Chapter 18

Advanced Design Techniques

Using Built-In Web Parts 577
Adding ASP.NET Server Controls 584
Installing and Using Office 2003 Web Parts and Components 588
Customizing Site Definitions 591

This chapter describes a number of techniques that are useful when designing SharePoint sites, but that require administrative privileges to install, enable, or deploy.

Windows SharePoint Services includes a number of built-in Web Parts that don't relate directly to lists, libraries, or data sources. In some cases, these might not even appear in your collection's Web Part gallery, and a collection administrator will need to add them. The first section in this chapter describes these Web Parts.

If you're developing Web Part Pages from scratch rather than using a predesigned template, you'll undoubtedly want your pages to include full-text search forms, Modify My Page links, Sign In buttons for anonymous sites, and "Up To" links. Windows SharePoint Services provides some ASP.NET server controls that are very useful in this regard, and this chapter explains how to use them.

The next major topic concerns the Office 2003 Web Parts and Components, a downloadable package that increases the integration between Windows SharePoint Services and Office 2003. This topic explains how a server administrator can download and install these components, and then briefly how team members can use them.

The last topic concerns site definitions, which behave in some respects like site templates but which, in fact, operate much closer to the internals of Windows SharePoint Services.

Using Built-In Web Parts

This section explains how to use some of the miscellaneous Web Parts that come with Windows SharePoint Services. None of these Web Parts makes any direct use of lists, libraries, or data sources on the server, or of ActiveX or Office 2003 components on the Web visitor's computer.

Occasionally, these Web Parts will appear with slightly different names. The Page Viewer Web Part, for example:

- Has the name Page Viewer Web Part in the site collection gallery.
- Has the name MSPageViewer.dwp on the Web Part Gallery page.
- Has the class name *Microsoft.SharePoint.WebPartPages.PageViewerWebPart* on the New Web Parts page.

- Has the suggested File Name PageViewerWebPart on the New Web Parts page.

Nevertheless, all these names point to the same Web Part. Similar variations in name occur for other Web Parts in this section.

Configuring the Title Bar Web Part

The Title Bar Web Part displays (what else?) the title bar of most pages having the standard SharePoint look and feel. In Figure 18-1, a Title Bar Web Part displays:

- The two light-colored lines.
- The icon.
- The caption, "Miscellaneous Web Parts."
- The title, "Title Bar, Image, Members, and Page Viewer."
- The Modify Shared Page link.

Figure 18-1. This page uses four built-in Web Parts: Title Bar, Image, Members, and Page Viewer.

Once you have a Title Bar Web Part on your page, you can configure the following properties:

- **Title** A one-line name that appears in a large font within the Web Part display.
- **Caption** One line of text that appears in a small font above the title.

- **Description** A sentence or two that appears as a tool tip when the mouse passes over the title bar.
- **Image Link** The URL of the icon picture that should appear just before the title.

> **The URL path for all the standard SharePoint icons is */_layouts/images/*, which equates on the server to the physical path C:\Program Files\Common Files\Microsoft Shared\web server extensions\60\TEMPLATE\IMAGES. However, you can locate custom icon files wherever you want.**

If the page contains any Web Part Zones that the current team member can modify, the Title Bar Web Part will automatically display a Modify Shared Page or Modify My Page link. However, Title Bar Web Parts don't automatically display Search Form controls. If anonymous access is enabled and the current visitor hasn't signed in, the Web Part will also display a Sign In button

> **For information on adding a Search Form control to a Web Part Page, refer to "Using the ViewSearchForm Control" later in this chapter.**

A Title Bar Web Part doesn't display the top navigation bar in Figure 18-1 or, for that matter, in any other page. To construct a top navigation bar by hand, open the page in FrontPage 2003 and then:

1. Create a one-row, three-column table with a width of 100%, and with a border thickness, cell spacing, and cell padding of zero.
2. Assign the CSS class name *ms-bannerframe* to the table.
3. For each cell, set the CSS class name to *ms-banner*, no wrap to *true*, and vertical alignment to *middle*.
4. Set the width of the middle cell to 100%, and the horizontal alignment of the right cell to *right*.
5. Display an icon in the left cell. For pages in the root folder of a SharePoint site, the URL of the default icon is *_layouts/images/logo.gif*.
6. Add a FrontPage Link Bar With Custom Links component to the middle cell. Configure this component to display the link bar named SharePoint Top Nav Bar.
7. Add the following tag to the rightmost cell. This displays the Up To links.

```
<SharePoint:PortalConnection runat="server" />
```

