

Insider Extra 6

Scripting Dynamic Tables of Contents

If your Web site changes rapidly, tables of contents and other kinds of menus might become outdated faster than Microsoft Office FrontPage 2003 components or your own most diligent efforts can update them. Often, these situations occur when external processes add files on no fixed schedule. Web visitors might upload files such as photographs or résumés, for example, or an automated process might deposit files representing business activity in your Web site.

Configuring the Web Server to Display Folder Contents

Most Web servers have a *directory browsing* feature that can display the content of a Web folder automatically. Use of this feature requires that both of the following conditions are true:

- The folder where the incoming files reside doesn't contain a Web page with the Web server's default page name.
- The Web server is configured to permit directory browsing. The Web server's administrator must configure this setting.

For information on configuring Internet Information Services to permit Directory Browsing for a given folder, refer to, "Configuring Home Directory Settings," in Appendix O.

If both of these conditions are true, browsing the given folder will display a listing like the one shown in Figure IE6-1. The Web server sets up a hyperlink surrounding each listed file name so that clicking on any file requests that file.

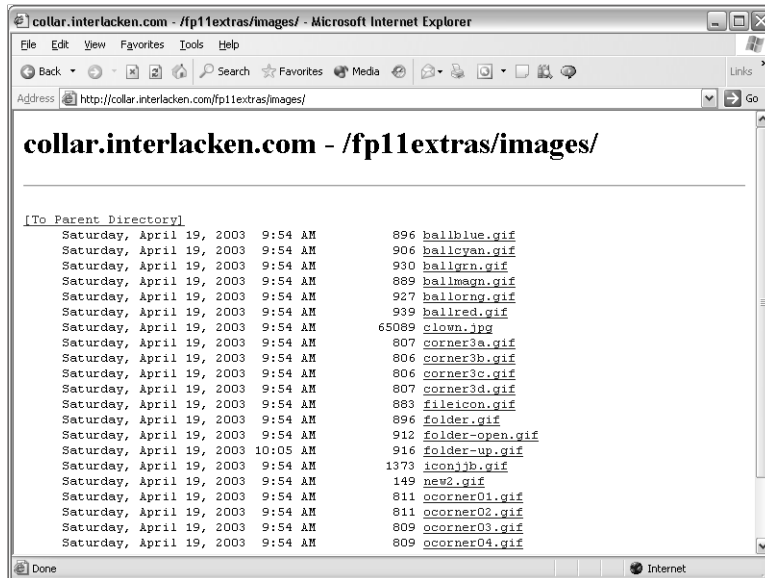


Figure IE6-1. Directory Browsing is a Web server feature that sends Web visitors a hyperlinked folder listing like this one. An administrator must enable this feature, and the Web visitor must browse a folder that contains no default home page.

Using ASP to Display Folder Contents

Straight folder listings like the one in Figure IE6-1 have their place, but for many applications, you'll want something better—something that looks a bit more professional and a lot less like it came out of somebody's bit bucket. As a starting point, Microsoft provides the `dyntoc.asp` ASP page shown in Figure IE6-2. This ASP page runs a script on the Web server that lists the contents of a specified folder and lists all the files name that exist there.



On the Web

For more information about the `dyntoc.asp` page, refer to article Q218606, "How To Use ASP and Scripting.FileSystemObject to Create a Dynamic Table of Contents Page," in the Microsoft Knowledge Base.

To view the source code for this page, first use FrontPage to open the Insider Extra Web site that the Sample Files setup program installs at [My Documents]\Microsoft Press\FrontPage 2003 Inside Out\fp11extras. Then, open the `dyntable` folder and finally the `dyntoc.asp` page. To view the ASP code, click the Code tab at the bottom of the Design view window.

