


Indexing SharePoint Site Collections Using the dtSearch Engine DataSource API

Bruno Terkaly, 30 May 2017

This article demonstrates using the dtSearch Engine as a solution for searching such data.

Editorial Note

This article is in the Product Showcase section for our sponsors at CodeProject. These articles are intended to provide you with information on products and services that we consider useful and of value to developers.

 **Download demo - 14.9 KB**

Business use SharePoint to archive and store a wide variety of enterprise documents. Large organizations typically have multiple business units. Each business unit generally creates one or more SharePoint Site Collections to organize its business documents and related content.

The ability to quickly find key business information in SharePoint is a significant business differentiator. However, searching across SharePoint documents to extract the necessary information poses challenges, both because of the potential volume of data and also because of the variety of possible file formats. This article demonstrates using the dtSearch Engine as a solution for searching such data.

dtSearch Engine Overview

The dtSearch Engine (see dtsearch.com) can perform lightning fast text searching across terabytes of data. And it does so for a wide variety of data formats. Supported data types include MS Office through current versions (Word, Excel, PowerPoint, Access, OneNote), other "Office" suite formats; compression formats (RAR, ZIP, GZIP, TAR); Exchange, Outlook and other popular email types (including multi-level nested attachments); PDF and PDF Portfolio; other web-based formats (HTML, XML/XSL, etc.); and a wide variety of databases (SQL, NoSQL, SharePoint, etc.).

There is no need to tell dtSearch what file format or other data type it is working with – dtSearch figures that out for itself. And dtSearch lets developers leverage over 25 full-text and metadata search options, including faceted searching and multiple advanced data classification options, as well as displaying search results with highlighted hits.

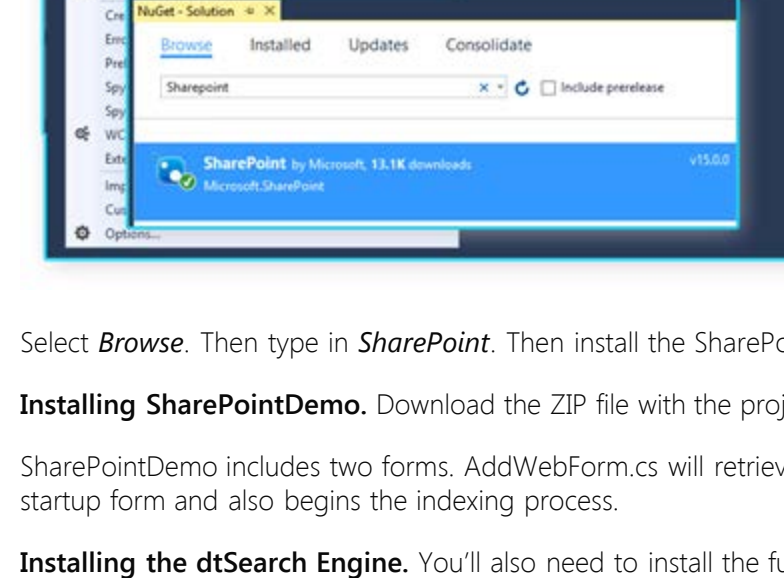
dtSearch products can index SharePoint data in two general ways. The first way is through the dtSearch Spider, built into the dtSearch product line and also available through the dtSearch Engine SDK. The second approach is through the dtSearch Engine DataSource API.

The dtSearch Engine DataSource API offers maximum flexibility for database content integration. This article's SharePointDemo is a Visual Studio Project that uses the dtSearch Engine DataSource API for indexing SharePoint Site Collections.

Getting Started

Central Administration Server Requirements. Every SharePoint Farm has one or more servers in the cluster designated as the Central Administration Server. This server acts as a front-end to the SharePoint Farm. All of the main work in this article occurs on the Central Administration Server. You will of course need admin rights to this server for most of this work.