8. Make sure the page contains this tag, before the <body> tag. Otherwise, the tag from step 7 will raise an error.

```
<%@ Register Tagprefix="SharePoint"
Namespace="Microsoft.SharePoint.WebControls" Assembly="Microsoft.SharePoint,
Version=11.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>
```

9. Be sure to save the page with a file name extension of .aspx.

Displaying Pictures with the Image Web Part

This Web Part simply displays a picture file. You specify the URL of the picture file, and the Web Part wraps it in an <img> tag and the usual Web Part border. Figure 18-1 shows an example of this Web Part in a page.

You could display the same picture by editing the page in FrontPage, but using the Image Web Part has two advantages:

- You can allow Web Visitors to add, modify, or delete Image Web Parts using only a browser.
- You can build Web Part connections from other Web Parts, and thereby control which picture the Web Part displays.

When you configure an Image Web Part, you can specify these properties:

- **Image Link** The URL of the picture you want to display.
- **Image Vertical Alignment** The vertical position the picture will occupy within the Web Part frame: Top, Middle, or Bottom. The default is Middle.
- **Image Horizontal Alignment** The horizontal position the picture will occupy within the Web Part frame: Left, Center, or Right. The default is Center.
- **Web Part Background Color** The background color that will fill the Web Part's frame. The default is Transparent.

To configure the title, size, frame appearance, visibility, and configurability of the Web Part, use the Appearance, Layout, and Advanced sections of its property sheet, just as you would for any other Web Part.

Configuring the Members Web Part

This Web Part displays a list of team members who have access to the current site, grouped by Windows Messenger status. An example appears in Figure 18-1. Clicking any member name displays either:

- That member's personal site under SharePoint Portal Server or, if none exists,
- That member's User Information page from Windows SharePoint Services.

The Web Part also displays a Windows Messenger icon. Clicking this icon displays a menu with options to connect via Windows Messenger, or to perform the usual assortment of Outlook actions: schedule a meeting, add or edit phone numbers, send mail, and so forth.

When you configure a Members Web Part, you can specify these properties:

- **Number Of Items To Display** The maximum number of members the Web Part will display.
- **Toolbar Type** The type of toolbar to display: summary or none.

In addition, the usual options appear in the Appearance, Layout, and Advanced sections of the Web Part's property sheet.

Displaying Content with the Page Viewer Web Part

This Web Part adds an <iframe> element to a Web Part Page. The <iframe> (and therefore the Web Part Page) can then display any file, folder, or Web page the team member's computer can access. Figure 18-1 shows a Page Viewer Web Part displaying a Web page from MSN.

When you configure a Page Viewer Web Part, you can specify these properties:

- **Web Page, Folder, or File** Select one of these options depending on the type of content you want to display.
- **Link** Specify the URL or path of the content you want to display.

If you want the Page Viewer Web Part to display a folder or file, be sure to specify a path that's valid from each team member's computer. In most cases, this will be a UNC path such as \\<*server*>\<*sharename*>\myfolder\myfile.doc.

If you specify a path like c:\localfiles\myfile.doc, the Web Part will look for that folder and file on each team member's computer.

Adding Forms with the Simple Form Web Part

Chapter 12 briefly mentioned this Web Part in its discussion about Web Part connections. It provides a way of adding simple HTML forms to a Web Part Page, and then using any form fields you create as input to a connection to another Web Part.

> **For more information about using the Simple Form Web Part, refer to "Connecting Form, List, and Image Web Parts" in Chapter 12.**

Configuring the XML Web Part

This Web Part uses eXtensible Stylesheet Language Transformation (XSLT) to transform an XML file into HTML, and then adds the resulting HTML to the Web Part Page for display. To use this Web Part, you add it as usual to a Web Part Page, and then display its properties. Figure 18-2 shows how this looks in FrontPage 2003.

Figure 18-2. This is how FrontPage 2003 displays the property sheet for an XML Web Part

Here are the specific XML Web Part properties you can modify. All are optional.

- **XML Editor** Click this button to display a Text Entry dialog box. This dialog box has a large text box where you can type or paste the XML input to the transformation.
- **XML Link** Type the URL of the XML source that provides the input to the transformation. If you make an entry in this field, it overrides any XML you may have supplied after clicking the XML Editor button.
- **XSL Editor** Click this button to display another Text Entry dialog box, this time to enter the XSLT code that will transform the source XML file.
- **XSL Link** Type the URL of the XSLT code that will transform the source XML. Any entry in this field overrides any XSLT code you may have supplied after clicking the XSL Editor button.