Scripting Dynamic Tables of Contents

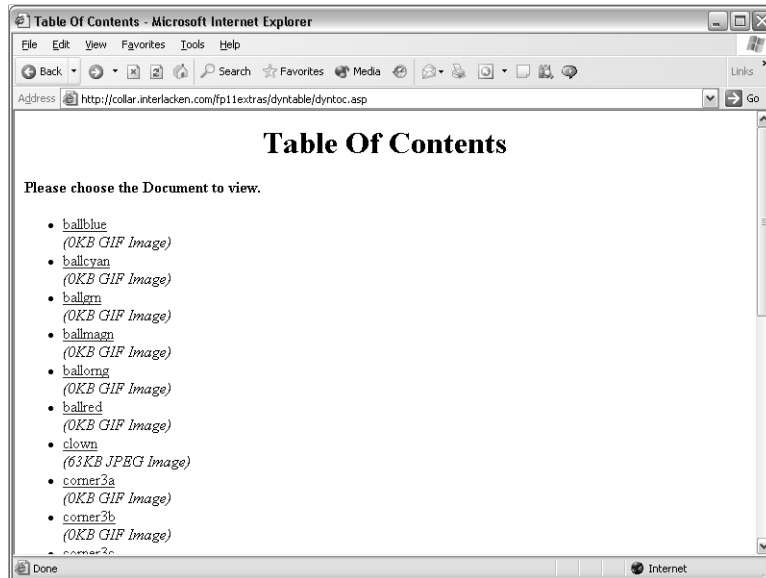


Figure IE6-2. This ASP page reads a folder on the Web server and displays a current table of contents.

A line-by-line explanation of this ASP page is beyond the scope of this book, but the following statements are the ones that do all the work:

```
strDocsPath = GetAppPath() & "images"
strDocsPhysicalPath = Server.MapPath(strDocsPath)
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")
Set objFolder = objFSO.GetFolder(strDocsPhysicalPath)
Set objFiles = objFolder.Files
For Each objFile in objFiles
' Code to display each file name goes here.
Next
```

These statements do the following:

- The first statement stores a URL path in a variable named *strDocsPath*. The ASP page will list the contents of this path. The *GetAppPath()* function returns the root URL of the current application.
- The second statement converts the URL path in *strDocsPath* to a physical file path on the Web server.
- The third statement creates an element named *Scripting.FileSystemObject*. This is a software component that provides a number of useful methods for inspecting and modifying the server's local file system.
- The fourth statement uses the *FileSystemObject's GetFolder* method to create a *Folder* object for the given physical file path. A *Folder* object provides access to the physical properties of the given folder.

Microsoft Office FrontPage 2003 Inside Out

- The fifth statement establishes *objFiles* as an alias for the *objFolder.Files* collection. The *Files* collection of any *Folder* object contains a *File* object corresponding to each file in the physical folder.

After this statement executes, the *objFiles* variable points to a list of *File* objects—one for each file in the given folder.

- The sixth and eighth statements set up a loop that successively points the *objFile* variable to each *File* object in the *objFiles* collection. Within this loop:
 - *objFile.Name* returns the file name (sans path) of the current file.
 - *objFile.Type* returns the file type of the current file.
 - *objFile.Size* returns the size of the current file.

Name, *Type*, and *Size* are properties that all *File* objects have.

The following is a listing of the code that displays each file name, somewhat reformatted from the Microsoft version for readability:

```
For Each objFile in objFiles
    strName = objFile.Name          ' get a file's name
    strFile = Lcase(strName)        ' make it lowercase for the URL
    strType = objFile.Type          ' get the file's type
    strName = MakeTitle(strName)    ' make the name a title for display
    lngSize = objFile.Size\1024     ' get the file size in KB
    ' output the filename and URL
    Response.Write "<li>" & _
        "<a href="" & strDocsPath & "/" & strFile & "">" & _
            strName & "</a><br>"
    ' output the file's size and type
    Response.Write "<em>(" & lngSize & "KB " & strType & ")</em>" & _
        "</li>" & vbCrLf
Next
```

Lines 2 through 6 copy various properties of the *objFile* object into ordinary variables and prepare them for display. The two *Response.Write* statements write into the output stream destined for the Web visitor's browser.

