Building Custom Search WebParts with Integrated SAP NetWeaver Portal Search for Microsoft Office SharePoint Server 2007 (MOSS)

White Paper

Authors:

Lei Liu, Spell GmbH, lei.liu@spell-gmbh.com
Tilo Böttcher, Microsoft Corp, tiloboet@microsoft.com

Summary:

With the public availability of Microsoft Office SharePoint Server 2007 (MOSS) Microsoft introduces an enterprise-level server for Compositon of Applicatons, Aggregation of Information, Collaboration and Document Management including Search as a significant evolution of the SharePoint Server platform. Next to the newly introduced enterprise content management (ECM) feature and the support of a variety of new client-based Office services, the new SharePoint server has significantly enhanced the search capabilities to provide the enterprise information workers with a consistent and quality search experience. It supports strongly the unification of enterprise data across unstructured documents, structured business data, and people to enable federated search across them. This whitepaper reviews the improvements concerning the search capabilities of the new MOSS server and demonstrates how to use them to provide enhanced search experience for the end user. Furthermore, this whitepaper checks up the possibilities and the constraints to build federated search with the new MOSS server and provides a simple sample to demonstrate how to integrate SAP NetWeaver Portal (also known as SAP Enterprise Portal) search into MOSS.

Applies to:

- Microsoft .NET Framework 2.0 / ASP.NET 2.0
- Office SharePoint Server 2007 Beta2
- Microsoft Visual Studio 2005 Team Suite

Keywords:

.NET Framework 2.0, ASP.NET, SharePoint, Microsoft SharePoint Portal Server 2007 (MOSS), SAP NetWeaver Portal, SAP Enterprise Portal (EP), WebPart, Search, Enterprise Search, Federated Search

Audience:

Technical consultants, Architects, Developers, IT Managers

For the latest information, please visit http://www.microsoft-sap.com

Contact

This document is provided to you by the Collaboration Technology Support Center Microsoft. For feedback or questions, you can contact the CTSC at ctsc@microsoft.com.

The information contained in this document represents the current view of the Editors on the issues discussed as of the date of publication. Because the Editors must respond to changing market conditions, it should not be interpreted to be a commitment on the part of the Editors, and the Editors cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. NEITHER OF THE EDITORS MAKES ANY WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Executive Summary

Enterprise search is crucial to the success of the daily business in each enterprise. It allows enterprise users to find the most relevant information from millions of documents stored in various business applications as well as document repositories. Today's business information is organized in a distributed manner: it can be in an enterprise content management system (e.g. word documents or HTML web site), in a backend Line-of-Business (LOB) system (e.g. reports from business intelligence, business data, etc) or in a shared network folder (e.g. documents of various types), either structured or unstructured. How to provide simultaneously the end users a federated search across the different content sources, above all from a single entry point with a consistent user interface, becomes a challenge for the portal server vendors.

As "a strong performer with strong future promise"[1] for the enterprise search platform, Microsoft has enforced the new common search architecture across its products, including the new **M**icrosoft **O**ffice **S**harePoint **S**erver 2007 from the Office server landscape. The new MOSS server exhibits several enhancements concerning enterprise search in comparison to its predecessors, the SharePoint Portal Server 2003. In addition to the improved manageability and extensibility, the increased relevance of search results and the consistent user search experience for the common content sources, the enterprise search platform in the new MOSS server provides the new ability to integrate the business data from LOB applications into the search scope of the MOSS server.

In this whitepaper, we will introduce the new search platform in the MOSS in combination with LOB systems by going through its architecture, its concept, and the ways to perform search operations programmatically. Furthermore, we will outline the principles and the architecture of federated search and how the MOSS server supports the federated search. A practical walkthrough with step-by-step instructions should show you a simple way to provide federated search through the contents of MOSS and SAP NetWeaver Portal. At the end, we conclude the whitepaper with some considerations concerning security, scalability and other facts that should give you some lift during your own implementation.

On the whole, this whitepaper will help decision makers as well as technical personals to understand the architecture, the new enterprise search capabilities from Office SharePoint Server 2007. Furthermore, it will give them an overview about federated search with a practical example of integrating SAP EP search into SharePoint server environment. With detailed walk-through for enabling federated search across MOSS and SAP EP, this whitepaper is also appropriate for developers as a hands-on lab for developing custom search applications, including a custom EP web application to enable deep search into SAP KM without TREX.

[1] Forrester Research: the Forrester Wave: Enterprise Search Platforms, Q2 2006

Table of Contents

Introduction

Today's Information workers are surrounded with increasing amount of business information available in various digital forms. However, most of the information workers are overwhelmed with information retrieval for making decisions, where each information worker spends on average 40% time of a working day for searching relevant business information (cited from the McKinsey Global Institute's research on the daily life of information workers). The possibility to run deep search of relevant business information from different content sources, including the structured and unstructured content resources, remains crucial for increasing the productivity of information workers.

With the Microsoft Office SharePoint Server (MOSS) 2007 as the significant evolution of the SharePoint Server platform, Microsoft introduces a set of new possibilities for performing deep enterprise search against the unification of different content resources. The predecessor of MOSS 2007, the SharePoint Portal Server 2003, supports already the search operations in unstructured content sources, such as documents, Exchange public folders, file systems, people, and all the SharePoint sites and workspaces. MOSS extends the relevancy as well as the user experience of the enterprise search from its predecessor. Furthermore, it enhances the search capabilities with the ability to search across not only unstructured data but also structured data in the enterprise, such as reports from BI or business objects from the underlying system.

In this whitepaper, we will review the search capabilities of MOSS - the new generation portal server from the SharePoint platform. Furthermore, we will provide some insight into the SAP's efforts in the enterprise search and outline how to combine the enterprise search products from both companies to provide the best user experience for information workers. With a simple web application on the side of SAP EP server, we demonstrate an easy way to integrate the SAP enterprise search into SharePoint with the help of the Business Data Catalog (BDC) features or a simple custom search WebPart. At the end of this whitepaper, we will draw a conclusion by providing some thoughts regarding our approach and some other aspects, such as authentication, authorization, etc.

The Anatomy of Enterprise Search

Before we begin with the capability review for the new SharePoint server, we try to provide a common view about the term "enterprise search" and the technical components associated with enterprise search.

In general, enterprise search is similar to the common Internet search service provided by e.g. Windows Live Search[2], the public Google search service or other search

[2] www.live.com

providers. To provide search services on a set of contents, these contents should be collected and aggregated from various resources. All the data being collected are then indexed to build up the index (or called "directory" in some case) as the basis for search later. Users have an interface to create queries and review search results that are organized in compliance with some predefined criteria. The major difference between the normal web search and the enterprise search is the scope of the search. While normal web search concerns itself with the public, unstructured content sources like HTML documents, Word documents, etc, enterprise search provides the users a deep insight into all the business information stored in the enterprise, including the unstructured and the structured ones.

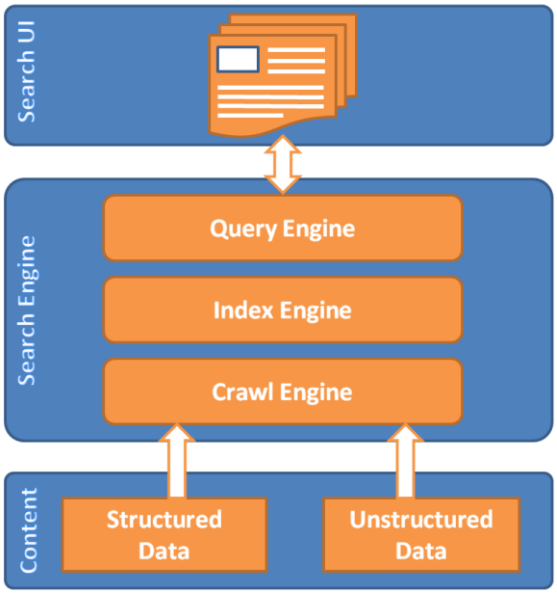


Figure 1: Enterprise Search Architecture

Figure 1 illustrates the typical architecture of enterprise search. The search engine of enterprise search consists of in general the following three components:

- **Crawling Engine**: the *crawling engine* (or called Crawler) traverses all the data from the various content sources in a methodical and automated manner. Since the data types of the content sources may vary from source to source, the *crawling engine* has the capability to extract text from different content sources. This is normally done by using dedicated content adapters for different content type, such as the IFilters used by SharePoint enterprise search. Usually, the *crawling engine* creates a copy of the content of all the visited data for later processing by the Indexing engine.
- **Indexing Engine**: The *indexing engine* is the next component in the pipeline to process the data being collected by the *crawling engine*. Its major responsibilities are to create an index on the data from the content sources, separate information of particular properties from the data, and create keyword

metadata of the data to build a more useful vocabulary for the *query engine* later.

