

Insider Extra 7

Scripting Web Searches

If the Web Search component seems close to what you want but lacks in some detail, you might want to consider developing your own search page. This is far from being the easiest, best-documented, and most intuitive aspect of Web development, but it works, and when all is said and done, it doesn't require many lines of code.

This method isn't applicable to the Microsoft Office FrontPage 2003 built-in full-text index. It works only with Microsoft Indexing Service and this means that it also requires a server-based Web site residing on a computer running Microsoft Windows 2000 or later.

The material in this chapter isn't intended as a complete reference. To find additional information, refer to "Locating Additional Documentation," later in this chapter.

Figure IE7-1 shows a Web page that performs a custom-coded Web search. Because it has some special debugging features, this probably isn't a Web page you'd choose to let your Web visitors see. It does, however, illustrate the important points. To examine this page in detail, use FrontPage to open the aspsearch/aspsearch.asp page in the Insider Extra Web Site. The Sample Files setup program installs this site at [My Documents]\Microsoft Press\FrontPage 2003 Inside Out\fp11extras.

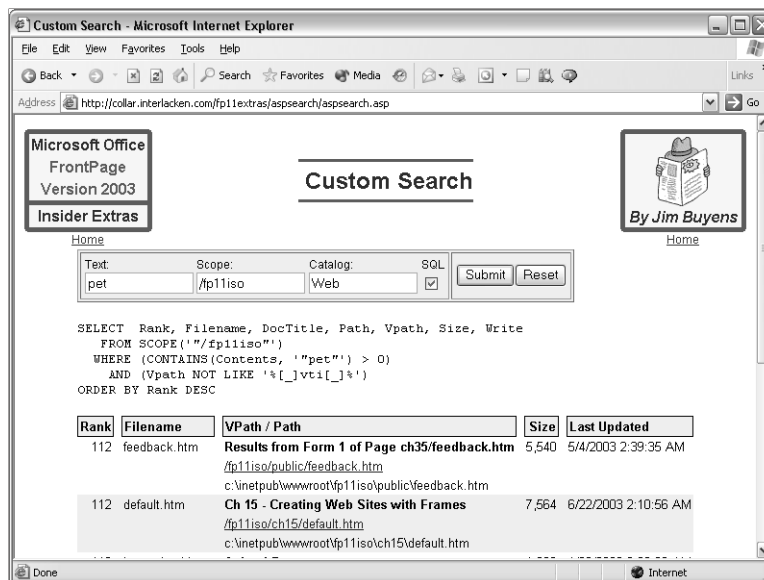


Figure IE7-1. This Web page queries an Indexing Service catalog and produces custom output.

The gray rectangle near the top of the page is an HTML form with these input elements:

- **Text** Specify the word or phrase you want to find.
- **Scope** Specify the URL of a folder on the Web server. This limits the search to the folder tree that begins at that point.
- **Catalog** Specify the name of the Indexing Service or Index Server catalog that indexes the server where the Web site resides. In a default installation, this is the *Web* catalog. The presence of this field is a debugging feature; in a real application, this isn't something you'd expect Web visitors to know or experiment with.
- **SQL** Select this check box to display the SQL statement (shown in Figure IE7-1) that locates the requested pages. This is also a debugging feature.

A form designed for Web visitors shouldn't have the Scope, Catalog, and SQL options shown in Figure IE7-1. Visitors wouldn't know what to enter in the Scope and Catalog fields and even if they did, there's no reason to let anonymous visitors search around your computer at will. The SQL statement appears only for debugging purposes, and again divulges more information that you probably want to share.

In a real search page, you'd either hard-code scope and catalog names in the page or let visitors select from a drop-down list.

Locating an Indexing Service Catalog

In the hope that additional detail will offer more clarification than confusion, Figure IE7-2 shows a list of Indexing Service catalogs on a Windows XP Professional computer named COLLAR.

