Module 2: Search customizations for SharePoint 2013 and SharePoint Online

# Introduction

This module describes search customization scenarios and sample search apps in SharePoint 2013 and SharePoint Online, including:

* Search-based site directory
* Personalized search
* Branding with display templates
* Export/import customized search configuration settings.

# Search-based site directory

SharePoint search enables you to create a search-based site directory without having to write any custom code. To create the site directory you need to complete these steps:

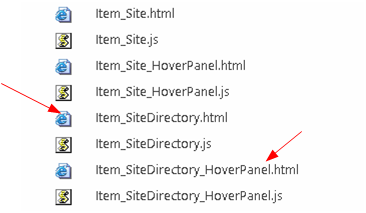
* Create the site directory display templates
* Define the site directory result type
* Create the results page
* Edit the Results web part properties.

## Create the site directory display templates

This example uses the Site related display templates with very little modification. If you wanted to change how the site directory results are displayed, you would make additional modifications to the display templates created in this step.

1. Open the mapped network drive to the **Master Page Gallery**. See [How to: Map a network drive to the SharePoint 2013 Master Page Gallery](http://msdn.microsoft.com/en-us/library/office/jj733519(v=office.15).aspx).
2. Make copies of the display template html files that most closely map what you are trying to do. For the site directory scenario, this will be **Item\_Site.html** and **Item\_Site\_HoverPanel.html**, located in the **\Display Templates\Search** folder in the mapped network drive.
3. Rename the copies **Item\_SiteDirectory.html** and **Item\_SiteDirectory\_HoverPanel.html**, as shown in figure 1.

**Figure 1. Site directory display templates**



1. Open the these files in an HTML editor, and make the following changes –
   * Item\_SiteDirectory.html
2. Change the **title** tag from:

<title>Site Item</title>

to:

<title>Site Directory</title>

1. Change the first **div** tag after the opening **body** tag from:

<div id="Item\_Site">

to:

<div id="Item\_SiteDirectory">

1. Change the hover panel display template js file name from:

var hoverUrl = "~sitecollection/\_catalogs/masterpage/Display Templates/Search/Item\_Site\_HoverPanel.js";

to:

var hoverUrl = "~sitecollection/\_catalogs/masterpage/Display Templates/Search/Item\_SiteDirectory\_HoverPanel.js";

* Item\_SiteDirectory\_HoverPanel.html

1. Change the **title** tag from:

<title>Site Hover Panel Test</title>

to:

<title>Site Directory Hover Panel</title>

1. Change the first **div** tag after the opening **body** tag from:

<div id="Item\_Site\_HoverPanel">

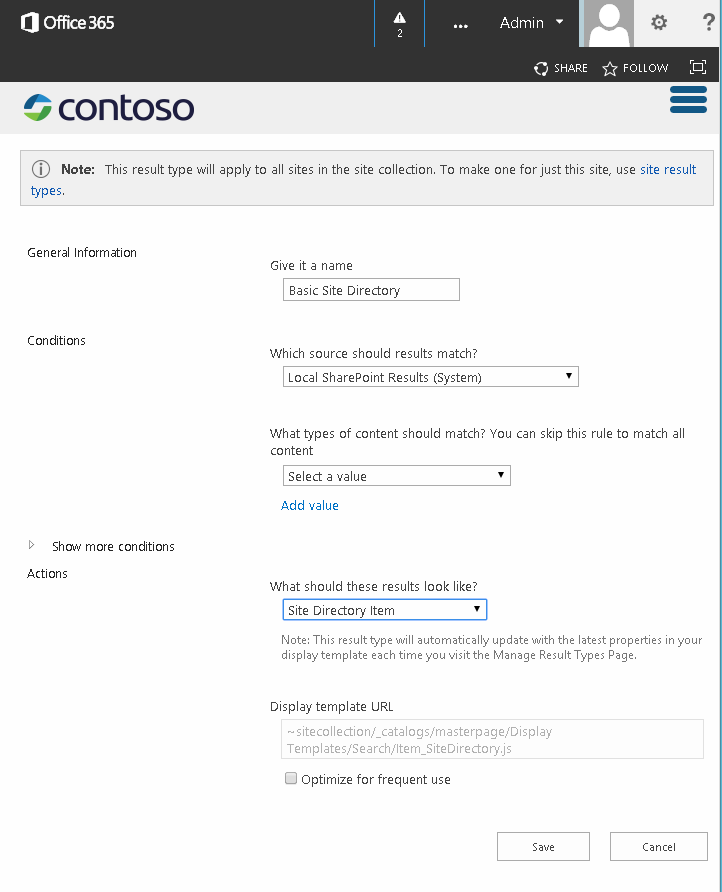
to:

<div id="Item\_SiteDirectory\_HoverPanel">

## Create the site directory result type

1. Go to **Site Settings** -> **Search** -> **Result Types**.
2. Click **New Result Type**.
3. Name it **Basic Site Directory**.
4. Select **Local SharePoint Results** for **Which source should results match?**
5. In the **Actions** section, select the **Site Directory** display template for **What should these results look like?**
6. Click **Save**.

**Figure 2. Basic Site Result configuration**



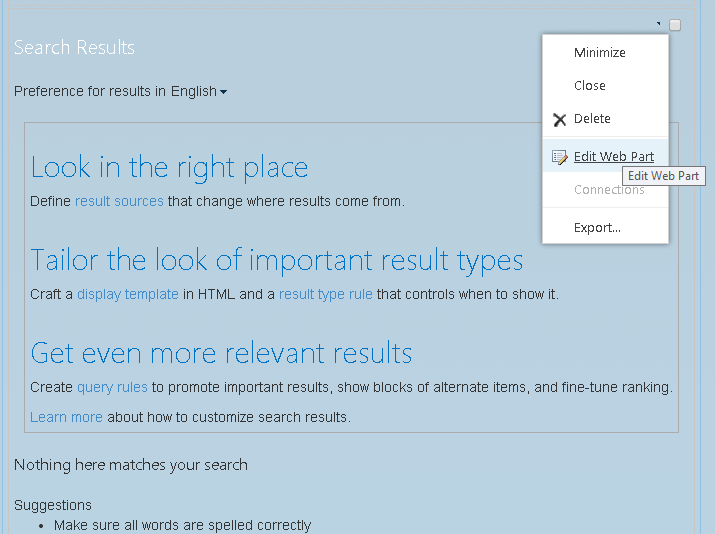
## Create the site directory results page

1. From the **Site Settings** menu, select **Site contents**.
2. Select **Pages**.
3. In the **Pages** library, select the **FILES** tab --> **New Document** --> **Page**.
4. On the **Create Page** page, specify **Site Directory** for **Title** and **sitedirectory** for **URL Name**.
5. Click **Create**.

## Edit the Results web part

1. On the **Site Directory** page click the **Settings** menu --> **Edit Page**.
2. In the Search Results web part on the page, click the web part menu, and then click **Edit Web Part**, as shown in figure 3.

