1: Term Store ID.

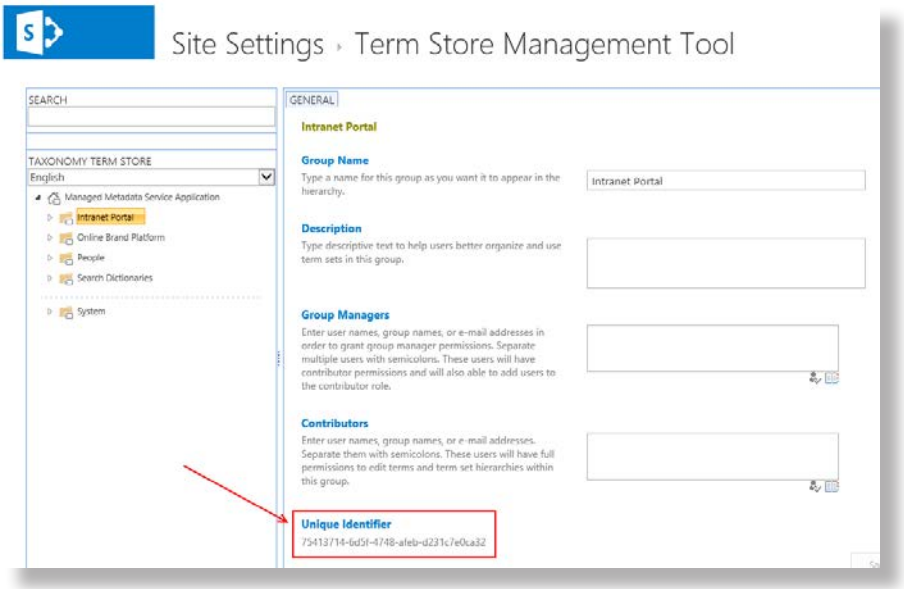

Figure 2: Term Group ID.

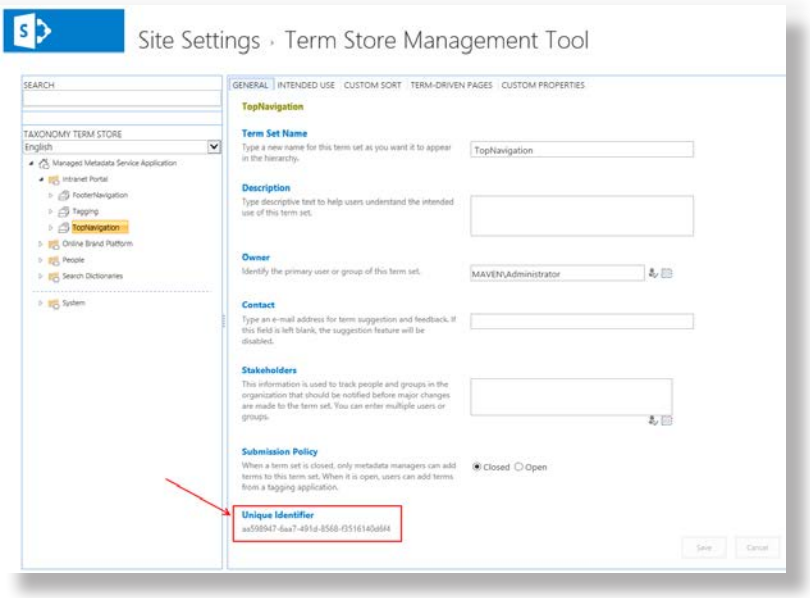

Figure 3: Term Set ID.

These ID's are used and stored when you define a SharePoint column of data type Managed Metadata.

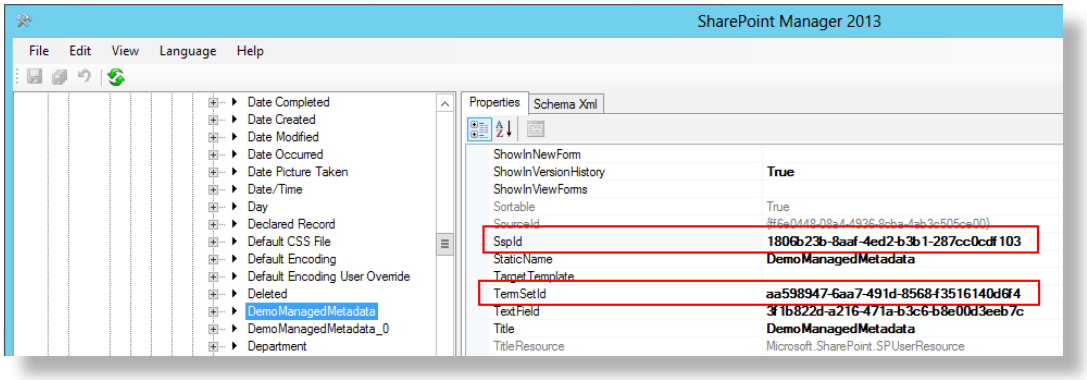

Figure 4: Properties of the Managed Metadata column.

In figure 4 the column DemoManagedMetadata is selected and its properties are shown on the right. Since this column is of type ManagedMetadata it has 2 important properties:

- SspId. This is the ID of the Term Store.
- TermSetId. This is the ID of the selected Term Set within the Term Store.

Just compare the IDs in figure 4 with those in figure 1 and 3, they are the same.

One final thing about ID's, the terms within a Term Set have ID's as well. These ID's are stored in the list items of your SharePoint list or library. When migrating content, you will have to take into account that these ID's must refer to the correct terms in the Term Store, which means that the ID's cannot change during the migration.

Let's focus on some migrate scenarios.

Scenario: Migrating the Term Store - one on one

Let's suppose you have to migrate from one SharePoint farm to another. In the destination farm there is no Managed Metadata Service Application yet. Perhaps this is the easiest scenario. First, create a backup of the Managed Metadata Service Application database and restore this database in the destination SharePoint farm's SQL Server. Then, create a new Managed Metadata Service Application and enter the name of the restored database as shown in figure 5.

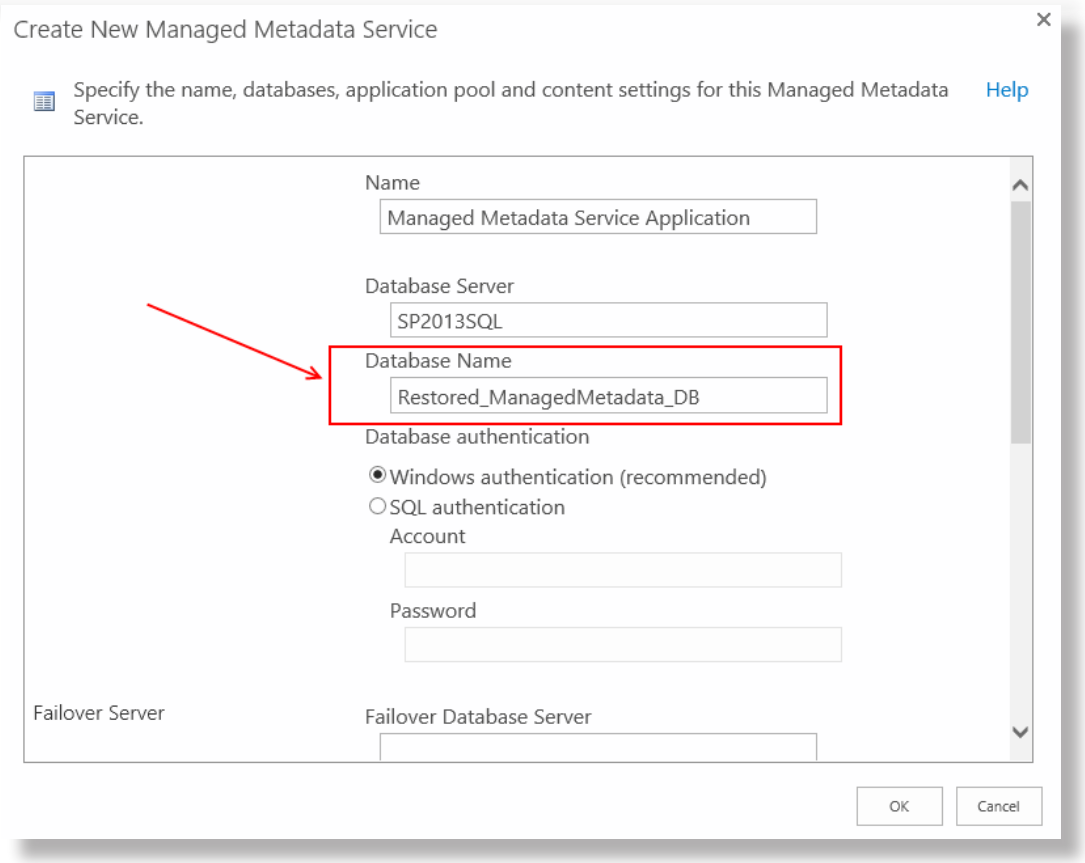

Figure 5: Using an existing MMS database.

After creation, start the Managed Metadata service instance and you're good to go.

In case you already do have a Managed Metadata Service Application, stop the service instance first, then go to the properties of the existing Managed Metadata Service Application (as shown in figure 6) and enter the name of the restored database. Start the service instance again and all should be fine now. Be aware though that the Managed Metadata Service Application can only use one database at a time, so you will now see all the terms from the copied database, but you will no longer see the terms from the original database of the destination environment. This fine if changes are only applied to the Term Store on the source environment. However if changes are applied to the Term Store of the destination environment as well this is probably not the best approach.

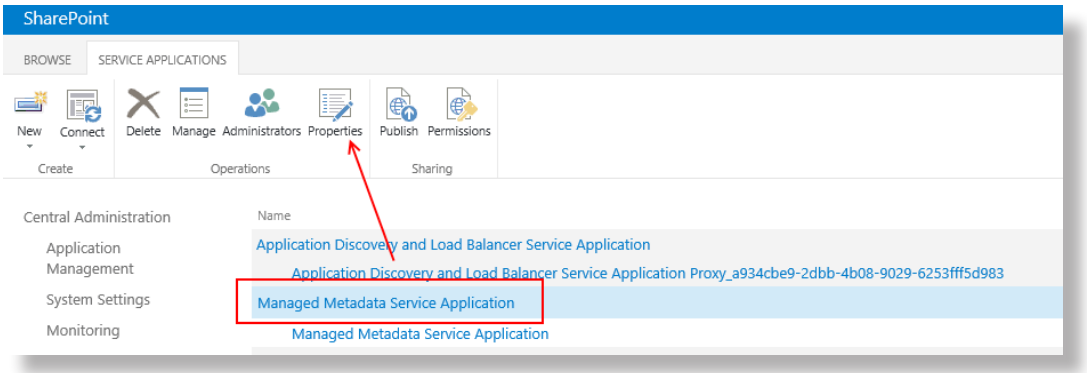

Figure 6: Properties of the Managed Metadata Service Application.

Scenario: Migrating specific Term Sets

What if it is not possible to migrate a complete Term Store as is? Maybe you have several (production) farms each having its own Managed Metadata Service Application with different content (that is, different Term Sets). Or, even more likely to be the case, you already have a MMS SA with some Term Sets and you need to preserve that. At this point Microsoft lacks a bit of tooling to manage your SharePoint Term Store contents. Yes, there is an option to import a Term Set, but it expects an Excel file in a specific .csv format. And, it always creates new ID's. There is no option to use existing ID's. You could search the net for tooling that you are comfortable with. I have tried some, but many of them were based on SharePoint 2010. Then I decided to create a script with PowerShell. My best friend. Period.

At Mavention I work with great minds, great colleagues, and we did some brainstorming to come up with a script. Two scripts even: Export-TermSets.ps1 and Import-TermSets.ps1. I wanted to achieve the following for now:

- Export one Term Set
- Export a group of Term Sets
- Import one or more Term Sets based on the export
- Optionally retain the IDs
- Import should be possible on any SharePoint farm

For the sake of this article I will focus on above requirements. Currently we have scripted lots of other functionality as well, such as Intended Use (navigation), sorting Term Sets and terms, custom properties, local site collection Term Groups, but to cover all of that would require several articles.

Connecting to a Term Store

Whether we need to export or import, connecting to the required Term Store is the first thing that must be done. From a site collection object we can get a TaxonomySession object. This TaxonomySession object allows us to get the required Term Store. Take a look at the code in listing 1 to see how this is done exactly.

```
Write-Host "Creating taxonomy session..."
$taxonomySession = Get-SPSite $SiteUrl | Get-SPTaxonomySession

Write-Host "Accessing Term Store..."
$termStore = $taxonomySession.TermStores[$TermStoreIdentity]
```

Listing 1: Connecting to a Term Store.

The cmdlet Get-SPTaxonomySession takes an SPSite object to retrieve a session. Once you have the session you can browse through all mapped Managed Metadata Term Stores. To get the one you need, just provide the name of the Term Store, for example like shown in figure 1 "Managed Metadata Service Application".

For both the Export and Import script we need to provide two parameters to connect to the Term Store, the URL of the site collection the Identity of the Term Store (which could be the name) like in Listing 2.

```
Export-TermSet.ps1 -SiteUrl http://intranet.mavention.com -TermStoreIdentity
"Managed Metadata Service Application"
```

Listing 2: Executing export script.

Export a Term Set

The Term Store has Term Groups in it and a Term Group has Term Sets in it. In Listing 3 the groups of the store and the sets of a group are retrieved.

```
$group = $termStore.Groups[$GroupName]
$termSet = $group.TermSets[$TermSetName]
```

Listing 3: Accessing Term Groups and Term Sets.

Listing 4 shows how to browse all Term Sets in a group.

```
$group.TermSets | % { Write-Host $_.Name }
```

Listing 4: Displaying all Term Sets in a Term Group.

Every object has the properties ID and Name, which means we can build an XML document as shown in Listing 5.

```
<?xml version="1.0" encoding="UTF-8"?>
<Group Name="Intranet Portal" ID="75413714-6d5f-4748-afeb-d231c7e0ca32">
  <TermSet Name="TopNavigation" ID="205413ce-b0f2-4844-b787-c5d5de890993">
  </TermSet>
</Group
```

Listing 5: Basics of the XML output.

A Term Set has Terms and those Terms can have Terms and those Terms can have Terms etc. In other words, it can get a little bit complex now. The first level of Terms can be retrieved by using the Terms property of the Term Set. Each term also has a Terms property as well. Listing 6 shows some nifty code to recursively browse through all terms.

```
function Get-ChildTerms()
{
  param ( $TermsCollection )

  if ($TermsCollection -and @($TermsCollection).Count -gt 0) {

    $TermsCollection | % {
      $childTerm = $_

      Write-Host "Retrieving term $($childTerm.Name)..."

      Get-ChildTerms $childTerm.Terms
    }

  }
}
```

Listing 6: Recursive function for retrieving terms.

Now that the basics are in place an export XML file can be generated that looks like Listing 7.

```
<?xml version="1.0" encoding="UTF-8"?>
<Group Name="Intranet Portal" ID="75413714-6d5f-4748-afeb-d231c7e0ca32">
  <TermSet Name="TopNavigation" ID="aa598947-6aa7-491d-8568-f3516140d6f4">
    <Terms>
      <Term ID="c589b885-c162-4323-b51f-475052d996af">
        <Name><![CDATA[Home]]></Name>
      </Term>
      <Term ID="b557fcad-f86a-48d2-9386-9336f207c7ea">
        <Name><![CDATA[Departments]]></Name>
        <Terms>
          <Term ID="0d7993fe-c442-4173-9487-1a85eb1eaf0c">
            <Name><![CDATA[HR]]></Name>
          </Term>
          <Term ID="44411138-79b5-41ad-8870-5a922901b3f8">
            <Name><![CDATA[Legal]]></Name>
          </Term>
        </Terms>
      </Term>
    </Terms>
  </TermSet>
</Group>
```

Listing 7: Example final XML output.

Since Term Groups, Term Sets and Terms have many more properties, you can easily access them now and save them as well in the export XML if you need additional information for your specific solution.

Import a Term Set

With the XML file complete it is time to define the import. Now, it is important to have a way to indicate whether you want to retain the ID's. When calling the script this can be achieved by providing a switch like in Listing 8.

```
Import-TermSet.ps1 -InputFile "export _ termset.xml" -RetainIDs
```

Listing 8: Executing Import script.

Listing 9 shows how to define the switch in the script.

```
param (
  [parameter(Mandatory = $true)]
  [string]$SiteUrl,

  [parameter(Mandatory = $true)]
  [string]$TermStoreIdentity,

  [string]$InputFile = "export _ termset.xml",
  [switch]$RetainIDs
)
```

Listing 9: script parameters.

$RetainIDs has the value of True when the switch is provided and False when the switch is omitted from the import statement.

The XML export file can be read using the cmdlet Get-Content and can then be stored in an XmlDocument object like in Listing 10.

```
[xml]$termsXml = Get-Content $InputFile
```

Listing 10: Reading the XML file.

With the XmlDocument object it is very easy to access elements and attributes, like $termsXml.Group.ID and $termsXml.Group.TermSet.Name.

In order to do the actual import the script needs to connect to a Term Store again, just like described earlier in listing 1. When the connection is made, Term Store objects can be created based on the XML input file. Take a look at the next code in Listing 11.

```
[string]$groupName = $TermsXml.Group.Name
[Guid]$groupId = $TermsXml.Group.Id

$group = $termStore.Groups[$groupName]

if ($group) {
  Write-Host "Found the Term Group $($groupName)"
}
else {
  Write-Host "Creating Term Group $($groupName)"

  if ($RetainIDs) {
    $group = $termStore.CreateGroup($groupName, $groupId)
  }
  else {
    $group = $termStore.CreateGroup($groupName)
  }
}
```

Listing 11: Creating a Term Group.

First we try to get a reference to a Term Group. If the reference is valid (not NULL) then the Group exists. Otherwise, the Group does not exist yet and we can create it. Based on the switch whether or not to retain ID's we can create it with or without an ID.

The same logic applies for creating Term Sets and Terms using the methods CreateTermSet and CreateTerm as shown in Listing 12 and 13.

```
[string]$termsetName = $termsetXml.Name
[Guid]$termsetId = $termsetXml.ID

if($RetainIDs) {
  $termSet = $group.CreateTermSet($termsetName, $termsetId, $Locale)
}
else {
  $termSet = $group.CreateTermSet($termsetName, $Locale)
}
```

Listing 12: Creating a Term Set with or without an ID.

```
[string]$termName = $TermXml.Name.InnerText
[Guid]$termId = $TermXml.ID

if($RetainIDs) {
  $term = $parent.CreateTerm($termName, $Locale, $termId)
}
else {
  $term = $parent.CreateTerm($termName, $Locale)
}
```

Listing 13: Creating a Term with or without an ID.

If the XML input file has more Term Sets to import we can simply loop through the XML elements by piping into the PowerShell foreach-object cmdlet as shown in Listing 14.

```
$TermsXml.Group.TermSet | % { $termsetXml = $_ }
```

Listing 14: Looping through all Term Set elements.

And of course you can do the same with the Terms like in Listing 15.

```
$parent.Term | % { $termXml = $_ }
```

Listing 15: Looping through all Term elements.

Although we have imported everything into the Term Store, at this point these are just the objects. We need to commit all changes to the database. This done by using the CommitAll method in Listing 16. If we don't use this method none of the change that we've made are actually applied to the Term Store.

```
$termStore.CommitAll()
```

Listing 16: Saving all changes to the Term Store.

Ow, one more thing…

Yes, we have discussed both Export and Import, and both work like a charm, but frankly we are not quite there yet. Again it's all about ID's. Let me explain. The reason you needed to use the PowerShell variant is because you had to deal with multiple Term Stores. Because of that, we are exporting and importing Term Groups, Term Sets and Terms all with the ability to retain their IDs during import. The Term Store used during the import session is always different, and I mean its ID, than the source during export! And the reason you needed to migrate Term Sets, is probably because your content is migrated or moved. And it is your content that has a reference to a Term Store. Remember the sspId property of the Managed Metadata columns? Right, that is still referencing the source Term Store.

Figure 7 shows a Managed Metadata column before migration.

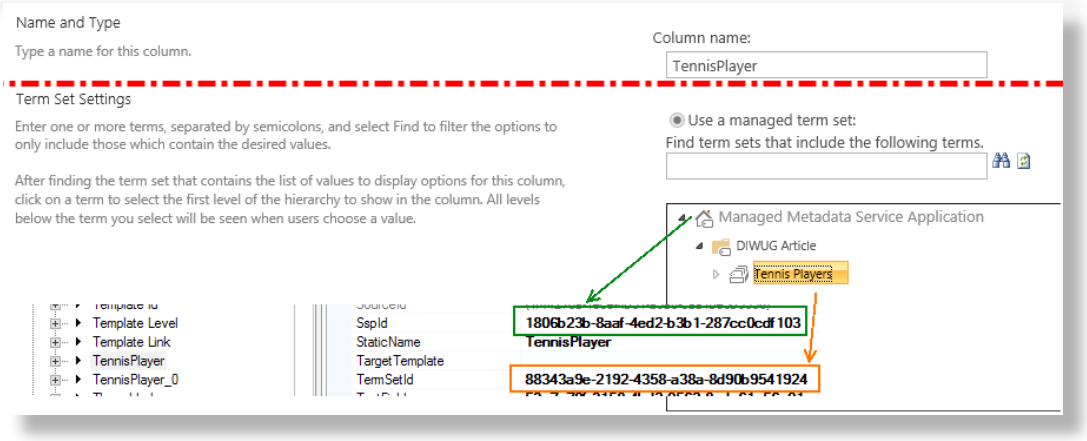

Figure 7: Properties of a Managed Metadata column before migration.

And the same site column after the migration and after the Import-TermSet script has run is shown in Figure 8.

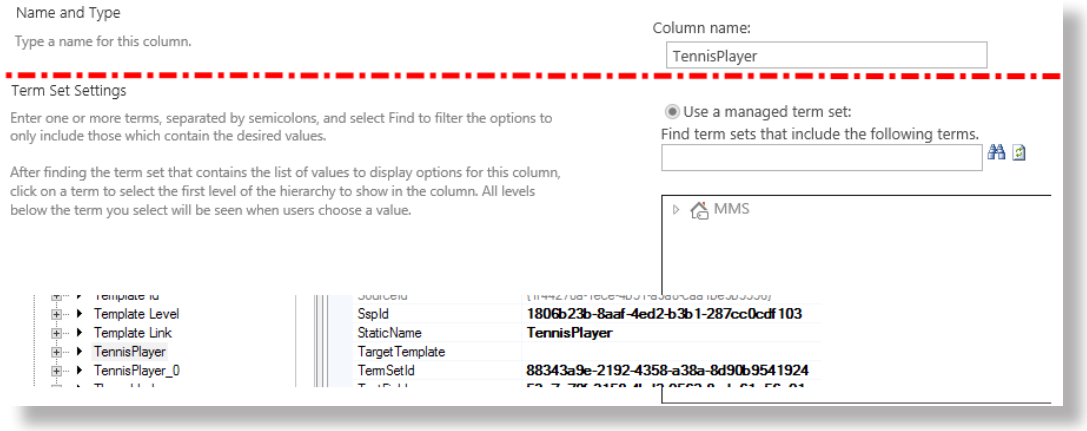

Figure 8: Properties of a Managed Metadata column after migration.

We need to update that sspId property, too. In the next example all Managed Metadata site columns of a site collection are updated. The script needs to connect to the Term Store like described in Listing 1. Then we loop through all Managed Metadata site columns and access the sspId property like in Listing 17.

```
$site.RootWeb.Fields | ? { $_.GetType().Name -eq “TaxonomyField” } | % {

Write-Host $_.Title `( sspID = $_.sspId `)

}
```

Listing 17: Looping through all Managed Metadata site columns.

To update the sspId property just use the code from Listing 18.

```
$taxField.sspId = $termStore.Id
$taxField.Update($true)
```

Listing 18: Updating Managed Metadata column.

With the TennisPlayer example the result is shown in Figure 9.