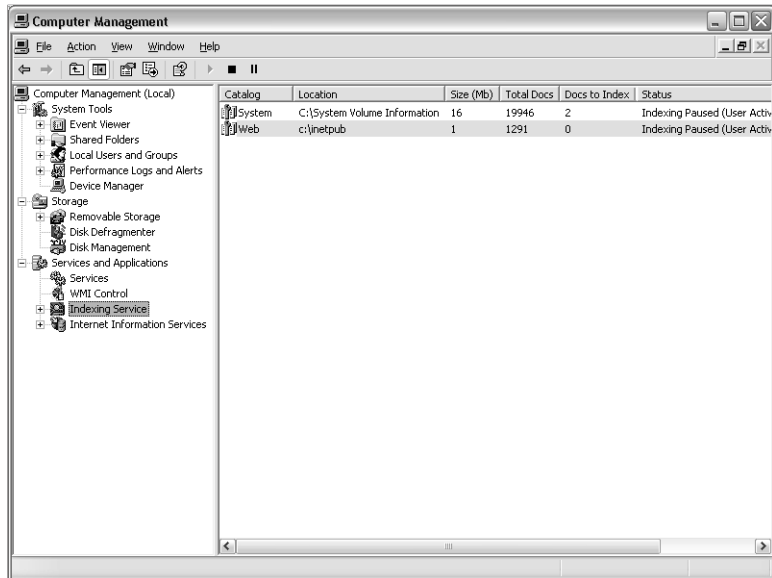


Figure IE7-2. To configure Indexing Service, open its entry in the Computer Management administrative tool.

Scripting Web Searches

To display this information on your own system:

- 1 Choose Computer Management from the Administrative Tools menu. The location of this menu varies somewhat by operating system but here are two ways of finding it:
 - Choose Programs from the Start menu.
 - Open the Control Panel and look for Administrative Tools.

If your Windows XP system doesn't have an Administrative Tools menu, right-click the Windows task bar, choose Properties. Then click the Start Menu tab, the Customize button, and the Advanced tab. Select Display Administrative Tools in the Start Menu Settings list, and then click OK.

This procedure will vary somewhat for other operating systems.

- 2 When the Computer Management window appears, click the plus sign next to Services And Applications.
- 3 Select the Indexing Service item.

Note If your system doesn't have Indexing Service installed, choose Start, click Control Panel, double-click Add/Remove Programs, and then click Add/Remove Windows Components. In the resulting dialog box, select the Indexing Service option, and then click Next on each remaining page of the wizard. Supply your Windows CD when prompted.

Figure IE7-2 shows that the computer COLLAR has two catalogs: *System* and *Web*. The Location column in the right pane shows where each of these catalogs physically resides. To view the folder trees that a particular catalog indexes:

- 1 In the left pane, click the plus sign to the left of Indexing Service.
- 2 Click the plus sign to the left of the catalog you're investigating.
- 3 Select the Directories icon that appears beneath the catalog in question.

Figure IE7-3 shows that the *Web* catalog indexes content in five folder trees. The first tree listed is the HTTP root for the default Web server on this computer. The remaining four trees are the physical locations of virtual directories on the default server. In general, each virtual server on the same computer will have its own Indexing Service catalog, and that catalog will index all physical or virtual directories that belong to that server.

If your Web server's HTTP root resides somewhere not indexed by Indexing Service, the server administrator will need to create a new catalog. (To start this process, right-click the Indexing Service entry in the Computer Management tree, choose New from the shortcut menu, and then choose Catalog). The name you give this catalog is the name you'd enter in the Catalog box of the form pictured in Figure IE7-1.