**Figure 3. Web part menu.**

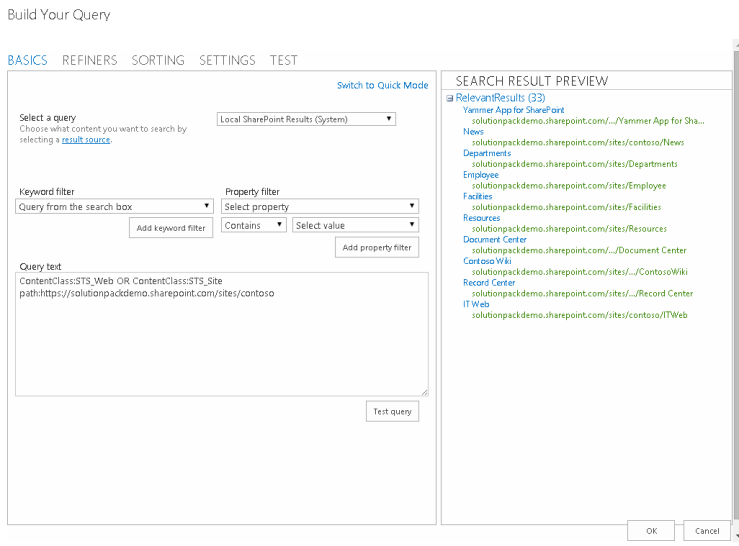


1. In the Web Part tool pane, click **Change query** to open the Query Builder.
2. In **Query text**, enter the following:

ContentClass:STS\_Web OR ContentClass:STS\_Site path:http://*<YourServer>*

1. Click **Test query** to confirm that the syntax is correct. The **Search Results Preview** pane should display subsites within the *<YourServer>* site you specified for path in the **Query text**.

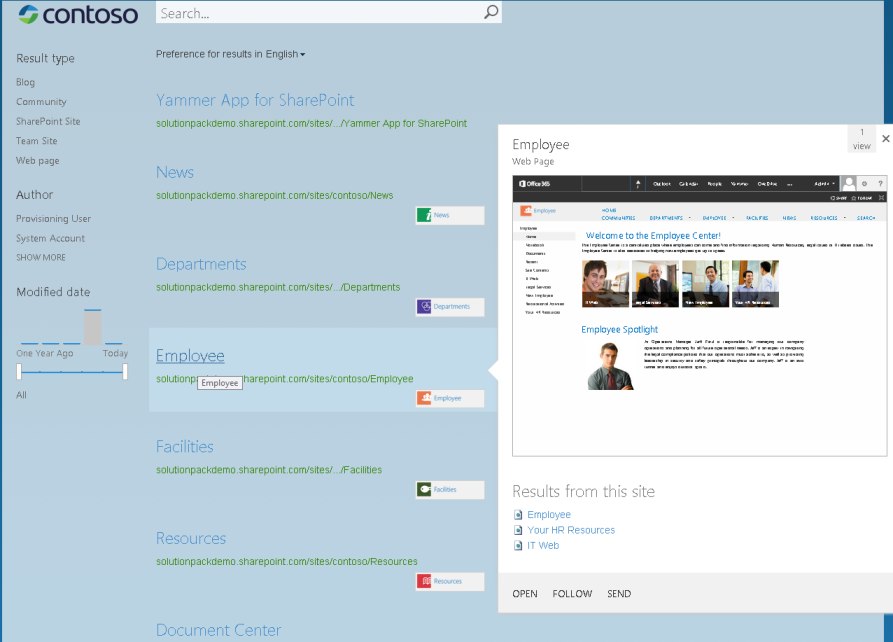
**Figure 4. Search results web part query builder**



1. Click **OK** to close the Query Builder
2. In **Display Templates**, select **Use result types to display items**.
3. Select **Basic Site Directory** from the **Result type for item** dropdown.
4. In the **Appearance** section, change the **Title** to **Sites I have access to**.
5. Click **OK** to save the changes to the web part and close the web part tool pane.

Figure 5 shows an example of a search-based site directory page created using these steps.

**Figure 5. Contoso search-based site directory example**



## Showing “Sites I Own”

With a simple modification, you can also display sites the user owns on the Site Directory page. To do this, you add a second result web part to this page, with the same configuration as the **Sites I have access to** results web part you customized in the previous section, with the addition of a filter on the author property to the query. To do this, change the Query text specified in step 4 in the previous section from:

ContentClass:STS\_Web OR ContentClass:STS\_Site path:http://*<YourServer>*

to:

ContentClass:STS\_Web OR ContentClass:STS\_Site path:http://*<YourServer>* Author:{User.Name}

This property filter and value means that the query will return only sites that are owned by the user browsing the page.