- **Query Engine**: The *query engine* executes the search query at runtime. It receives the search query from the UI and, in necessary, breaks the search query into several words or phrases. After that, it performs the search operation against the indexing engine by using the keywords. The search results returned from the search operation are then filtered and ordered according to the relevancy, before they are forwarded to the search UI.

Federated Search in Enterprise

The need for federated enterprise search derives from the fact that the business information is strongly distributed throughout the enterprise. The business information, either structured or unstructured, are normally disconnected from the associated business processes. To find the necessary information that an information worker may need to complete the tasks involved in the processes, it is necessary to perform the same search query in different search UIs and to combine the search results being returned manually. The motivation of federated enterprise search is to reduce the complexity for performing search operations by providing a unified single search UI for different search engines that are deployed in the enterprise.

The federated search is the key to increase the productivity of information workers and makes so that the information retrieval in enterprise more efficient and precise. It gives the information worker the possibility to find the meaningful business information directly from a single search UI. Furthermore, the search results being returned by the different search engines are consolidated and resorted according to the relevancy, the taxonomy or some other criteria. So the information worker can directly read the desired information from the matching list without performing the search separately in each system.

There are generally two ways to enable the federated search in enterprise. The first approach is to provide so called "*delegated*" search to integrate the third-party search engine(s) into the major search engine. This approach is normally easy to realize, but requires large efforts after the results have been returned from the third-party search engines. The largest challenge is to consolidate the search results depending on the different schema applied to the result, because each search engine may have its own and particularly very different schema for organizing the search result. Search engines may save the same property under different names. For instance, one search engine may save the file name under the property "name", while another engine saves it under the property "file". On the other side, it is also possible that two search engines save different properties under the same name. For instance, one may use the property "date" to indicate the creation date, while another search engine uses the property "date" to indicate the last-modification date. Therefore, the requirement to eliminate the inconsistency of the search results from different search engines by using a common taxonomy remains the largest challenge for the delegated search.

The second approach for federated search is to provide deep integration by enabling a generic indexing- and crawling engine to integrate third party contents. In this approach, the third-party search engine works like a content provider rather than a search provider. The generic indexing- and crawling engine allows for including third-party contents for searching at runtime. The challenge of this approach is to enable the access to the third-party contents through some dedicated connector. Such connector must be able to extract text information from the external content and makes it available for the indexing engine. In comparison to the first approach being introduced, this approach enables an integrated search more efficiently based on the common indexing- and crawling engine. With this approach, the IT professionals do not have to care for the inconsistent taxonomies from different search engines. Furthermore, the same search algorithm implemented by the major search engine is applied consistently to the contents from different content providers. However, this approach has some technical problems to solve, too. By using this approach, the content may be indexed several times by different search engines, which causes additional operation load on the backend system as redundant indexing data. Furthermore, the search results returned by the major search engine and the third-party search engine may be inconsistent, because both the search engines may have different schedule for crawling contents at runtime. This situation with inconsistent search result may get worse, if the contents are refreshed frequently. Therefore, it is one of the major responsibilities of the search engine to keep the index up-to-date.

The Scenario in the Whitepaper

In this whitepaper, we cover part of the aforementioned federated search with the sample scenario between Office SharePoint Portal Server and SAP NetWeaver Portal Server. The focus in this whitepaper is to provide an easy way to integrate SAP enterprise search into SharePoint portal without a lot of efforts. Furthermore, this whitepaper introduces also the query object model and demonstrates the extensibility and the programmability of the new SharePoint portal. Among other things, this whitepaper covers the following topics (illustrated in Figure 2 as well):

- Development of custom search WebPart: The original enterprise search WebPart of SharePoint Server allows for a set of customizing possibilities to change the UI[3]. However, in this whitepaper, we will build our own custom search WebPart to get the customized functionality.
- Search programmability: SharePoint portal provides a set of APIs for extending your custom application. In this whitepaper, we will demonstrate the usage of the Query Object Model, which allows you to perform search queries programmatically against the search component.

[3] The components in the new Search Center of SharePoint, such as Search UI, Search Results, Search Navigation Part, can be customized by changing the XSLT script bound to the components. Please refer to the product documentation to get more information on this topic: http://msdn2.microsoft.com/en-us/library/ms545957.aspx .

- Business Data Catalog: one of the most important improvements in the SharePoint Server is the Business Data Catalog, which provides an extremely easy way to integrate the structured data in the enterprise into the portal system, such as Pos, BI reports, etc. In this whitepaper, we will use the BDC WebPart to perform search against the SAP NetWeaver Portal.
- Search with SAP NetWeaver Portal: The search in the SAP NetWeaver Portal comprises both structured data as well as unstructured data. In this whitepaper, we use a custom J2EE search application to enable the link between the SharePoint portal search and the NetWeaver portal search. The custom J2EE search application consumes the search API exposed by the Knowledge Management (KM) to enable a delegated search from the point-of-view of the SharePoint portal.

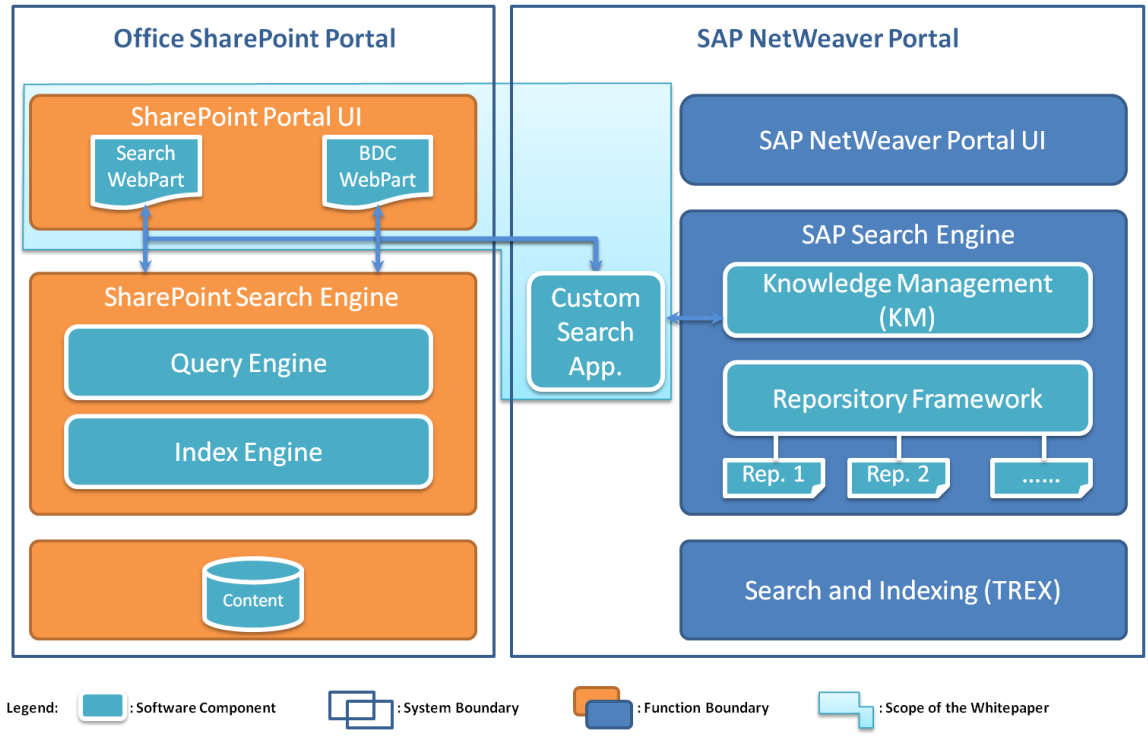


Figure 2: The Scenario covered in the Whitepaper

Intended Audience

This whitepaper aims at the technical decision makers and architects. With detailed but concise technical overviews, this whitepaper helps them to understand the technical concept, the functionalities and the use of SharePoint Portal Server. With detailed step-by-step development instructions for building the integrated enterprise search WebPart, this whitepaper is also useful for developers and technical consultants who work with SharePoint Portal Server and SAP NetWeaver Portal. Because the solutions are developed in Visual C#, it is assumed that the readers have fundamental understanding about Visual C# programming with .NET framework as well as the basic understanding about the ASP.NET programming.