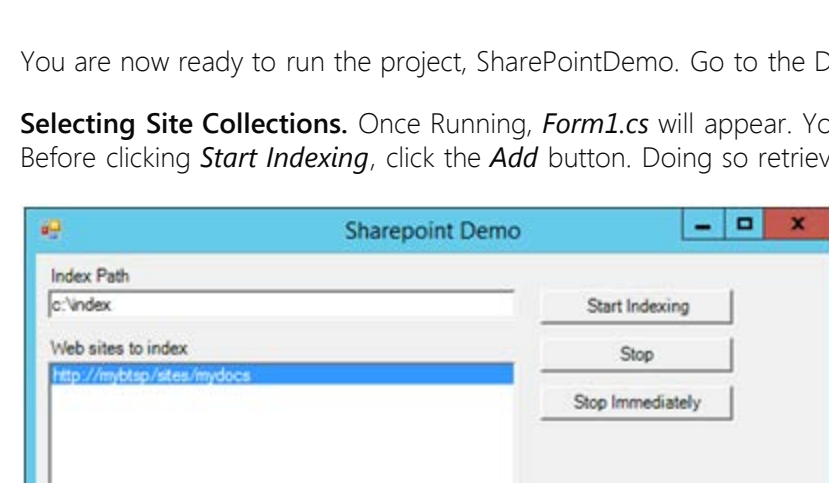
The integration has two categories of work: (1) using the SharePoint API to crawl SharePoint web sites and to enumerate documents to index in each site, and (2) implementing the dtSearch Engine's DataSource interface so the documents can be passed to the dtSearch Engine for indexing.



Visual Studio and Windows Server (64-Bit) Installation. After verifying that you have the appropriate admin rights, install the current version of Visual Studio and the current 64-bit version of Windows Server including patches and OS drivers on the Central Administration Server.

Installing SharePoint DLLs. The SharePoint DLLs in this project require the Windows Server 64-bit operating system. To simplify the installation of the SharePoint DLLs, you can leverage the NuGet capability in Visual Studio. NuGet is available through Visual Studio menu selections.

Select Tools -> NuGet Package Manager -> Manage NuGet Packages for Solution...



Select **Browse**. Then type in **SharePoint**. Then install the SharePoint DLLs.

Installing SharePointDemo. Download the ZIP file with the project files from the link attached to this article at CodeProject.com.

SharePointDemo includes two forms. AddWebForm.cs will retrieve all the Site Collections within the SharePoint Farm. Form1.cs is the startup form and also begins the indexing process.

Installing the dtSearch Engine. You'll also need to install the full dtSearch Engine for Win & .NET package. You can request a fully-functional 30-day evaluation version at dtsearch.com/contact.html. Your job as a developer is to inherit from the DataSource API and use method override for **GetNextDoc()** and **Rewind()**. More details below.

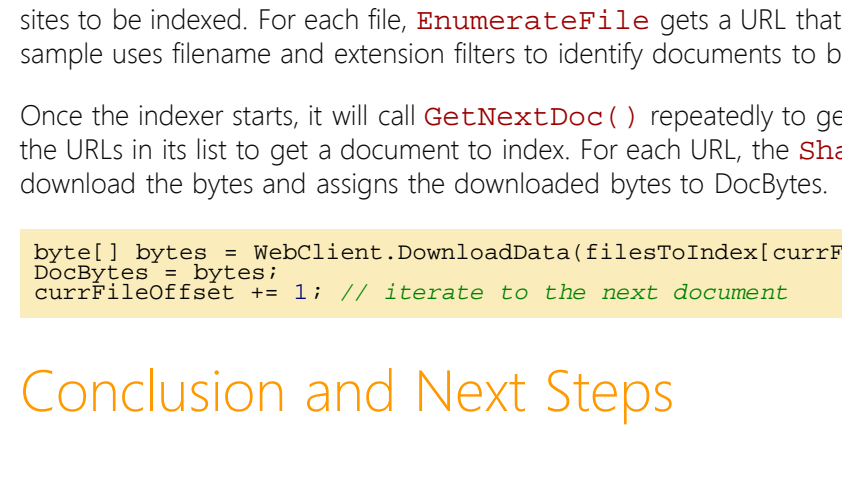
The dtSearch Engine's default installation directory is **C:\Program Files (x86)\dtSearch Developer\bin64**. (Note that bin64 indicates 64-bit.) This article's SharePointDemo already includes References to the dtSearch Engine default installation location. If you install the dtSearch Engine in a different directory, right click on the References section of the SharePointDemo to select a different location.