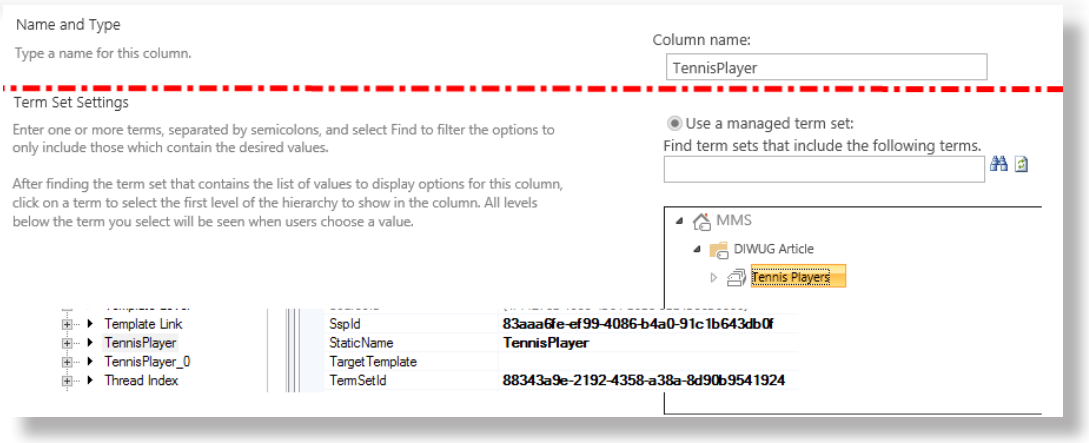

Figure 9: Properties of a Managed Metadata column with updated sspId.

Although this seems straight forward, you need to think about the logic which Managed Metadata columns need to be updated. You can update all columns, like in the example described, or only those that are part of a specific Site Column Group or maybe even something more complex. Take a moment to think about what your specific requirements are and you should be all right.

Summary

Migration scenarios are almost always more complex than they seem to be at first glance. This article focused on migrating Term Sets and what to use in which scenario. Migrating the complete service application database is the easiest option, but using this approach isn't always feasible. Third party tools or handcrafted PowerShell script are often the solution. Many of them lack the option to retain the IDs of the Term Store objects though, which is functionality you really need if you want to retain the links between the content (columns) and the Term Sets and Terms. It is not that difficult to create your own PowerShell script and I have shown how to deal with the ID's and why it is important.

The complete scripts can be found at http://bit.ly/OUfYKX

Implementing Kerberos delegation in SharePoint

by Jasper Siegmund

Kerberos is often perceived as painful to setup and hard to debug. And to be honest: in my opinion it's not one of the most user friendly techniques out there. But once you get the hang of it, the bigger picture starts to become clear and things get easier. In this article I will explain how to use Kerberos in combination with Business Connectivity Services (BCS) to delegate a users' credentials to SQL Server. It is important to note that this will only work with Web Applications that are running in Classic mode, Web Applications that are running in Claims mode cannot use Kerberos for BCS data sources. Web Applications in Claims mode can use Kerberos and the Claims To Windows Token Service (C2WTS) for other scenarios, like Excel Services or Performance Point.

An example case

Let's start with a short case description for some context. Suppose you're contacted by a business user, John, who wants to surface data from a SQL Server database in SharePoint. The data in the database is of sensitive nature. Based on the logged on user, John wants the solution to filter the records so only the appropriate ones are shown. This means there is a need for an additional layer of security, to make sure the correct data is visible for each user.

There are some routes you may consider:

1) Create a web service which queries the database and adds a layer of security to make sure the user only sees the appropriate data. Connect your BCS entities to this web service, instead of directly to SQL.
2) Create views within SQL which use the current logged on user to pre-filter the table and return only relevant data. This is sometimes referred to as view-time security.
3) Use reporting services to create a report which filters data based on the current user.

The real creative thinkers amongst you might think of more ways to do the same, but that's not the point here. The point is that SharePoint is not going to magically filter the data, so you need to make arrangements to make sure that your security requirements are met. And for that, you will need to be able to know which user is requesting the data, no matter which solution you choose.

Kerberos authentication, the basics

As the name implies, Kerberos will help us with authenticating the user. It's important to understand that authentication is not the same as authorization. Authentication will help you to identify the user, making sure their identity is validated. Authorizing is the process of granting a user access to resources, based on that same identity. I will not go into the details of authentication; it suffices to know that Kerberos implements a way of checking your credentials to validate your identity. In Microsoft Active Directory environments, Kerberos or NTLM is used as the authentication method, where Kerberos is the more secure one.

In the mentioned case, our user's identity will be used in several places:

- First, SharePoint will authenticate the user.
- It will then use the user's identity to check if he / she is authorized (there's the difference!) to view the page, the BCS entity, etc.
- Now, assuming our user is authorized, BCS is going to query the SQL database to get the actual data. SQL is going to ask for credentials as well. That is where delegation comes in.
- SharePoint has to delegate the user's credentials to SQL, so that SQL can authenticate the user using those credentials

Why Kerberos?

When you search for the combination of SharePoint and Kerberos online, you will find a lot of people having problems with it (myself included by the way). Based on that, you might ask: why should I use Kerberos at all? Well for starters, your domain admins might have specified it's required. Secondly, NTLM doesn't support double hop authentication. Double hop means your credentials are passed on to a second service / server, like we're doing in this case, you log into SharePoint and SharePoint has to pass you credentials on to SQL Server. If NTLM is used this is not possible. SharePoint then won't be able to send your credentials to SQL Server. Lastly and arguably the most important reason: it's the more secure than NTLM. So, Kerberos it is…

Step 1: SPNs

To be able to use Kerberos, the service using it should be registered in Active Directory. This is done using a service principal name (SPN). To register an SPN, we need some detail on the service which will be using Kerberos as an authentication provider.

- Which service you'll be using. When you initially enabled Kerberos for SharePoint, you used HTTP as the service type. Since we're enabling SQL Server to use Kerberos, we use:

  MSSQLSvc

- The fully qualified domain name (FQDN) of the SQL server:

  sqlserver.contoso.com

- The port your server is using, or in case of a named instance: the named instance itself.

  1433 (default)

  SP (named instance)

- The service account which is used to run the SQL service:

  CONTOSO\sqlserver_svc

We use these parameters as input for the setspn command like in Listing 1.

```
setspn –A MSSQLSvc/sqlserver.contoso.com:1433 CONTOSO\sqlserver _ svc
setspn –A MSSQLSvc/sqlserver.contoso.com:SP CONTOSO\sqlserver _ svc
```

Listing 1: Registering SPNs for SQL.

The setspn command should be executed by a domain admin. When the details are correct, this will register the SPN in AD.

TIP

Ask your DBA for these details. And should he / she not know (or there is no DBA...): Microsoft has a tool which finds out for you. It's called "Microsoft Kerberos Configuration Manager for SQL Server". With this tool, you can easily check if your configuration is OK and it can also generates the script to set the correct SPNs when needed. This way you don't have to worry about getting the syntax right.

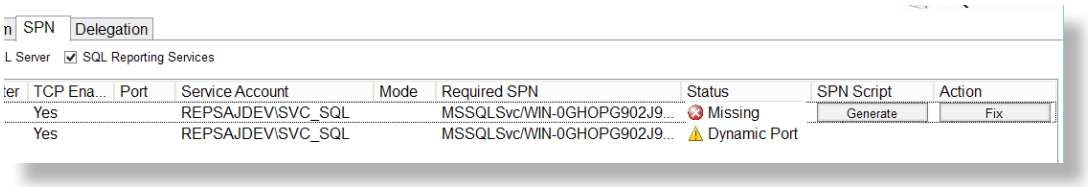

Figure 1: The Kerberos Configuration Manager shows missing SPNs and allows to generate a script for SPN creation.

Note: when you're running SQL 2012 AlwaysOn, you will need to register SPNs for all of the servers in your availability group. You do not have to register SPNs for your listener or aliases. Use the Kerberos Configuration Manager to find out exactly what you need to register.

Step 2: delegation

Setting the SPNs will not yet have enabled the double-hop scenario. We now need to tell Active Directory that our SharePoint application pool accounts are allowed to delegate credentials to the previously added SPNs.

To do so, open up the Active Directory Users and Computers tool and find the account used for the SharePoint application pool. Note that this is the application pool which serves the site in which you want to use the external content type (ECT). When you're going to use it in multiple web applications, you might have to do these steps for multiple accounts as well.

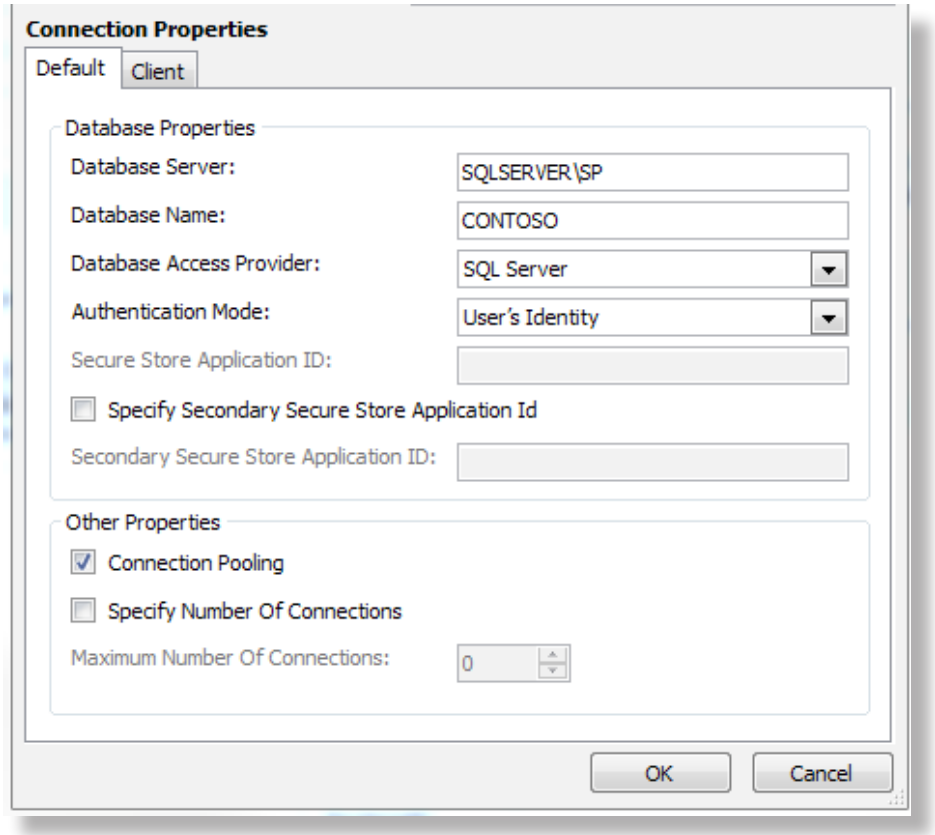

Figure 2: the Delegation tab shows to which SPNs this account is allowed to delegate credentials.

On the Delegation tab (Figure 2), you will find all of the SPNs to which this account is allowed to delegate. Of course you could also select "Trust this user for delegation to any service (Kerberos only)", which would be the easier, but less safe option. To add an SPN, click "Add" and search for your SQL Server service account (NOT the server itself). You will now find all SPNs (servers) registered for that particular service account; select the ones you need to enable delegation for. At this time, Kerberos will now allow delegation of user credentials by the SharePoint application pool, to the SQL server hosting the database.

Step 3: Configuring ECT for delegation

When you create an External Content Type (ECT), the connection properties (figure 1) have an "Authentication Mode" property. Here, we can choose between the following values:

- User's Identity
- BCS Identity
- Impersonate Windows Identity
- Impersonate Custom Identity

The two impersonate options use a credential which is stored in the secure store and identified with a secure store application ID. This will usually be a service account stored in Active Directory, or a SQL account. The "BCS Identity" option uses the application pool account. This option is disabled by default and in most cases you should leave it that way. Reusing such an account for multiple purposes adds security risks and complicates your security scheme.

The first option is the one we're interested in: using the User's Identity. This means BCS will connect to the SQL server by delegating (acting on behalf of) the currently logged on user. Figure 3 shows the BCS connection properties in our demo scenario. This way, we can use this information to filter data.

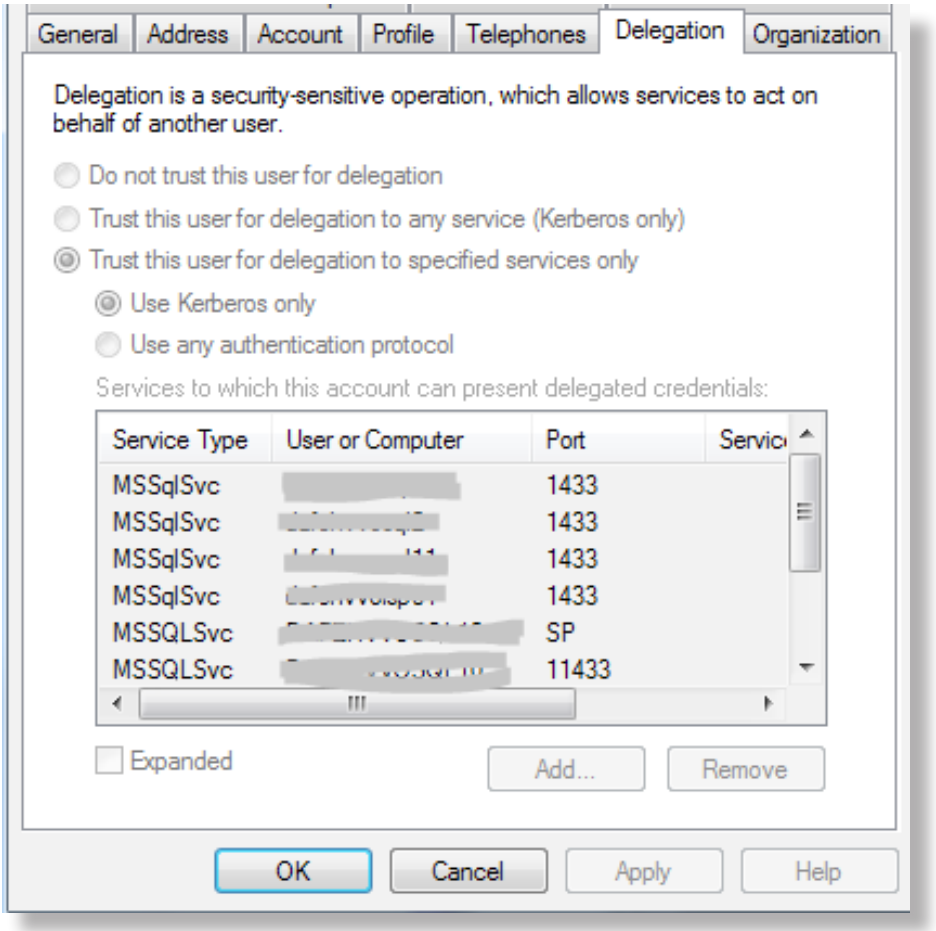

Figure 3: BCS connection properties.

That's it! You should now be able to create a new External List based on your previously created content type. When everything is configured, SharePoint now passes along the identity of the currently logged on user to SQL. You can check if your connections are using Kerberos by using the SQL statement from Listing 2.

```
SELECT auth_scheme FROM sys.dm_exec_connections
```

Listing 2: Show the current user in a SQL query.

The query in Listing 3 shows the CURRENT_USER variable, which you can use to include the current user in your view

```
SELECT CURRENT_USER
```

Listing 3: Show the current user in a SQL query.

Of course, these same steps apply to any service for which you want to enable Kerberos. Make sure you register the correct service type, server name and accounts; and you're good to go!

Used sources / links

Identity delegation for Business Connectivity Services

http://technet.microsoft.com/en-us/library/gg502600(v=office.14).aspx

SharePoint 2010 and Kerberos

http://www.harbar.net/archive/2010/03/31/sharepoint-2010-and-kerberos.aspx

Authenticating to Your External System

http://blogs.msdn.com/b/bcs/archive/2010/03/12/authenticating-to-your-external-system.aspx

Setspn overview

http://technet.microsoft.com/nl-nl/library/cc773257(v=WS.10).aspx

Microsoft® Kerberos Configuration Manager for SQL Server®

http://www.microsoft.com/en-us/download/details.aspx?id=39046

State of SharePoint Statistics

by Nikander Bruggeman and Margriet Bruggeman

SQL Server uses all kinds of statistics about database usage in order to be able to optimize SQL queries. Examples of such statistics are: number of records, density of pages, and histograms. SharePoint is responsible for keeping the statistics of (almost all) of its own databases up to date, which SharePoint leaves up to specific SharePoint timer jobs. As a result, explicit action at the database level can but doesn't need to be taken by administrators in this regard.

Having said that, outdated statistics can have serious performance repercussions, and it's very useful to check periodically if SharePoint database statistics are recent. If they are not, this may indicate a problem with the timer jobs that needs to be addressed, but no direct action should be taken at the database level. This article explains how SharePoint manages SQL statistics and explains how a SQL DMV query can be used to check if statistics are up to date.

Statistics and DMVs

In SharePoint 2010 and 2013 environments, it is recommended to let SharePoint take care of keeping the statistics of various SharePoint databases up to date. SharePoint does this by letting SharePoint timer jobs run nightly and call the following stored procedures:

- proc_DefragmentIndices
- proc_UpdateStatistics
- proc_UpdateStatisticsNVP

Please Note

When SharePoint leverages these SharePoint stored procedures to keep SharePoint database statistics up to date, the database option AUTO_CREATE_STATISTICS should be set to OFF (please refer to http://blogs.msdn.com/b/chunliu/archive/2011/11/17/auto-update-statistics-and-auto-create-statistics-on-or-off-for-sharepoint-2010-databases.aspx for more information). This prevents the two mechanisms from working against each other, both trying to take care of keeping SharePoint database statistics up to date. The results of such a fight are usually predictable: you lose.

Proc_DefragmentIndices

This stored procedure determines the fragmentation percentage of the associated SharePoint database and rebuilds indexes for large tables (row count > 10,000) when the fragmentation percentage is > 30%. When SQL Server Enterprise Edition is used, indexes are rebuilt online; otherwise indexes are rebuilt offline which locks the affected database table. Index fragmentation is the normal result of SQL insert, update, and delete operations and causes suboptimal disk I/O when accessing database tables.

Proc_UpdateStatistics

Updates index statistics. The formula for updating statistics on indexes is:

(sys.sysindexes.rowmodctr * 100) / (sys.sysindexes.rowcnt + 1) > 1

Where:

- rowmodctr counts the total number of inserted, deleted, or updated rows since the last time statistics were updated for the table.
- Rowcnt returns the total number of rows in a table and may also contain uncommitted data. For big tables, reading this column is far more efficient than calculating actual table size by querying current table row count.

The result of this formula is that indexes are updated nightly in most cases, a little less frequently once the database table gets larger (at least 1M+ rows of data). Results > 1 lead to index partitioning. Results of 0 or 1 lead to table partitioning.

Proc_UpdateStatisticsNVP

Updates index statistics or "NameValuePair" database tables, which are only present in SharePoint content databases. Please refer to http://support.microsoft.com/kb/2635071/en-us for more information. "NameValuePair" database tables are used when indexing columns in a SharePoint list (at the software level, not at the database level). As a result of the indexing of a column of a SharePoint list, the NameValuePair_[SQLCollation] table is filled with all values of the columns that are indexed, as well as a reference to the corresponding list item.

Please Note

Every time a list item is modified, the associated NameValuePair row is modified as well. Please refer to http://apmblog.compuware.com/2009/01/28/sharepoint-list-performance-how-list-column-indices-really-work-under-the-hood/ for more information.

The proc_UpdateStatisticsNVP stored procedure is only present on SharePoint content databases; the other stored procedures are present on the following types of SharePoint databases:

- Configuration database
- Content database
- User Profile Service Application Profile database
- User Profile Service Application Social database
- Web Analytics Service Application Reporting database
- Web Analytics Service Application Staging database
- Word Automation Services database
- WSS_Logging database

The stored procedures are run nightly and called by SharePoint timer jobs executed by the following database maintenance Health Analyzer rules:

- Databases used by SharePoint have fragmented indices (Daily)
- Database used by SharePoint have outdated index statistics (Daily)

The Health Analyzer framework is a SharePoint feature (for SharePoint 2010 and up) that enables administrators to schedule built-in jobs that identify problems with SharePoint and potentially automatically fix those problems. The timer job responsible for executing mentioned Health Analyzer rules is the Health Analysis Job (Daily, Microsoft SharePoint Foundation Timer, Any Server). This can be checked by following the next procedure:

1. Start SQL Server Profiler
2. Run the various Health Analyzer rules on demand
3. Watch stored procedures getting executed as the result of each Health Analyzer rule
4. Then, check the Monitoring page in SCA to review timer job definitions until you see one of the stored procedures mentioned in this section
5. Run the timer jobs on demand until SQL Server Profiler output matches the output of the Health Analyzer rule mentioned above. Limit your search to timer jobs that are run Daily with a scope of Any Server.

Please Refer to

http://www.mssqltips.com/sqlservertip/2648/sharepoint-2010-databases-maintenance-health-analyzer-rules/ for more information.

Okay, so that's SharePoint taking care of things as far as SQL Server statistics are concerned. But what about DMVs?

DMV stands for Dynamic Management Views referring to views on internal SQL Server metadata and are an integral part of SQL Server. DMVs have minimal impact on SQL Server system performance and are in that regard far superior over SQL Server Profiler. DMVs provide a great resource that can help diagnosing and monitoring SharePoint performance issues, since most performance issues in SharePoint are related to database interaction anyway.

Internal SQL Server metadata used by DMVs is not persisted, but resides in memory. This means that restarting the SQL Server instance leads to resetting of that metadata. During SQL Server analysis by way of DMVs, the SQL Server instance must not be restarted. If this happens, the analysis needs to be performed anew.

Please Note In times of performance issues, a DMV query can be executed at any time. If DMV queries are executed in the context of a performance test, DMV queries should only be executed after the performance test has been completed.

Listing 1 shows a query to find the state of the statistics for a specific database.