Prerequisites

In this whitepaper, we use the most recent software available at the development time to build up the WebParts and the Web services. Please note that, some of the software is still in Beta status and may cause occasionally application crash. Therefore, do not install these builds on the machine you operational depend on.

To build the applications introduced in this whitepaper, you need the following software components:

- **Development Environment**: you can use the Visual Studio 2005 full retail version as your development environment. In this whitepaper, the whole demo scenario is built using Visual Studio 2005 Team Suite, which is available to MSDN subscribers. As an alternative development environment, you can use the Visual Studio 2005 Express Editions with limitation, too. The free Visual Studio 2005 Express Editions can be downloaded from http://msdn.microsoft.com/vstudio/express/default.aspx.
- **Windows Workflow Foundation 2.2**: Office SharePoint Server 2007 provides the out-of-box support for workflow-based form-driver business processes with XML-based electronic forms; therefore it requires Windows Workflow Foundation (WF) as one of the prerequisites for the deployment. To get the latest WF, please visit the Windows Workflow Foundation portal site http://wf.netfx3.com/ and download the installation kit from there.
- **Microsoft Office SharePoint Server (MOSS) 2007**: You can download the installation kit for the Office SharePoint Server 2007 directly from http://www.microsoft.com/office/preview/beta/overview.mspx. Please note that, you need not only the installation kit for the Windows SharePoint Services (WSS) 3.0, but also the one for the Portal Server. In our scenario, we will use the functionalities from the Business Data Catalog (BDC), which are only available with the SharePoint Portal Server. To try out the new functionalities to create, customize and contribute to the SharePoint portal sites, you can also install the SharePoint Designer 2007, which is based on the previous Office FrontPage technologies. However, the SharePoint Designer is only optional for the implementation described in this whitepaper. For more information about the deployment of SharePoint Server either in a single server or in a server farm, please refer to the Office SharePoint Server technical library under http://technet2.microsoft.com/Office/en-us/library/3e3b8737-c6a3-4e2c-a35f-f0095d952b781033.mspx?mfr=true.

Until now, we have introduced the motivation and the business scenario described in this whitepaper. In the following sections, we will introduce the implementation of the scenario with a top-down approach. We will begin with the development of the WebParts and describe how to create a custom enterprise search WebPart in the new SharePoint portal. Afterwards, we move the focus from MOSS to SAP NetWeaver portal, and describe how to create a simple J2EE application that exposes KM search functionalities through Web interface. At last, we unify both approaches and

demonstrate the way to provide federated search comprising both SharePoint search and SAP enterprise search. Furthermore, we also demonstrate how to use the new BDC WebPart as the UI for the SAP search without any coding effort. The last section summarizes the whitepaper and provides some thoughts regarding security, restrictions, and other aspects of the approach introduced in this whitepaper.

Building a Simple Federated Search WebPart with Integrated SAP Search

In this section, we focus on the technical aspects of the business scenario we described in the introduction section. Among other things, we will show how to use the WebPart and a simple J2EE Web application to integrate SAP search into SharePoint. Furthermore, we introduce another codeless approach to integrate the SAP search into SharePoint by utilizing the BDC functionalities.

Basic for Building WebPart

In the world of SharePoint, WebParts are the basic integral parts for constructing the pages of a Web site. You can simply imagine WebParts as building blocks that are specifically designed to provide personalized and customized UI to users. From the technical point-of-view, each WebPart is an ASP.NET control with additional properties and functions that make them to be able to interact with the SharePoint framework. The base WebPart class provides the basic properties that are common to all WebParts, such as title, rendering properties, and so on. To create a custom WebPart, you just need to declare a new .NET Web custom control deriving from the base WebPart class. In this way, you can add custom properties to the WebPart, which will be merged at runtime to a single set of WebPart properties.

In general, a WebPart comprises two parts: a .NET Web control assembly and a WebPart description file which can either be in ASP.NET 2.0 format with the file extension *.webpart* or in the dedicated SharePoint format with the extension *.dwp*. The .NET Web control assembly is a .NET assembly that must be present and registered on the server where the SharePoint portal runs. The description file contains metadata about the instance of the WebPart in XML format. We will come back to this topic later when we are going to deploy the WebParts that we developed to the server.

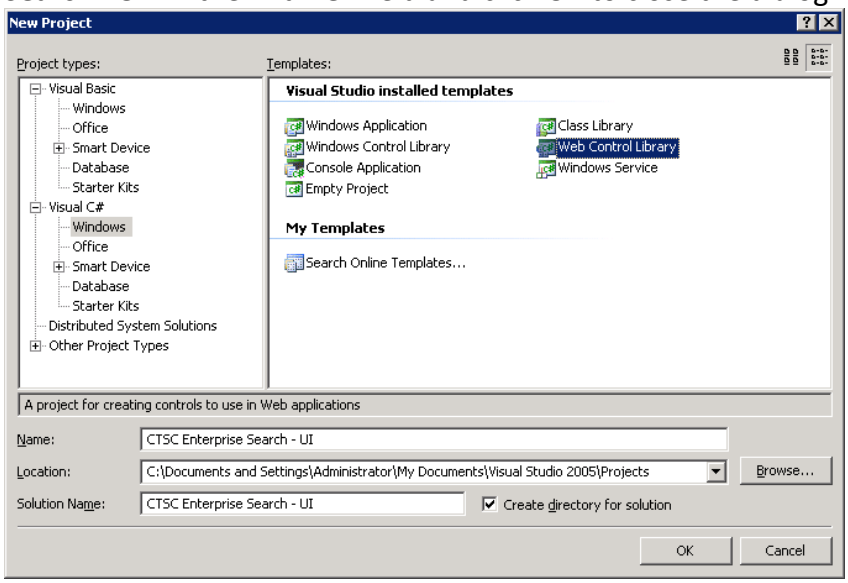
In comparison to ASP.NET 1.0, ASP.NET 2.0 comes with a built-in portal framework that also brings the concept of WebParts with it. This makes it easier to build ASP.NET portal solutions, because the built-in framework with WebPart concept provides a consistent model for building portal solutions. So far so good, but what are in fact the differences between ASP.NET WebPart and SharePoint WebPart? In essence, SharePoint WebParts also inherit from the ASP.NET WebPart base class. It does mean that SharePoint WebParts contains all the functionalities that ASP.NET 2.0 WebParts provide at runtime. In addition, SharePoint WebParts provide further possibilities to interact with the SharePoint portal framework. In other words, a custom WebPart that inherits from the base SharePoint WebPart class has some additional features, for instance, to create cross-page connections in SharePoint sites or create connections between WebParts from different WebPart zone. However, because of the additional functionalities, which functionally depend on the SharePoint framework, SharePoint WebParts can only be

used in a SharePoint site. Therefore, if you are planned to ensure the independence of your custom WebParts, you should try to implement them by inheriting from the ASP.NET base WebPart class, instead of the SharePoint base class.

In our business scenario with integrated SAP search, we want to get the search results from SharePoint and SAP NetWeaver portal displayed in WebParts side-by-side. Therefore, we need altogether three WebParts: one contains the search mask where the user can enter the queries, and the other two WebParts should display the search results from both portals. In the following, we will provide a step-by-step guide on how to develop a custom WebPart for the new SharePoint portal server. In the following, we mainly address the following three tasks in the WebPart development: set up the WebPart project, implement the WebPart, and deploy the WebPart.

To set up a new WebPart project, please follow the instructions below:

1. Create a new project in Visual Studio 2005. Select "**Windows**" from the "**Project types**" under "**Visual C#**", select the project template "**Web Control Library**" from the available templates, and specify the project name as "**CTSC Enterprise Search –UI**" in the "**Name**" field and click OK to close the dialog.



2. In this step, you will add the necessary assembly references to the WebPart project. On the Project menu, select "**Add Reference**". In the popup dialog, select the following .NET assemblies from the list: **System.Data**, **System.XML**, **Microsoft.SharePoint**, **Microsoft.SharePoint.Portal**, **Microsoft.Office.Server**, **Microsoft.Office.Server.Search**. If you cannot find the SharePoint assemblies from the list, you may have to add them manually. In the popup dialog, switch to the "**Browse**" tab and navigate to the folder "**C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\12\ISAPI**". There you can find all the .NET assemblies that you need for the project. Select the corresponding assemblies from the folder and click "**OK**" to close the dialog.