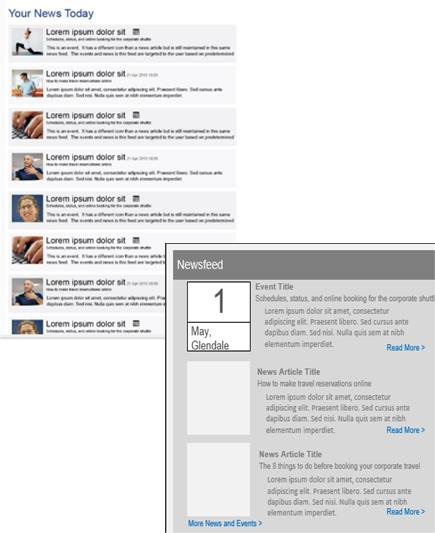
# Personalized search results

Personalized search is when you show search results targeted to the user submitting the search; the results are personalized in some way. This section describes some scenario examples for personalized search and how you might implement them.

## Your News

In this scenario you create a search app that shows relevant content targeted to the user such as news or news and events combined.

**Figure 6. Your News personalized search scenario**



To implement the news scenario, you could use the SharePoint search results web part and default display templates to display the news information with title, description and rollup image, and then show the first ten news items. When the user clicks the rollup image, title, or **Read More** link, the news article page is loaded.

Alternatively you could create a search app and using the query API (CSOM or REST) you could also do the same thing. You could make the number of news items to be displayed configurable through the search app properties.

Another option using the query API, but without requiring a search app would be to add the query API code that retrieves the search results directly to the page layout.

To display the news and event information specific to the user you would:

* Modify the query to filter news and event results based on user profile properties like business unit, region and language.
* Retrieve the Title, Description, rollup image, and URL properties for the news or event items.
* You could then implement sorting logic for the combined news and events based on the **LastModifiedDate** property.

## Upcoming Events

In this scenario, the search app shows relevant events target to the user.

**Figure 7. Upcoming Events personalized search scenario**



To implement this, you could configure the SharePoint search results web part to change the query to only retrieve upcoming event information. For this, you could specify **ContentClass:STS\_ListItem\_Events** for the web part's query text.

To change how the event results are displayed, you would create custom display templates to render the event information.

You could modify the item display template so that when the user clicks the image, title, or **Read More** link, the event information page is loaded.

You could modify the control display template so that when the user clicks **See More**, the next ten event results are displayed in the web part.

In this scenario, you could also do the same thing by creating a search app that uses the query API to retrieve even results. You could configure the search app to show only the latest ten upcoming events by default, but make this setting configurable through the search app properties.

Whether you choose to configure the out-of-the-box search web parts or create a search app for scenarios like these depends on the complexity required, and also configuration changes needed.

## Featured News

In this scenario the search app shows search results as featured content targeted to your users in places such as corporate intranet and divisional landing pages.

You could implement this with an app part that contains a jQuery plugin with HTML, that uses the search REST service or the query CSOM to get search results from SharePoint and writes out the HTML to display the results.

## Personalized search results code sample

The [SharePoint 2013: Personalizing search results in an app for SharePoint](http://code.msdn.microsoft.com/SharePoint-2013-Personalizi-fb6ddcf9) code sample demonstrates both a basic search example as well as a personalized search results example using the search query CSOM.

The basic search example allows the user to provide a search filter to be used for a tenant-wide search and is looking for sites that apply to the user-supplied filter.

The example code starts by getting SharePoint context using the **SharePointContextProvider** class:

var spContext = SharePointContextProvider.Current.GetSharePointContext(Context);

(See [Announcing the new SharePointContext helper in Apps for SharePoint 2013](http://blogs.msdn.com/b/officeapps/archive/2013/11/07/announcing-the-new-sharepointcontext-helper-in-apps-for-sharepoint-2013.aspx) for more information about this class.)