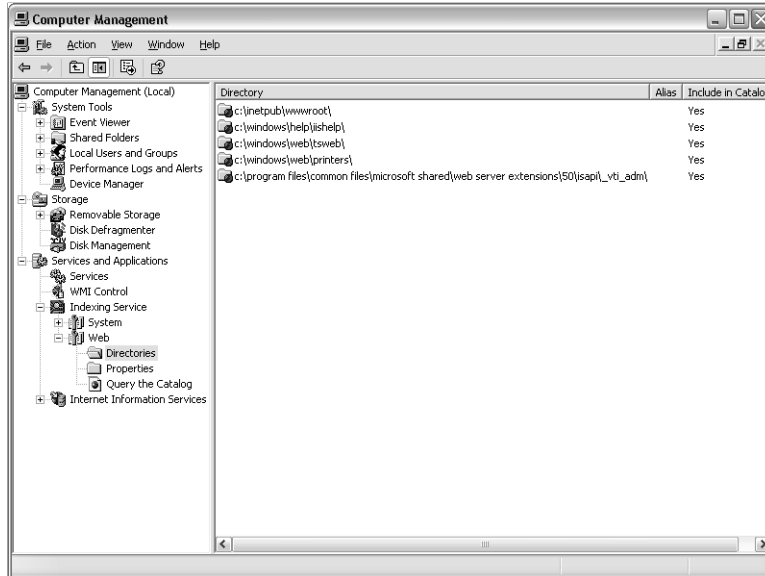


Figure IE7-3. A single Indexing Service catalog can index multiple folder trees. In this case, each tree is a virtual directory belonging to the same Web server.

Virtual Servers, Virtual Directories, and Indexing Service

In the most basic case, a computer providing Web services has one IP address and one DNS name. As a result, it appears to be one Web server. This is eminently simple, but installing a new computer for each Web site is an expensive proposition when the sites are small. Most providers therefore offer *virtual servers*.

A virtual server is a single computer that responds to multiple IP addresses and delivers content from a different HTTP root depending on the IP address. Each IP address has a different DNS name and so appears to be a stand-alone site. Each virtual server that requires Indexing Service thus also requires its own catalog. Otherwise, content that normally appeared on two distinct Web sites would appear mixed together in search results.

A virtual directory, on the other hand, is a folder tree that appears to be part of a Web server's folder tree but physically resides elsewhere. In other words, the logical folder tree a Web visitor sees might be larger and more complex than the folder tree that physically resides on the Web server. The added branches come from virtual directories defined in the Web server's configuration.

Scripting Web Searches

For Indexing Service to produce complete results, each catalog must index not only the Web server's physical folder tree, but all of the virtual folder trees as well. That's why additional physical directories appear in the display shown in Figure IE7-3. Each of the extra *physical* directories hosts a *virtual* directory defined by the Web server.

Indexing Service normally detects the need to list additional virtual directories by means of interfaces to the Web server software. Indexing a new virtual server, however, requires manual configuration.

Coding an Indexing Service Query

The code required to open a connection to Indexing Service, submit a query, and read through all the matching records (the hits) is very similar to that for accessing a database. Here are the salient points:

- The following statement creates a connection string named *cnStIx*. A connection string is the specification for opening a connection to a data provider, such as a database or Indexing Service. In this connection string, *msidxs* means Microsoft Indexing Service. The *source=* setting specifies the name of the Indexing Service catalog. The *qcatalog* variable contains the value entered in the Catalog box of the form in Figure IE7-1.

```
cnStIx = "provider=msidxs;data source=" & qcatalog
```

- The next two statements create and open an *ADODB.Connection* object named *cnIx*. *ADODB* stands for ActiveX Data Objects Database. Notice that the *Open* method on the second line specifies the *cnStIx* connection string as an argument:

```
Set cnIx = Server.CreateObject("ADODB.Connection")
cnIx.Open cnStIx, "", ""
```

- The next three statements create and open an *ADODB.Recordset* object. For a normal database, this type of object contains all records that match a given database query. For an Indexing Service catalog, the object contains one record for each matching file (that is, one record for each search hit):

```
Set rsIx = Server.CreateObject("ADODB.Recordset")
sql = "SELECT ... "
rsIx.Open sql, cnIx, adOpenForwardOnly
```

The SQL statement in Line 2 of this code is obviously incomplete. To view the code that creates the SQL statement shown in Figure IE7-1, open the `aspsearch/aspsearch.asp` page in the sample Web site, and switch to Code view. For more information about the format of an Indexing Service SQL statement, refer to the next section.