If you don't specify any source XML, or if you specify an empty file, no error will occur except that the result will be empty. If you don't supply any XSLT code, no transformation will occur; the output will be identical to the input XML.

> **To appreciate what the XML Web Part can do, try using it to syndicate RSS content (blogs) into a Web Part Page. To get the necessary XSLT file, download the Syndication (RSS) tool from *http://www.asaris-matrix.com/sweber/playground/default.aspx?id=15*.**

Displaying HTML with the Content Editor Web Part

This Web Part displays static HTML obtained from a URL or entered by hand. Figure 18-3 shows an example in use.

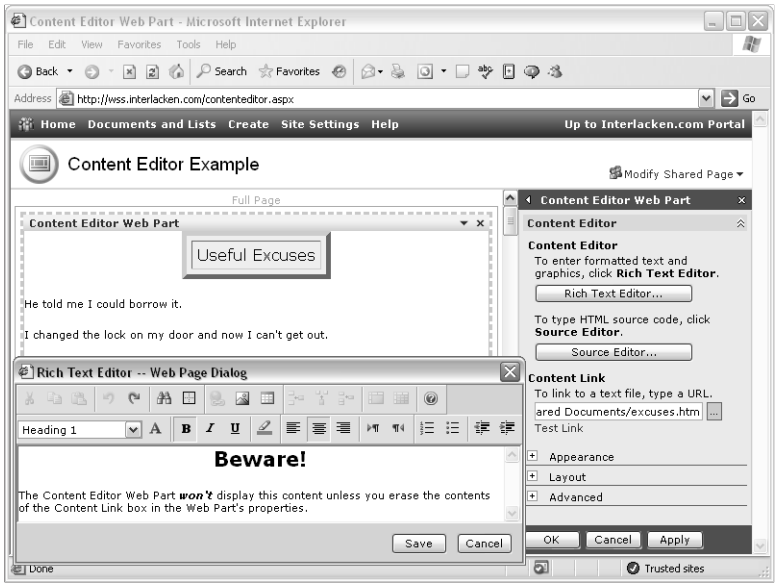

Figure 18-3. The Content Editor Web Part can display its own content or a remote URL, but it can edit only its own content.

In this figure, a Web designer:

1. Added the Content Editor Web Part to the page and configured its Content Location property to display an HTML file in a document library.
2. Browsed the page and verified that data from the Content Link location appears in the Web Part.
3. Clicked the Web Part's down arrow and chose Modify Shared Web Part.
4. Clicked the Rich Text Editor button.

If you try this yourself, you'll discover that:

- Neither the Rich Text Editor button nor its close cousin, the Source Editor button, edits the contents of the document at the Content Link location!

  Instead, these two buttons modify a block of HTML code that the Web Part stores as part of its own definition. Initially, this block of code is empty, and that's why the Rich Text Editor comes up empty.

- Clicking the Rich Text Editor's Save button *doesn't* update the document at the Content Link location, nor does it update the content editor Web Part's display.

  As long as the Content Link property contains data, the Web Part displays the data from that location, and ignores any data you may have entered through the Rich Text Editor, or by clicking the Source Editor button.

Got that? If not, here's another way of looking at the same information.

- The Content Editor Web Part displays data *either* from a content location URL *or* from its own Web Part definition.
- The Web Part displays data from its own Web Part definition *only* if the Content Location field is empty.
- The Rich Text Editor and Source Editor Buttons edit *only* the HTML stored in the Web Part definition.
- The Rich Text Editor and Source Editor Buttons *never* save data back to the document at the content location.
- The Rich Text Editor and Source Editor buttons edit the same HTML data: the data stored in the Web Part definition. You can click one, save, click the other, save, and continue in that mode as long as you like.

Adding ASP.NET Server Controls

This section describes four ASP.NET server controls that come with Windows SharePoint Services. These aren't Web Parts, so to use them you must have access to a page's HTML. For example, you must open the page in FrontPage, Visual Studio, or Notepad. Nevertheless, they're useful for adding certain common features to your pages.