Next it builds the query based on what the user entered and also restricts the query to site collections, and then calls the **ProcessQuery** method, passing the context and the query in the method call, and returns the **ProcessQuery** results as a result table, which is then parsed by the **FormatResults** method:

using (var clientContext = spContext.CreateUserClientContextForSPHost())

{

string query = searchtext.Text + " contentclass:\"STS\_Site\"";

ClientResult<ResultTableCollection> results = ProcessQuery(clientContext, query);

lblStatus1.Text = FormatResults(results);

}

The **ProcessQuery** method builds a [**KeywordQuery**](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.search.query.keywordquery(v=office.15).aspx) object representing the search query:

KeywordQuery keywordQuery = new KeywordQuery(ctx);

keywordQuery.QueryText = keywordQueryValue;

keywordQuery.RowLimit = 500;

keywordQuery.StartRow = 0;

keywordQuery.SelectProperties.Add("Title");

keywordQuery.SelectProperties.Add("SPSiteUrl");

keywordQuery.SelectProperties.Add("Description");

keywordQuery.SelectProperties.Add("WebTemplate");

keywordQuery.SortList.Add("SPSiteUrl", Microsoft.SharePoint.Client.Search.Query.SortDirection.Ascending);

The search query is then submitted to SharePoint by calling the [**ExecuteQuery**](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.search.query.searchexecutor.executequery(v=office.15).aspx) method of the [**SearchExecutor**](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.search.query.searchexecutor(v=office.15).aspx) class. Results are returned to the [**ClientResult**](http://msdn.microsoft.com/en-us/library/office/ee545298(v=office.15).aspx) object as shown below:

SearchExecutor searchExec = new SearchExecutor(ctx);

ClientResult<ResultTableCollection> results = searchExec.ExecuteQuery(keywordQuery);

ctx.ExecuteQuery();

The **FormatResults** method iterates through the results and constructs an HTML table to display the result values:

string responseHtml = "<h3>Results</h3>";

responseHtml += "<table>";

responseHtml += "<tr><th>Title</th><th>Site URL</th><th>Description</th><th>Template</th></tr>";

if (results.Value[0].RowCount > 0)

{

foreach (var row in results.Value[0].ResultRows)

{

responseHtml += "<tr>";

responseHtml += string.Format("<td>{0}</td>", row["Title"] != null ? row["Title"].ToString() : "");

responseHtml += string.Format("<td>{0}</td>", row["SPSiteUrl"] != null ? row["SPSiteUrl"].ToString() : "");

responseHtml += string.Format("<td>{0}</td>", row["Description"] != null ? row["Description"].ToString() : "");

responseHtml += string.Format("<td>{0}</td>", row["WebTemplate"] != null ? row["WebTemplate"].ToString() : "");

responseHtml += "</tr>";

}

}

responseHtml += "</table>";

The personalized search results example loads your user profile properties and passes the **AboutMe** profile property to the **ResolveAdditionalFilter** method:

PeopleManager peopleManager = new PeopleManager(clientContext);

PersonProperties personProperties = peopleManager.GetMyProperties();

clientContext.Load(personProperties);

clientContext.ExecuteQuery();

string aboutMeValue = personProperties.UserProfileProperties["AboutMe"];

string templateFilter = ResolveAdditionalFilter(aboutMeValue);

The **ResolveAdditionalFilter** method checks for “Apptest”. If it is found, a list of site templates of any type is returned in the search results. If it is not found, only STS web templates are returned in the search results.

private string ResolveAdditionalFilter(string aboutMeValue)

{

if (!aboutMeValue.Contains("AppTest"))

{

return "WebTemplate=STS";

}

return "";

