

Insider Extra 11

Developing Custom Web Components

With only a little coding effort, you can create your own Web Components and integrate them seamlessly into the Microsoft Office FrontPage 2003 environment. Figure IE11-1, for example, shows the Insert Web Component dialog box, displaying five components developed just for this book.

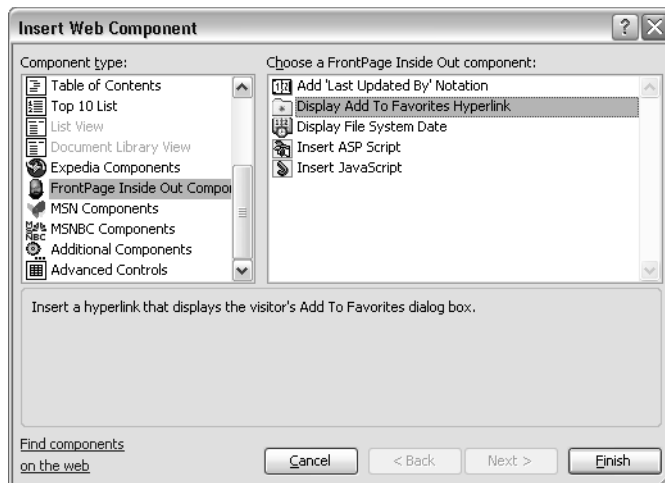


Figure IE11-1. The five Web Components listed in the Choose A FrontPage Inside Out Component list on the right are unique to this book.

Here are the five Web Components. The first is interactive; the rest are not.

- **Add 'Last Updated By' Notation** Adds a message containing your name and the current date.
- **Display Add To Favorites Hyperlink** Adds a browser-side script that determines whether the current browser is Microsoft Internet Explorer and, if so, displays a hyperlink to the Web visitor's Add To Favorites dialog box. This makes it easier for Web visitors to bookmark your page.
- **Display File System Date** Adds a browser-side script that displays the last date the Web page was changed, as indicated in the Web server's file system.

Microsoft Office FrontPage 2003 Inside Out

- **Insert ASP Script** Adds an empty block of ASP script code to a Web page at the current insertion point. By running this component and then double-clicking the resulting icon to display the Server Script dialog box, you can add server-side script code to your Web page without ever switching to HTML view.
- **Insert JavaScript** Adds an empty block of browser-side JavaScript code to a Web page. To add JavaScript code to your Web page without using HTML view, run this component, double-click the resulting icon to display the HTML Tag dialog box, and then enter the script code.

For more information about Add To Favorites hyperlinks, refer to “Helping Visitors Set Up Shortcuts,” on page 423.

For more information about displaying a Web page’s file system date, refer to the Frequently Asked Question sidebar “How Can a Web Page Display Its File System Date?” on page 722.

For more information about the Server Script dialog box and about the HTML Tag dialog box, refer to Insider Extra 8 “Scripting Time-Sensitive Content.”

Installing Web Components

To try these components on your own system, you must first locate the Microsoft Office webcomp folder. The exact path varies somewhat depending on your system and the version of Office or FrontPage that you’re using, but it will probably resemble the following:

C:\Program Files\Microsoft Office\Office11\1033\webcomp

Note 1033 is the numeric language identifier for English. If your copy of Office is localized for another language, this number will be different.

One way to find this folder is to search all local hard drives for a folder named webcomp. This folder should be inside a \Program Files\Microsoft Office\ folder tree, and it contains a small collection of INI files and like-named folders. For example, you’ll probably see file names like expedia.ini, msn.ini, and msnbc.ini, plus matching folders named expedia, msn, and msnbc.

To install the Insider Extra components, proceed as follows:

- 1 Run the Sample Files setup program from the companion CD
- 2 Copy all the files and folders from the [My Documents]\Microsoft Press\FrontPage 2003 Inside Out\webcomp folder to the Office webcomp folder.
- 3 If the path to your webcomp folder isn’t `C:\Program Files\Microsoft Office\Office11\1033\webcomp`, open the fpiso.ini file, and modify the last line to reflect the path on your system.
- 4 If FrontPage is running, quit and restart it.

That’s it! The next time you display the Insert Web Components dialog box, the components should be there. (If not, renew your search for the real webcomp folder.)

Configuring Web Components

As you've no doubt noticed, the Insert Web Component dialog box contains two lists: one on the left that lists groups of components, and another on the right that lists components within each group. The next section will explain how to define an entry that appears in the list of components groups on the left. Later sections will explain how to configure noninteractive components and then interactive components that appear in the list on the right.