Verifying SharePoint Site Collections. At this point, you'll need to verify you have a SharePoint Farm with Site Collection documents to index. Appendix A at the end of this paper walks you through the creation of one or more Site Collections including the addition of documents, if this is not already part of your SharePoint setup. Note that everything in SharePoint is a web resource represented by a URL; physical file paths are not available through SharePoint.

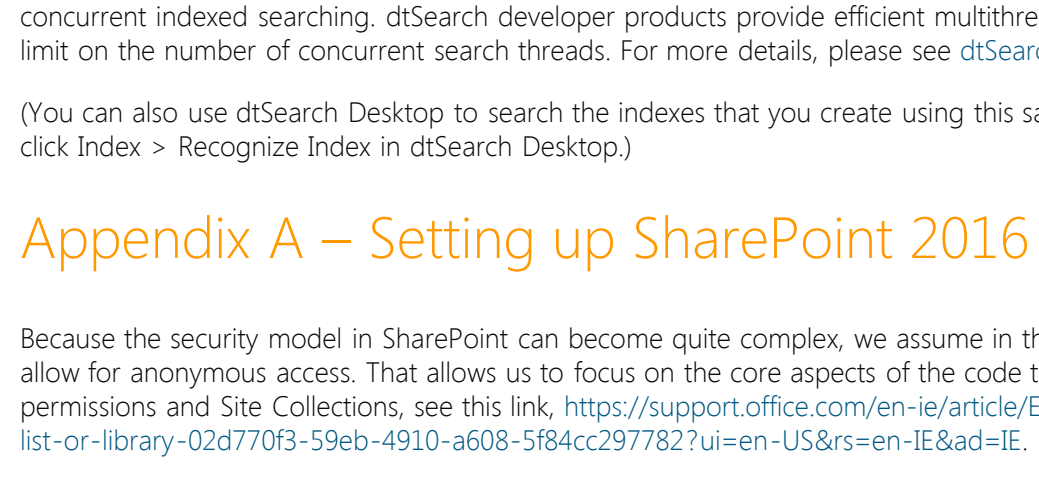
Running SharePoint Demo

You are now ready to run the project, SharePointDemo. Go to the Debug Menu and select **Run**.

Selecting Site Collections. Once Running, **Form1.cs** will appear. You will notice that the application defaults to Index Path **c:\index**. Before clicking **Start Indexing**, click the **Add** button. Doing so retrieves **AddWebForms**.



AddWebForm.cs includes a ListBox along with a button, **FindSharePoint Sites**. This button will look for Site Collections. The sample below includes a couple of Site Collections.



The code that populates the ListBox contains a loop within a loop. SharePoint Site Collections are children of SharePoint Web Services, necessitating the nested loops.

```
private void FindSharepointSitesButton_Click(object sender, EventArgs e)
{
    SPFarm farm = SPFarm.Local;
    //Get all SharePoint web services
    SPWebService service = farm.Services.GetValue(SPWebService "");
    foreach (SPWebApplication webapp in service.WebApplications)
    {
        foreach (SPSite site in webapp.Sites)
        {
            this.WebSiteListBox.Items.Add(site.Url);
        }
    }
}
```

After selecting one or more Site Collections to index in **AddWebForm.cs**, return to **Form1.cs**.

Indexing. Next, select **Start Indexing** in **Form1.cs** to begin the dtSearch Engine indexing process.

The DataSource Interface

The sample code uses a **SharePointDataSource** class which implements the dtSearch DataSource interface to provide documents to the dtSearch indexer.