}

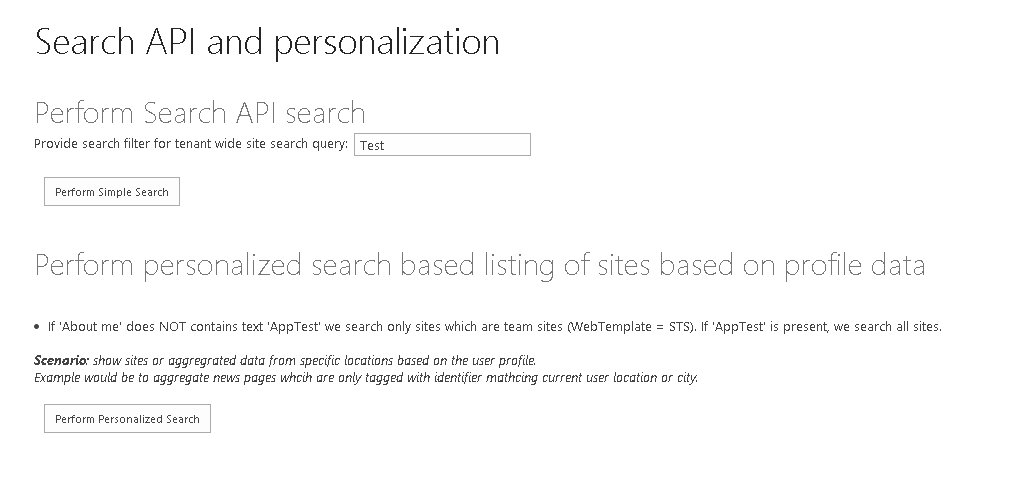
The personalized search results example code then constructs the query and calls the **ProcessQuery** and **FormatResults** methods to retrieve, format, and display the search results the same as the basic search example:

string query = "contentclass:\"STS\_Site\" " + templateFilter;

ClientResult<ResultTableCollection> results = ProcessQuery(clientContext, query);

lblStatus2.Text = FormatResults(results);

**Figure 8. Personalized search results sample UI**



# Search configuration portability

In SharePoint 2013 and SharePoint online you can export and import customized search configuration settings between site collections and sites.

For customized search configuration settings at the SSA level, you can only export these settings, and you must use the search APIs to do this programmatically, as the export option is not available using the SharePoint UI.

The [SharePoint 2013: Import and Export search settings for SharePoint Online](http://code.msdn.microsoft.com/SharePoint-2013-Import-and-6287b5ac) sample demonstrates how to import and export search settings for a SharePoint Online site using the search CSOM in a console application.

## Search configuration settings that can be exported and imported

When you export customized search configuration settings, SharePoint 2013 creates a search configuration file in XML format. This search configuration file includes all exportable customized search configuration settings at the SSA, site collection, or site level from where you start the export. A search configuration file for a site collection does not contain search configuration settings from the individual sites within the site collection.

When you import a search configuration file, SharePoint 2013 creates and enables each customized search configuration setting in the site collection or site from where you start the import.

Table 1 shows the settings that you can export and import. For each setting, the table indicates any dependencies on other customized search configuration settings. If the customized search configuration settings depend on a customized search configuration setting at a different level, for example, if a site query rule depends on a result source at site collection level, you must export and import settings at all of the relevant levels.

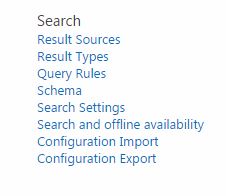
**Table 1. Search settings you can import and export**

|  |  |
| --- | --- |
| **Customized search configuration setting** | **Dependency on other customized search configuration settings** |
| Query rules. These include result blocks, promoted results, and user segments. | Result sources, result types, search schema, ranking model |
| Result sources | Search schema |
| Result types | Search schema, result sources, display templates |
| Search schema | None |
| Ranking model | Search schema |

It is also possible to export customized search configuration settings from a Search service application (SSA) and import the settings to site collections and sites, but you cannot import customized search configuration settings to an SSA. You also cannot export the default search configuration settings.