Configuring Web Component Groups

To see how Web Components work, open the `fpiso.ini` file with Notepad or your text editor of choice. The first few lines contain the following code:

```
[Component]
Name="FrontPage Inside Out Components"
Caption="C&hoose a FrontPage Inside Out component:"
Sorted=True
Type=IconAndText
ImageFile="icojbb.ico"
```

Here's what each statement in this block of code does:

- **[Component]** This is a required entry that indicates the start of the Web Component definition.
- **Name** Specifies the text that appears in the Component Type list on the left in the Insert Web Component dialog box. For proof, compare the value shown in this code to the highlighted entry in Figure IE11-1.
- **CommentText** This optional setting applies only to components that don't have suboptions. In that case, it specifies the text that will appear in the comment surrounding the component's HTML, and also in the Undo drop-down list.
- **Caption** If the component has suboptions, this setting provides a title that appears over the selection list on the right in the Insert Web Component dialog box. Again, you can verify this by comparing the code to the figure.
- **Sorted** Tells FrontPage whether to sort the list of components alphabetically or to present them in the order in which they appear in the INI file:
 - **True** Specifies sorting.
 - **False** Specifies no sorting.
- **Type** Tells FrontPage how to display the group's entry in the Component Type list:
 - **IconAndText** Tells FrontPage to display an icon followed by a text description.
 - **ImageFile** Tells FrontPage to display a picture file.

If this setting is absent or set to any other value, FrontPage displays a text description only.

Microsoft Office FrontPage 2003 Inside Out

- **ImageFile** If *Type* is *IconAndText*, this setting specifies the name of the Windows icon file. If *Type* is *ImageFile*, it specifies the name of the picture file.

In either case, the file you specify must reside in a subfolder of the webcomp folder, with the same name as the INI file less the .ini file extension. For example, any icons that you specify in the file webcomp\fpiso.ini must reside in the folder webcomp\fpiso\.

You can specify any size icon you want, but FrontPage expands or shrinks it to 16 pixels square. Therefore, you might as well specify a 16x16 pixel icon.

Note If you don't have a graphics editor that works effectively with ICO files, try Axialis AX Icons. You can download and purchase this software from www.axialis.com/order/index.html.

The next group of statements provides an internal name for each Web Component in the current group:

```
[Component.Options]
Option1=InsAddFavs
Option2=InsFsDate
Option3=InsASP
Option4=InsJscript
Option5=LastUpdBy
```

Here's what each of these lines does:

- **[Component.Options]** Tells FrontPage that the following statements enumerate the internal names of the Web Components in this group.
- **Option1 through Optionn** Supply the internal name of one component each. Number these statements sequentially, and use as many of them as you have components. Pick any names you want, but keep them short, free of punctuation, free of spaces, and meaningful.

Configuring Noninteractive Web Components

The remaining sections in the fpiso.ini file define the individual components. The first four components—InsJscript, InsASP, InsFsDate, and InsAddFavs—are all noninteractive and therefore quite similar. The following INI section, for example, describes the InsAddFavs option:

```
[InsAddFavs]
Name="Display Add To Favorites Hyperlink"
Description="Insert a hyperlink that displays the visitor's Add To Favorites dialog box."
ImageFile="favorite.ico"
```

Developing Custom Web Components

Here's the function of each of these lines of code:

- **[InsAddFavs]** Tells FrontPage that this block of statements describes the InsAddFavs option.
- **Name** Specifies the text that appears in the component selection list on the right in the Insert Web Component dialog box. (Again, compare the value shown in this code to the highlighted entry in Figure IE11-1.)
- **Description** Specifies the text that appears in the large gray box near the bottom of the Insert Web Component dialog box. Enter this text on a single line.
- **ImageFile** Specifies the name of a Windows icon file. The picture in this file precedes the component's name in the component selection list.

A second INI section supplies the HTML that FrontPage adds to the Web page every time you insert an InsAddFavs component. Here's an example:

```
[InsAddFavs.HTML]
HTML1=<script language="JavaScript">
HTML2=if (navigator.appName.substring(0,9) == "Microsoft"){
HTML3= document.write("<p><a href='" +
HTML4=   "javascript:window.external.addFavorite" +
HTML5=   "(document.location,document.title)'">" +
HTML6=   "Add to favorites</a></p>")
HTML7=}
HTML8=</script>
```

This code is fairly self-explanatory, but here, nevertheless, are the explanations:

- **[InsAddFavs.HTML]** Identifies this block of statements as the HTML that constitutes the component. The section name must be the internal name of the component, followed by a period (.) and the letters *HTML*.
- **HTML1 through HTMLn** Provides the required HTML or script statements.