To index the SharePoint data, **SharePointDataSource**'s **EnumerateFiles** method is first called to generate a list of files in the sites to be indexed. For each file, **EnumerateFiles** gets a URL that **SharePointDataSource** can use to download the file. The sample uses filename and extension filters to identify documents to be skipped.

Once the indexer starts, it will call **GetNextDoc()** repeatedly to get documents to index. On each call, **GetNextDoc** will use one of the URLs in its list to get a document to index. For each URL, the **SharePointDataSource** uses a .NET WebClient class to download the bytes and assigns the downloaded bytes to DocBytes.

```
byte[] bytes = WebClient.DownloadData(filesToIndex[currFileOffset]);
docBytes = bytes;
currFileOffset += 1; // Iterate to the next document
```

Conclusion and Next Steps

SharePointDemo as an Indexing Starting Point. This paper has introduced a new Visual Studio Project which brings together two separate and powerful APIs. By combining the SharePoint APIs and the dtSearch Engine, the sample project makes it possible to index key business data in SharePoint Site Collections. Developers can use SharePointDemo as a starting point to build their own custom indexing solutions.

Search Options and Appendices. After indexing, developers will typically turn to dtSearch developer product to enable web-based concurrent indexed searching. dtSearch developer products provide efficient multithreaded Internet or Intranet-based searching, with no limit on the number of concurrent search threads. For more details, please see dtsearch.com/PLF_engine_2.html.

(You can also use dtSearch Desktop to search the indexes that you create using this sample. To access the indexes in dtSearch Desktop, click Index > Recognize Index in dtSearch Desktop.)

Appendix A – Setting up SharePoint 2016

Because the security model in SharePoint can become quite complex, we assume in this article that the documents in the Site Collection allow for anonymous access. That allows us to focus on the core aspects of the code that do the indexing. To learn more about permissions and Site Collections, see this link: <https://support.office.com/en-us/article/Edit-and-manage-permissions-for-a-SharePoint-list-or-library-d217703c-55eb-451b-ad08-5f84cc297782?ui=en-US&rs=en-IE&ad=IE>.