Using the ViewSearchForm Control

This control displays the search form that appears on the home page of most SharePoint sites, provided that the site is using a SQL Server database with full-text searching in effect. If the database is WMSDE or if full-text searching is disabled, the control displays nothing. Figure 18-4 illustrates the control in its visible state.

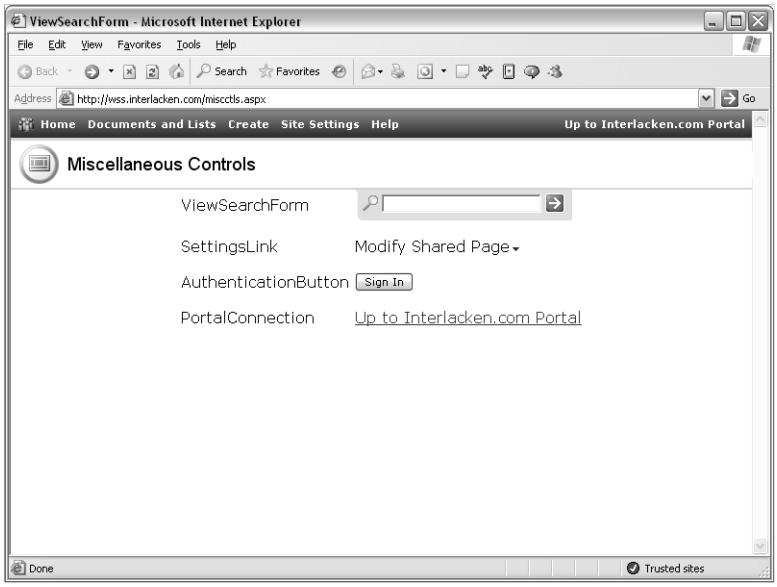

Figure 18-4. SharePoint ASP.NET server controls display all four elements in the right column of this Web Page. In practice, the SettingsLink control and the AuthenticationButton control would never both be visible.

Even if searching is available, the Title Bar Web Part doesn't display the search form. If you want your page to contain a search form, you have to add some HTML like the following to it:

```
<table cellspacing="0" cellpadding="0" border="0">
  <tr>
    <td></td>
    <SharePoint:ViewSearchForm
      id="SearchView"
      action="searchresults.aspx"
      go="Go"
      prompt="Search this site"
      runat="server" />
  </tr>
</table>
```

When the ViewSearchForm control produces output, it writes four <td> tags, the second of which is coded colspan="2". It doesn't write any <table> or <tr> tags. For this reason, it's usually best to enclose the control in your own <table> and <tr> tags, plus an empty pair of <td> and </td> tags. The empty <td></td> pair avoids the illegal HTML sequence <table><tr></tr></table> in case the View Search Form control produces no output.

Like any other ASP.NET control, ViewSearchForm requires a unique id= attribute and a runat="server" attribute. In addition, you'll probably want to code these attributes:

- **Action** The name of the page in the _layouts folder that will perform the search and display the results. Typically, you code this attribute action="searchresults.aspx".

  If you omit this attribute, the ViewSearchForm will connect to each List View Web Part on the same page, and tell those Web Parts to search their respective lists.

  If you omit the Action attribute and the page contains no List View Web Parts, submitting a search creates a JavaScript error.
- **Go** The tooltip text that will appear if the team member hovers the mouse over the search button. Typical coding would be go="Go" or go="Search".
- **Prompt** The tooltip text that will appear if the team member hovers the mouse over the search terms text box. For example, this might be prompt="Search this site".

The search form itself is designed to reside immediately below a Title Bar Web Part, at the right margin of the page. This placement will ensure that the background shape blends into the rest of the page layout.

Any page that uses the ViewSearchForm control must also:

- Contain the following tag somewhere before the <body> tag. This prevents errors that occur when loading the control on the server.

  ```
  <%@ Register Tagprefix="SharePoint"
  Namespace="Microsoft.SharePoint.WebControls" Assembly="Microsoft.SharePoint,
  Version=11.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c" %>
  ```

- Contain this tag in the <head> section, to avoid JavaScript errors that prevent the search from working.

  ```
  <SCRIPT language='javascript' src='/_layouts/1033/ows.js'></SCRIPT>
  ```

- Contain the following tag, also in the <head> section, so that the form appears with colors and fonts that match the rest of the page:

  ```
  <link rel="stylesheet" type="text/css" href="_layouts/1033/styles/ows.css">
  ```

Any Web Part Page that Windows SharePoint Services creates will already contain these tags. So will any page you create in FrontPage 2003 by displaying the Page Templates dialog box and choosing a template from the Web Part Pages tab.