3. The default class file contains some attribute declarations that are unnecessary for the WebPart implementation. Therefore, it is recommended to remove the default class file and add a new class file to the project. In **Solution Explorer**, simply right-click the default class file **WebCustomControl1.cs** and then click **Delete** to remove the default class from the project. Then, on the **Project** menu, click **Add New Item** to use the **Class** template to add a new class to the project.

In the next step, you will modify the class file to render the WebPart UI:

1. At first, you have to add the namespace directives to the class file:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Web.UI;
using Microsoft.Office.Server;
```

2. We are going to create a SharePoint WebPart, therefore, we inherit our class from the SharePoint WebPart base class:

```
namespace CTSC.Enterprise.Search.UI
{
    public class SearchUI : Microsoft.SharePoint.WebPartPages.WebPart
     {
     }
}
```

3. Now we have to override the rendering method of the WebPart class for our WebPart. You can either override the Render/RenderWebPart methods or the CreateChildControls method to create the UI for the WebPart. Which method should be overridden depends on your implementation details: the Render/RenderWebPart allows you to work directly with HTML markups, and the CreateChildControls method let you to work with ASP.NET Web controls. In this WebPart, we first use the Render-method to create the UI, as shown as follows:

```
/// <summary>
/// render the UI in the WebPart
/// </summary>
/// <param name="output"></param>
protected override void Render(HtmlTextWriter output)
{
    string keywords = "";

    if (this.Page.IsPostBack)
    {
keywords = this.Context.Request["keywords"];
    }

    // render the HTML page
    new LiteralControl("<div style=\"padding-left:7px\">").RenderControl(output);
    new LiteralControl("<span style=\"color:Black;font-size:12pt;font-
family:Tahoma\"><b>CTSC Enterprise Search with Integrated SAP Search</b></span><br
/><br />").RenderControl(output);
    new LiteralControl("<span style=\"color:DodgerBlue;font-size:8pt;font-
family:Tahoma\"><b>Find all documents with the following
keywords:  </b></span>").RenderControl(output);
    //search box
    new LiteralControl("<input type=\"text\" name=\"keywords\" value=\"" +
keywords + "\"/><br/><br/>").RenderControl(output);
    //search button
    new LiteralControl("<input type=\"submit\" value=\"Search\"
class=\"submitbutton\">        ").RenderCo
ntrol(output);
```

```
    new LiteralControl("<input type=\"reset\" value=\"Reset\"
class=\"submitbutton\">").RenderControl(output);
    new LiteralControl("</div>").RenderControl(output);