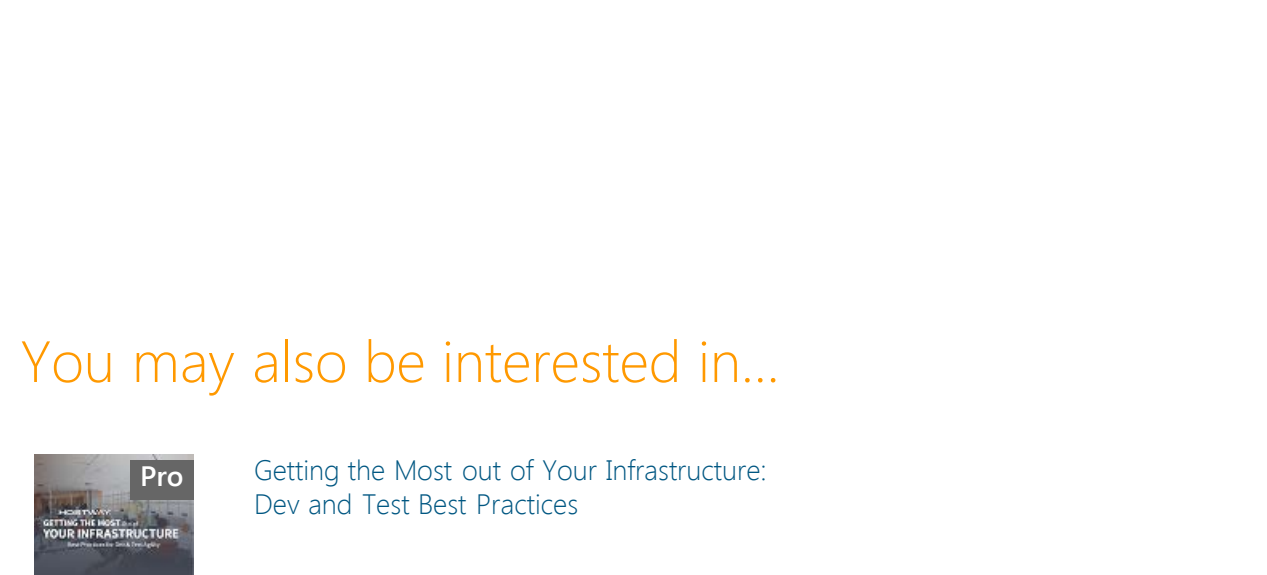
Creating the Site Collection



The central administration pages available for SharePoint administrators allow them to create Site Collections. From the central administration page, select Application Management -> Site Collections -> Create Site Collections. A web form will appear allowing you to define the attributes for the new Site Collection.

To create the site, enter a title and description, a URL, and a site template (i.e., Team, Blog, Developer Site, etc.). A message will appear, "Site was created successfully with the specified URL." The example in this article is <http://mybtsp/sites/mydocs>

The **Site Collection** page will look something like this:



Adding documents to the new Site Collection (http://mybtsp/sites/mydocs)

Next, add documents to the Site Collection. These could include any dtSearch-supported file types such as Word, Excel, PowerPoint, Access, OneNote, ZIP, HTML, XML/XSL, PDF, etc.

Setting Anonymous Access



to set anonymous access, please see:

[https://technet.microsoft.com/en-us/library/m791266\(v=office.16\).aspx](https://technet.microsoft.com/en-us/library/m791266(v=office.16).aspx)

You can also find more information about managing permissions here, [https://technet.microsoft.com/en-us/library/f607739\(v=office.16\).aspx](https://technet.microsoft.com/en-us/library/f607739(v=office.16).aspx).

More on dtSearch

- dtSearch.com
- A Search Engine in Your Pocket – Introducing dtSearch on Android
- Blazing Fast Source Code Search in the Cloud
- Using Azure Files, RemoteApp and dtSearch for Secure Instant Search Across Terabytes of A Wide Range of Data Types from Any Computer or Device
- Windows Azure SQL Database Development with the dtSearch Engine
- Faceted Search with dtSearch – Not Your Average Search Filter
- Turbo Charge your Search Experience with dtSearch and Telerik UI for ASP.NET
- Put a Search Engine in Your Windows 10 Universal (UWP) Applications

License

This article, along with any associated source code and files, is licensed under The Code Project Open License (CPOL)

Share

About the Author

Bruno Terkaly

Canada 🇨🇦


Starting with Turbo C and 8086 Assembler in the late 80s, Bruno has kept busy teaching and writing code in a multitude of platforms, languages, frameworks, SDKs, libraries, and APIs.

Bruno's depth of knowledge comes from years of experience in the field, where he can bring real-world knowledge and combine it with forward thinking that is required for his current role as a Developer Evangelist at Microsoft. Prior to evangelism, Bruno was a Microsoft Premier Field Engineer, helping customers in remote locations on a moment's notice to help with extreme troubleshooting scenarios, including problem isolation and correction, live and post-mortem debugging, on-the-fly application design and code reviews, performance tuning (IS, SQL Server, .NET), application stability, porting / migration assistance, configuration management, pre-rollout testing and general development consulting.

As an evangelist, Bruno spends time writing code and giving live presentations on building cloud based applications, specifically using the Windows Azure Platform. He also takes a strong interest in Mobile Computing and is convinced that both mobile and cloud platforms, separately and together, are poised for huge growth over the next 10 years.

Bruno is very optimistic about the potential for new interactions with software using Kinect. The software industry has been languishing in aging approaches to human interactions and software. Kinect opens up an brave new world of possibilities to next generation software engineering.

You may also be interested in...

-  Getting the Most out of Your Infrastructure: Dev and Test Best Practices
-  A Search Engine in Your Pocket – Introducing dtSearch on Android
-  Internal Site Search Engine

Comments and Discussions

0 messages have been posted for this article Visit <https://www.codeproject.com/Articles/1189807/Indexing-SharePoint-Site-Collections-Using-the-dt-to-post-and-view-comments-on-this-article-or-click-here-to-get-a-print-view-with-messages>.