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
SELECT
  ss.name AS SchemaName
  , st.name AS TableName
  , s.name AS IndexName
  , STATS_DATE(s.id,s.indid) AS 'Statistics Last Updated'
  , s.rowcnt AS 'Row Count'
  , s.rowmodctr AS 'Number Of Changes'
  , CAST((CAST(s.rowmodctr AS DECIMAL(28,8))/CAST(s.rowcnt AS
  DECIMAL(28,2)) * 100.0)
  AS DECIMAL(28,2)) AS '% Rows Changed'
FROM sys.sysindexes s
  INNER JOIN sys.tables st ON st.[object_id] = s.[id]
  INNER JOIN sys.schemas ss ON ss.[schema_id] = st.[schema_id]
WHERE s.id > 100
  AND s.indid > 0
  AND s.rowcnt >= 500
ORDER BY SchemaName, TableName, IndexName
```

Listing 1: Query to find the state of statistics for a specific database.

This query lets you find an overview of all statistics for a specific database. The primary source of information for this DMV query is sys.indexes, containing information about indexes, row counts, number of rows that have changed since statistics were last updated, etc.

Please Note

The query only shows statistics for indexes that have more than 500 rows, because statistics are only relevant for larger indexes.

Figure 1 shows an example result of the State of Statistics query.

Results | Messages

	SchemaName	TableName	IndexName	Statistics Last Updated	Row Count	Number Of Changes	% Rows Changed
1	dbo	Objects	IX_Objects_ClassId_ParentId_Name	2014-02-10 00:01:10.170	1630	0	0.00
2	dbo	Objects	IX_Objects_Version	2014-02-10 00:01:10.123	1630	848	52.02
3	dbo	Objects	PK_Objects	2014-02-10 00:01:10.130	1630	0	0.00
4	dbo	TimerJobHistory	IX_TimerJobHistory_Id	2014-02-10 00:01:10.567	539123	29626	5.50
5	dbo	TimerJobHistory	IX_TimerJobHistory_JobId_Id	2014-02-10 00:01:11.737	539123	29624	5.49
6	dbo	TimerJobHistory	IX_TimerJobHistory_ServerId_Id	2014-02-10 00:01:12.357	539123	29624	5.49
7	dbo	TimerJobHistory	IX_TimerJobHistory_ServiceId_Id	2014-02-10 00:01:13.460	539123	29623	5.49
8	dbo	TimerJobHistory	IX_TimerJobHistory_Status_Id	2014-02-10 00:01:14.810	539123	29623	5.49
9	dbo	TimerJobHistory	IX_TimerJobHistory_WebApplicationId_Id	2014-02-10 00:01:14.423	539123	29623	5.49
10	dbo	TimerScheduledJobs	IX_TimerScheduledJobs_JobId_ServerId	2014-02-10 00:01:14.817	553	570	103.07
11	dbo	Tombstones	IX_Tombstones_Version	2014-01-23 00:01:34.220	8553	39	0.46
12	dbo	Tombstones	PK_Tombstones	2014-01-23 00:01:34.273	8553	39	0.46

Figure 1: Result of the State of Statistics query

The DMV query returns the following information:

- SchemaName, the name of the database schema. A schema provides a way to logically group database objects, such as tables, views, and stored procedures. The schema name "dbo" is the name of the default schema in SQL Server.
- TableName, the name of the database table.
- IndexName, the name of the table index.
- Statistics Last Updated, the date and time when index statistics were updated for the last time.
- Row Count, the number of rows in the table.
- Number of Changes, the number of changes that have taken place since the last statistics update.
- % Rows Changed, percentage of changes related to total number of rows in the database table.

A quick glance at the Statistics Last Updated column will tell you if statistics are up to date or if further investigation is warranted.

Conclusion

When using the discussed SQL DMV query for statistics, you should look out for a couple of things:

- Outdated statistics. Take a look at the date and time when the statistics were updated for the last time. Usually, these statistics are updated nightly. Older statistics indicate problems with SharePoint timer jobs responsible for maintaining statistics.
- Inspect the Number of Changes and % Rows Changed. If huge amounts of changes take place in SharePoint databases, it may be necessary to execute SharePoint statistics timer jobs more frequently either manually or by adjusting their schedules. Please Note It's extremely rare that this is necessary.

How to organize meetings in SharePoint 2013

by Frank op 't Landt

How can we organize meetings in SharePoint 2013? Should we use an App? Or custom lists? Microsoft has deprecated the Meeting Workspace template in SharePoint 2013 and suggests alternatives such as OneNote and Lync. How can we use these alternatives for our meetings? In this article I describe some of the scenarios, including screenshots. I will start describing the suggested alternative by Microsoft, the SharePoint App from AvePoint and finally a custom alternative.

Lync/OneNote/SharePoint

This solution has the following requirements:

- SharePoint 2013 Team site
- Agenda App
- Outlook 2013 client
- OneNote client

I will describe step by step how to create an appointment and store your notes.

Creating the appointment

We create the appointment in the SharePoint Calendar App by using the Outlook client. In this way the meeting is available in every attendee's calendar and on the SharePoint Team site calendar. First we have to connect the Calendar App with Outlook by clicking on the ribbon and Connect to Outlook button as shown in Figure 1.

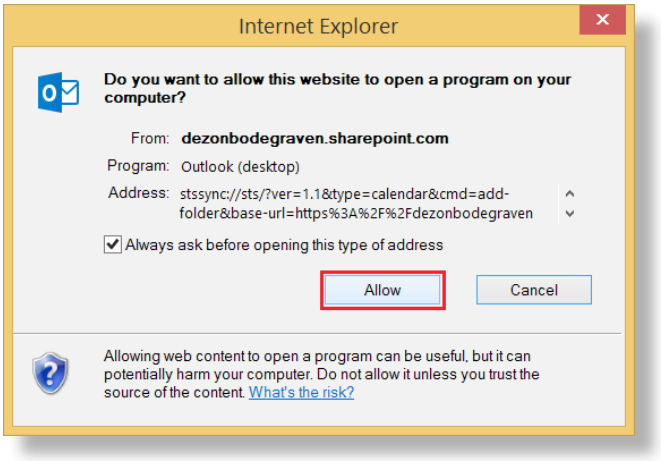

Figure 1: Shows Oulook Calendar.

Click on Allow in the pop-up as shown in Figure 2.

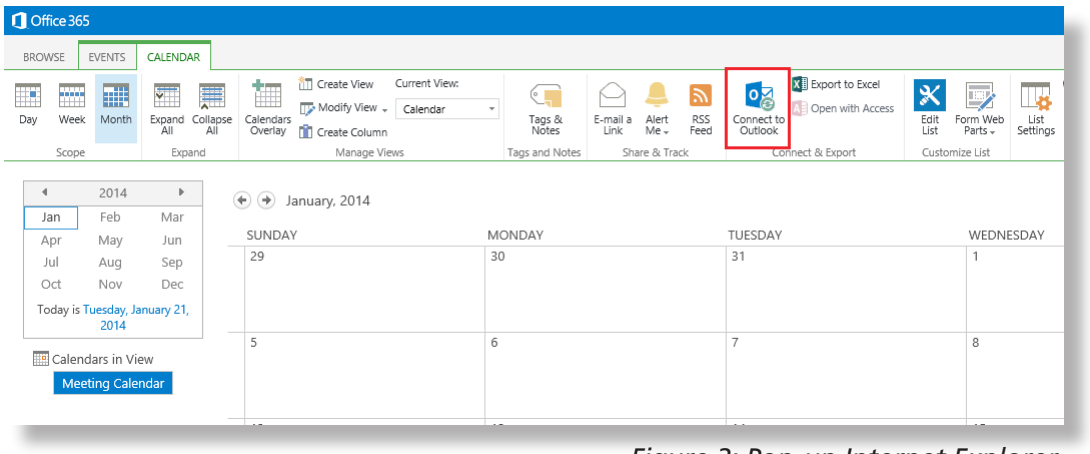

Figure 2: Pop-up Internet Explorer.

Click on Yes as shown in Figure 3.

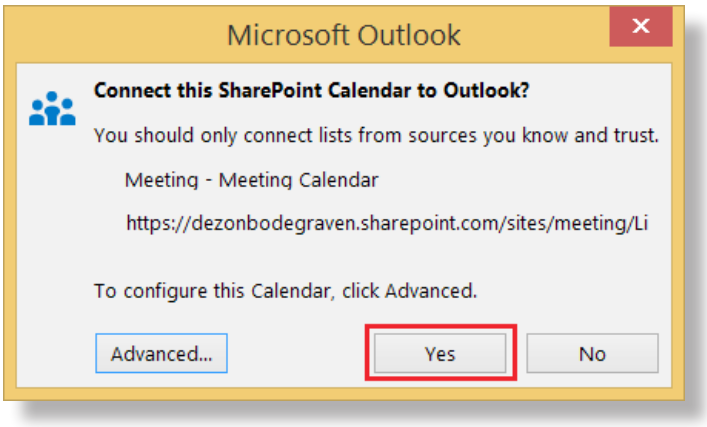

Figure 3: Connect to Outlook.

Right-click in the calendar of your Team Site, within Outlook, at the meeting time and date. Click on New Appointment as shown in Figure 4.

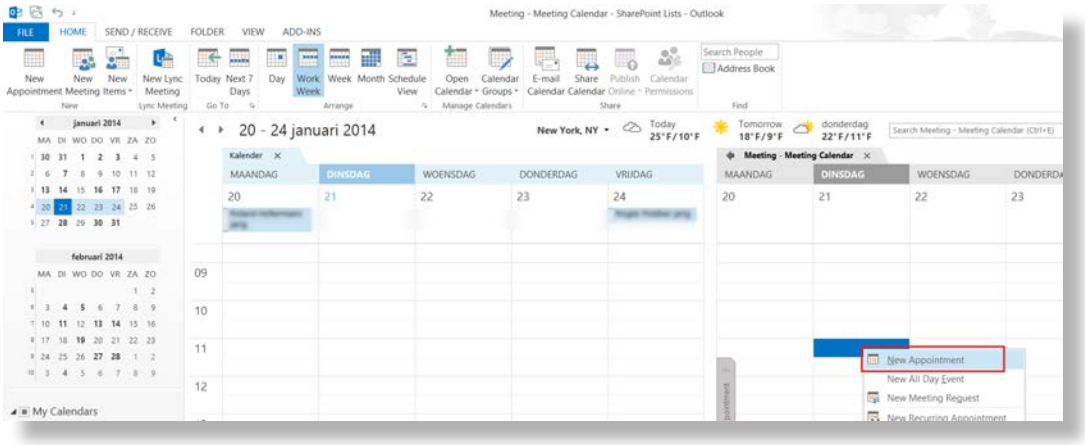

Figure 4: Add new appointment in SharePoint calendar.

Click on Lync Meeting if you want to make this an online meeting accessible for all the attendees by using Lync.

Enter the attendees in the To field. Because you are using the To field, the Appointment is located in the personal calendars of your colleagues and the calendar in your SharePoint Team Site. Don't forget to send the invitation to yourself. If you don't send it to yourself the meeting doesn't appear in your personal calendar, because you sent it from your SharPoint Team site.

The appointment is now available in my personal calendar and the SharePoint Team Site calendar.

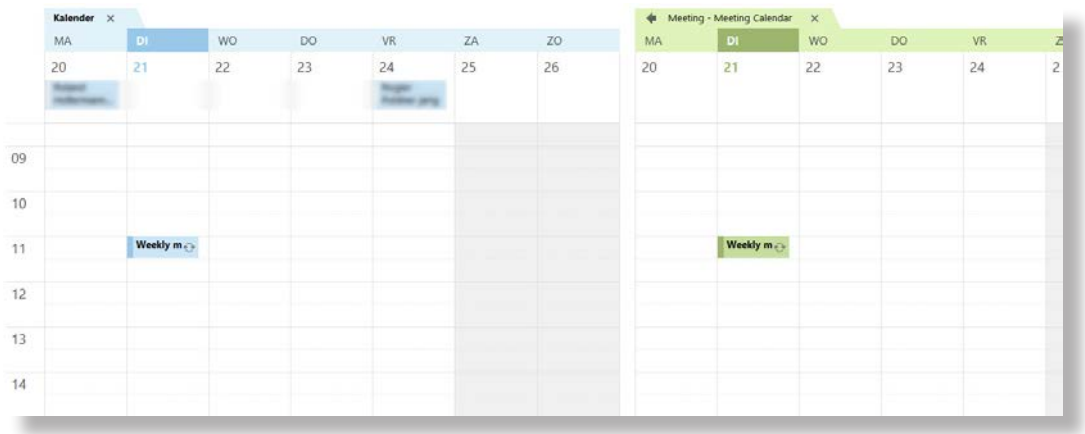

Figure 5: Shows appointment in both calendars.

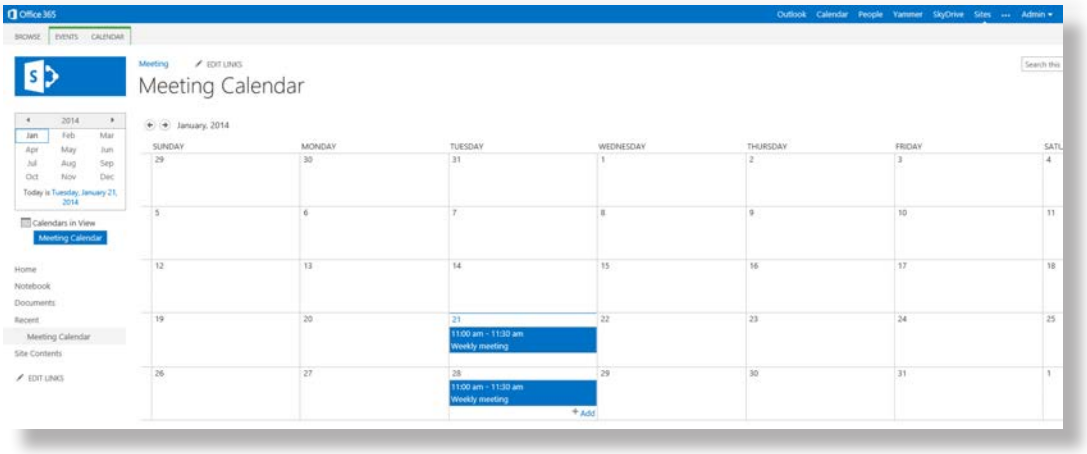

Figure 6: Shows calendar in SharePoint.

Meeting Notes

Now we're going to share the meeting notes by using OneNote. Click on Meeting Notes and choose Share notes with the meeting series as shown in Figure 7.

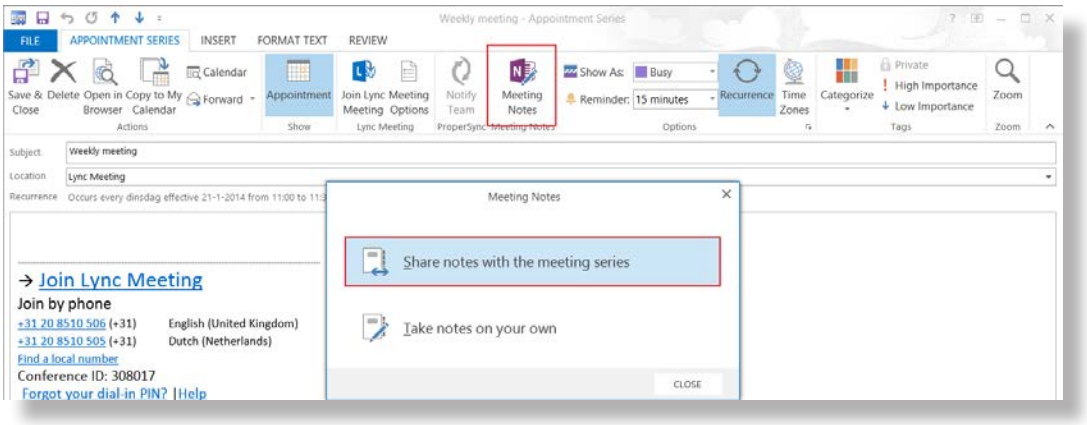

Figure 7: Create notes in Outlook.

Choose New Notebook as shown in Figure 8 and the location of the SharePoint Team Site Document library. Enter the filename for the notebook.

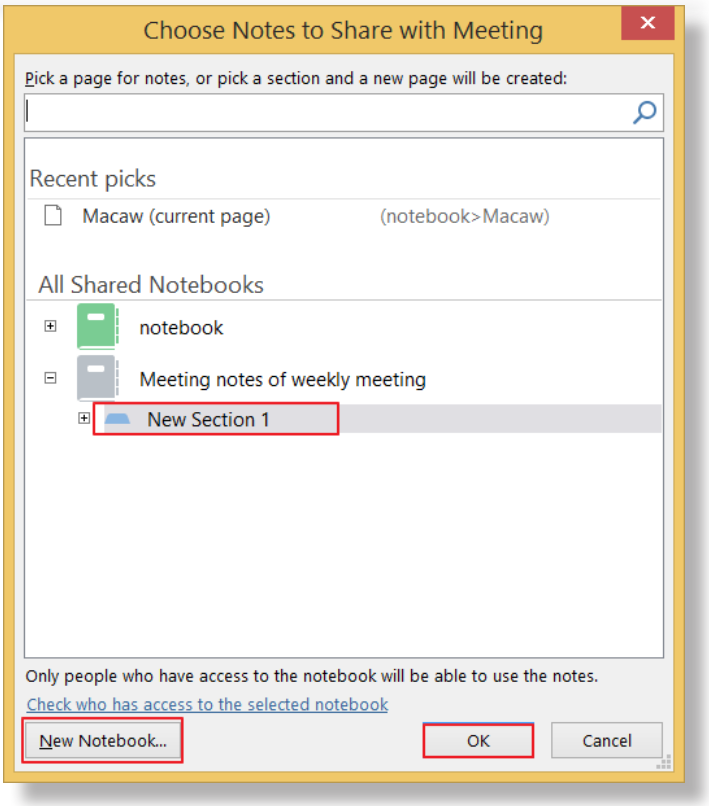

Figure 8: Choose Notes to Share or create a new notebook.

You are asked to invite people, use this to share the notes with your colleagues. Close OneNote and go back to Outlook.

Save and send the appointment to attendees. Your colleagues can access the notes by using Outlook, OneNote or go to the SharePoint Team Site.

Try to find a good way to sort your meeting notes in OneNote. For example you can create pages for every meeting and subpages for every topic. The example is shown in figure 9.

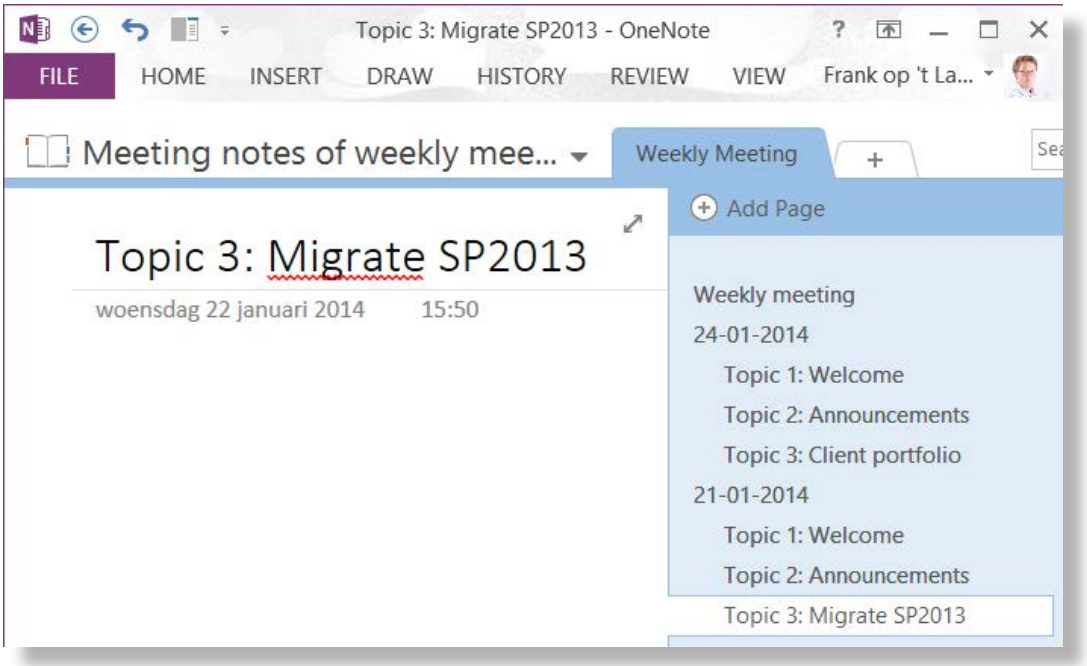

Figure 9: Example of meeting notes in OneNote.

N.B.:

Changing the appointment afterwards is still possible, but it will be changed only on the SharePoint Team Site and you cannot forward these changes to the attendees.

To prevent this problem, there is a third party tool that can synchronize your SharePoint Team Site calendar with your Outlook calendar.

This tool called ProperSync (http://www.propersync.com/) and works very well.

AvePoint Meetings App

I am not going to describe how to use the App step by step, because this App works very well out of the box and is user friendly.

When we open the App we see an overview of all the meetings and the option to create new meetings as shown in Figure 10.

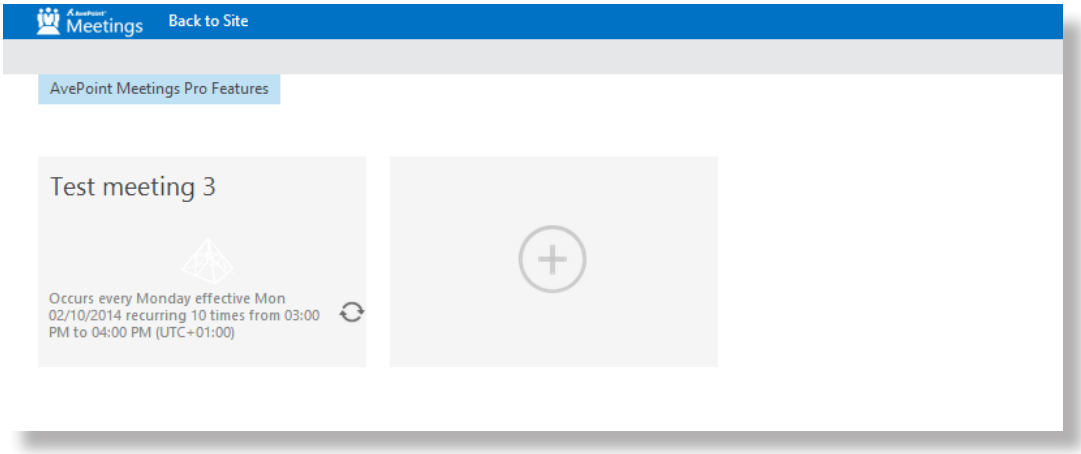

Figure 10: Start page AvePoint Meetings app.

In the meeting you can create agenda items, add actions, decisions, notes and attachments. This enables you to organize the most important elements of a meeting. Figure 11 shows the app's meeting details page.

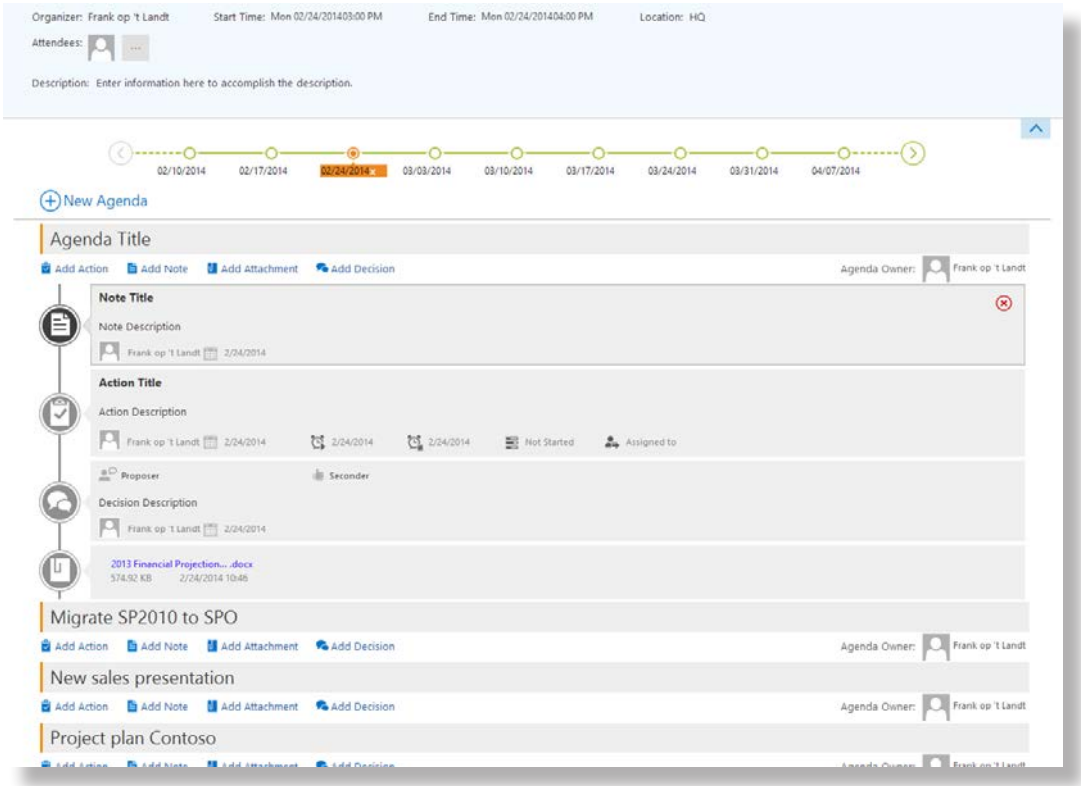

Figure 11: detail page of a meeting.

All the data stored in the App, such as attachments or calendar items, are actually stored in standard SharePoint lists. That's very cool! You are now able to get the data out of the App and do with it that you want. For example, create a workflow on an assigned agenda item. An overview of the lists with meeting data is shown in figure 12.

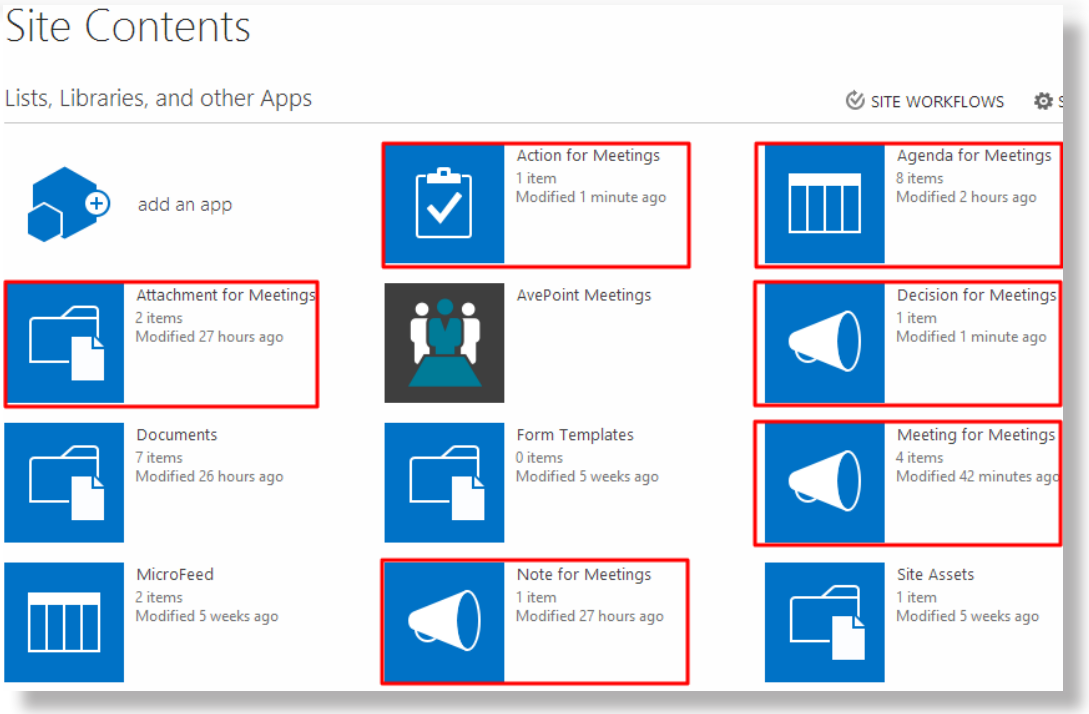

Figure 12: Site Contents with overview of all lists for the meetings app.

There are a couple issues I discovered while using this App.

The first issue is the 'You deserve more…' box. Every time I start the App it shows up, even when I checked the 'Don't show me this again' checkbox. As shown in Figure 13.

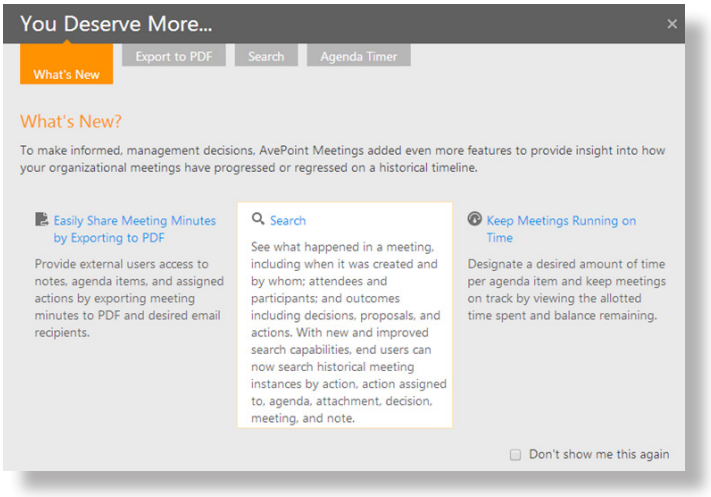

Figure 13: "you deserve more" pop-up.

The second issue is that external users have to register when they want to use the App. When they open the app they see a register form, I don't know what the purpose is of registering external users. This is a barrier for these users, because it isn't user friendly. We don't want that, especially if they aren't from our organization and we can't force them to use our tools.

The last issue is that the App sometimes breaks. This means sometimes one or more meetings become accessible and in other cases the app even crashes completely, to the point where it has to be removed and re-added (the version that was used was the free version of 2.4.4.0).

The figure 14 shows an example of an error.

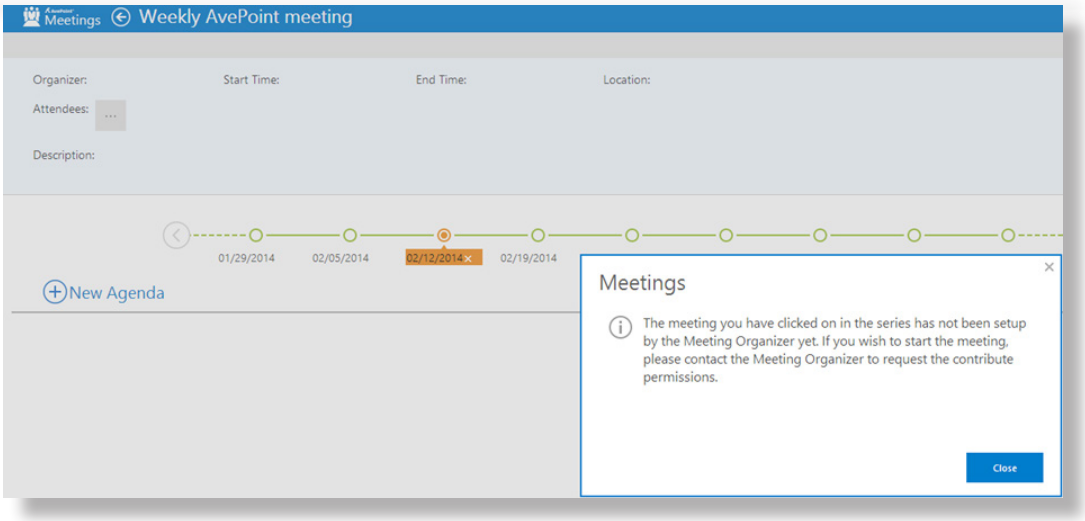

Figure 14: Error message of a broken meeting.

There is also something to keep in mind about SharePoint Apps in general. Apps, such as this meeting App, are hosted outside of SharePoint. This is really good on one side, because it means your SharePoint environment will be more stable and because you don't have to upgrade all your customizations every time you upgrade SharePoint, but it also has some downsides. Depending on how the App is surfaced in SharePoint it might either show up in an "App Part" shown on your SharePoint site, or it might be a completely different page. If the App is shown on a separate page this can be confusing for the business users because when they open the App, the top navigation bar is gone. This is something to keep in mind and it will help users to explain to them how to navigate between Apps and your SharePoint team site.