The value *adOpenForwardOnly* is a constant defined in a file named *adovbs.inc*. To include this file in any Web page that needs it, add the following line near the top of the <head> section:

```
<!-- #include file="adovbs.inc" -->
```

As coded, this statement assumes that the *adovbs.inc* file resides in the same folder as the *aspsearch/aspsearch.asp* page. Be sure to adjust the *file=* URL if the two files reside in different relative locations.

- The first statement in the following code defines the start of a loop that continues until the current record position of the *rsIx* recordset is past the last record. The *rsIx.MoveNext* statement advances to the next record. Between these two statements you would place whatever code is necessary to create the HTML that displays the search hit.

```
Do While Not rsIx.EOF
    ' Code to display each record goes here.
    rsIx.MoveNext
Loop
rsIx.Close
cnIx.Close
```

The *Loop* statement marks the end of the loop, and the last two statements close the recordset and the connection.

Coding SQL Statements for Indexing Service

Take another look at the SQL statement shown in Figure IE7-1:

```
SELECT Rank, Filename, DocTitle, Path, Vpath, Size, Write
FROM SCOPE('"/fp11iso"')
WHERE (CONTAINS(Contents, "pet") > 0)
AND (Vpath NOT LIKE '%[_]vti[_]%' )
ORDER BY Rank DESC
```

In this statement:

- The field names listed in line 1 are field names in the Indexing Service catalog, not in a database table. To view a list of the available properties, locate the Indexing Service catalog using the Computer Management tool (see Figure IE7-2). Then click the plus sign to the left of the catalog name, and select the Properties folder. The list of available fields appears in the Friendly Name column, shown on the right in Figure IE7-4. (The *Rank* field isn't part of the catalog; it's generated as part of running a query.)
- The *FROM SCOPE* clause specifies the folder tree to search.

Scripting Web Searches

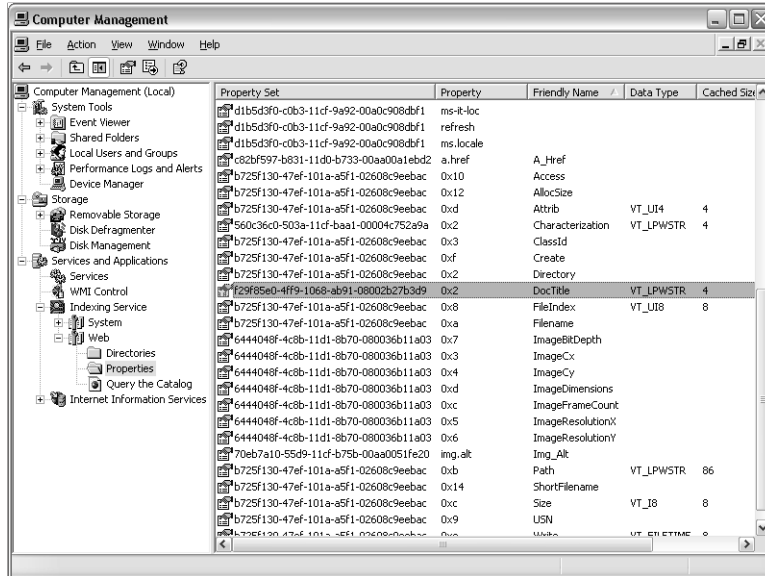


Figure IE7-4. The Friendly Name column in the right pane lists the field name you can use in SQL statements.

- The *WHERE* clause in this example specifies two conditions. The first condition demands that the *Contents* field contain the word *pet*. This field contains the main content of a file, such as the textual portion of a Web page. The second condition demands that the file's virtual path (that is, its URL) not contain the string *_vti_*. This excludes FrontPage system files from the search results.

The percent signs are wildcard characters that mean “any string of text.” Underscore characters are normally wildcard characters that mean “any single character,” but the square brackets tell the data provider to treat them as ordinary characters.

- The *ORDER BY* clause specifies how to sort the results.

Accessing a field in an Indexing Service result set follows the same pattern as accessing a field from a database query. The following expression, for example, retrieves the value of the *filename* field of the *rsIx* recordset and writes it into the outgoing HTML:

```
<%=rsIx("filename")%>
```

Locating Additional Documentation

If your Web server runs Windows 2000, you can get additional information about Indexing Service at www.microsoft.com/windows2000/en/server/help. To locate the Indexing Service topic, first open the Files And Printers topic, as shown in Figure IE7-5.