Using the SettingsLink Control

This control displays the Modify Shared Page or Modify My Page link that team members click when they want to modify a Web Part Page through the SharePoint browser interface. It also displays the accompanying drop-down menu. Figure 18-4 illustrates this control.

> **Tip** A Title Bar Web Part displays a link identical to the one that a SettingsLink control displays. Therefore, if your page contains a Title Bar Web Part, you probably don't need a separate SettingsLink control.

A SettingsLink control displays nothing unless the current team member has sufficient rights to modify the page. If the site permits anonymous access, for example, the Modify Shared Page or Modify My Page link won't appear until the team member signs in. To permit easy sign-in, you should add an AuthenticationButton control to the page. The options on the drop-down menu, if it appears, will vary depending on the team member's permissions.

To display either a Modify Shared Page or Modify My Page link, as appropriate, simply add the following tag to your Web Part Page:

```
<WebPartPages:SettingsLink runat="server"/>
```

Any page that uses a SettingsLink control must also include the following tag preceding the <head> tag:

```
<%@ Register Tagprefix="WebPartPages"
Namespace="Microsoft.SharePoint.WebPartPages" Assembly="Microsoft.SharePoint,
Version=11.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c"%>
```

However, because Web Part Zones also require this tag, it's almost certain to be present in any Web Part Pages you encounter.

Using the AuthenticationButton Control

If your SharePoint site permits anonymous access, team members who need to use secured features need a way to sign in explicitly. The Title Bar Web Part displays a Sign In button for this purpose but if necessary, you can get the same button by adding the following tag to your page.

```
<WebPartPages:AuthenticationButton runat="server"/>
```

Figure 18-4 shows a sample of the resulting button. However, the button appears only if the current team member *isn't* signed in.

Any page that uses an AuthenticationButton control must provide the <%@ Register %> tag already described for the SettingsLink control.

Using the PortalConnection Control

Within a top-level Web site, this control displays a link to the portal site, if any, specified after clicking Configure Connection To Portal Site on the site's Top-Level Site Administration page. Within a subsite, it displays a link to the parent site.

> **For more information about configuring a connection to a portal site, refer to "Configuring a Connection to a Portal Site" in Chapter 17.**

Figure 18-4 shows two examples of this link. One is in the top navigation bar, and the other appears in the body of the page. The top navigation bar is the usual position. To display this link, add a tag like the following to your page.

```
<SharePoint:PortalConnection runat="server" />
```

As with the ViewSearchForm control, any page that uses the PortalConnection control must also contain the following tag somewhere before the <body> tag:

```
<%@ Register Tagprefix="SharePoint" Namespace="Microsoft.SharePoint.WebControls"
Assembly="Microsoft.SharePoint, Version=11.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c" %>
```

However, as before, most Web Part Pages already contain this tag.

Installing and Using Office 2003 Web Parts and Components

Office 2003 Web Parts and Components is a collection of Web Parts and Great Plains Web Part Pages that work closely with Microsoft Office 2003 and Windows SharePoint Services 2.0.

Installing the Microsoft Office 2003 Web Parts and Components

To use the Microsoft Office 2003 Web Parts and Components, you must first obtain and install them on your SharePoint server. To do this, proceed as follows:

1. Browse *http://www.microsoft.com/downloads.*
2. In the Search For A Download section:
   - Set Product/Technology to Office System.
   - Set Keywords to Web Part.

   Then click Go.

3. When, the Search Results page appears, click Office 2003 Add-in: Web Parts and Components.
4. When the Download Details page appears, click the Download button and save the ststpkpl.exe file in a location accessible to your SharePoint server.
5. Run ststpkpl.exe on the server running Windows SharePoint Services. (You must be a server administrator to run this step.)
6. Add the Web Parts to the Web Part gallery for your SharePoint site collection. To do this, you must be an administrator of the collection.

> **Fro instructions on adding a Web Part to a site collection's Web Part gallery, refer to "Managing the Web Part Gallery" in Chapter 17.**

Running setup for the Office 2003 Web Parts and Components affects all virtual servers on the same computer. It doesn't affect:

- Other servers in the same farm. To install Office 2003 Web Parts and Components on multiple servers, you must run setup on each server.
- Virtual servers you create in the future. To add the Office 2003 Web Parts and Components to a new virtual server, you must rerun the original setup program or run Reinstall or Repair from Control Panel, Add Or Remove Programs.