Figure 15 shows a view of the team site and Figure 16 shows a view of the full page App.

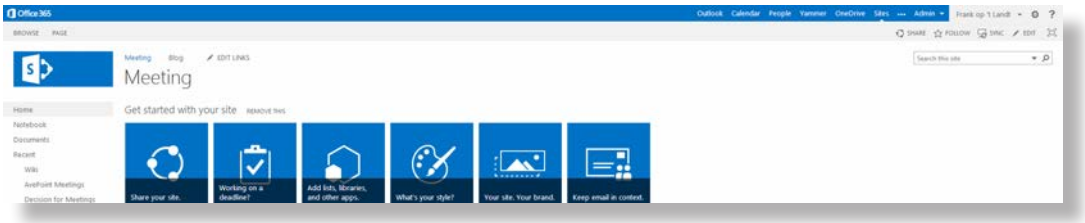

Figure 15: default SharePoint Team site homepage.

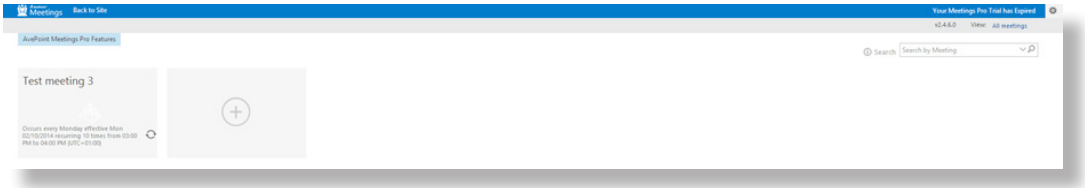

Figure 16: Start page AvePoint Meetings app.

Have you ever visited a DIWUG event (in Dutch)?
Just register on the DIWUG site to receive the monthly newsletter.

Create your own alternative

Recently I created a custom alternative for Meeting Workspaces. I would like to give you the step by step description.

Create lists and views

A Meeting Workspace in SharePoint 2010 has the following Apps: agenda items, decisions, calendar, documents and tasks. We start by creating these Apps. Let's start with the calendar App:

- Create the default calendar;
- Create a new view called FutureItems;
- Uncheck recurrence and workspace at Columns;
- Create the filter Start Time is greater than or equal to [Today];
- Create another view called PastItems
- Uncheck recurrence and workspace and Sort by Start Time and choose for descending order;
- Create the filter 'Start Time is less than [Today]'.

To synchronize your team calendar with your personal calendar follow the instructions from the first solution at the beginning of this article

The next step is creating the Agenda App:

- Add a custom list App and call it Agenda items;
- Go to List settings;
- Change the name of the Title column in Subject;
- Create the columns Time (single line of text), Comments (multiple lines of text) and Owner (person or group);
- Now click on Add from existing site columns and Start Date;
- Change the name of this column in Meeting Date, set it at required and set defaults to None.

Now create the Decisions App:

- Add a custom list App and call it Decisions;
- Go to List settings;
- Create the columns Description (multiple lines of text) and Owner (person or group);
- Now click on Add from existing site columns and Start Date;
- Change the name of this column in Meeting Date, set it at required and defaults to None.

The Documents App probably already exists:

- Go to Library settings and click on Add from existing site columns and Start Date;
- Change the name of this column in Meeting Date, set it at required and default values to None.

Finally the Tasks App:

- Add a custom list App and call it Tasks;
- Go to List Settings;
- Change the name of the Title column in Task name;
- Create the columns Start date (date and time), Due date (date and time), Assigned to (person or group), Description (multiple lines of text), Status (choice) and Priority (choice);
- Now click on Add from existing site columns and Start Date;
- Change the name of this column in Meeting Date, set it at required and set defaults to None.

Creating the dashboards

The first dashboard we create is on the home page. For the dashboard, I choose a page layout with three columns. Add in the first column I add the Calendar web part. In the second column I add the Agenda items and Decisions web parts. In the last column I place the web parts for Documents and Tasks.

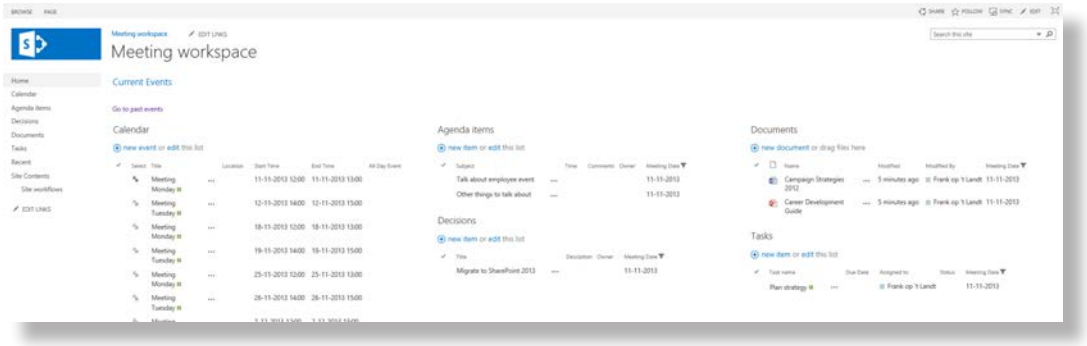

Figure 17: Dashboard.

Edit the Calendar web part:

- Change the view in FutureItems;
- Click on Send Row of Data To and then Agenda items as shown in Figure 18.

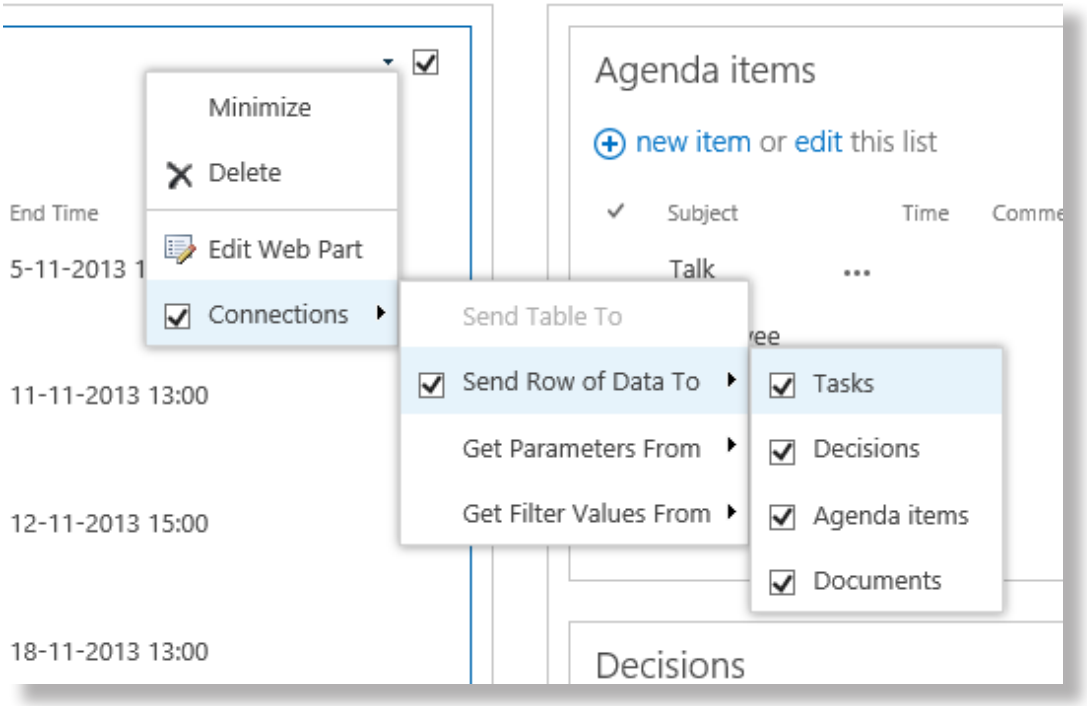

Figure 18: Connect web part.

- Select for Get Filter Values From, and then click on Configure. As shown in Figure 19

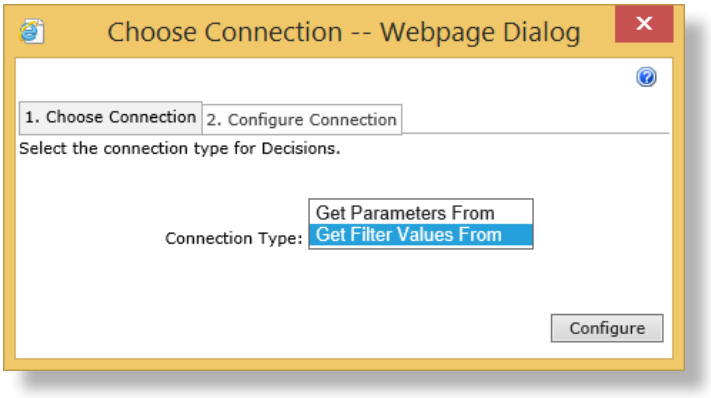

Figure 19: Connection type.

- Click on the Provider Field Name for start time and consumer field name at for meeting date. As shown in Figure 20.

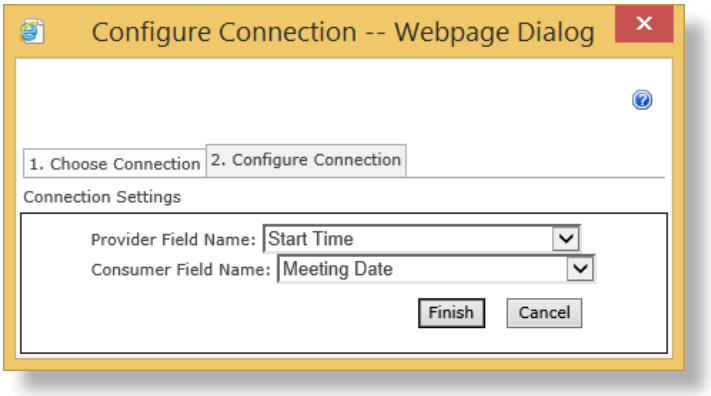

Figure 20: Connection settings.

- Click on Finish.

Repeat these steps for decisions, documents and tasks.

The next dashboard revolves around the items from the past. Repeat the steps from the previous instruction. The only difference is the view of the calendar web part. Set up the view on PastItems. Link from the first dashboard to the second and vice versa.

Make the workflow

The Calendar App in SharePoint offers the possibility for recurring items. One item is created and shown several times in the App or web part. There is a problem with this solution, because connecting the web parts doesn't work with recurring events. There is only one item and therefore only one start time of the event. Filtering out the other dates of the event won't work.

To solve this problem, we create a workflow which create recurring events for us. In this example I create a workflow in SharePoint Designer 2013 that creates the same weekly event at the same time.

Create the variable Counter with as type Number. Then create the following Form parameters:

- Quantity: Type Number
- Title: Type Single line of text
- Start date and time: Type Date and Time. Uncheck Allow blank values and set Display format as Date and time
- End date and time: type Date and Time. Uncheck Allow blank values and set Display format as Date and time

Copy the workflow settings as shown in Figure 21.

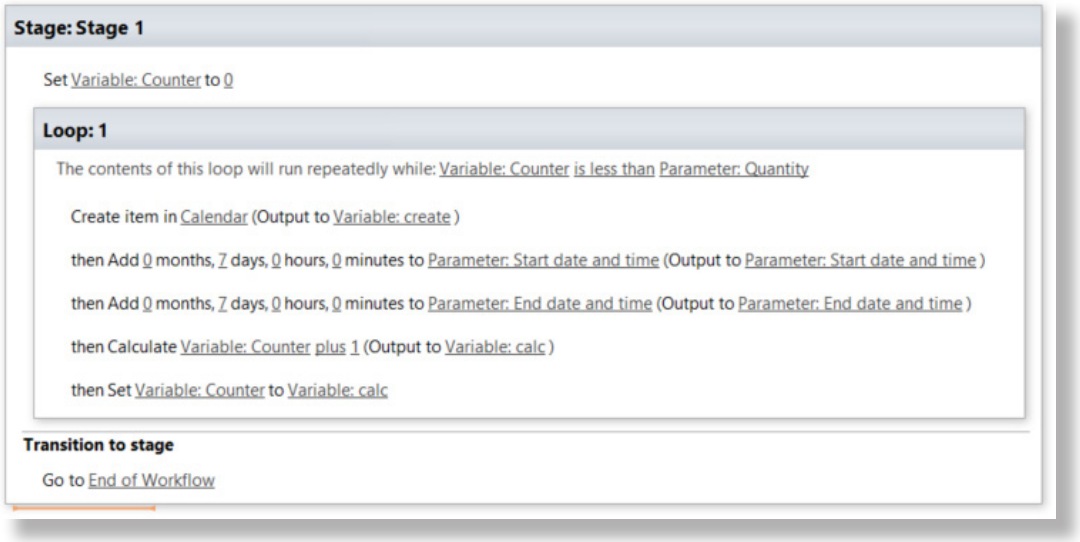

Figure 21: Workflow.

This workflow, created using SharePoint Designer 2013, has one disadvantage. When you create multiple items and you create some of them in daylight-savings time that can cause an issue where the incorrect time is used for the meeting.

When you use a SharePoint On-Premises environment with Nintex workflows you can solve this problem. You can use the 'Calculate date' action and specify that the time must be stored in 'local time' instead of UTC. This option isn't available in Nintex Workflow for SharePoint Online (yet).

Conclusion

The three scenarios described all provide solutions to organize your meetings in SharePoint. Each solution has its own pros and cons. You will have to choose the best solution for your specific requirements.

The first scenario I describe require a huge change in the way the end users work, but on the other side it uses tools the users already know.

The second scenario uses a third party tool that could use some improvement, but really is an opportunity.

The last scenario requires quite a lot of configuration. The benefit of this scenario is that it gives you the possibility to fully customize the solution to your requirements.

MCMS2002 Migration to a SP 2013 metadata driven environment

by Vincent Verbeek

As any SharePoint enthusiast will acknowledge, migrating an existing website to a SharePoint environment is no easy task. This task becomes increasingly difficult when the website is a non-SharePoint website containing 100,000+ pages and documents. Tagging and organizing content in such a way that users will still be able to find it, either through search or through site navigation, can be very difficult – especially when the source content is unstructured. And finally – how can you ensure that all existing URLs are properly converted to their new ones while maintaining the search ranking in the major search providers?

In this article I'll show you, as well as provide you with examples on, how we managed to overcome these challenges.

Case explanation

Our customer had a website based on Microsoft Content Management System 2002 SP1, which had been in place for over ten years. In those ten years, the website had been modified to meet all customer requirements. It worked like a charm. But, because MCMS and SQL 2000 are no longer supported by Microsoft, the customer had to move to a new platform. Since publishing content through their website is one of the customers core businesses, they needed a robust platform that had extensive publishing capabilities. They chose SharePoint 2013.

Along with the platform change, the new website also has a very different way of presenting content to the visitor. Instead of browsing articles though a hierarchic structure, we went for a search driven site through the use of metadata. This meant that the use of metadata would be of vital importance. Because without it, how could you find an article?

This presented us with a major challenge. More than ten years of publishing content meant that over 70,000 pages and 30,000 documents had to be migrated to SharePoint 2013 and had to be provided with additional metadata. Our goal was after all to make the content meaningful, relevant and, most of all, easily searchable.

Underneath you will read in detail how we approached this challenge.

Although there are several existing migration tools that are able to migrate MCMS2002 content into SharePoint, all of these tools had limitations due to the heavily customized website. Because consistency and accuracy of the migrated data was so important for the customer, we decided to build our own migration tooling.

Exporting the existing content

The first step towards migrating the website was to export the existing content. Because we had to deal with different sorts of content (website pages, documents and images), we decided on a generic approach that would work for all file types. In order to meet this challenge, we created a custom tool that exported all content into an XML based file, so there would be one XML file per URL. This XML file contained all the information we needed to be able to create new content into our SharePoint 2013 environment. All webpages and documents were ordered in a logical folder based structure, so we could import based on the year of publication. Webpages published on May 2006 would be converted into an XML file with a folder structure like this: {export location}\{language}\2006\05\{articlename}.xml . Although the same principle was applied to documents and images, they were located in an extra subfolder named Documents or Images.

The XML files created were the only source of input we had for creating the new pages and their properties. That is why we had to make sure that these files contained as much information as possible. Along with the basic properties, such as publication date and title, we also added a keywords property, which was filled with the most important keywords on the page. As well as the keywords, the theme property was also of great importance for the new website. This theme property could be derived from the URL of the existing content. A small snippet of the final XML is displayed in Listing 1.