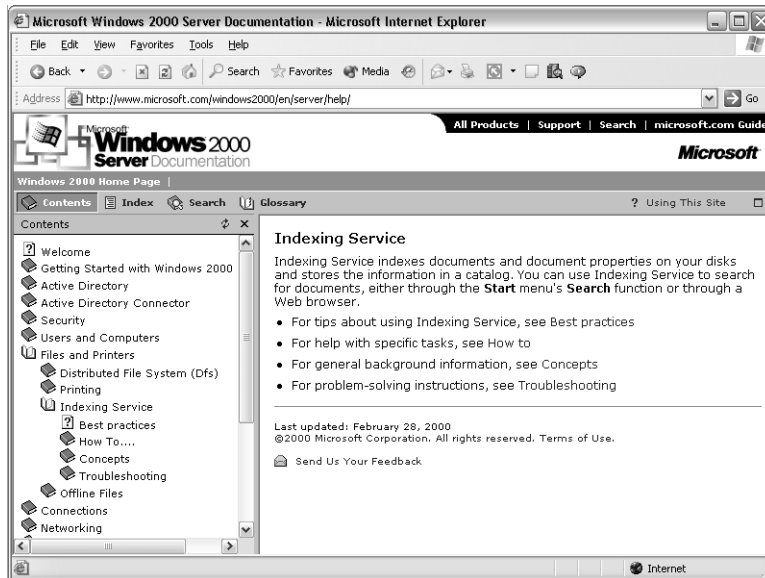


Figure IE7-5. Information about the Windows 2000 Indexing Service is available from Microsoft's Web site.

To find out more about ADO access to the Indexing Service catalog, browse the MSDN Library, at msdn.microsoft.com/library. Expand the following topics in the order listed:

- Data Access
- Reference
- Microsoft ActiveX Data Objects (ADO)
- ADO Programmers Guide
- Appendixes
- Appendix A: Providers
- Microsoft OLE DB Provider for Microsoft Indexing Service

Figure IE7-6 illustrates this path. Be forewarned that the path changes somewhat from time to time.

Scripting Web Searches

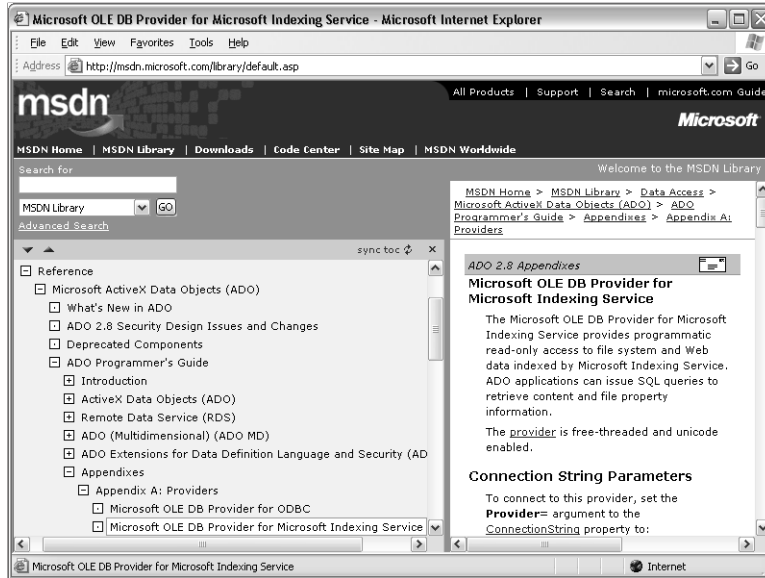


Figure IE7-6. Documentation on using ADO to search Indexing Service catalogs is available from the MSDN Library location shown here.

Another method for finding information about this topic is to browse the search page at search.microsoft.com/us/dev/default.asp, searching for the phrase *Microsoft OLE DB Provider for Microsoft Indexing Service* or for the provider name *msidxs*.