Installing the Office 2003 Web Components

The Microsoft Office 2003 Web Parts and Components are usable only on client computers that have Microsoft Office 2003 installed. The Web Components run within the browser window, and not on the server. You can install them as part of Office 2003, as a separate install from the Office 2003 CD, or from the Microsoft Office Online Web site. To install from the Web site, proceed as follows.

1. Browse *http://office.microsoft.com/*.
2. Click Downloads.
3. Search Downloads for, "Office 2003 Web Components."
4. Click Office 2003 Add-in: Office Web Components.
5. Download and run the owc11.exe file.

Using the PivotView, Spreadsheet, Web Capture, and Quick Quote Web Parts

The Office 2003 Web Parts and Components includes several Web Parts. For detailed information on each one, add it to a Web Part Page and then click Help on the Web Part's menu. The following is a summary description of each Web Part.

- **PivotView Web Part** This Web Part displays, analyzes, and reports data in any of three related views: PivotTable view, PivotChart view, or Datasheet view. This data comes from a source you specify in either of two file types:
  - An Office Database Connection (*.odc) file. This connects to SQL Server and other database systems using database drivers on the Web visitor's computer.
  - An Office Data Retrieval Service Connection (*.uxdc) file. This connects to SharePoint lists and other data sources accessible via the data retrieval services on a SharePoint server.

  Significantly, these files reside on each team member's computer. This means that the same Web page could display completely different data for each team member.

  PivotView Web Parts participate in many kinds of Web Part connections, including Chart Data From, Chart Data To, Filter Data With, Get Data From, Provide Data To, and Provide Row To.

  To save any results you obtain using this Web Part, display the Web Part toolbar and click the Export To Microsoft Office Excel button.
- **Spreadsheet Web Part** This Web Part adds the Office 2003 Spreadsheet Component to a Web Part Page. Using a classic, two-dimensional spreadsheet grid, you can then import data, enter data by hand, enter formulas, and so forth.

  Like the PivotView Web Part, this Web Part imports data based on an Office Database Connection or Office Data Retrieval Service Connection file. As to Web Part connections, it supports only Chart Data To.

  You can save any result you obtain using this Web Part by displaying its toolbar and then clicking the Export To Microsoft Office Excel button.
- **Web Capture Web Part** This Web Part retrieves an entire Web page, and then displays all or selected portions within the Web Part border. For example, you can choose to display only selected tables, images, or Web Parts.
- **Quick Quote Web Part** With this Web Part, you can enter the symbol of a stock, fund, or index and retrieve a brief summary quote for that symbol. Any such information will be at least 20 minutes old.

For more information about any of these Web Parts, add the Web Part to a page and then click the Help button on its toolbar.

Previewing Microsoft Office 2003 Integration for Great Plains

The Office 2003 Web Parts and Components include a number of Web Parts that provide read-only access to information in Microsoft Business Solutions Great Plains business systems. For example, you can analyze sales, inventory, and employee earnings data.

Installing Office 2003 Web Parts and Components also installs Great Plains Site sample data, and this is what the Web Parts initially display. To access data from a production Great Plains system, you must install the following software:

- Microsoft Office 2003 Integration for Great Plains. You can download this software from the Microsoft Business Solutions Web site.
- Microsoft SQL Server 2000 with Service Pack 3 (SP3) or later.

Customizing Site Definitions

Chapter 17 described how to save an existing SharePoint site as a template, and how to manage the site template gallery for a Web site collection.

A *site definition* is a second, more fundamental way of defining and creating SharePoint sites. If fact, when you first install Windows SharePoint Services, no site templates are available. Instead, a site definition describes each type of Web site a default installation can create: a Team Site, a blank site, a Document Workspace site, and five kinds of Meeting Workspace sites. That's eight site definitions in all.

Ghosting Site Template Files

When you create a site based on a site definition, Windows SharePoint Services doesn't copy a complete set of Web pages into the SharePoint content database. Instead, it *ghosts* the pages in the site definition. This means that if, for example, you create a hundred Team Sites, those sites would behave as if the content database contains a hundred copies of the Team Site pages. But in fact, all hundred sites would share one set of pages, and that set of pages would reside in the Web server's file system. This avoids the overhead of storing, retrieving, and caching duplicated pages in the content databases.