Coding Quotation Marks As Literal Values

Visual Basic requires that you enclose all literal string values in quotation marks. For example, to send the characters `<html>` to a browser, for example, you must code:

```
Response.Write "<html>"
```

The quotation marks in this context aren't part of the value; they indicate the start and end of the value. This presents a problem if the value itself must contain a quotation mark. Here's an example:

```
Response.Write "Tom exclaimed, "Oh no!" as the cake exploded."
```

Scripting Dynamic Tables of Contents

If you tried running this line of code, Visual Basic would interpret

```
Response.Write "Tom exclaimed, "
```

as a correct statement and

```
Oh no!" as the cake exploded."
```

as extra garbage. Fortunately, Visual Basic provides a solution. To indicate that a quotation mark is part of a literal value, code two quotation marks instead of one. Here's the correct way of coding the previous example:

```
Response.Write "Tom exclaimed, ""Oh no!"" as the cake exploded."
```

This statement sends the browser a total of two quotation marks: one before *Oh* and one after *no!*

Microsoft obviously designed this example to illustrate the ASP code and not to represent the state of the art in Web design. The Web page shown in Figure IE6-3 provides a somewhat better appearance, the ability to move up and down a folder tree, and the ability for a query string to specify a folder location, but these are minor enhancements rather than significant new concepts. To view the ASP code or any other details relating to this page, open the `dyntable/dyntable.asp` file in the Insider Extra Web site.

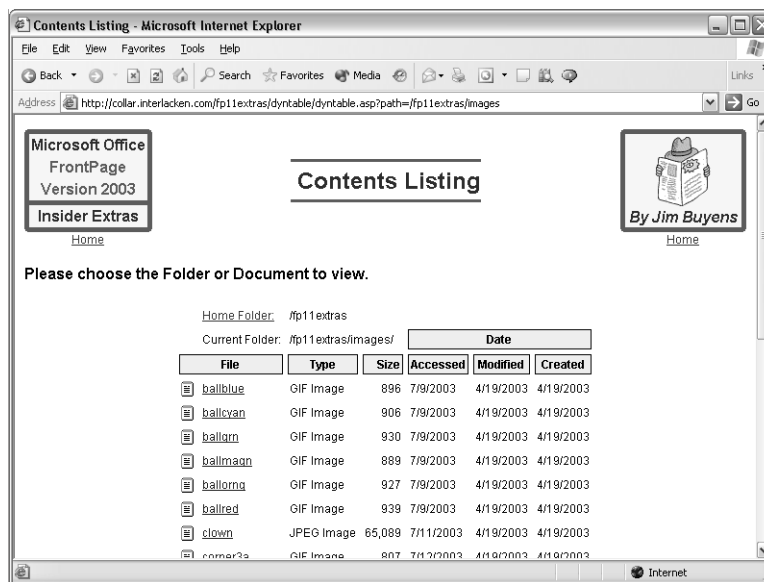


Figure IE6-3. This Web page looks more like something you'd want visitors to see, but file names still don't provide much guidance as to what each hyperlink displays.

Modifying the Contents Listing Page

You are, of course, free to make whatever changes you want to the `dyntable.asp` page. Just remember that doing so requires knowledge of ASP programming, which is something you need to acquire on your own. The supplied code does have special provisions for making two kinds of changes:

- To restrict `dyntable.asp` so that it can display only files in a given folder tree, remove the comment indicators from the following lines, and set the `basePath` variable equal to the path you consider legal. This path must be relative to the server root, and it must begin and end with slashes (/).

```
'dim basePath
'basePath = "/fp11Extras/"
'if Lcase(Left(strHome, len(basePath))) <> basePath then
'    strHome = basePath
'end if
'if Lcase(Left(strPath, len(basePath))) <> basePath then
'    strPath = basePath
'end if
```

- To restrict `dyntable.asp` so that it displays only files named with certain extensions, remove the comment indicators from the following lines, and then edit the second *If* statement to select the extensions you want. The sample code displays only the file extensions `.htm`, `.html`, `.asp`, and `.aspx`.

```
' If showFile Then
'     If (fileExtn = "htm") Or (fileExtn = "html") _
'       Or (fileExtn = "asp") Or (fileExtn = "aspx") Then
'         Else
'             showFile = False
'         End If
'     End If
```

Unlike the browser listings that Web servers generate, directory listings that ASP pages create can filter the file listings, display only those informational fields you want, and let Web visitors switch to other folders if they want. But the fact remains that file names, even with the best of naming conventions, usually make poor menu choices.