    base.Render(output);
}
```

Now you have finished implementing the WebPart. To compile your WebPart project, on the **Build** menu, click **Build Solution** from the menu. In the next step, we are going to deploy the WebPart to the SharePoint portal site. In general, there are two folders where you can deploy the WebPart assembly to: the Global Assembly Cache (GAC) or the **_app_bin** directory of your site. Before you can deploy your assembly to the GAC, you have to sign it with a strong name. Please refer to the MSDN library for more information[4] about how to sign a .NET assembly with a strong key. In our scenario, we will deploy our assembly into the GAC.

1. Before you can start with the deployment, you have to determine the root application path of the site. Open the **Internet Information Services (IIS) Manager**[5] console, expand the **Web Sites** node, right-click the site of your SharePoint portal, and select **Properties** from the context menu. Then click on the **Home Directory** tab of the pop-up window, and the **Local Path** field contains the root application path of the target site.
2. Copy the assembly to the GAC. Please rename the assembly to "CTSC.Enterprise.Search.UI.dll" and sign it with a strong name. Then open a new console windows, and use the following command to import the assembly into the GAC:

   ```
   C:\>gacutil -i CTSC.Enterprise.Search.UI.dll
   ```

3. Each WebPart must be explicit declared as a "safe control", before they can be deploy to the SharePoint portal. To do it, open the Web.config file that you can find in the root application path of the site. Then add the following directive into the <SafeControls>…</SafeControls> tag in the Web.config file:

   ```
   <SafeControl Assembly="CTSC.Enterprise.Search.UI, Version=1.0.0.0,
   Culture=neutral, PublicKeyToken=a226358969cf4b6c"
   Namespace="CTSC.Enterprise.Search.UI" TypeName="*" Safe="True" />
   ```

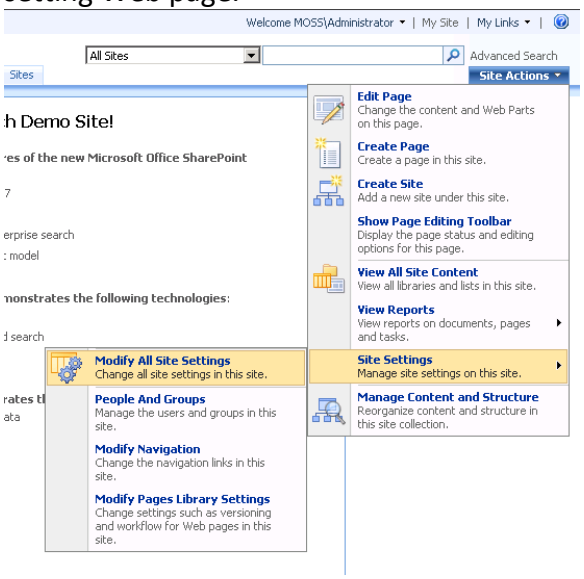
   You can find out the assembly version and the public key token of your .NET assembly by navigating to the directory "C:\Windows\Assembly", finding the .NET assembly and reading the values from the corresponding fields in the list. Please make sure that the assembly version and public key token match up with the values you get in the GAC, otherwise you will not have the WebPart displayed in the SharePoint portal later.

[4] .NET Framework Developer's Guide: Signing an assembly with a strong name - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconassigningassemblystrongname.asp.

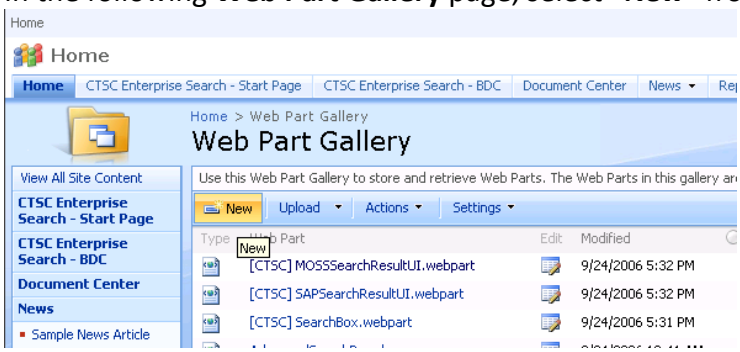
[5] You can find the Internet Information Services (IIS) Manager in the Administration Tools of your Windows Server.

After you have deployed the WebPart assembly, you are ready to test it now. To do it, open the SharePoint portal site in your browser and follow the instructions below:

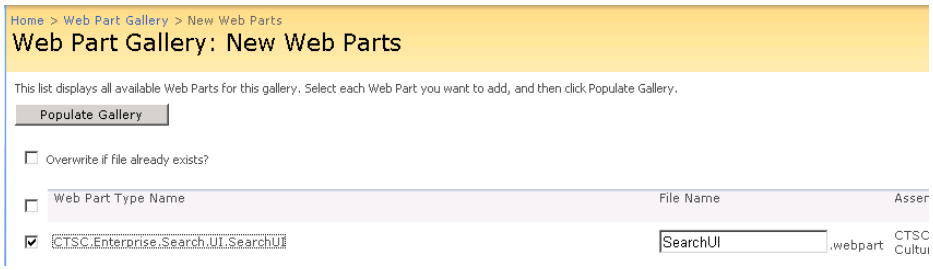
1. In the portal site, you can find the menu "**Site Actions**" on the right side. Navigate to "**Site Actions| Site Settings | Modify All Site Settings**" to open the setting Web page.



2. In the **Site Settings** page, click **Web Parts** from the group **Galleries.**
3. In the following **Web Part Gallery** page, select "**New**" from the menu bar.



4. In the New Web Parts page, you can find the WebPart "CTSC.Enterprise.Search.UI.SearchUI" from the list. Check the type in the list and click "Populate Gallery" to add it to the public Web Parts gallery. If you cannot find the WebPart in the list, please make sure that the WebPart assembly has been successfully added to the GAC and the values in the <SafeControl> tag, especially the assembly version and the public key token, are identical with the ones listed in the GAC.



Now you have now successfully added the WebPart to your SharePoint portal site. To add the WebPart to a Web page, simply open the page in edit mode, select a Web Part zone on the page, click the **Add a Web Part** link to open the Add Web Parts dialog, and select our WebPart from the list to add it to the page. Please refer to the documentation of SharePoint Portal Server 2007 to get more information on it[6].

In this section, we have together implemented a WebPart for the search UI. In a similar way, you can implement the WebPart for the search results UI of SAP NetWeaver portal search as well as Microsoft SharePoint search. Figure 3 illustrates how it may look like. You can get the source code for the WebParts directly from the Web site of the Whitepaper.

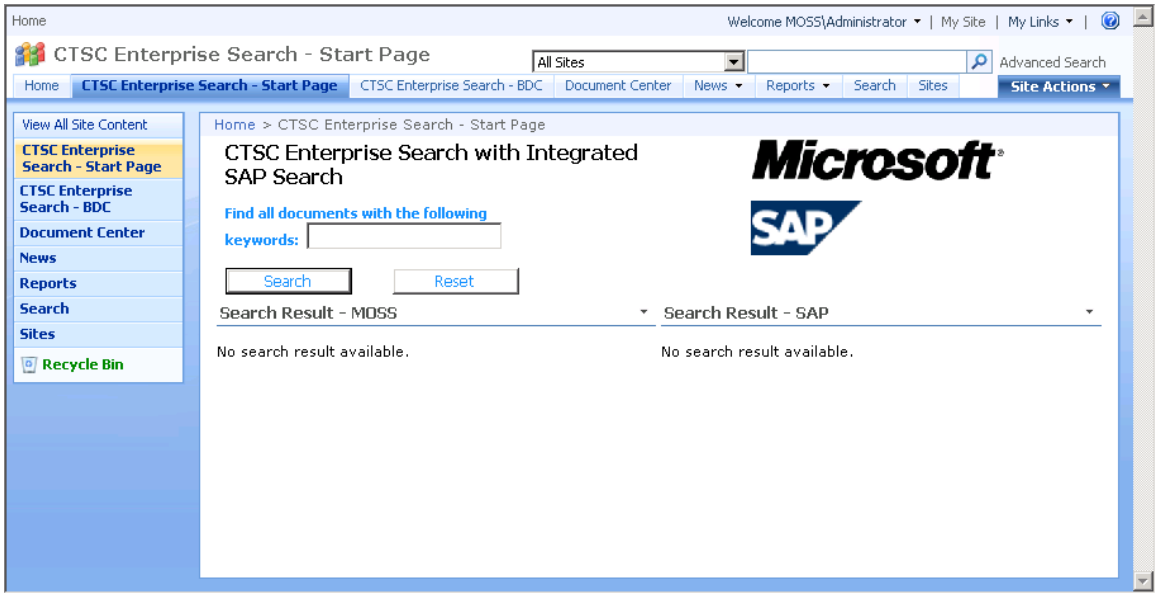


Figure 3: The WebParts for the Enterprise Search Scenario

Another issue to be figured out is the communication between the WebParts. In our enterprise search scenario, we have the search UI that performs the search operation against SharePoint/NetWeaver portal at runtime. On the other side, we have the other two search results UI that display the search results at runtime. Therefore, we need the communication between the search UI WebPart and the search results UI WebParts. In SharePoint, you can share information between different WebParts by creating connections between them. Because this topic is out of the scope of this whitepaper, please refer to ASP.NET 2.0 book for more information. MSDN provides several articles on this topic:

- Introducing the ASP.NET 2.0 Web Parts Framework: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvs05/html/webparts.asp
- Introducing ASP.NET Web Part Connections: http://msdn.microsoft.com/msdnmag/issues/06/02/BasicInstincts/

[6]Microsoft Office SharePoint Server 2007 help: http://office.microsoft.com/en-us/sharepointserver/FX101211721033.aspx.

Building a Custom Search Application for SAP EP

One of the important announcements at SAP's TechEd this year is the project Argo, the codename of the SAP's enterprise search project. With a homogeneous federation layer, similar with the BDC in SharePoint portal, the project Argo introduces mechanisms to integrate content management system, file shares, business data and other business information into the enterprise search that are accessible from multiple channels, such as NetWeaver portal, Office application, or even mobile devices. Currently, there is a Project Argo Preview Release available, with which you can get an early try with the product. Please go to SAP Develop Network to get the installation kit for the preview release of Argo: http://sdn.sap.com (Search for ARGO since this will not be a static link during the next couple of month where Argo will be developed).

In this whitepaper, we focus on the existing search infrastructure of the SAP NetWeaver portal. Together with the Knowledge Management (KM) and TREX (Search and Classification), user can perform enterprise search against one or several NetWeaver portals through all the unstructured business data. However, with the APIs exposed by KM, you can also create your custom J2EE search application that provides the full functionalities of SAP search to the external world, i.e. to a SharePoint portal. In this section, we are going to shortly introduce how you can utilize the KM APIs to implement the custom J2EE search application.

Running as a Java servlet running on the SAP NetWeaver AS, the custom search application communicates over HTTP with the SharePoint Portal Search Engine (as shown in Figure 4). The search query and its meta information are transferred as URL parameters to the search servlet for further processing. After having got HTTP request from the client application – in our case the custom Search WebPart from SharePoint – the servlet logs in to the SAP NetWeaver Portal using a set of predefined username/password combination, executes the search operation using the APIs provided by KM and returns the search results as XML document to the SharePoint WebPart via HTTP again.

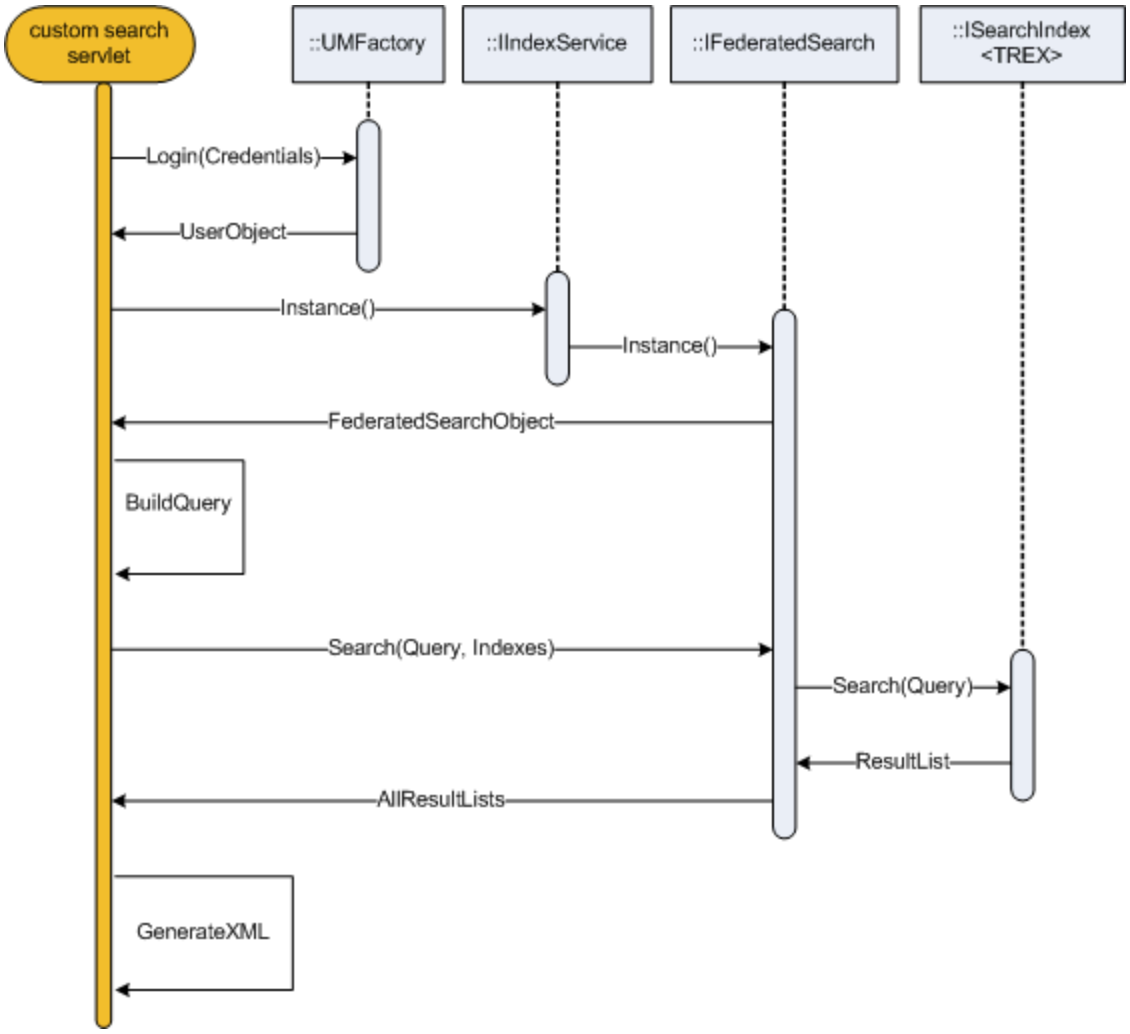


Figure 4: Custom Search Servlet Execution

Performing Search against MOSS

With SharePoint Portal Server 2003, you can use both the Query Object Model and the enterprise search Web services to perform search against SharePoint portal. MOSS improves the search capabilities from SharePoint Portal Server 2003 and ensures compatibility of the Query object model and the Web services with the old SharePoint portal. In other words, any custom search applications that are based on these technologies can continue to use without any modification.

The Query Object Model allows you to create custom search applications, including custom search WebPart, ASP Web application, and so forth, to execute search queries against a SharePoint Server. To consume the APIs of the Query Object Model, you need to include references in your project to the assemblies: **Microsoft.Office. Server.Search.dll**, **Microsoft.Office.Server.dll**, and **Microsoft.SharePoint.dll**. Using the Query Object Model, you use execute either queries in SQL syntax or keyword queries against the SharePoint portal. To determine which type of queries should be used in

your application, you should consider the complexity of the search queries you want to support in your applications. The keyword queries give you the possibility to directly forward the search terms to the Query Object Model API as keywords. In this way, you do not have to parse the search terms to build the search query. However, keyword query is not sufficient any more, if you would like to support more complex search queries. In this case, you need the search queries in SQL syntax. The Query Object Model supports several SQL operators by using this type of search query. For example, you can use the operator *ORDER BY* to specify how the search results being returned should be sorted. MSDN provides a technical article that describes the syntax of queries in SQL syntax detailed. You can refer to this article to get more information about it: http://msdn2.microsoft.com/en-us/library/ms493660.aspx.

The other way to perform search operations against SharePoint portal is to use the enterprise search Web services. The search Web services can be invoked remotely just like any normal Web services from your custom applications by consuming the proxy classes[7]. And if no anonymous access is allowed on the remote SharePoint site, you have to specify the credentials to be used for the access. Just like the Query Object Model, you can use both queries in SQL syntax or keyword queries to perform search against the SharePoint portal. And you can also specify how the search results should be returned by the portal, either in XML string or in ADO.NET object. Next to the search Web methods, the search Web services also offers other Web methods to provide i.e. runtime status or search metadata of the search service on the remote SharePoint portal.

In this whitepaper, we will simply demonstrate the programmability of the SharePoint portal for enterprise search by execute keyword queries:

1. To perform a keyword search against the SharePoint portal, you just need to create a KeywordQuery object, set up the properties of the object and execute it, as shown below:

```
//retrieves the Search Context for the current site
ServerContext context = ServerContext.Current;
KeywordQuery query = new KeywordQuery(context);

query.QueryText = keywords;
//to return relevant results
query.ResultTypes |= ResultType.RelevantResults;
//do the search
ResultTableCollection result = query.Execute();
```

2. To read the search result from the query object, you just need to create a new DataTable object and load the results objects from the result table to it, as follows:

```
if ((int)ResultType.RelevantResults > 0)
{
    //read the result table from the search runtime
```

[7] Please refer to the .NET Framework programmer guide to learn how to generate Web service proxy classes for the search Web services: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconCreatingWebServiceProxy.asp.

```
    ResultTable relevantResults = result[ResultType.RelevantResults];

    //copy the result to a DataSet
    DataTable resultsDataTable = new DataTable();
    resultsDataTable.TableName = "MOSS Search Results";
    DataSet resultSet = new DataSet("mossresultsset");
    resultSet.Tables.Add(resultsDataTable);

    resultSet.Load(relevantResults, LoadOption.OverwriteChanges,
resultsDataTable);

    this.MOSSsearchResult = resultSet;
}
else //find no result
{
    this.MOSSsearchResult = new DataSet();
}
```

Another Easy Way to Integrate SAP Search with BDC

The Business Data Catalog introduced in Office SharePoint Server 2007 allows portal administrators to integrate structured business data for end-user accesses by registering Web services and data sources from a variety of backend systems. As the key infrastructural component for the business data features in the SharePoint portal, BDC enables auto-provision WebParts with respect to the user profiles in the portal. With the BDC enabled WebParts, user can simply edit business data from the backend business systems through Web services or databases. Furthermore, BDC provides the out-of-box support for indexing structured business data, so that the business data can be included in the search results while user just need to use a single search UI to perform search queries against the SharePoint portal.

Figure 5 illustrates the high-level architecture of the Business Data Catalog in combination with the SharePoint portal UI and the content sources. As already mentioned before, structured business data can be accessed by BDC either through Web services or database connections. The key shift of BDC is the change from coding to metadata. BDC uses a declarative metadata model together with a consistent client object model to get a homogeneous access to the underlying content sources. In the development lifecycle of BDC, an IT professional or developer enter the information about the backend business application into an *application definition* file and deploy it to the metadata repository in BDC. Then, business analysts as well as application developers can interact with the business application indirectly by using the features from BDC. In this way, they get a more consistent view about various business applications and can interact with them consistently. MSDN provides a detailed guide on Business Data Catalog. Please refer to the MSDN Web site to get more information on this topic: http://msdn2.microsoft.com/en-gb/library/ms563661.aspx.

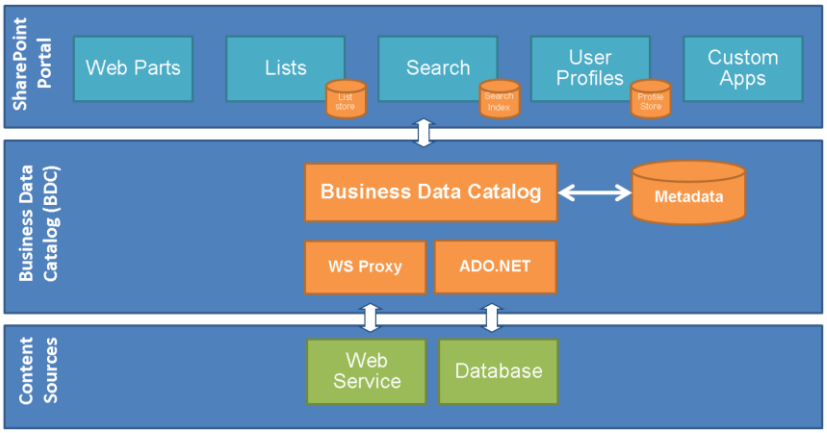
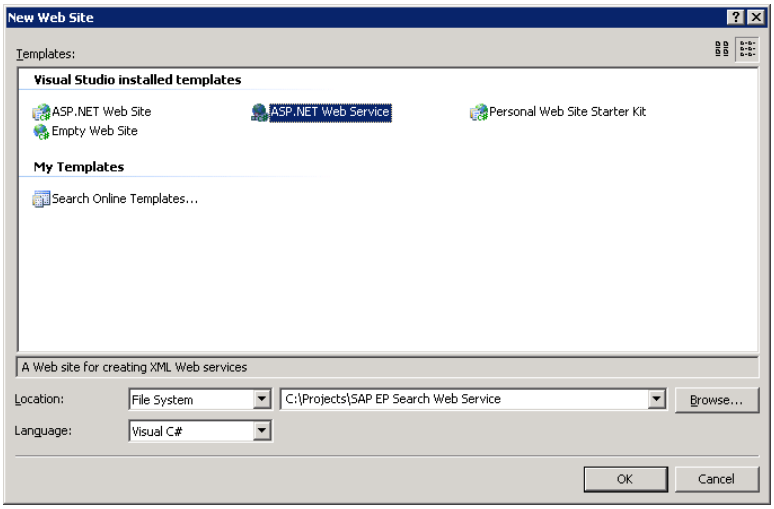


Figure 5: High-level Architecture of Business Data Catalog

In the last sections, we have together developed a set of custom WebParts that have integrated both SharePoint portal search and NetWeaver portal search. However, we can also simplify the development effort by using the features of BDC. In the following, we will demonstrate the features of BDC by integrating the NetWeaver portal search into SharePoint portal with the help of BDC WebPart. The key idea of the implementation is to treat the search results as a business object. Therefore, we need a Web service that can forward the search request to the SAP portal search application and return the search results as business objects to BDC. In other words, we need a wrapper Web service for our J2EE search application. After that, we will develop the application definition for BDC and deploy it to the SharePoint portal. At last, we will use the BDC WebPart to integrate the search application into the portal.

To implement the wrapper Web service for the J2EE search application, simply set up a new ASP.NET Web Service project as described in the following:

1. Create a new Web site in Visual Studio 2005. Select the project template "**ASP.NET Web Service**" from the available templates, and specify the project name as "**SAP EP Search Web Service**" and click OK to close the dialog.



2. It is recommended to remove the default Web service and add a new Web service to the project. In **Solution Explorer**, simply right-click the default class file

Service.cs as well as Service.asmx and then click **Delete** to remove them from the project. Then, on the **Project** menu, click **Add New Item** to use the **Web Service** template to add a Web service to the project. Give the new Web service a meaningful name, i.e. “EPSearchService.asmx”.

Now, you have successfully set up the Web service project. In the following, we are going to implement the Web service:

1. At first, we need a new class for the business object “SearchResult”. Add a new class file to the project and add the following code to the class file:

```
/// <summary>
/// Summary description for SearchResult
/// </summary>
public class SearchResult
{
    public string DisplayName;
    public string URL;
    public string Size;
    public string ContentType;
    public string CreatedBy;
    public string CreationDate;
    public string ModifiedDate;

    public SearchResult()
    {
    }

    public SearchResult(string displyname, string url,
        string size, string contenttype, string createdby,
        string creationdate, string modifieddate)
    {
        DisplayName = displyname;
        URL = url;
        Size = size;
        ContentType = contenttype;
        CreatedBy = createdby;
        CreationDate = creationdate;
        ModifiedDate = modifieddate;
    }
}
```

2. Then, you are ready to implement the wrapper Web service. In our implementation, we are going to use a set of APIs from .NET framework, therefore, please add the following namespace directives to the Web service class at first:

```
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Collections.Generic;
using System.Net;
using System.IO;
using System.Xml.Serialization;
using System.Xml;
```

3. At last, we are going to implement the Web service by forwarding the search query to the custom J2EE search application and convert the search results from SAP to the business objects defined before. The source code may look like as follows:

```
[WebService(Namespace = "http://www.microsoft-sap.com/enterprise/search/EP")]
```

```
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class EPSearchService : System.Web.Services.WebService
{
    public EPSearchService () {
    }

    [WebMethod]
    public SearchResult[] GetResults(string query) {
        List<SearchResult> searchresults = new List<SearchResult>();

        results sapresults = this.GetSAPResults(query);

        for (int i = 0; i < sapresults.Items.Length; i++)
            searchresults.Add(CopyValue(sapresults.Items[i]));

        return searchresults.ToArray();
    }

    private SearchResult CopyValue(result sapresult)
    {
        return new SearchResult(sapresult.displayname, sapresult.url,
                 sapresult.size,sapresult.contenttype, sapresult.createdby,
                 sapresult.creationdate, sapresult.modifieddate);
    }

    /// <summary>
    /// get the result XML from the SAP EP
    /// </summary>
    /// <param name="query"></param>
    /// <returns></returns>
    private results GetSAPResults(string query)
    {
        // forward the search request to SAP
        // Please refer to the source code of the sample project for more
        // information
    }
}
```

Now, we have created the wrapper Web service that interacts with the custom J2EE search application at runtime and returns the search results as business objects to the client. In the next step, we are going to define the application definition metadata file for BDC. In the metadata file, we will only define a single **Finder** method delivering the search results as business objects:

1. In the first step, we will define the meta-level information to connect to the wrapper Web service implemented in the last step. In the meta-model of BDC, a **LobSystem** object is always the top-level metadata element that describes a particular business application. It contains information like connection string to the Web service or the authentication information for BDC. The **LobSystem** definition for our wrapper Web service may look like as follows:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<LobSystem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.microsoft.com/office/2006/03/BusinessDataCatalog
BDCMetadata.xsd" Type="WebService" Version="1.0.0.0" Name="SAPSearchWebService"
xmlns="http://schemas.microsoft.com/office/2006/03/BusinessDataCatalog">
  <Properties>
    <!-- The Web Service proxy namespace name you specify here will be used by the
Business Data Catalog when it generates the proxy.-->
    <Property Name="WebServiceProxyNamespace"
Type="System.String">SAPSearchWebServiceProxy</Property>
    <!-- Enter the wildcard character that your Web Methods support. Business Data
Catalog will use the wildcard character with filters.-->
    <Property Name="WildcardCharacter" Type="System.String">$</Property>
    <!-- URL to WSDL or ASMX.-->
    <Property Name="WsdlFetchUrl"
```

```
Type="System.String">http://localhost/SAPSearchService/EPSearchService.asmx</Property>
  </Properties>
  <LobSystemInstances>
    <!-- The LOB instance name is used by business data clients in the entity picker,
etc..-->
    <LobSystemInstance Name="SAPSearchWebServiceInstance">
      <Properties>
        <Property Name="LobSystemName"
Type="System.String">SAPSearchWebService</Property>
      </Properties>
    </LobSystemInstance>
  </LobSystemInstances>
  <Entities>
  </Entities>
</LobSystem>
```

Please save this application definition metadata under the file name "**SAPSearchBDCDefinition.xml**".

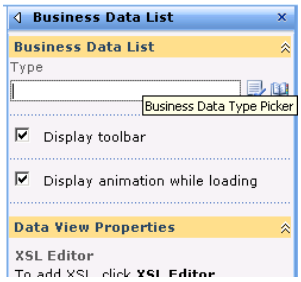
2. In the meta-model of BDC, each LOB application contains several entities. Every entity in the definition is a business object, such as Customer or Search Result in our case. Each entity contains one or more methods that can be executed on the entity. Furthermore, a method may contain filters to enable end-user filtering by limiting the entity instances returned by the method. In a nutshell, an entity belongs to a single LOB application and contains next to the identity also methods and filters. In this step, we will define an entity called **SAPSearchResult**. This entity contains a single Finder-method "**GetResults**" that contains an input-parameter "**query**" and returns the output-parameter "**SAPResults**", which is a set of **SearchResult** objects. The following XML segment demonstrates the definition of the **SAPSearchResult**-entity. Please add this XML segment into the application definition file created in last step.

```
<!-- The SAPSearchResult entity has only finder-method. -->
<Entity EstimatedInstanceCount="10000" Name="SAPSearchResult">
  <Properties>
    <Property Name="Title" Type="System.String">DisplayName</Property>
  </Properties>
  <Identifiers>
    <Identifier TypeName="System.String" Name="URL" />
  </Identifiers>
  <Methods>
    <Method Name="GetResults">
      <FilterDescriptors>
        <FilterDescriptor Type="Wildcard" Name="SAPQueryString" />
      </FilterDescriptors>
      <Parameters>
        <Parameter Direction="In" Name="query">
          <TypeDescriptor TypeName="System.String" AssociatedFilter="SAPQueryString"
Name="query" />
        </Parameter>
        <Parameter Direction="Return" Name="SAPResults">
          <TypeDescriptor TypeName="SAPSearchWebServiceProxy.SearchResult[],
SAPSearchWebService" IsCollection="true" Name="ArrayOfSearchResult">
            <TypeDescriptors>
              <TypeDescriptor TypeName="SAPSearchWebServiceProxy.SearchResult,
SAPSearchWebService" Name="SearchResult">
                <TypeDescriptors>
                  <TypeDescriptor TypeName="System.String" IdentifierName="URL"
Name="URL" />
                  <TypeDescriptor TypeName="System.String" Name="DisplayName" />
                  <TypeDescriptor TypeName="System.String" Name="Size" />
                  <TypeDescriptor TypeName="System.String" Name="ContentType" />
                  <TypeDescriptor TypeName="System.String" Name="CreatedBy" />
                  <TypeDescriptor TypeName="System.String" Name="CreationDate" />
                  <TypeDescriptor TypeName="System.String" Name="ModifiedDate" />
```

```
                </TypeDescriptors>
              </TypeDescriptor>
            </TypeDescriptors>
          </TypeDescriptor>
        </Parameter>
      </Parameters>
      <MethodInstances>
        <MethodInstance Type="Finder" ReturnParameterName="SAPResults"
ReturnTypeDescriptorName="ArrayOfSearchResult" ReturnTypeDescriptorLevel="0"
Name="SearchSAP" />
      </MethodInstances>
    </Method>
  </Methods>
</Entity>
```