Of course, when someone opens an individual site and uses a program like FrontPage to customize one of its pages, the SharePoint server can no longer ghost that file for that site. Instead, it saves the new file version in the content database, and that version overrides the ghosted file from disk. (This is called *unghosting* the file).

Taken as a unit, the files that make up a site definition describe every detail necessary to create a new site. A site template, however, contains only the differences between a site definition and the state of the site at some later time (with or without list and library content). When you create a site from a site template, Windows SharePoint Services:

1. Creates a site based on the site definition that created the original site.
2. Applies any changes that the site template specifies.

Working with Site Definition Files

When you first install Windows SharePoint Services, a file named WEBTEMP.XML identifies each site definition. (TEMP in this context means *template,* not *temporary*.) You can find this file at:

C:\Program Files\Common Files\Microsoft Shared\web server extensions\60\TEMPLATE\1033\XML

where 1033 is the current locale ID. A version of this file (with greatly shortened Description values) appears below:

```
<?XML version="1.0" encoding="utf-8" ?>
<!-- _lcid="1033" _version="11.0.5510" _dal="1" -->
<!-- _LocalBinding -->
<Templates xmlns:ows="Microsoft SharePoint">
  <Template Name="STS" ID="1">
    <Configuration ID="0" Title="Team Site"
      Hidden="FALSE" ImageUrl="/_layouts/images/stsprev.png"
      Description="description">
   </Configuration>
    <Configuration ID="1" Title="Blank Site"
      Hidden="FALSE" ImageUrl="/_layouts/images/stsprev.png"
      Description="description">
    </Configuration>
    <Configuration ID="2" Title="Document Workspace"
      Hidden="FALSE" ImageUrl="/_layouts/images/dwsprev.png"
      Description="description">
    </Configuration>
 </Template>
 <Template Name="MPS" ID="2" >
    <Configuration ID="0" Title="Basic Meeting Workspace"
      Hidden="FALSE" ImageUrl="/_layouts/images/mwsprev.png"
      Description="description">
    </Configuration>
    <Configuration ID="1" Title="Blank Meeting Workspace"
      Hidden="FALSE" ImageUrl="/_layouts/images/mwsprev.png"
      Description="description">
    </Configuration>
    <Configuration ID="2" Title="Decision Meeting Workspace"
      Hidden="FALSE" ImageUrl="/_layouts/images/mwsprev.png"
      Description="description">
    </Configuration>
    <Configuration ID="3" Title="Social Meeting Workspace"
```

```
      Hidden="FALSE" ImageUrl="/_layouts/images/mwsprev.png"
      Description="description">
    </Configuration>
    <Configuration ID="4" Title="Multipage Meeting Workspace"
      Hidden="FALSE" ImageUrl="/_layouts/images/mwsprev.png"
      Description="description">
   </Configuration>
  </Template>
</Templates>
```

Each <Template></Template> block identifies a set of similar Web site definitions. Within each <Template> tag:

- The Name attribute supplies a mnemonic name.
- The ID attribute supplies a unique identifier.

Within each <Template></Template> block there's one <configuration> tag for each site definition. Within this tag:

- The ID attributes provide a unique identity within each <Template> node.
- The Title, ImageUrl, and Descriptions fields specify values that the Template Selection page will display when a team member creates a site.

Working with ONET.XML Files

Further information about each site definition appears in a file named ONET.XML. Because the WEBTEMP.XML file specified two template names (STS and MPS), there are two ONET.XML files. To find these files, first navigate to C:\Program Files\Common Files\Microsoft Shared\Web server extensions\60\TEMPLATE\1033\ and then to these locations:

STS\XML\ONET.XML
MPS\XML\ONET.XML

Each ONET.XML file defines a series of common elements for individual sites to use. This includes navigation bars such as the top navigation bar and the quick launch bar, list templates, document templates, base types, configurations, and modules.

A *configuration* is the structure that describes an individual site definition.

- The STS\XML\ONET.XML file contains one configuration structure for each ID value in the <Template Name="STS"> node of the WEBTEMP.XML file.
- The MPS\XML\ONET.XML file contains one configuration structure for each ID value in the <Template Name="MPS"> node of the WEBTEMP.XML file.

A *module* is a block of XML code that specifies all the files associated with a definition and the location of those files in the new site. In the case of a Web Part Page, for example, the module definition specifies not only the Web Part Page file, but also the specific Web Parts, List Views, and files to include on the page.