```
<?xml version="1.0"?>
<Root>
  <Properties Guid="{357C9063-17D4-425C-BE40-6BD69FBFA9D0}"
    Url="/en-gb/menu/theme/prices/publications/sampleFile.htm" />
  <CustomProperties PublicationType="Webmagazine" Theme="Prices"
    Keywords="Economic,inflation" />
  <HtmlTitle><![CDATA[Why inflation is so high]]>
  </HtmlTitle>
  ...
</Root>
```

Listing 1: Exported XML snippet.

Executing the export tool using the settings and configuration described earlier, resulted in a total of ~105,000 XML files. These files could then be used by the content creation tool for creating new content in our SharePoint 2013 environment.

Creating new content in SharePoint

The next step in the process was to create new content in our SharePoint 2013 environment using the exported XML files. For this process we created custom tooling that used a dynamic approach, with which our customer has the ability to easily control the output generated by the content creation tool. In order to be able to achieve this dynamic behavior, we combined two XML files that form the heart of the content creation tool.

The first XML file describes a mapping that, based on the URL property of the export, determines which contenttype is used for creating the new content, as well as which tag is added for search optimization. Listing 2 contains a snippet of this XML file.

```
...
<record>
  <PartOfUrl>menu/methods/publication-analysis</PartOfUrl>
  <ContentType>Methods Publication and Analysis</ContentType>
  <SearchTag>Validated Methods</SearchTag>
</record>
...
```

Listing 2: UrlToContentTypeMapping.xml snippet.

The second XML file describes which property in the exported XML file is mapped to which SharePoint Field. Listing 3 contains a snippet of this XML file.