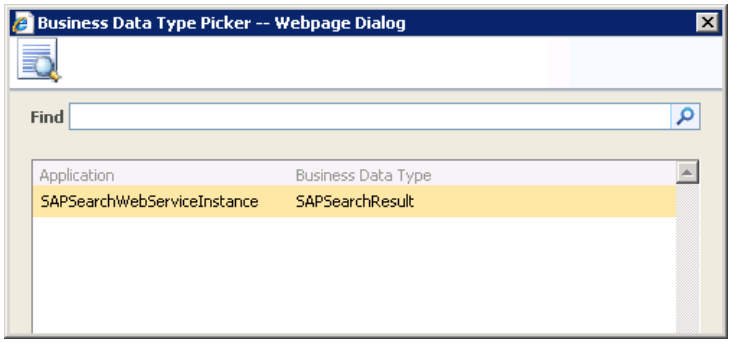
You have successfully defined the application definition for the SAP search wrapper Web service. Now, you are ready to deploy the application definition to the BDC. It's assumed that you have set up your SharePoint environment with all the necessary components, including the **Office SharePoint Server Shared Service** for your site. To deploy the application definition to BDC, please open the Central Administration site of your SharePoint portal in a browser and follow the instructions below:

1. In the Central Administration site, you can find the Shared Service provider in the left navigation pane. Click the Shared Service provider in the list.
2. In the following page, click **Import application definition** in the **Business Data Catalog** section.
3. In the Import application definition page that opens, browse to the XML file that contains the application definition you created in last step and then click **Import** to upload the file to BDC.

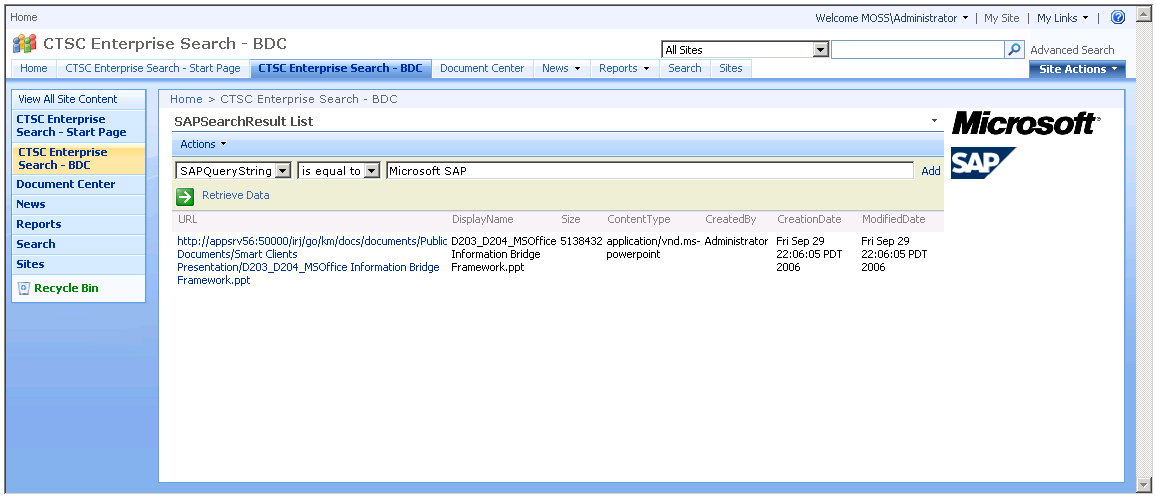


In the next step, you are ready to test the wrapper Web service by using the BDC features. We will use a **Business Data List** Web part to consume the wrapper Web service of SAP search from SharePoint portal. To add it to a Web page, simply open the page in edit mode, select a **Web Part zone** on the page, click the **Add a Web Part** link to open the **Add Web Parts** dialog, and select the **Business Data List** WebPart from the list to add it to the page. Then, open the tool pane of the **Business Data List** WebPart to configure it, as illustrated in the figure on the right side.

In the next step, you have to choose the Business data type that should be used with the WebPart. Click the browse button ("") to browse the business data types that are available on the SharePoint portal and double-click the type "SAPSearchResult" from the list to specify the business data type to use.



Now, you have finished implementing the BDC application for SAP enterprise search. You can test it by entering the query string in the filter field and click “Retrieve Data” to perform the search. Furthermore, you can also customize the UI of this Web part by modifying the XSL script of it. The BDC WebPart may look like as follows:



Summary

Enterprise search allows information workers to find the most relevant information from millions of documents and business data stored in various business applications as well as document repositories; therefore, it remains a crucial task for the success of the daily business of every enterprise. With the availability of new products, all software vendors try to provide the comprehensive support for federated search in enterprise that allows users to use a single entry point to execute search query against several search content sources, regardless of structured data like business objects, or unstructured data like Office documents. Microsoft introduced the new feature Business Data Catalog in the Office SharePoint Server 2007 that aims to provide a unified application layer to deal with structured business data, including executing search queries on these business data consistently. With the announcement of the project Argo in the SAP TechEd this year, SAP also provides a similar federated search platform for enterprise search across structured and unstructured enterprise data.

In this whitepaper, we have introduced the concept of enterprise search and the importance to provide federated search across all the content sources in the enterprise. Furthermore, we have implemented the enterprise search scenario with delegated search to SAP to demonstrate the programmability and the extensibility of the new SharePoint portal server. In this way, we have shown an easy way to provide integrated search across SAP and Microsoft with reasonable development efforts. The full implementation is done with Visual Studio 2005 and Office SharePoint Server 2007 Beta 2 TR. You can download the full implementation from the web site http://www.microsoft-sap.com/interop.aspx . Since the implementation is just a proof-of-concept implementation and has its focus on the feasibility of the approach we introduced in the whitepaper, we have not taken security and some other aspects into account during the implementation. Therefore, we will give some considerations in the following.

At first, we haven't tried to merge the search results returned from the SharePoint search and the NetWeaver portal search. In the implementation, we have only displayed the search results side-by-side in two different WebParts to demonstrate the search programmability of SharePoint and NetWeaver portal. However, in a real-world search application, it is normally desired to merge the search results in order to get a single list of search results out of them. In this case, you have to make some considerations to deal with the different taxonomies of the search results. This can be normally done by providing a mapping between the different taxonomies. The other technical challenge that you have to take into consideration is the order of the consolidated search results. In this case, some of the common properties, such as the relevance of the particular search result, can be used as the criteria to sort the consolidated search results.

The other aspect that we haven't considered in our application is security, in particular the authentication and authorization of the search request on both SharePoint portal and NetWeaver portal. If we check out our business scenario: because we directly perform search operations from inside of SharePoint portal by utilizing the enterprise search APIs, the authentication and authorization of the search requests are performed directly by the SharePoint portal itself. In other words, the SharePoint portal is responsible to provide the current user the corresponding search result, to which he has access. Hence, the security of search requests on the SharePoint portal side is not critical in our scenario.

In our implementation, we have realized a delegated search to the SAP NetWeaver portal; therefore, this remains the critical part in the whole implementation from the point-of-view of security. In the demo-implementation, we have utilized the middle-tier pattern for authentication and authorization. In other words, the middle-tier application- in our case the custom J2EE search application- authenticates to the backend KM component with a fixed identity and the backend KM never comes to known the user's real identity on the portal layer. This security pattern should be sufficient in the most cases. However, sometimes it is desired to pass the user's identity to the backend KM component. In this case, you should consider using the back-end pattern for authentication and authorization. In this model, the WebParts in the SharePoint portal delegates authentication to the middle-tier application- in our case the custom J2EE search application, which impersonate the WebParts and authenticate to the backend KM component on behalf of the WebParts. With the custom WebPart we implemented in this whitepaper, you can simply read the user credentials from the current SharePoint context and forward the credentials to the J2EE application for authentications. In the second implementation, where we have utilized the BDC features to integrate the SAP search functionality, you can use the Impersonation/Delegation mode in BDC to realize this scenario. In this case, please refer to the BDC documentation on MSDN to get more information about it[8].

[8] SharePoint Server – Business Data Catalog Security Model: http://msdn2.microsoft.com/en-us/library/ms543340.aspx.