At the site or site collection level, you can export or import search configuration settings using the SharePoint user interface (UI). These settings are located in the **Search** section of the **Site Settings** page, as shown in figure 9.

**Figure 9. Site Settings - Search**



They are also available in the **Site Collection Administration** section.

Alternatively, you can programmatically import and export these settings using the SharePoint 2013 search CSOM.

## Search configuration files

The search configuration file XML format is described in the [Share Point search settings portability schemas](http://msdn.microsoft.com/en-us/library/office/dn627953(v=office.15).aspx). The schemas available are listed in table 2.

**Table 2. Search settings portability schemas**

|  |  |
| --- | --- |
| **Schema** | **Description** |
| [SPS15XSDSearchSet1](http://msdn.microsoft.com/en-us/library/office/dn639116(v=office.15).aspx) | Specifies XML that represents result sources. |
| [SPS15XSDSearchSet2](http://msdn.microsoft.com/en-us/library/office/dn639118(v=office.15).aspx) | Specifies XML that represents administrative types and members for managing an SSA's search instance; this includes settings like result item types, property rules, etc.... |
| [SPS15XSDSearchSet3](http://msdn.microsoft.com/en-us/library/office/dn639120(v=office.15).aspx) | Specifies XML that represents settings which include query rules, result sources, managed properties, crawled properties, and ranking models. |
| [SPS15XSDSearchSet4](http://msdn.microsoft.com/en-us/library/office/dn639117(v=office.15).aspx) | Specifies XML that represents enumerations used in other schemas. |
| [SPS15XSDSearchSet5](http://msdn.microsoft.com/en-us/library/office/dn639119(v=office.15).aspx) | Specifies XML that represents enumerations like ResultType which are used in other schemas. |
| [SPS15XSDSearchSet6](http://msdn.microsoft.com/en-us/library/office/dn639115(v=office.15).aspx) | Specifies XML that represents enumerations used in the [Microsoft.Office.Server.Search.Administration](http://msdn.microsoft.com/en-us/library/office/microsoft.office.server.search.administration(v=office.15).aspx) schema. |

You can download the schemas from <http://download.microsoft.com/download/1/2/2/12204CDE-56A6-4B2F-9719-4EA25FDA7743/SP15_search_settings_portability_schema.zip>.

## Exporting and importing search configuration settings using the search CSOM

The [SearchConfigurationPortability](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.search.portability.searchconfigurationportability(v=office.15).aspx) class in the [Microsoft.SharePoint.Client.Search.Portability](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.search.portability(v=office.15).aspx) namespace contains the methods that can be used to export or import search configuration settings using the search CSOM.

The following code shows how to export a site’s search configuration settings:

private static void ExportSearchSettings(ClientContext context, string settingsFile)

{

SearchConfigurationPortability sconfig = new SearchConfigurationPortability(context);

SearchObjectOwner owner = new SearchObjectOwner(context, SearchObjectLevel.SPWeb);

ClientResult<string> configresults = sconfig.ExportSearchConfiguration(owner);

context.ExecuteQuery();

string results = configresults.Value;

System.IO.File.WriteAllText(settingsFile, results);

}

The following code shows how to import a site’s search configuration settings:

private static void ImportSearchSettings(ClientContext context, string settingsFile)

{

SearchConfigurationPortability sconfig = new SearchConfigurationPortability(context);

SearchObjectOwner owner = new SearchObjectOwner(context, SearchObjectLevel.SPWeb);

sconfig.ImportSearchConfiguration(owner, System.IO.File.ReadAllText(settingsFile));

context.ExecuteQuery();

}

You can download a code sample demonstrating search configuration portability using the CSOM from

[SharePoint 2013: Import and Export search settings for SharePoint Online](http://code.msdn.microsoft.com/SharePoint-2013-Import-and-6287b5ac).