```
...
<record>
  <FieldName>ISBN</FieldName>
  <MappedProperty Name="Publication" Attribute="ISBN"></MappedProperty>
</record>
<record>
  <FieldName>Publicationperiod</FieldName>
  <MappedProperty Name="period" Attribute=""></MappedProperty>
</record>
...
```

Listing 3: FieldToXmlPropertyMapping.xml snippet.

The flow of the content creation tool, which is executed for every input XML file, is displayed in Figure 1.

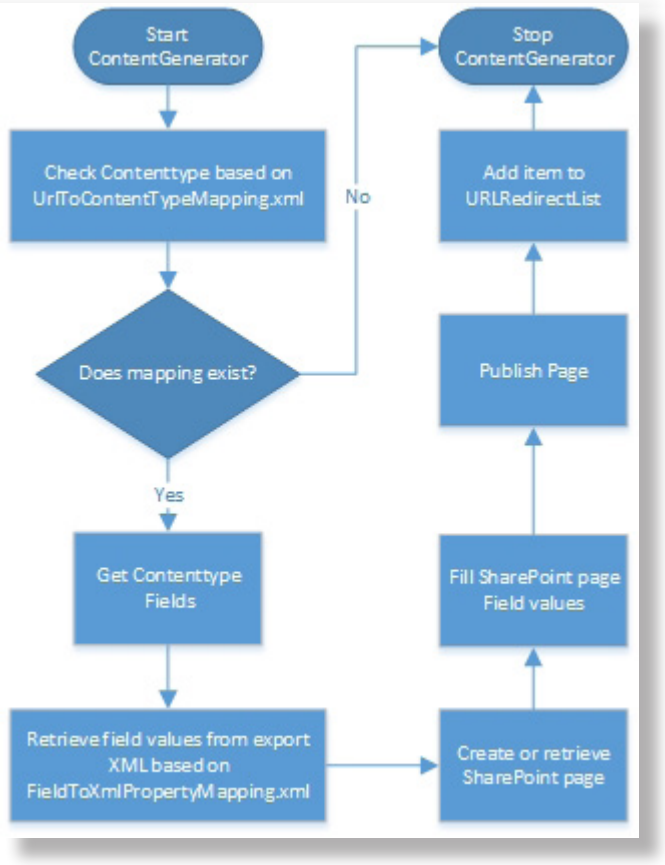

Figure 1: Content creation flow, per exported item.

First, the contenttype is chosen based on the mapping file shown in Listing 2. Each contenttype has its own collection of fields that need to be supplied with information. For each field in this collection, we retrieved the appropriate value, based on the mapping shown in Listing 3. To read the data from the XML mapping files, we created a generic method which is displayed in Listing 4. After all field values were retrieved, a new page was created based on the mapped contenttype, and the field values were set to their appropriate values. After that, the page is saved and published. Finally, an item was added to the URLRedirectList. This list can then be used to perform user redirection as described in the next paragraph.

The content generator processes all XML files that are located at a user specified path. This means that multiple instances of the content generator can be started at the same time, each with a different starting path. In our case, we started a different instance for each year-folder of the export. This dramatically decreased the duration of a complete content import from two days to approximately 3 hours.

SharePoint Search Tips

The search box in the SharePoint user interface is a lot more powerful than you might expect!

To search for all people whose last name is Smith use: **LastName:Smith**

To search for all people whose first name starts with S: **FirstName:S***

To find out what communities a user belongs to:
Smith ContentType="Community Member"

```
internal string GetXMLPropertyValue(XElement xElement, string
  propertyName, string attributeName)
{
  string mappedPropertyValue = string.Empty;
  if (string.IsNullOrWhiteSpace(attributeName))
  {
    var selectedNode = xElement.DescendantsAndSelf(propertyName).Select(
      node => node.Value).ToArray();
    if (selectedNode.Length != 0)
    {
      mappedPropertyValue = selectedNode[0].ToString();
    }
  }
  else
  {
    var selectedNode = xElement.DescendantsAndSelf(
      propertyName).Attributes(attributeName).Select(
      node => node.Value).ToArray();
    if (selectedNode.Length != 0)
    {
      foreach (var selected in selectedNode)
        mappedPropertyValue = selected.ToString();
    }
  }
  return mappedPropertyValue.Trim();
}
```

Listing 4: Generic function for reading XML values.

Search provider friendly URL redirection

Migrating the website into SharePoint 2013 meant that all existing URL's were invalid, since SharePoint stores pages and documents in a completely different directory. This would be easy enough to change in the import tooling for all internal links, however in this case we also had to consider the hundreds of thousands of external links to the website that have been created by various sites over the internet. We also had to make sure that all of these links would still be valid! And we wanted to perform the redirection to the new pages in such a way that search providers would maintain the page rating.

To tackle this problem, we developed a simple, yet very powerful, solution. We created a SharePoint list, containing two columns namely the old and new URL. As discussed earlier, each time a new page or document was created during the execution of the content creation tool, a new list item was added to this list. After the content creation was completed, we had a full list of URL's to which we want our users to be redirected to when they visit an old URL.

In order to make sure that we could intercept a request before the user is returned a 404 error code, we had to create a custom HttpModule. The downside to this was that HttpModules execute on every request, even if the resulting page would not be a 404 error. To make the HttpModule as lightweight as possible, we first checked whether or not the request would end in a 404 status. Because we knew that all existing URL's end in the .htm extension, this was our second check. Only if both comparisons are true, we would query the URLRedirect list to see if the user had requested an old URL.

Since we were migrating 100,000+ pages and documents, we initially ran into the list resource throttling limits. To overcome this issue, we had to change the oldURL field type to 'Single line of text' and make this an indexed column.

The end result of the HttpModule is shown in Listing 5. Please note that URLRedirectBE is a business entity class used for working with the URLRedirect list items in a strong typed manner. It retrieves a SharePoint listitem based on the oldURL column and maps the field values to public properties.

```
private HttpApplication app;

public void Init(HttpApplication context)
{
  app = context;
  app.PreSendRequestContent += context _ PreSendRequestContent;
}
public void Dispose() { }

private void context _ PreSendRequestContent(object sender, EventArgs e)
{
  HttpResponse res = app.Response;
  HttpRequest req = app.Request;

  //Only attempt a redirect if the page is not found
  if (res.StatusCode == 404)
  {
    //We only want to check for htm files
    if(req.Url.AbsolutePath.EndsWith(“.htm”)
    {
      string siteUrl = req.Url.Scheme + “://” + req.Url.Host;
      string oldUrl = req.Url.ToString();

      //Check if the requested URL is included in the URLRedirect list
      URLRedirectBE urlRedirect = new URLRedirectBE(siteUrl, oldUrl);
      if (urlRedirect.ID != Guid.Empty)
      {
        //Set the response code to 301 moved permanently
        //and perform redirect
        res.Status = “301 Moved Permanently”;
        res.AddHeader(“Location”, urlRedirect.NewUrl);
        res.End();
      }
    }
  }
}
```

Listing 5: HttpModule code example.

By using a redirection with a 301 status, we not only made sure that any existing ranking with search providers remained in place, but also when the user visits this URL through the same search provider again, they will no longer use the HttpModule.

In order to register the new HttpModule in SharePoint we created a web application scoped Feature with a feature event receiver (see Listing 6).

```
public override void FeatureActivated(
  SPFeatureReceiverProperties properties)
{
  SPWebApplication webApp = (SPWebApplication)properties.Feature.Parent;
  SPWebConfigModification modification = new
  SPWebConfigModification(“add[@name=’URLRedirectHttpModule’]”,
  “configuration/system.webServer/modules”);

  modification.Sequence = 0;
  modification.Type =
    SPWebConfigModification.SPWebConfigModificationType.EnsureChildNode;
  modification.Value = string.Format(
    @”<add name=””URLRedirectHttpModule””
    type=””URLRedirect.URLRedirectHttpModule, {0}”” />”,
    Assembly.GetExecutingAssembly().FullName);

  webApp.WebConfigModifications.Add(modification);
  webApp.Update();
  webApp.WebService.ApplyWebConfigModifications();
}
```

Listing 6: Registering the custom HttpModule in SharePoint.

The result is a user friendly redirection mechanism that takes the existing pages search provider ranking into account.

In SharePoint 2010 we used a similar mechanism to create "Friendly URLs". For example, when a user browsed for /search, they would be redirected to /searchcentersubsiteurl/Pages/default.aspx .

Conclusion

Migrating a website into a SharePoint 2013 environment is never an easy task. In this article we have discussed our dynamic approach using custom tooling and configuration XML files. The key point for every migration is to focus on what is important for the customer, and make sure that is translated to the best possible migration approach. For us, the generation of metadata throughout the content generation process was of vital importance. This ultimately allowed us to create a new website that had conceptually completely changed the visitors perspective, from a website based on a hierarchal structure to a search driven website through the use of metadata.

If you have any questions please don't hesitate to contact me by email at vverbeek@conclusion.nl, or through twitter @vverbeek87.

Version: SharePoint 2013

What: Indexing a database through BCS: Unable to display any value for BCS Managed Properties if we crawl a nvarchar(MAX) column with a large data inside.

Cause:

When database content is crawled through a BCS External Content Type that contains both 'Read Item' (SpecificFinder) 'Read List' (Finder) methods, SharePoint Search creates 2 crawled properties for each field in the External Content Type:

- Read ItemElement.<fieldname> (from ECT SpecificFinder method)
- Read ListElement.<fieldname> (from ECT Finder Method)

After you crawled your date, You'll find that all values are stored in the 'Read ListElement' crawled properties, and 'Read ItemElement' properties remain blank. This is because all data is normally gathered from the ECT 'Finder' method. After managed properties are created and mapped to the Read ListElement' crawled properties, data can be displayed in search results.

However, if a nvarchar(MAX) column contains large data, the ECT 'SpecificFinder' method is called for that particular record, and then those values are placed in the 'Read ItemElement' crawled properties instead.

Workaround:

To show all values in the search results, you need to map both the above crawled properties to the corresponding managed properties.

Richard Joosten

Wat doe jij volgende maand?

Jij bouwt het wereldwijde intranet voor een grote multinational, waar de meer dan 25.000 gebruikers dagelijks met veel plezier mee werken. Je opent de poorten naar kritische ERP, BI en HR systemen en integreert die met de modernste Document Management en Social Networking applicaties.

Jij bent namelijk een specialist die wel raad weet met SharePoint, Nintex Workflow en WSF. Iemand die altijd op zoek is naar nieuwe kennis en als eerste met de technologie van morgen wil werken, maar ook iemand die geniet van het succes dat de klant vandaag boekt met jouw innovatieve oplossingen.

Jij bent ondernemend en deelt graag je visie en kennis met je omgeving, zowel fysiek als online, en werkt samen met Designers, Architecten, Developers en Consultants aan de beste oplossingen voor je klant. Je weet hoe je werk en privé vlekkeloos kunt combineren en onafhankelijk van tijd en plaats de optimale bijdrage levert aan je team.

Volgende maand werk jij bij Macaw.

SharePoint Infrastructure Specialist
SharePoint Consultant
SharePoint Developer

Macaw
tomorrow's IT, today

Met vestigingen in Schiphol-Rijk, Barendrecht en Apeldoorn is Macaw Nederlands grootste, uitsluitend op het Microsoft-platform gespecialiseerde IT-dienstverlener. Onze kennis is gebundeld in drie Solution Centers en twee Service Centers die zich richten op het realiseren van collaboratieve portalen, ECM-systemen, internet en commerce sites, business solutions, CRM systemen, applicatiebeheer en hosting diensten.

Bezoek voor meer informatie over deze en andere vacatures onze website www.echtleukwerk.nl, volg ons op www.facebook.com/MacawNL, twitter.com/MacawNL of neem contact op met onze afdeling Recruitment via 020-8510510.

Using ASP.Net SignalR within your SharePoint apps

by Bas Lijten

With the introduction of the app model in SharePoint 2013, now about one year ago, this finally gave us the possibility to adopt "new" technology in our apps, like, for example, .Net 4.5, MVC5.0 and WebAPI 2.0. Another example is SignalR, which is a "real-time web" functionality, an ability to have your server-side code push content to the connected client, in real-time. Remember those applications, where users refresh their browsers to receive new data or, where those long polling mechanisms were used to update the user interface? This is where SignalR comes in as a solution. And the greatest thing of all: it's very easy to use with just a few lines of code! A very common scenario where SignalR is used, is a chat application, but in this article I will show, step by step, how to really integrate with SharePoint: Update the user interface of a SharePoint app whenever a SharePoint list receives an update. It uses the HTML5 WebSocket API that enables bidirectional communication between server and browser and gracefully falls back to other technologies.

Preparations

For this example, I start out with a new project. I chose to use an autohosted project, based on the MVC template, but a provider-hosted app would do as well. After your project has been created, you can open the Nu-Get Package manager console (it can be found under Tools -> Nu-Get Package manager -> Package manager console) and execute the command shown in Listing 1.

```
Install-Package Microsoft.AspNet.SignalR
```

Listing 1: Install the required nu-get packages to work with SignalR.

This will install a bunch of other packages as well, like jQuery, Newtonsoft.Json and Owin. Owin is the Open Web Interface for .Net. It defines a standard interface between .Net web servers and web applications. Using this interface makes it possible to decouple an application from a specific web server, which means that the application can run on several different platforms, for example on the Kayak webserver which runs on Linux/Unix servers using Mono. SignalR implements this interface, as shown later in this article.

Note:

Currently, there are some dependency conflicts between SignalR 2.0 and OWIN 2.0. This can be resolved by following the steps as described here: http://stackoverflow.com/questions/21439756/SignalR-2-0-2-and-owin-2-0-0-dependancy-conflict

Creating the SignalR Hub in your remote app

As the preparations are done, the next step is to enable your web application to use SignalR. This consists of 2 simple steps

Loading the SignalR middleware

First, the SignalR module needs to be loaded, therefore we need to create a new class which is used as the OWIN startup module for this assembly. From Visual Studio, add a new item to the project and select "OWIN Startup class" under the general category

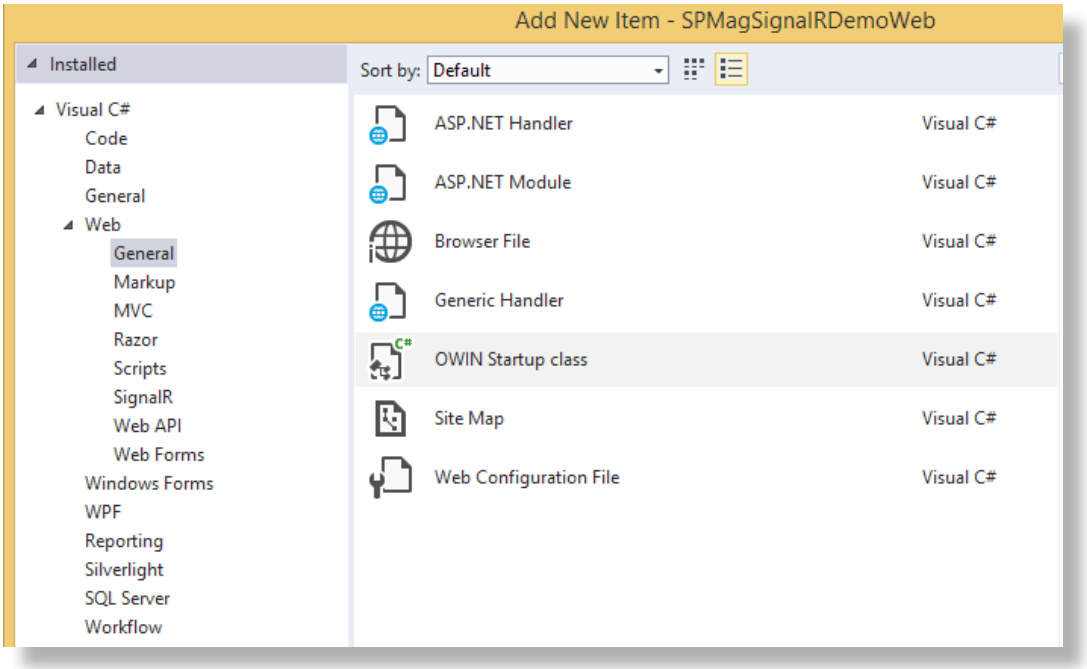

Figure 1: Create your OWIN Startup class.

Next, add the code as shown in listing 2 to the generated configuration method.

```
app.MapSignalR();
```

Listing 2: Load the SignalR module.

You should end up with the startup class as shown in figure 2. On line number 4, this class is marked as the OWIN startup class for this assembly. In the configuration method (line 9), the modules are loaded with their configurations, if they exist. And that's is all that is needed to load the SignalR module.

```
1  using Microsoft.Owin;
2  using Owin;
3
4  [assembly: OwinStartup(typeof(SPMagSignalRDemoWeb.Startup))]
5  namespace SPMagSignalRDemoWeb
6  {
       1 reference
7      public class Startup
8      {
           0 references
9          public void Configuration(IAppBuilder app)
10         {
11             app.MapSignalR();
12         }
13     }
```

Figure 2: the startup class.

In figure 3 an example of a more advanced configuration is shown (not required for this article), which enables cross-domain calls for SignalR, using jsonp and CORS. Please note that to use the CORS, the nu-get package Microsoft.Owin.Cors is needed: it's build on OWIN as well.

```
6   [assembly: OwinStartup(typeof(SPMagSignalRDemoWeb.Startup))]
7   namespace SPMagSignalRDemoWeb
8   {
        1 reference
9       public class Startup
10      {
            0 references
11          public void Configuration(IAppBuilder app)
12          {
13              app.Map("/signalr", map =>
14              {
15                  map.UseCors(CorsOptions.AllowAll);
16                  var config = new HubConfiguration
17                  {
18                      EnableDetailedErrors = true,
19                      EnableJSONP = true,
20                      EnableJavaScriptProxies = true
21
22                  };
23                  map.RunSignalR(config);
24              });
25          }
26      }
27  }
```

Figure 3: a more advanced startup class for cross-domain calls.

Creating your SignalR Hub

The next step is to create a SignalR Hub. Add a new item to your project and select SignalR Hub Class under the Web -> SignalR category and name it "InventoryHub". Replace the contents of the InventoryHub with the code as shown in Figure 4.

Every hub acts as a "channel" that clients can connect to and takes care of sending and receiving messages. When the SignalR module is loaded, a JavaScript proxy will be generated for every hub (it's located on /SignalR/hubs). From this proxy, it's possible to call the sendUpdate method. Whenever this happens, this hub takes care of the message that has been sent by the client, and instructs to send this message to all clients that are connected, except the client that sent the message (line 10).

```
        1 reference
6       public class InventoryHub : Hub
7       {
            0 references
8           public void SendUpdate(string message)
9           {
10              Clients.Others.ClientNotification(message);
11          }
12      }
13  }
```

Figure 4: SignalR InventoryHub.

Sending out messages

Whenever an event takes place, a message can be send to the connected clients. For this inventory example, this happens whenever an update to the inventory takes place. First, create a new list in your SharePoint app, with the following fields: Title<String>, Amount<Number> and Price<Number>, as shown in Figure 5.

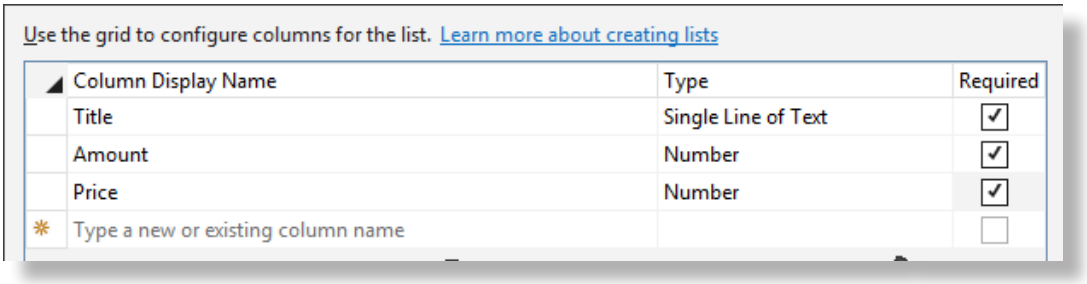

Figure 5: The inventory list.

To be able to notify the clients when changes to this inventorylist happen, a remote event receiver needs to be added. For this example, I used the itemAdded event.

The code that was generated within the ProcessOneWayEvent method, can be removed, as it isn't needed: that code needs to be replaced with a little bit of logic to update all the connected clients. Before this logic can be written, add a new class "InventoryItem", as shown in Figure 6, has to be created. This model will be used to send the new item to all clients.

```
6  namespace SPMagSignalRDemoWeb
7  {
       0 references
8      public class InventoryItem
9      {
           0 references
10         public string Title { get; set; }
           0 references
11         public int Amount { get; set; }
           0 references
12         public decimal Price { get; set; }
13     }
14 }
```

Figure 6: InventoryItem Model.

The code provided in Figure 7 takes care of the item added event and sends the new item to all connected clients. Please note that the SendUpdate method (Figure 4, line 8) could be used here. Whenever you don't want to provide a proxy that every client can send messages to, and only want to send updates from your web application, this method shouldn't be used.

On line 37, the hubcontext for our InventoryHub is picked, which is used on line 47 to broadcast an update to all clients using the newItem function. This newItem function is a dynamic function and can have any name. At the end of the article we'll see how to subscribe to the newItem event.

```
35          public void ProcessOneWayEvent(SPRemoteEventProperties properties)
36          {
37              var inventoryHub = GlobalHost.ConnectionManager.GetHubContext<InventoryHub>();
38              var itemProperties = properties.ItemEventProperties.AfterProperties;
39
40              var inventoryItem = new InventoryItem
41              {
42                  Title = itemProperties["Title"].ToString(),
43                  Amount = Int32.Parse(itemProperties["Amount"].ToString()),
44                  Price = Decimal.Parse(itemProperties["Price"].ToString())
45              };
46
47              inventoryHub.Clients.All.newItem(inventoryItem);
48          }
49      }
50  }
```

Figure 7: remote event receiver code.

This was all the ASP.Net code that is needed to get a working SignalR solution. To prove it, a user interface needs to be created which can receive notifications.

Creating the user interface

Before the user interface can be updated with new inventory items, items should of course be added first. Using the CSOM or REST API would be a smart idea to use, but for the sake of simplicity, I just added a link to the menu that points to newform.aspx using razor. The code can be found in Figure 8.

```
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("About", "About", "Home")</li>
        <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
        <li><a href="@(Request.QueryString["SPAppWebUrl"] + "/Lists/Inventory/NewForm.aspx")">Create New</a></li>
    </ul>
```

Figure 8: Add a link to newform.aspx.

The next, and last step, is to write some logic to update the user interface whenever an event takes place. Two actions are needed here:

- Adding and referencing JavaScript libraries
- Update HTML and add some JavaScript logic

Referencing JavaScript libraries

First, add the knockout library using Nu-get to your project. This enables easy databinding in our view. Second, reference the SignalR and knockout libraries. Add the code from Listing 4 to BundleConfig.cs.

```
bundles.Add(new ScriptBundle("~/bundles/SignalR").Include(
                        "~/Scripts/jquery.SignalR-{version}.js"));
bundles.Add(new ScriptBundle("~/bundles/knockout").Include(
                "~/Scripts/knockout-{version}.js"));
```

Listing 4: Add the JavaScript bundles.

Next, load those JavaScript libraries in the header of your layout file (_Layout.cshtml). Please note that I moved the jquery library to the header as well. Figure 9 shows the code.

```
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>@ViewBag.Title - My ASP.NET Application</title>
7      @Styles.Render("~/Content/css")
8      @Scripts.Render("~/bundles/modernizr")
9      @Scripts.Render("~/bundles/jquery")
10     @Scripts.Render("~/bundles/signalR")
11     @Scripts.Render("~/bundles/knockout")
12 </head>
```

Figure 9: Load the JavaScript files.

Update HTML and add some JavaScript logic

The last step is to write some HTML and add the JavaScript logic to subscribe to the SignalR updates. Open up Index.cshtml and replace the contents with the content that is shown in Figure 10. It contains a layout with a table inside, with 3 columns, and enables knockout to databind the items from my viewmodel. In addition to the table, a list is created, which will hold all notifications from other clients.

```
4   <h2>Inventory</h2>
5
6   <div class="row">
7       <div class="col-lg-12">
8           <div class="span8">
9               <p class="lead"><u>Inventory List</u></p>
10              <table class="table table-striped">
11                  <thead>
12                      <tr>
13                          <th>Title</th>
14                          <th>Amount</th>
15                          <th>Price</th>
16                      </tr>
17                  </thead>
18                  <tbody data-bind="foreach:viewModel.items">
19                      <tr>
20                          <td data-bind="text: Title"></td>
21                          <td data-bind="text: Amount"></td>
22                          <td data-bind="text: Price"></td>
23                      </tr>
24                  </tbody>
25              </table>
26              <p class="lead"><u>Client Notifications</u></p>
27              <ul data-bind="foreach:viewModel.clientNotifications">
28                  <li data-bind="text: $data"></li>
29              </ul>
30          </div>
31      </div>
32  </div>
```

Figure 10: HTML to display all new items.

The last step is to add the JavaScript logic, the code is provided in Figure 11. The very first line loads the SignalR proxy that was automatically generated for the InventoryHub that was created before (Figure 4, line 6). On the document load event, the viewModel is databound and the InventoryHub is registered for use. Lines 42-55 are the lines where the magic happens: Two anonymous functions are bound to the newItem event and to the clientNotifications event which can be fired on the server. The newItem event is fired whenever an action on the SharePoint list takes place, the clientNotifications event is fired whenever a client connects to Hub, as can be seen on line 53 of Figure 11.

The anonymous events contain logic to add the notifications to the viewModel, which is databound via knockout. The last step is to instantiate the connection, using the code as seen in Figure 11, line 52.