That's it! With no more work than this you can add an option to the Insert Web Component dialog box that inserts whatever HTML or script code you want. There are, however, three precautions regarding any HTML you insert:

- In almost every case, the easiest way to create and test the HTML is in a stand-alone Web page. Get the code working first, and then make a Web Component out of it.
- FrontPage writes no line endings into the HTML that a Web Component inserts. In the previous example, no line ending occurs between the HTML1 code and the HTML2 code. This usually isn't a problem for HTML and JavaScript, which treat line endings the same as spaces anyway. Line endings are significant, however, in all forms of Microsoft Visual Basic. Take this into account when you enter VBScript code. Separate statements with colons, and don't code line continuation characters.

Microsoft Office FrontPage 2003 Inside Out

- Updating the HTML code for a noninteractive Web component has no effect on component's you've already added to Web pages.

The code for the remaining noninteractive Web Components follows the same pattern: the `InsFsDate` option has `[InsFsDate]` and `[InsFsDate.HTML]` sections, the `InsASP` option has `[InsASP]` and `[InsASP.HTML]` sections, and so forth.

Configuring Interactive Web Components

The Add 'Last Updated By' Notation component is interactive, and thus requires a somewhat different configuration. For starters, it requires only one section in the `fpiso.ini` file:

```
[LastUpdBy]
Name="Add 'Last Updated By' Notation"
Description="Add a visible comment that you updated this page today."
ImageFile="LastUpdBy.ico"
URL="C:\Program Files\Microsoft Office\Office11\1033\webcomp\FPISO\LastUpdBy.htm"
```

The section name and the *Name*, *Description*, and *ImageFile* settings have the same meanings as they do for noninteractive components. The *URL* setting gives the location of the first Web page that FrontPage will display when configuring the component. Figure IE11-2 illustrates this page for the `LastUpdBy` component.



Figure IE11-2. This dialog box is actually a Web page that configures an interactive Web component.

Although this example uses a Web page that resides on the designer's local disk, you can also specify Web pages that reside at an `http://` location.

Developing Custom Web Components

For additional simplicity, this example uses a single Web page rather than a series of Web pages that constitute a wizard. If you decide to use multiple pages, it's your responsibility to pass data from one page to the next, to create Next, Back, Cancel, and Finish as appropriate on each page, and to program the behavior of those buttons. Typically:

- The first page in a wizard contains Next and Cancel buttons.
- Any interior pages contain Back, Next, and Cancel buttons.
- The last page contains Back, Finish, and Cancel buttons.

When the designer clicks the Finish button, you should arrange for a series of JavaScript statements like this to execute:

```
window.external.WebComponent.PreviewHTML = <value 1>;
window.external.WebComponent.HTML = <value 2>;
window.external.WebComponent.Tag = <value 3>;
window.external.Close(true);
```

Here's what these statements do:

- The first statement sends FrontPage the HTML it should use when displaying the component in Design view. Replace *<value 1>* with a variable or literal that contains the HTML you want.
- The second statement sends FrontPage the HTML it should add to the Web page every time you save the page.
- The third statement appears to serve no official use. Curiously, however, it defaults to *"body"*. In practice, you can save whatever value you want in this property, and that value will then be available the next time the designer configures the component.
- The fourth statement closes the window and returns control to FrontPage.

JavaScript statements like these normally can't affect software or settings outside the browser window. They work in this case only because the Web page is running in a special FrontPage environment.

Given this information, you can now understand how the Web page shown previously in Figure IE11-2 works. The page contains an HTML form named *form1* with a text box named *YourName*. The following HTML defines the Finish button:

```
<button onclick="insertHTML();">&nbsp;Finish&nbsp;</button>
```

Microsoft Office FrontPage 2003 Inside Out

Clicking this button runs the following JavaScript function. The first line denotes the start of the function and gives it a name. The last line marks the end of the function.

```
function insertHTML(){
    tempname = document.form1>YourName.value;
    tempname = tempname.replace(/(^\\s*|\\s*$)/g, "");
    if (tempname == ""){
        document.form1>YourName.value = ""
        msgName.innerText = "Required!";
    }else{
        tagOut = "<p>Updated " + fmtDate + " by " + tempname + "</p>";
        window.external.WebComponent.PreviewHTML = tagOut;
        window.external.WebComponent.HTML = tagOut;
        window.external.WebComponent.Tag = tempname;
        window.external.Close(true);
    }
}
```

Line 2 retrieves the contents of the text box and saves it in a variable named *tempname*. Line 3 uses a regular expression to remove any leading or trailing spaces.