```
34  @section scripts {
35      <script src="/signalr/hubs"></script>
36      <script>
37          jQuery(document).ready(function ($) {
38              ko.applyBindings(viewModel);
39              registerHub();
40          });
41
42          function registerHub() {
43              var inventoryHub = $.connection.inventoryHub;
44              inventoryHub.client.newItem = function (inventoryItem) {
45                  viewModel.items.push(inventoryItem);
46              };
47
48              inventoryHub.client.clientNotification = function(message) {
49                  viewModel.clientNotifications.push(message);
50              };
51
52              $.connection.hub.start().done(function() {
53                  inventoryHub.server.sendUpdate("client connected");
54              });
55          }
56
57          var viewModel = {
58              items: ko.observableArray(),
59              clientNotifications: ko.observableArray()
60          };
61      </script>
62  }
```

Figure 11: JavaScript.

Testing your code

This was all the code that you need to build your first SignalR application. When you chose to use an autohosted application, one more action is needed: configure the Azure service bus to be able to debug remote events. Therefore, go to your Azure management portal (http://manage.windowsazure.com) and create a new service bus. Copy the contents from the connectionstring, as show in Figure 12.

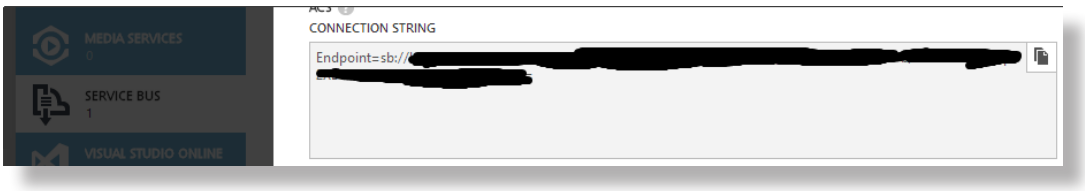

Figure 12: Azure service bus connectionstring.

Next, open up your SharePoint project properties and select the SharePoint tab. In the Debugging section, this connectionstring can be entered. Save your project settings and all you have to do is to fire up the application. Make sure to open two windows and see the notification showing up in the first window, when the second window has been opened. This is caused by the code "Clients.Others.ClientNotification", which updates all clients except the sending client. (Figure 13).

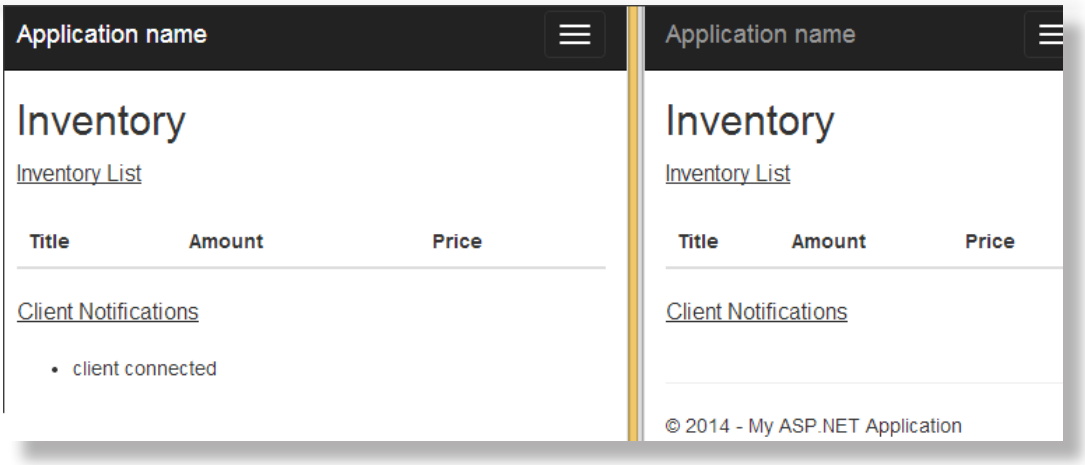

Figure 13: a client connects and notifies the other clients.

Whenever you open the "Create new" window, and add a new item, both windows get updated with the new item. (Figure 14)

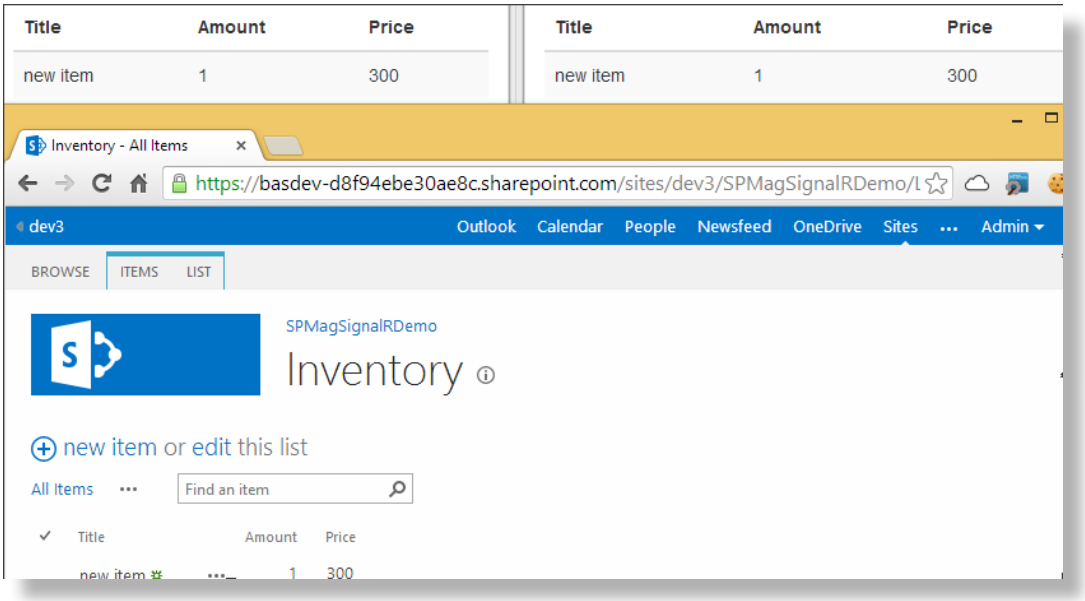

Figure 14: both windows get updated when a new item has been added.

Conclusion

SignalR is a very easy to use real time web technology. With literally just a few lines of code, this real time communication module can be integrated with SharePoint and your custom App, to provide real time updates. I have shown how to send notifications from one client to all other clients, and how to update all clients with information, whenever an action within SharePoint happens. I hope that I have inspired you to create some stunning SharePoint apps using SignalR!

User generated metadata

by Albert Hoitingh

In order to make any information more retrievable, you want to add some form of metadata. In the most primitive scenario's a relevant document-title will suffice or maybe a description. But with the avalanche of information which we must process on a daily basis, this kind of metadata simply is not enough.

Platforms like SharePoint offer us a multitude of options to enhance our information further. Some of these options are integrated into the platform or work alongside Microsoft Office.

In this article I want to look at some of these options. To be more specific, to look at the ways in which a platform like SharePoint supports us in adding our own metadata. Can we use SharePoint to make information better available in a way we, the end-user, want to? And what are the consequences of this for the organization?

Should all metadata be mandatory and centrally managed or should we go for a free format, optional and even folksonomy metadata model? These are questions many enterprises are dealing with and for which platforms like SharePoint, with their multitude of options, don't have a definite answer.

I'll try to give you mine in this post.

First things first though. What is metadata?

Metadata

Quote

Metadata is "data about data". The term is ambiguous, as it is used for two fundamentally different concepts (types). Structural metadata is about the design and specification of data structures and is more properly called "data about the containers of data"; descriptive metadata, on the other hand, is about individual instances of application data, the data content.

Metadata is traditionally found in the card catalogs of libraries. As information has become increasingly digital, metadata is also used to describe digital data using metadata standards specific to a particular discipline. By describing the contents and context of data files, the quality of the original data/files is greatly increased. For example, a webpage may include metadata specifying what language it is written in, what tools were used to create it, and where to go for more on the subject, allowing browsers to automatically improve the experience of users.

This quote is from Wikipedia. So, in my own words; Metadata is additional data fields used to enhance an object. Take a document for example. It will have an author and perhaps a subject. But metadata provides us with an opportunity to link more information to this document. The type of document for example or the version number or even the number of likes. In general, metadata is used to be able to search or (more general) to be able to locate your information more easily.

Metadata is everywhere. Even when just writing a blog article using Microsoft Word, metadata is collected. Properties like the author of the document and the creation date are automatically stored with the document, and thus enabling platforms like Microsoft SharePoint to store these as metadata. But these are not all the metadata possibilities.

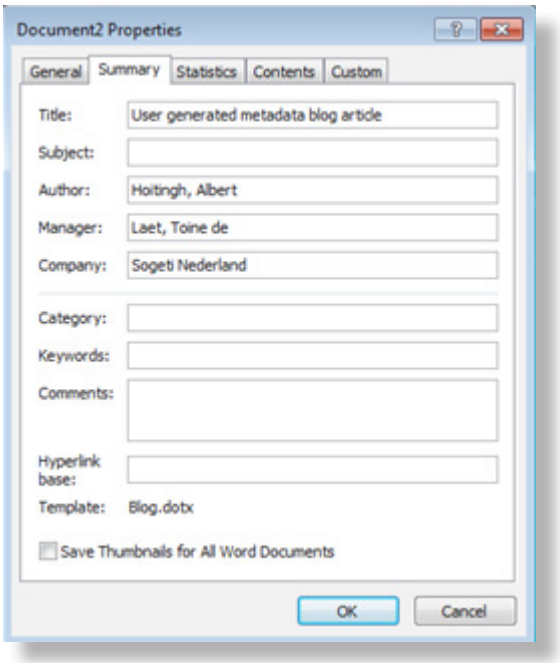

Figure 1: Document properties in MS Word.

Types of metadata in document management

In this article I'll be discussing the type of metadata that describes or enhances content. And basically when looking at this from the SharePoint perspective, there are four types of metadata:

1. System metadata.
2. Standard metadata.
3. Organization specific metadata (formal metadata).
4. Informal metadata.

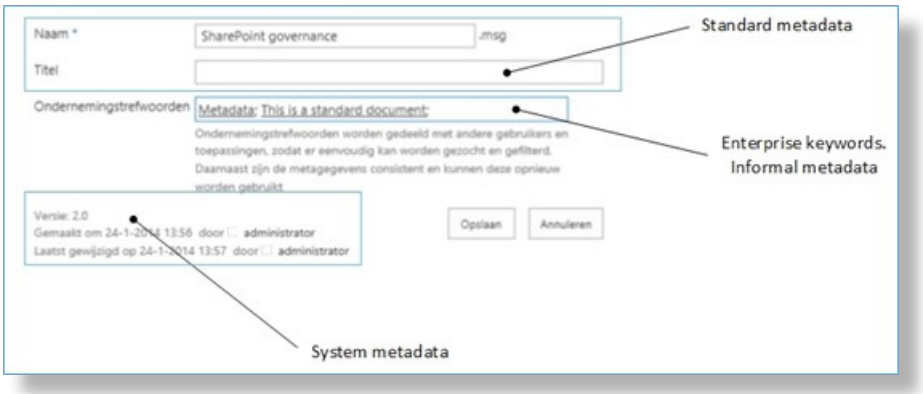

Figure 2: SharePoint 2013 document metadata.

System and standard

The first two categories are the ones that we, as a user, cannot modify. System metadata is items like “created by” or “modified by”. The system (i.e. SharePoint) provides these free of charge and automatically. We can use this metadata, but we won’t be able to change it.

Next is the standard metadata. And I use the term standard because this is metadata that is regarded as standard in the content management arena. Take the Dublin Core document metadata standard for example. When adding a document to SharePoint you should at least expect an author field or a title field. These can be optional or mandatory or even filled automatically by Microsoft Office.

Either way, the system and standard metadata is out of the box. You don’t have to change these and you can use them right away.

Organization specific

But in most cases, just using system metadata is not enough. As an enterprise you will need to add your own metadata. And perhaps even allow your employees to add values of their own. That’s when things become tricky.

I won’t go into the solutions to store this metadata, as I assume we are all aware of the options (site columns, content types, content type hub, and etcetera). Nor will I go into the nice additional feature you can have using metadata fields. So I won’t focus on default values based on the location of the document, input validation and allowing separate values in a selection field.

When you do decide to provide your own metadata, there are a lot of different options available. To name but a few:

- Text: A free-format way to add text. Either one line or multiple. Either plain or rich text.
- Managed metadata: They epiphany of formal metadata. Presented as a list of managed values.
- Choice: Presented as a list of managed values, but within the column itself.
- Number: A number.
- Date/time: A field that can store a date and/or time. Can be selected using a date picker.
- People: A field that can store a link to a person. Can be selected using the SharePoint Peoplepicker.

Some of these fields are no problem at all. A date/time field will limit the users input somewhat. But when do you need a text field? And should you go for choice fields or managed metadata?

So when you’re deciding on the use of metadata, you almost automatically need to decide if you want to provide mandatory and choice only metadata or a more lenient model.

This all depends on your enterprise. In my experience, you should at least have a basic metadata model which you use to create your columns. Based on this, you go on to discuss formal and informal metadata.

Formal metadata - taxonomy

Formal metadata is the metadata that your organization would like you (and expects you) to use. A taxonomy normally comes into play here. A taxonomy contains a centrally managed collection of metadata terms. These terms can be used in a multilingual environment, can be linked to other terms and can have synonyms (in a nutshell). In SharePoint you can delegate this taxonomy and provide a separate one for each department for example. Or you can go for an enterprise wide taxonomy.

A taxonomy has at least one problem: it's centrally managed (even if this has been delegated). This seems odd, but let me explain. A taxonomy will never be able to contain every term which might be useful in the enterprise. Employees will have their own values which they find useful. You might consider opening up the taxonomy. Employees can then simply add their own values. But this is contradicting to the philosophy behind the taxonomy.

So if you need to provide these employees the means to add their own values, look a bit further. And delve into the realm of more informal metadata.

Informal metadata – bring your own metadata (BYOM)

Ok, here's where metadata like tags, notes, likes, keywords, comments, well… you get the drift, come into play. In my opinion: informal metadata.

By the way. You can have long discussions on whether metadata is formal or informal. Enterprise keywords are part of the metadata of a document, so these could be seen as formal. But by my definition, formal metadata is mandatory and can only be selected. Informal metadata allows people to add their own values.

Free text and "Allow fill-in choice"

One of the easier ways to add your own value to a metadata field, is to use the "allow fill-in choice" option when using a choice field. A user will be able to add his/her own value to the list. The problem is, this value is only available to this particular user. It won't show up when a different user selects the metadata. So perhaps this is not the way to go.

Another option is to add a "comments" metadata field to documents. Most of the time this is a free text field. I won't go in on the subject of usefulness. When versioning is enabled, every version of the document has the system metadata field "comments", so adding the same field might be overkill.

But this kind of field does enable your employees to add their own values. But this type of field is not supposed to be used to categorize documents. It's needed to provide extra information on the document. When people use this kind of field as a metadata field, you might run into problems.

First of all, it's error prone. Typos are easily made and SharePoint will not correct them or suggest an alternative. Second, not all documents will be retrieved when searching, because of these typos and the information in text fields will not be used as search refiners automatically. And third, there is no way of managing the content of these fields.

So from a metadata perspective, free text fields (including fill-in choices) have a limited usability. So what are other options? In all, we have two. And these two have similarities but are also very different. We're talking about enterprise keywords and labels.

Enterprise keywords

Enterprise keywords allow users to add their own metadata to a documents. If it is a new keyword, SharePoint will store this in the managed metadata Keywords section. If the keyword already exists, then the user is prompted with the similar keyword. But that's not all. When the user types their value, SharePoint will check if the keyword already exists in the taxonomy as-well! So you get a best-of-both-worlds scenario.

Figure 3: SharePoint 2013 enterprise keywords .

The keywords are part of the document management environment. This means that you can add them when adding a document to a document library or when creating the document in Microsoft Word. The keywords also appear (if you choose so) in the view(s) of the document library.

Some other features of enterprise keywords:

- Can be added and/or changed from within the document library;
- Can be changed using the Quick Edit mode in the document library (SharePoint 2013);
- They can be used to filter documents in the document library;
- They appear (alongside tags) in the tag cloud (see below);
- You can "follow" a keyword or add the keyword to your profile;
- The keyword will not appear on the social newsfeed.

Tags

Another option to add your own metadata values is tagging. This is somewhat similar to the enterprise keywords in that SharePoint will also store the tags in the managed metadata Keywords section. It will also check if the tag already exists in the taxonomy. But that's basically were the similarity ends.

Figure 4: SharePoint 2013 newsfeed and tag cloud including tags.

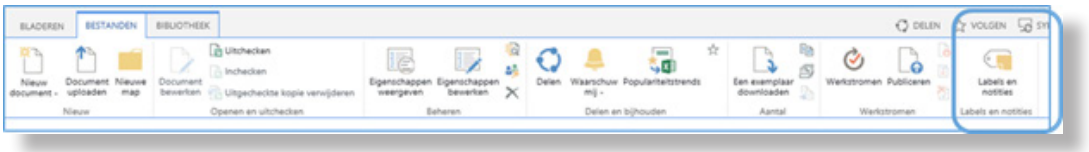

Figure 5: SharePoint 2013 tagging option in ribbon.

Tagging is mostly a social feature. It's the most informal of all metadata options. When someone tags information (e.g. a document), this tagging action will be displayed in the social newsfeeds. But the tags won't show up as metadata in the document library. You won't see a column "tags", for example.

Tagging works differently from a document management perspective. From this perspective, you would expect all document management options to be available when you're adding documents. But tags can only be added to existing documents and you will need to select the Tags and notes option from the ribbon.

Tags are therefore not part of the document library.

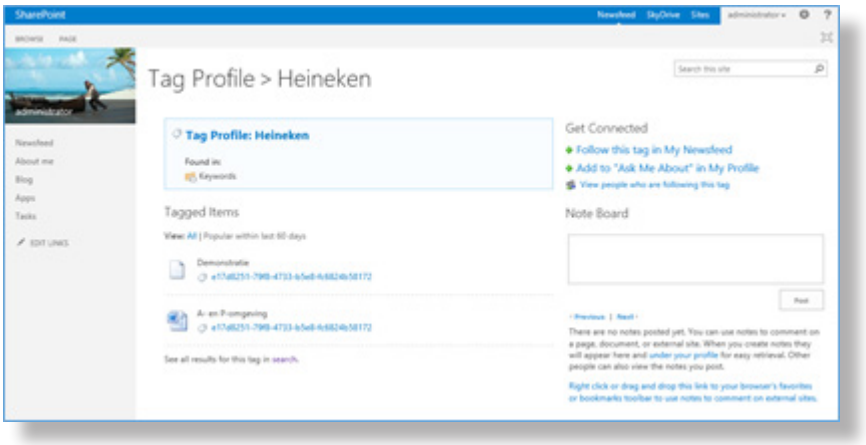

Figure 6: SharePoint 2013 tag profile.

Some other features of tagging:

- Tags don't show up in the view(s) of the document library;
- Tags cannot be added or modified using the Quick Edit option;
- Tags will appear on the tag cloud;
- You can "follow" a tag or add the tag to your profile;
- Tags show up in the newsfeeds;
- You cannot add or modify tags from Microsoft Office;
- Tags can be personalized
  (other people won't be able to see what items you tagged).

Manageable

Most enterprise will be skeptical on implementing keywords or tags. Most will be wondering if these types of metadata can be managed. Well..... Yes and no. With keywords or labels you won't get the nice managed metadata management options. You can't add synonyms to a tag or have different languages for a keyword. Let face it: these are informal (user generated) metadata!

But you do get an overview of all keywords and tags within the organization. These are stored in the managed metadata service, under Keywords. Let's say the scenario unfolds where you decide that a tag is "worthy" of transforming from the informal to the formal side. In that case, you can move the tag to the taxonomy. All management option will become available and the tag is still available as a tag. Nice!

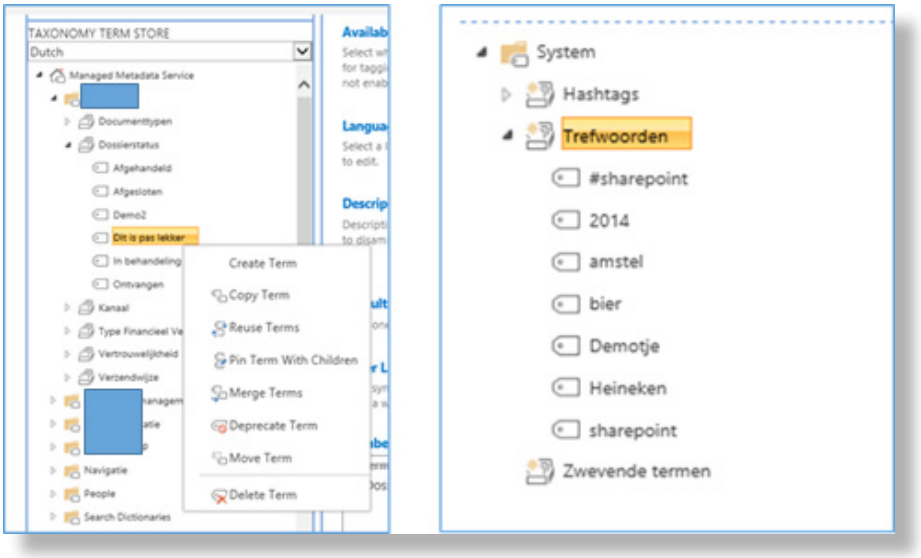

Figure 7: Taxonomy and keyword management in SharePoint 2013 .

Overview

At this moment I believe I've told you a lot about metadata. But perhaps it's a bit too much. So here's an easier table:

	Formal metadata		Informal metadata	
	Text/number/date	Taxonomy	Keywords	Tags
Option	Selectable/fill-in	Selectable	Fill-in	Fill-in
Shown	In views	In views	In views In tag cloud Keyword Property page	Tag cloud Newsfeed Tag Property page
Searchable	Yes	Yes	Yes	Yes
Can be made mandatory	Yes	Yes	No	No
Risk of typo	Yes	No	Yes	Yes
Will move to record center	Yes	Yes	Yes	No
Manageable	No/minimal	Full	Minimal	Minimal
Microsoft Office integration	Yes	Yes	Yes	No
Add to profile or subscribe	No	No	Yes	Yes

Conclusion

Having an option for the users to add their own metadata values does have its rewards. It can help to professionalize the information management within the enterprise. Let's face it: two know more than one and by using keywords or tags, you will soon discover hidden knowledge within the organization. Users will be able to find more information more quickly and are themselves found more easily when adding keywords or tags to their profile.

Implementing

But it does have its drawbacks. For one: what to choose? You will need to have a clear understanding of the workings of a taxonomy and a folksonomy in order to create the most optimal metadata solution for your organization. Tags thrive in a social environment with lots of newsfeeds, profiles and interaction. In a document management environment you should use keywords instead of tags. In some organizations it would be best to keep the formal metadata to a minimum and use keywords instead.

But then again, this does depend on the enterprise. In a highly regulated environment you would probable like to have specific taxonomy fields which are mandatory. In a more social environment (an R&D department for example) you might want to go for less taxonomy and more tagging.

It also depends on the expected effort needed to implement the required metadata model. And this is not just the technical side of it. Yes, we will need a social database, user profile synchronization and the distributed cache service to get the most out of the platform. But that's where the SharePoint architect comes in. But implementing also means getting the people to use it. It needs to be clear to them what they will get out of this.

Adoption

Just adding 25 mandatory fields to a document just because the Document & Information Management department has lost track of document usage within the enterprise is not a valid reason or a good solution.

On the other hand, adding informal metadata to a document library should have its reasons as well. Because you still have to explain why this is necessary when you already have (for example) 10 formal metadata fields as well. So, communication and adoption has to be part of the total implementation.

Drawback

Another possible drawback is the tagging overload. Because of the tsunami of tags, people can't find their information anymore. Which might be a very ironic thing indeed. And there might be a small community of people tagging, which will give this community some form of control of the metadata.

I don't agree with some of these points. The best way to get some experience with all types of metadata is simply: start using them. See what fits. Accelerate this by motivating your users to use you metadata model. And evaluate this in a couple of months.

SharePoint 2013 and Enterprise Search

by Jaap Zwart

For a huge national client in the Netherlands a new enterprise search solution had to be build. SharePoint 2013 was chosen as the main candidate, but it soon it became clear that there is more to it than just choosing SharePoint 2013 when implementing an enterprise search solution.

Introduction

In this article we try to give a brief description of some SharePoint 2013 search elements and their functionality. Some basic farm architectures are presented which we use for the small and medium farm solutions. The article will continue to describe some additions needed when using SharePoint 2013 in an enterprise search solution. The article will finish with a conclusion and global road ahead for this company.

Search in SharePoint 2013 has some great new features and additions. At the heart of SharePoint 2013 search there are still FAST-like technologies. Most of the code seems to have been changed or is rewritten. What is interesting for us is that several technologies and ideas from Bing have been added. This provides us with a set of enterprise search capabilities that gives us some space for customizations. Furthermore it includes a rule based query parsing framework. When we search content that mostly resides in SharePoint itself, probably SharePoint 2013 search offers enough functionality for us. We also want to define enterprise search projects where SharePoint 2013 is used to search other platforms and repositories, and therefore add-on capabilities will probably be needed to fit the bill. That situation will be described later in this article.

The Search Architecture overview

Before diving deeper into the subject of enterprise search with SharePoint 2013, it is good to first describe some of the most important elements of the search architecture. Figure 1 shows an overview of the different components of the SharePoint 1013 search architecture.

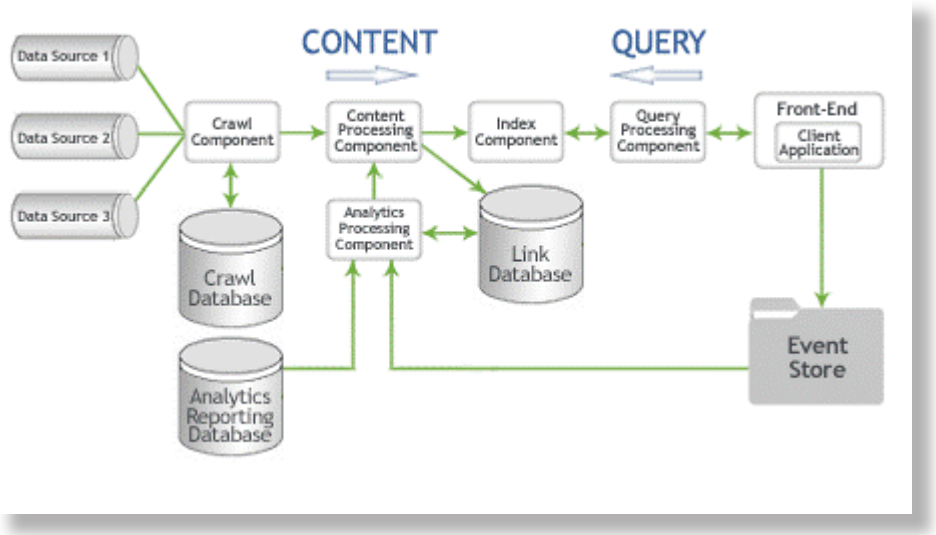

Figure 1: SharePoint 2013 search architecture components.

The crawl component will crawl our content sources. It is possible for us to crawl many content sources. We will crawl file shares, SharePoint content, line of business applications and others. To retrieve the information, our crawl component connects to the content sources by invoking the appropriate indexing connector or protocol handler. Our crawled items are transported to the content processing component. It is possible for us to create different content sources specific for our needs.

Figure 2 shows the screen in which we can set up new content sources. We can even create a BCS connection to an external source and create an LOB content source to be crawled. For our SharePoint 2013 enterprise search requirements we need to expose LOB applications through this procedure.

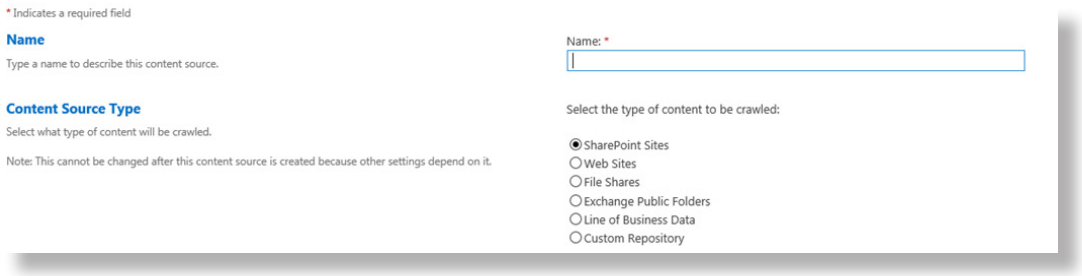

Figure 2: Setting up a new content source.

What is important for us is that the content processing component processes the crawled items and sends them to the index component. The important tasks of document parsing, property mapping, and linguistics processing and entity extraction are all done for us through the content processing component. Our crawled items are transformed into artifacts and added to the search index. The link database is updated with the relevant information about links and URLs.

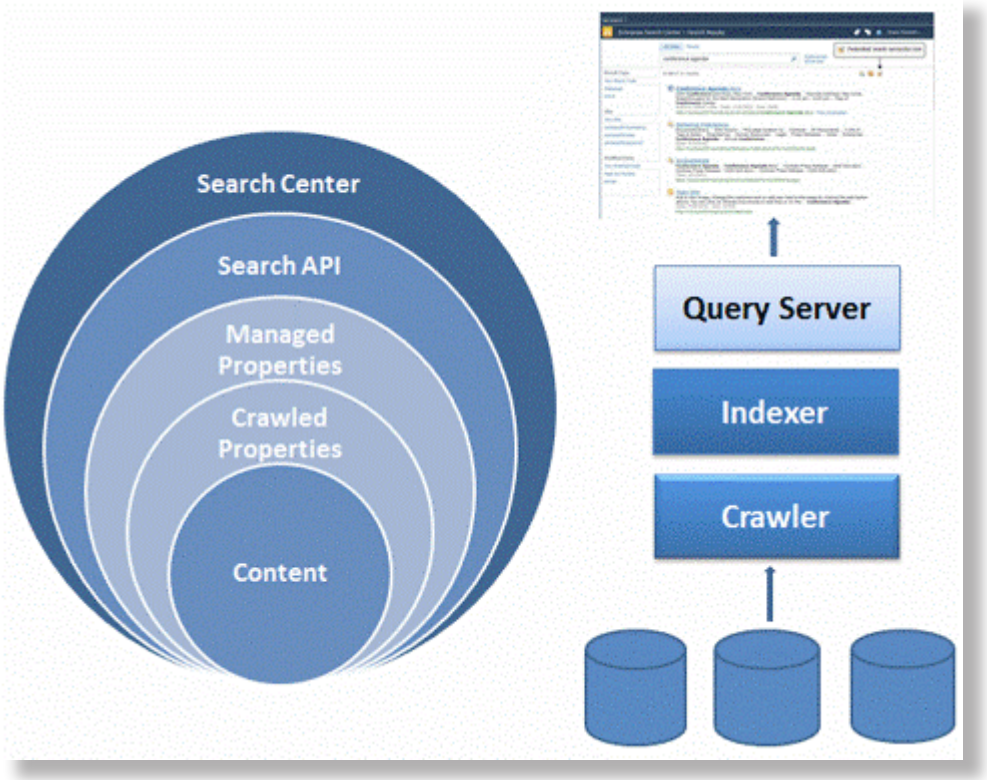

Figure 3: Schematic overview of SharePoint 2013 search crawling.

We also want to check some analytics around search and usage. This is done through the analytics processing component. It will help us to improve the search relevance and create search reports. From this we are able to generate recommendations and deep links. Usage analytics will help us to analyze usage log information received from the front-end via the event store and it will generate usage and statistics reports for us.

Search analytics will help us to extract information from the link database, such as links, the number of times an item is clicked, anchor text, data related to people, and metadata. This information is important to predict the relevance of found items. The results from these analyses are added to our items in the search index. The results from the usage analytics are stored in the analytics reporting database. This will help us tremendously in fine tuning the search results to the behavior of our users.

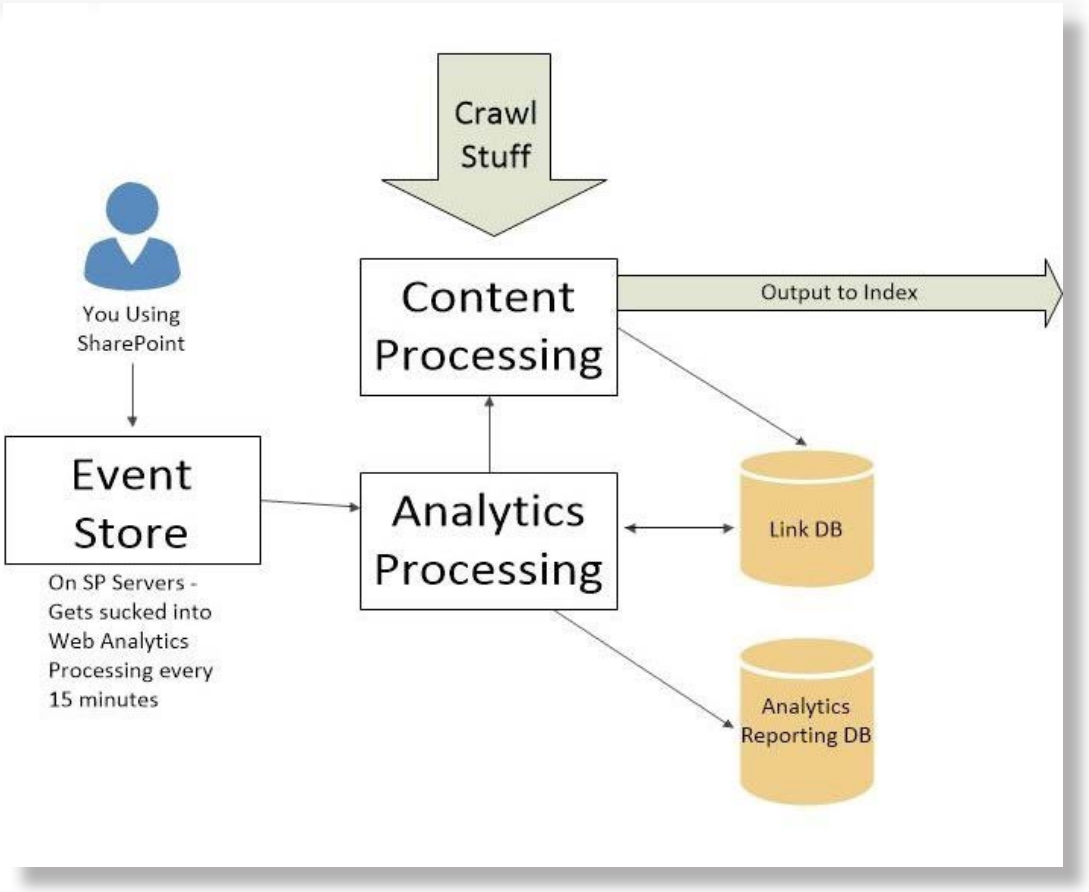

Figure 4: Content processing and analytics.

We will divide the search index into discrete portions, called index partitions. Our search index is the aggregation of all index partitions where each index partition will hold one or more index replicas that contain the same information. We want to achieve fault tolerance and redundancy and so we create additional index replicas for each index partition and distribute the index replicas over multiple servers. Our index component is the logical representation of an index replica. In our search topology we have to provision one index component for each index replica.

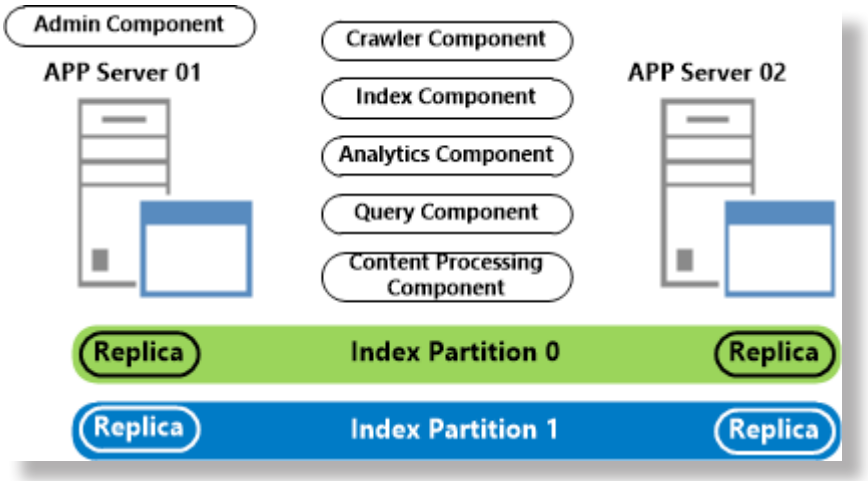

Figure 5: SharePoint 2013 index partitions and replicas.

Our index component receives processed items from the content processing component and writes those items to an index file. These Index files are stored on a disk of the server that hosts the index component. It will receive queries from the query processing component and returns result sets.

The query processing component analyzes and processes our queries and results. It will also perform linguistics processing like word breaking and stemming. When this query processing component receives a query from our search front-end, it analyzes and processes the query to optimize precision, recall and relevance. The processed query is then submitted to our index component. Our index component returns a result set based on the processed query to the query processing component, which in turn processes that result set, before returning it to the search front-end. The search administration component in our architecture runs the system processes for search. It supports provisioning like adding and initializing instances of the other search components.

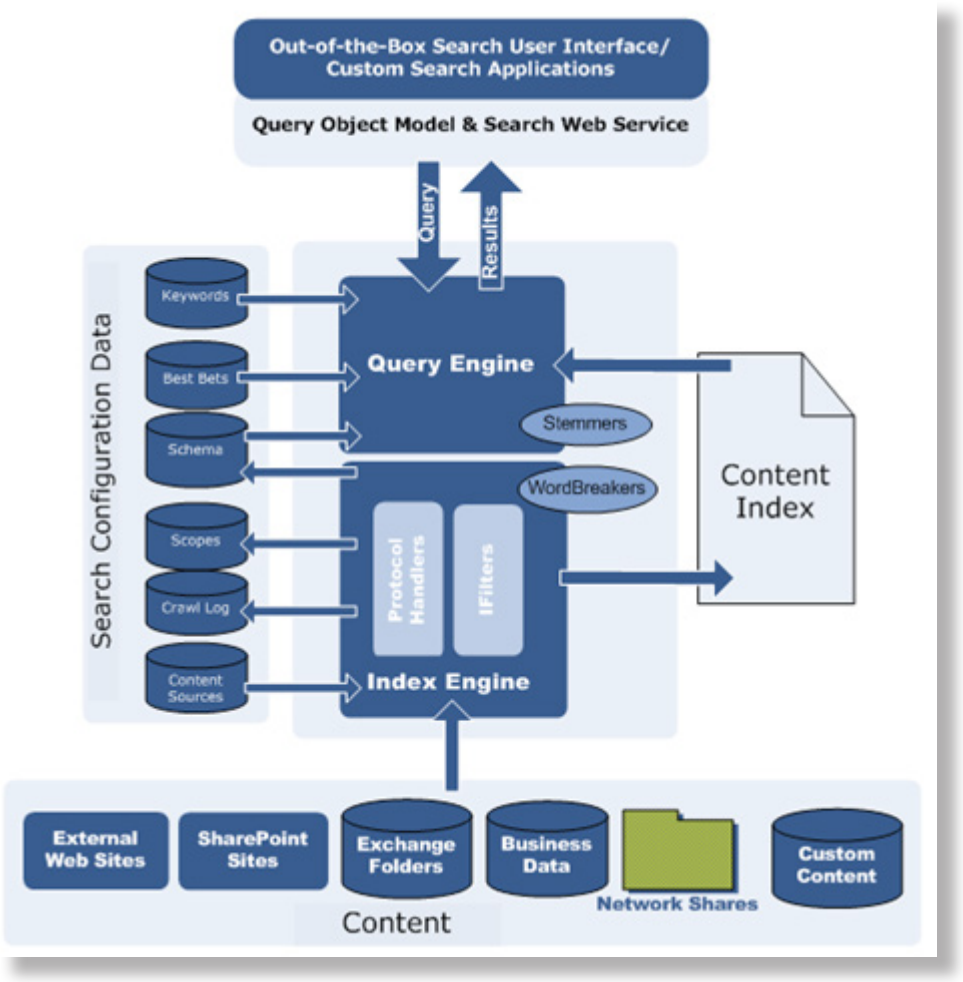

Figure 6: SharePoint 2013 search component interaction.

Next our crawl database stores tracking information and historical information about crawled items within our organization and stores information about the last crawl time, the last crawl ID and the type of update during the last crawl. Our link database stores information extracted by the content processing component. It also stores information about search clicks concerning the number of times our employees click on a search result from the search result page. This information is stored unprocessed, to be analyzed by the analytics processing component.