Line 4 determines whether the resulting value is empty. If so, it clears any spaces or other garbage out of the text box and displays an error message. If not, the code following the *else* takes these actions:

- 1 Stores a `<p>` tag in a variable named *tagOut*. This `<p>` tag contains the keyword *Updated*; the date, in a variable named *fmtDate*; the keyword *by*; and finally, the name retrieved from the text box.
- 2 Sends this tag to FrontPage as the HTML to display in Design view.
- 3 Sends the same tag to FrontPage as the HTML to put in the page during each Save.
- 4 Sends the name retrieved from the text box to FrontPage as the value to save in the *Tag* property.
- 5 Closes the window.

This explanation begs two questions: Where did the date in *fmtDate* come from, and why bother saving the designer's name in the *Tag* property?

In answer, it turns out that the `<body>` tag of this Web page contains the following code:

```
<body bgcolor="#f0f0f0" scroll="no" onload="setDefaults()">
```

Developing Custom Web Components

The `bgcolor="#eeeeee"` attribute sets the background color, and the `scroll="no"` attribute prevents the appearance of scroll bars. The `onload="setDefaults()"` attribute is much more important to this discussion. It runs the following JavaScript function:

```
function setDefaults() {
    curDate = new Date();
    months = "JanFebMarAprMayJunJulAugSepOctNovDec";
    mo = curDate.getMonth();
    mon = months.substring(mo * 3, (mo * 3) + 3);
    fmtDate = curDate.getDate() + "-" + mon + "-" + curDate.getYear();
    lastUpd.innerHTML = fmtDate;

    tempName = window.external.WebComponent.Tag;
    if (tempName == "body"){
        document.form1.YourName.value = "";
    }else{
        document.form1.YourName.value = tempName
    }
}
```

As before, the first line marks the beginning of the function and gives it a name. The last line marks the end of the function.

Line 2 creates a new *Date* object named *curDate*, initialized by default to the current date and time. Line 3 defines a string variable named *months* that contains the three-letter abbreviation for each month in the year. The *getMonth* method on line 4 returns the current month as a number (0 for January, 1 for February, and so forth) and saves this in a variable named *mo*. Line 5 uses the *mo* value, some arithmetic, and the *substring* method to extract the corresponding three-letter month abbreviation from the *months* variable. Line 6 creates the long-awaited *fmtDate* value using the day of the month from the *curDate* object, the month abbreviation just obtained, and the year from the *curDate* object. Line 7 displays this value in an HTML tag named *lastUpd*. This is an HTML table cell just to the right of the cell that contains the text *Last Updated*.

The code following the blank line retrieves the *Tag* property, if any, that the *insertHTML* function saved when the designer last configured the component. If the resulting value turns out to be *body* (the curious default value), the code initializes the *YourName* text box with an empty string. Otherwise, it initializes that text box with the value from the *Tag* property. This saves designers from reentering their names each time they configure the component.

Because this page loads from disk, JavaScript code embedded in the page must perform any validation, and the Finish button invokes the *insertHTML* function directly. In a more typical situation, the component would invoke an ASP or ASP.NET page on your Web server, and the Finish button would submit a form request across the network. Code in the ASP or ASP.NET page would validate the form input against databases or other resources on

Microsoft Office FrontPage 2003 Inside Out

the server. If validation failed, the code would resend the page with an error message. If validation passed, the server-side code would trigger execution of the browser-side *insertHTML* function, perhaps by adding code such as the following near the end of the Web page:

```
<script language="JavaScript">
insertHTML();
</script>
```

To view all the code in this Web page, open the LastUpdBy.htm file in the [My Documents]\Microsoft Press\FrontPage 2003 Inside Out\webcomp\fpiso folder that the Sample Files setup program installs from the companion CD.

To create your own groups of components, just add an INI file and a matching folder to the webcomp folder, and create INI file entries like the ones described here. For more examples, open and inspect any other INI files in the webcomp folder.

For more information about these techniques, browse msdn.microsoft.com/library, and open the following series of topics:

- Office Solutions Development
- Microsoft FrontPage
- Microsoft FrontPage 2003
- SDK Documentation
- Developing Solutions
- Web Components

The exact names and organization will, of course, change over time. If all else fails, search for *FrontPage SDK*. If you can't find FrontPage 2003 information, consult the FrontPage 2002 documentation; the implementation of Web Components doesn't seem to have changed.