Our analytics reporting database stores the results of usage analytics. It also stores statistics information from the analyses. This information will be used to create Excel reports showing different statistics. And finally our search administration database stores search configuration data like the topology, crawl rules, query rules, and the mappings between crawled and managed properties. It also stores the access control list (ACL) for the crawl component. It is not possible to have more than one search administration database per search service application.

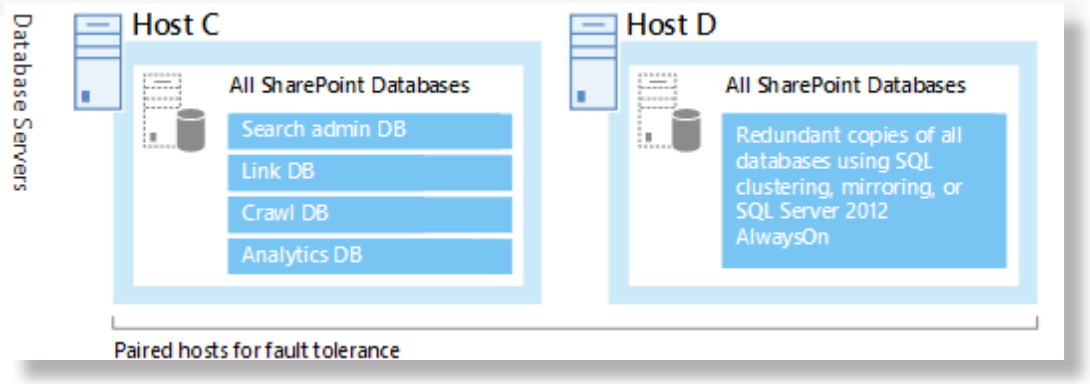

Figure 7: SharePoint 2013 search databases.

Plan our Farm Architecture

Before we set up our enterprise search architecture, there are many things that require careful planning. Step by step we need to plan for a small, a medium, or a large-size enterprise search architecture. We need to check if we have people who are familiar with the components of the search system in SharePoint 2013 and how they interact with each other, like described in the previous paragraphs, so these people can setup, manage and maintain our search architecture.

The volume of content that we have in our search index will also affect what resources we need to host our farm. We need to work out approximately the number of items that we plan on making searchable. Examples of items are documents, web pages, SharePoint list entries, and images. It is important to realize that each entry in a SharePoint list counts as one item. More complexity will be added when we search external content.

When we have established a content baseline we have to multiply it by what we think the expected growth of that content will be over the next 6 to 12 months. For example, when we start out with 10,000 indexed items, and we expect the volume of that content to triple over the next 6 months we should plan for 30,000 or 60.000 (12 months) searchable items. This can also be done for external content, but the math will be a bit more complex.

It's not easy to determine how big or small the search architecture should be. The size of our search architecture depends highly on the volume of our content, the crawl rate, the query throughput, and the level of high availability that we require. We can use sample search architectures that were tested by Microsoft which can be used as a basis to plan our own farm. The sample search architecture that we choose depends on how much content has to be indexed and searched.

If we have up to 10 million items, the small search farm will probably be a suitable farm for us. Microsoft tested this search architecture, and measured that it can crawl 50 documents per second, and serves 10 queries per second. With a crawl rate of 50 documents per second, it takes search 55 hours to crawl 10 million items in the first full crawl.

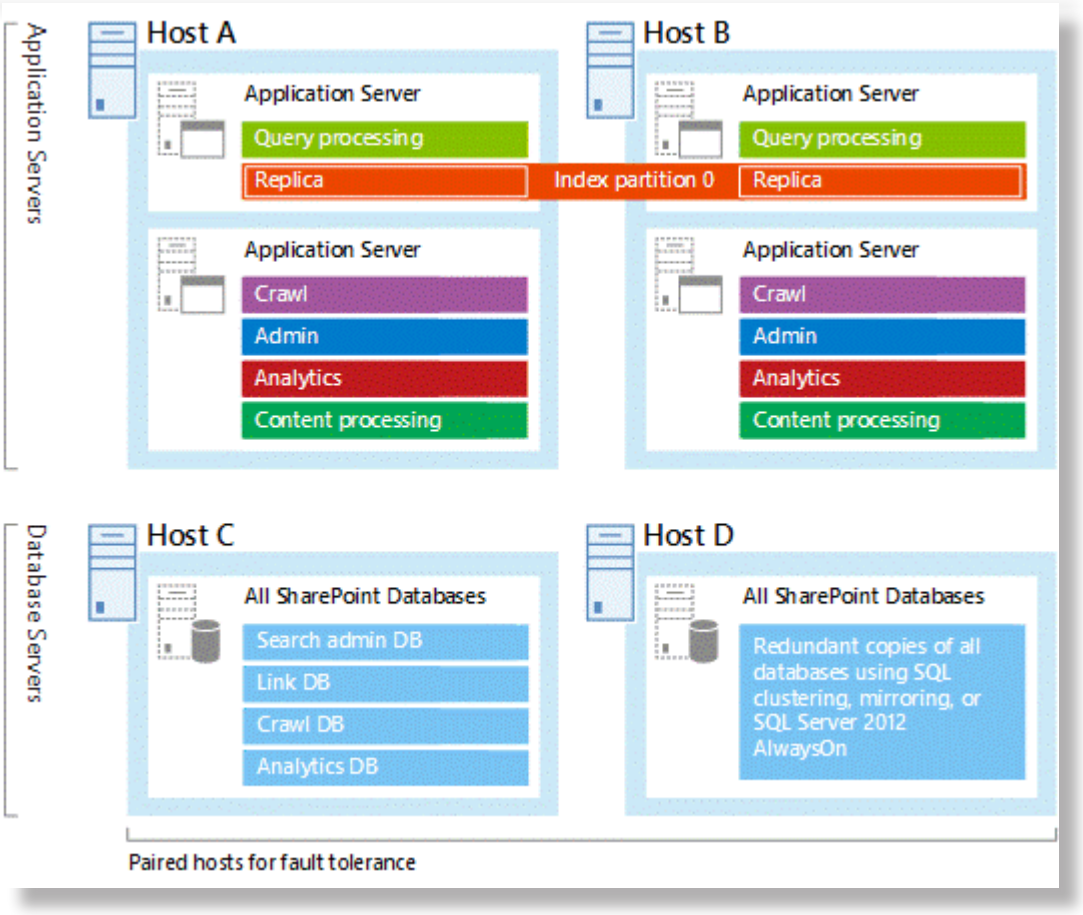

Figure 8: Small SharePoint 2013 search architecture.

If we have between 10 and 40 million items, the medium search farm will probably be the most suitable farm for us. Microsoft tested this search architecture, and measured that it can crawl 100 documents per second, and serves 10 queries per second. With a crawl rate of 100 documents per second, it takes search 110 hours to crawl 40 million items in the first full crawl.

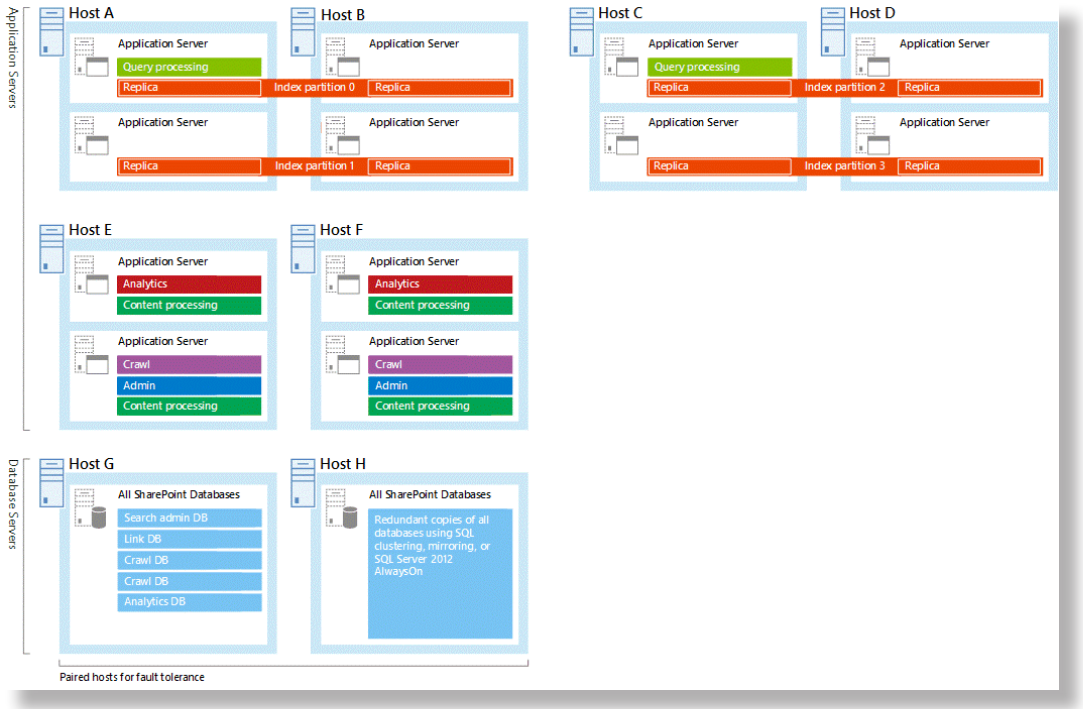

Figure 9: Medium SharePoint 2013 search architecture.

There is also a large search architecture which we will not include in this review. It can be found on the relevant Microsoft sites and we advise you to look at it if you are likely to end up with more than 40 million searchable items. If we use one of the small or medium architectures that Microsoft tested for us, then we will be running our search architecture on virtual machines.

For us running our search architecture on virtual boxes is not always allowed because of confidential data. Luckily for us it's also possible to run our search architecture on physical servers. In the sample farm architectures we just have to move the search components from the virtual machines to the host and take away the virtual machines. Although a virtual environment is easier to manage, its performance level can sometimes be slightly lower than that of a physical environment. A physical server can host more search components on the same server than a virtual server. These are very important advantages for us.

Each of our search components and search databases requires a minimum amount of hardware resources from the host server to perform well. The more hardware resources we have, the better the performance of our search architecture will be. These facts are important elements for our scale out strategy.

It is also important for us to make sure that each of our host servers has enough disk space for the base installation of the Windows Server operating system and for the SharePoint 2013 program files. Our host server also needs free hard disk space for diagnostics such as logging, debugging, and creating memory dumps, for daily operations, and for the page file. Normally, 80 GB of disk space is enough for the Windows Server operating system and for the SharePoint 2013 program files.

Another important fact for us is that the speed of the storage affects the search performance. We have to make sure that the storage we have is fast enough to handle the traffic from the search components and databases. The way we decide to distribute search data and operating system files across our storage, has an impact on our search performance. Recommendations are to distribute the base installation of the Windows Server operating system, the SharePoint 2013 program files, and diagnostics across three separate storage volumes or partitions with normal performance. Furthermore it is important to store the search component data on a separate storage volume or partition with high performance. Of course it is possible to do this our own way, but tests have showed us that this will have a negative impact on some components down the road.

Our servers that host the index, analytics processing, and the search administration components, or search databases, require storage that can maintain low latency, while providing sufficient I/O operations per second (IOPS). We have also deployed shared storage like SAN/NAS and the peak disk load of one search component typically coincides with the peak disk load of another search component. To get the number of IOPS search requires from the shared storage, we need to add up the IOPS requirement of each of these components. We started the calculations without these important facts in mind and came up with useless and untrustworthy calculations. Inclusion of these elements gave much better results. Our search architecture will support high availability when we host redundant search components and databases on separate fault domains. That's not the case at this moment and the implications are important to realize.

Enterprise Search with SharePoint 2013

We want to implement an enterprise search system with SharePoint 2013. Enterprise Search systems can index multiple, disparate content repositories, heterogeneous in nature and the typical topology is distributed across the corporate network. These systems are serving a range of user requirements from deep research to simple fact checking, and supporting business-critical processes with customized search capabilities. Finally enterprise search is respecting all document-level security restraints imposed by the originating content repositories involved.

Thanks to the FAST legacy, SharePoint 2013 contains a solid, scalable search engine, capable of coping with very large data sets. When all content that has to be indexed resides within SharePoint, then the native tools provided with SharePoint 2013 work well for us, and in many cases this can be set up and administered by our general IT staff.

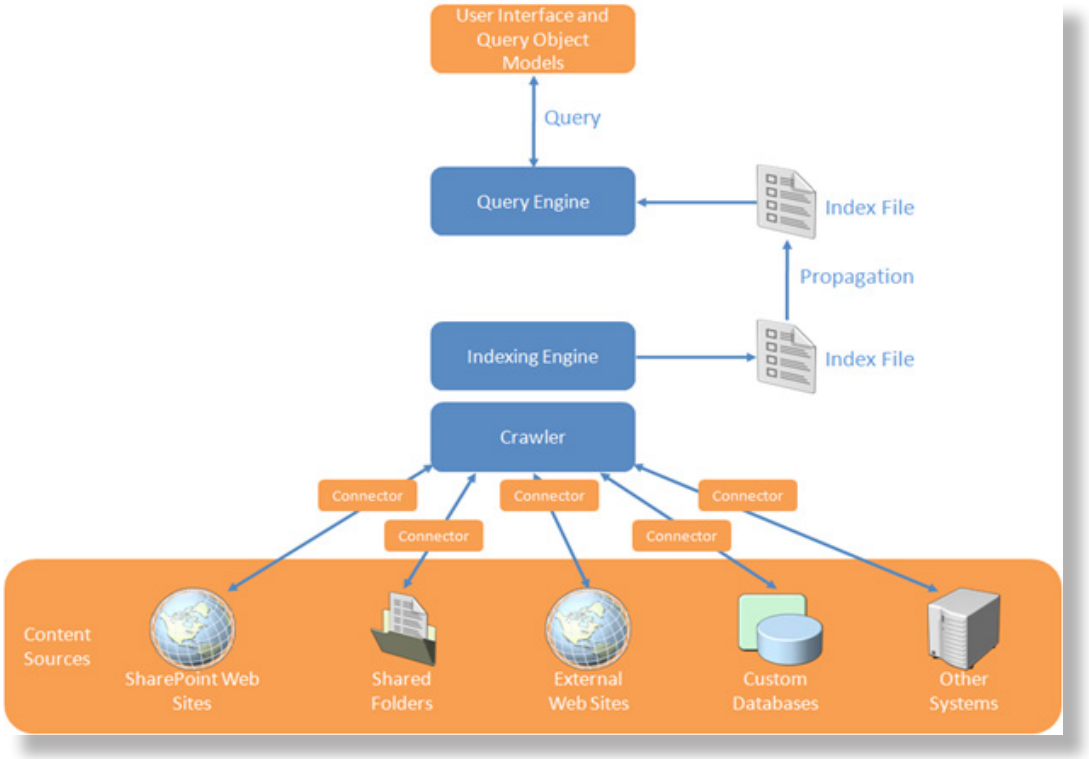

Figure 10: Indexing external data sources using SharePoint 2013 search.

However, our corporate-wide search system will be more complex, because of the need to index content from other content repositories, business systems, and file shares. Also these systems must be capable to normalize and enrich our content, enable search algorithms to sensibly compare and rank disparate document types and formats (and normalize metadata into a consistent schema) and they must provide an efficient search experience to a range of our user types and applications, whose requirements are potentially very diverse. This means that we need to attract people specialized in SharePoint 2013 search and in how to configure external content sources for it. If we do not use specialized people, then we introduce a serious risk to the quality, time, maintainability and usability of our search solution.

SharePoint 2013 Enterprise Search and External content

We face a challenge because we also want to search external content that is not crawled by our search server. For this we used the approach of Federated Search and are forwarding the query to external content repositories where we have it processed by that repository's own search engine. This repository's search engine then returns the result set to our search server. Our search server formats the results and renders it from the external repository for displaying it on the SharePoint search results page. The predefined display rules are then applied. We tried to display the results from the external repositories the same as we display the results from our own search index.

Using this technique has some advantages. First we don't require any additional capacity for the content index, because the content is not crawled by Search in SharePoint 2013. Secondly we can take advantage of the repository's existing search engine. An example could be Trim Records Management version 8 where we could federate to its search engine to search the Records stored in Trim. Another example could be to federate to an internet search engine to search the web. We are also considering to federate search results from Meridian, but this is still in the analysis phase. There are of course other possibilities. We are also looking to define a service bus to handle multiple result sets in a timely and uniform manner.

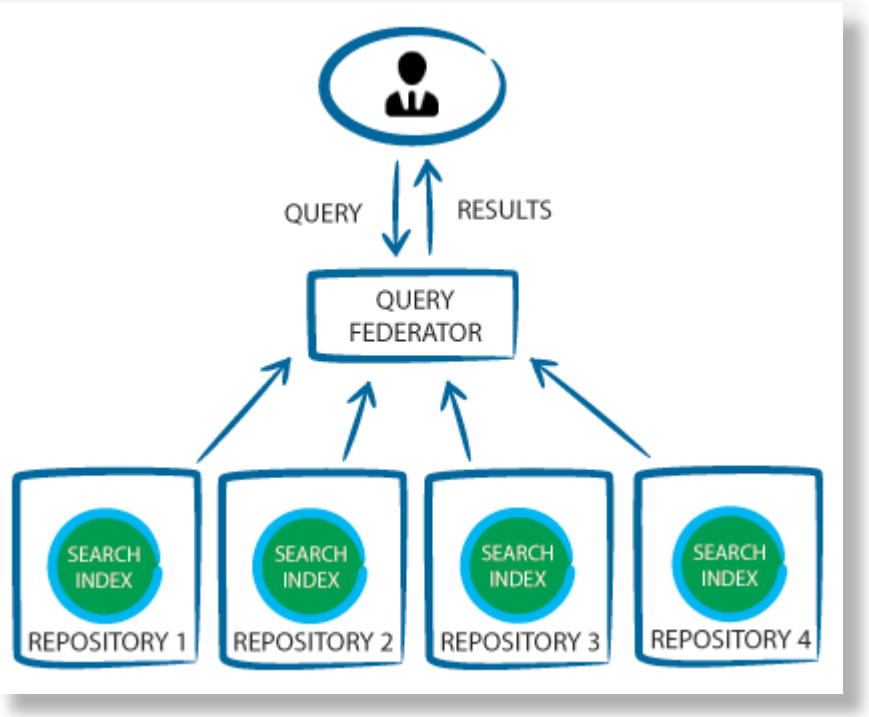

Figure 11: Federating search results.

Federating the search to other repositories also has the advantage of optimizing the content repository's search engine for its specific set of content. This might provide better search performance on the content set. Furthermore, and this is an important one for us when dealing with secured environments, we can access repositories that are secured against crawls but which we can access by search queries.

Finally we also want to use the connector framework in Search. It is the last example in this article. With the connector framework we are able to crawl external data and make it available in search results through BCS indexing connectors. Our BCS indexing connector is used by the crawler to communicate with the external data sources. When the crawl happens, the crawler calls our BCS indexing connectors to fetch the data from the external systems and this data is passed back to the crawler.

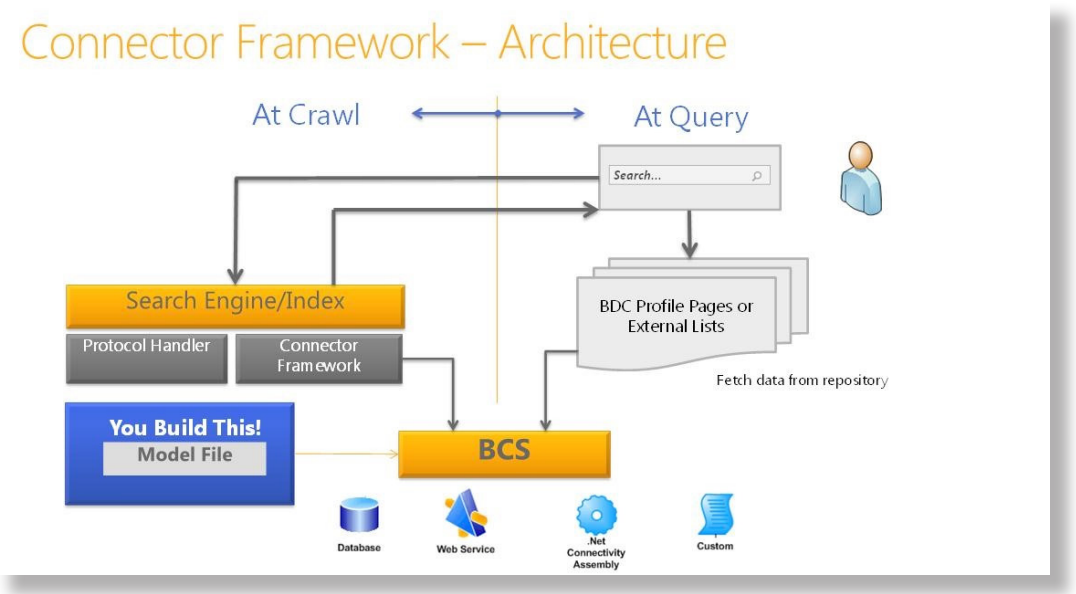

Figure 12: SharePoint 2013 search Connector Framework architecture.

Our BCS indexing connectors are composed through the BDC model files and the connectors. The BDC model files provide the information about the connections to the external systems and the structure of the data. The connectors on the other hand contain the code that connect to the external systems and parses the access and BCS

identifiers. The BDC metadata model offers us several properties that are applicable to Search and many of them are required to support BCS indexing connector crawling. Please turn to the proper information pages at Microsoft to investigate this yourself. There are ways to optimize the BDC model files to improve performance when crawling our external data. We use this but a thorough description of it goes beyond the scope of this article.

Conclusion and road ahead

It is possible to use SharePoint 2013 Search as our Enterprise Search Engine. For us Enterprise Search means searching SharePoint and non-SharePoint data sources and present the results in a uniform and timely manner. Added to this is the way people want to search. The "search feeling" must be the same for all the sources behind the queries. We described the most important elements of this SharePoint 2013 Enterprise Search solution being the architectural possibilities and decisions, the different search components and what they can offer us and how we can include external data sources to our search experience in a uniform and timely manner. Although this article is just the tip of the Search iceberg, we tried to give you a useful foundation to build upon when using SharePoint 2013 as your Enterprise Search solution of the future.

For us it is important to pay even more attention to improving the techniques SharePoint 2013 offers us to include external data sources into our Enterprise Search solution. Not only do we have to make even better decisions about our physical and virtual architectures by constantly measuring different results and interactions between the different architectural elements, but also do we have to improve the Federated Search capabilities and the Connector Framework. Our SharePoint 2013 Enterprise Search solution is very flexible and we can adjust our components if and when we feel we need to. To ensure that our search solution will continue to be able to crawl and query all content in a timely and consistent manner we will need to continuously monitor all components of the solution. Approaching it this way will give us the biggest chance of a successful and stable implementation and happy search customers throughout our organization.

Rest us to say: Happy SharePoint 2013 searching!

Creating a reusable cross site collection navigation in SharePoint

by Radu Tuț

One of the most demanding aspects of SharePoint is navigation. Not only because it has an impact on the way users, developers or consultants transform the platform by using or developing it, but also because of its impact on user adoption, engagement and overall platform performance.

Navigation affects the way users browse through the existing content that – at a fundamental business level – is tightly linked with the way information hosted on the platform is capitalized by employees.

Real world scenarios

Real world scenarios are compatible with more complex SharePoint custom applications (consisting of different site collections and web applications) for which consultants and developers must find solutions that are easily adaptive to a constantly evolving SharePoint portal, complexity-wise.

Related to navigation, this requires a solution that specifically allows a consistent global navigation (for the entire SharePoint portal).

If we go on looking for a solution, out-of-the box features seem to be an option for configuring navigation, but they only allow configuration at the site collection level, and not across many site collections.

We need to find a suitable, reusable solution that will allow us to have a consistent navigation across the entire portal. Using a reusable solution places us in the position of having a different mind-frame in finding the answers for more complex scenarios.

Firstly, reusable components are usually easily pluggable in a new SharePoint environment. The solution does not only solve an important usability issue, but it also solves a common problem that SharePoint specialists have when setting up a SharePoint portal: creating a great solution in as little time as possible. In addition, having reusable components for essential features like navigation allows us to be more flexible and quicker in the overall deployment process.

The context

Configuring navigation depends on distinct elements like quick launch, top navigation and breadcrumbs. Top navigation, for example, is usually consistent throughout the entire portal, while the quick launch may change depending on the given context. Related to users and UI friendliness, there are two aspects that we could consider when enhancing navigation:

- When browsing a SharePoint site, users should know where they are (this is usually achieved by breadcrumbs);
- Users can easily reach to essential access points (landing page of the intranet portal, department site, and so on).

The standard navigation in SharePoint 2010 was configurable by a site administrator, offering a two-level depth, which - considering the complex potential of SharePoint at the structural level, is usually not enough for real SharePoint implementations. With the introduction of the new managed metadata navigation feature in SharePoint 2013, many of the existing issues seemed to be solved: it supports more than two-level depth navigation, easy multilingual configuration and SEO-friendly URLs.

For SharePoint 2010:

- Custom site map provider which retrieves navigation data from:
  - XML file
  - Navigation list
  - Navigation elements of a “central” site collection
- Creating – from scratch - a custom navigation control.

For SharePoint 2013:

- Using a source term set for which you define your own navigation structure.
  - Create a new term set and pin it to the source term set.

    This option may be suitable for a portal with a small number of site collections, but in a large SharePoint portal, with many site collections it will also lead to a large number of duplicated terms in your term store (each site collection will have its own associated navigation term set);
- Creating a custom navigation control;
- The same options as for SharePoint 2010.

The solution? - Navigating through the core

Specifically, the solution for implementing reusable cross-site collection navigation is all about retrieving navigation information, what it looks like is a secondary concern.

The main levers we can use in order to control this are:

1. Using the term store to get the navigation for all existing site collections.
2. Developing a site map provider which reads all terms from a term set, and delivers them as navigation nodes.

   This allows us to reuse previously gained knowledge (SharePoint 2010), but also to integrate the new features (SharePoint 2013) in the concept.
3. This solution can be installed in a new SharePoint environment simply by deploying it in Central Administration, activating a feature at the web application level, and configuring the term set that should be used by the navigation provider.

Final words

Creating a reusable solution isn’t only effective because it saves time, but it also allows for a high quality SharePoint solution to be delivered. Creating and implementing reusable solutions can have the effect of setting new standards of performance related to all SharePoint sites we set up. In our case, consistent global navigation supports adding more portal sites while assuring a user-friendly browsing experience.

About the Authors

Implementing eDiscovery in SharePoint 2013

My name is Maarten Eekels and I am CTO at Portiva, one of the larger SharePoint implementation partners in the Netherlands. In that role I am responsible for all knowledge and technology related matters, and for our entire project delivery business.
Already since version 2003 I am involved in implementing SharePoint in areas like web content management, document management, knowledge management, business process support, business intelligence, social and search.
As a speaker I have done presentations about various SharePoint related topics on SharePoint Connections, the SP24 conference, SharePoint Saturdays, DIWUG user group meetings, and SharePoint & Sushi events.
Company: Portiva

Migrating Term Sets

Octavie van Haaften is a driven and passionate SharePoint technical consultant at Mavention. I work both on cloud (IaaS and Office365) and on-premises solutions. Usually architecting and developing solutions with other Mavens. I share my experiences on my blog http://blog.octavie.nl and like to discuss on Twitter and Facebook.
Company: Mavention
Blog: http://blog.octavie.nl

BCS Kerberos Delegation

Jasper Siegmund is a SharePoint Solution Architect working for Atos in the Netherlands. Jaspers' goal in his (working) life is: making things simpler. Utilizing SharePoint, he advises enterprises on how to get the most out of what they usually already have. Being a developer by heart, he does this by trying to find the perfect mix between out of the box and custom software to do so.
Company: Atos

How to organize meetings in SharePoint 2013

Frank op 't Landt works for Macaw Workplace Solutions as SharePoint Consultant. His focus is on functional level and he likes to help the business and end user to a practical solution. Important are usability, accessibility and analytics.
Company: Macaw Workplace Solutions

MCMS2002 Migration to a SharePoint 2013 metadata driven environment

Vincent Verbeek is a SharePoint Lead Developer and Consultant at Conclusion ICT Projects. Working with and leading a development team of around 10 people, he has been helping companies implement their SharePoint intranet and extranet solutions since 2010. He is very passionate about helping companies get the most out of their SharePoint implementation, either by cleverly combining SharePoint out-of-the-box functionalities or through custom coding.
Company: Conclusion ICT Projects

State of SharePoint Statistics

Nikander Bruggeman is an experienced web developer who started out working with the very first versions of HTML, JavaScript, Java, and later .NET and SharePoint. In the past, Nikander has been given the prestigious SharePoint Most Valuable Professional (MVP) award three times. He has worked on 13 SharePoint books as either author, co-author or contributing author for publishers such as MS Press and APress. Nikander has written .NET/SharePoint articles for magazines such as .NET magazine, XML Developer, ASPToday, and Smart Access. Nikander has presented at conferences such as TechNet and HeliView and is a technical reviewer and technical proofreader for Manning. Currently, Nikander works freelance as co-founder of Lois & Clark IT Services (http://www.loisandclark.eu) working on a variety of SharePoint/.NET related projects.

Margriet Bruggeman is a software development enthusiast, has been working in the industry for over 15 years, and was among the first 11 people world-wide ever to receive the SharePoint MVP award. Since then, Margriet has worked on 13 SharePoint books as either author, co-author or contributing author for publishers such as MS Press and APress. Margriet has written dozens of SharePoint articles, has contributed to SharePoint courseware, has presented at conferences such as TechNet, and is a technical reviewer/ proofreader for Manning. You may also know Margriet from the TechNet forums, where Margriet tries to help people struggling with SharePoint questions. Margriet is a Forums Moderator, Wiki Ninja and one of the six non-MS members of the TechNet Wiki Community Council. Margriet works freelance as co-founder of Lois & Clark IT Services (http://www.loisandclark.eu) working on SharePoint, SharePoint and SharePoint. Margriet is a current SharePoint MVP.

Company: Lois & Clark IT Services
Web: http://www.loisandclark.eu

Using ASP.Net SignalR within your SharePoint apps

Bas Lijten is a SharePoint Architect working for Achmea, a leading insurance company based in the Netherlands. As a SharePoint Architect, he works on both the internet facing WCM sites, as well as on the intranet solutions. Because of his experience with the product, he is a valuable source for all of his colleagues. Bas has a special interest in Search, Windows Phone, BI, SharePoint Apps and ALM and speaks about these subjects on community events.

Company: Achmea
Blog: http://blog.baslijten.com

User generated metadata

Albert Hoitingh is senior consultant working for the Microsoft business unit at Sogeti Netherlands. He's been working with information worker solutions from the late 1990's, starting with IBM Lotus Notes. He has extensive experience with Microsoft SharePoint (2001-2013) as-well-as other Microsoft platforms.

Albert enjoys inspiring organizations as a SharePoint evangelist and works closely with business users to successfully implement SharePoint solutions. He's presented at SharePoint Connections in Seattle and in Amsterdam as well as the Sogeti SharePoint events.

Company: Sogeti
Blog: alberthoitingh.wordpress.com.

SharePoint 2013 and Enterprise Search

Jaap Zwart is an independent IT Professional and specialized in SharePoint implementations. His focus is besides the technical side of the product more academic towards leading innovation and change around SharePoint implementations. With a profound technical background Jaap tries to combine this with the academic aspects to align all the layers within an organization when implementing SharePoint. This is done to avoid a mismatch between the operational, tactical and strategic layers when SharePoint is used because all layers will eventually be influenced. This mismatch is often the reason why SharePoint implementations fail.
Company: Micro-Touch & Consult

Creating a reusable cross site collection navigation in SharePoint

Radu Tuț is a SharePoint Consultant at Accesa, Romania. With more than 3 years of experience in working with Microsoft technologies, he is a SharePoint 2010 MCTS and MCDP. His interests related to SharePoint are Web Content Management and Search.
Company: Accesa

Notes & Remarks

Notes & Remarks

Technology

People

Fun

SharePoint

Kom werken aan uitdagende SharePoint projecten!

avanade®
Results Realized

Passion

Connection

Heb jij passie voor Microsoft-technologie en ben je een echte SharePoint expert?
Bekijk jouw nieuwe uitdaging op werkenbijavanade.nl en neem contact op met Duygu Ciftci of Nicole Holla via nl.recruitment@avanade.com of 036 547 51 07.

avanade®
Results Realized

From Accenture and Microsoft