Working with SCHEMA.XML Files

Another XML file named SCHEMA.XML defines the views, forms, and fields for each type of list. The file for any of the standard list types resides at a location such as

C:\Program Files\Common Files\Microsoft Shared\Web server extensions\60\TEMPLATE\1033\STS\LISTS\ANNOUNCE\SCHEMA.XML

where 1033 is the culture ID, STS is the template name from the WEBTEMP.XML file, and ANNOUNCE is a mnemonic name identifying the type of list (Announcements, in this case).

Customizing Site Definitions

Making changes to the WEBTEMP.XML, ONET.XML, and SCHEMA.XML files that come with Windows SharePoint Services is generally a bad idea. Here are the two main reasons:

- Microsoft considers these to be system files. As a result, any reinstallation, repair, or upgrade to Windows SharePoint Services might replace them, overlaying any changes you've made.
- Any site created from a site definition continues to use the WEBTEMP.XML, ONET.XML, and SCHEMA.XML files indefinitely. Therefore, an incorrect change, even if it works for new sites, could break hundreds or thousands of existing sites.

The correct approach, therefore, is to create new site definitions rather than change existing ones. To do this, you create a new site definition file in the same folder as WEBTEMP.XML, and append a suffix to the filename base. Here, for example, are some possible site definition files names:

WEBTEMPIDAHO.XML
WEBTEMPLOGBOOKS.XML
WEBTEMPSALES.XML

When Windows SharePoint Services starts up, it scans the C:\Program Files\Common Files\Microsoft Shared\web server extensions\60\TEMPLATE\1033\XML folder for filenames that begin with WEBTEMP and end in .XML (both all caps) and consecutively loads them into memory. This provides a way of adding new site definitions without modifying the Microsoft-supplied WEBTEMP.XML file. The content below would be typical for a WEBTEMPSALES.XML file.

```
<?XML version="1.0" encoding="utf-8" ?>
<Templates xmlns:ows="Microsoft SharePoint">
   <Template Name="SALES" ID="10011">
      <Configuration ID="0" Title="Sales Tracking Site" Type="0"
         Hidden="FALSE"
         ImageUrl="images/salestrack.jpg"
         Description="A site for tracking new and potential
         customers, for recording customer visits and
         conversations, and for reporting orders.">
      </Configuration>
   </Template>
</Templates>
```

Note that the template name SALES on line 3 matches the characters you appended to the WEBTEMP filename base. This is a good practice but not a requirement. Note as well that the template ID is greater than 10,000. Microsoft numbers its templates starting at zero and, to avoid conflicts, suggests that customers number their templates starting at 10,001.

Next, you would create the C:\Program Files\Common Files\Microsoft Shared\Web server extensions\60\TEMPLATE\1033\SALES\XML\ path and provide an ONET.XML file. Because the ONET.XML file is long and complex, you'll probably want to copy and modify an existing file rather than type one from scratch.

Finally, for each list, you'll need to create the C:\Program Files\Common Files\Microsoft Shared\Web server extensions\60\TEMPLATE\1033\SALES\LISTS\<*listname*>\ path and add a SCHEMA.XML file that defines that list.

Although the WEBTEMP.XML, ONET.XML, and SCHEMA.XML files provide the essence of a site definition, you'll need to supply many other files as well. For example, you'll need to provide copies of all the Web pages and associated files the site will use. To observe the type and location of these files, inspect the folder tree for any of the Microsoft-supplied site definitions.

Choosing between Site Templates and Site Definitions

With two methods of designing sites that team members can create, you need some guidelines for deciding which method is best in a particular situation. Here are some factors to consider.

- **Ease of Use** Site templates are easy to create and deploy. You can find or create a site with the features you want, save it as a template, deploy the template, and start creating new sites, all without leaving the browser interface or FrontPage 2003.

  Creating a site definition requires working with XML code in a text editor. Then, when you deploy it, you must do so on each front-end Web server in the same farm. Without a doubt, site definitions are harder to create and deploy.

- **Who Performs the Work?** Any SharePoint Web designer can create a sample site and save it as a template. Then, any site collection administrator can deploy it.

  Because site definitions require access to the Web server's file system, development requires administrative access to test Web servers, and deployment requires the cooperation of production server administrators.

- **Resource Dependence** Site templates don't include all the information required to create a site. Instead, they specify a site definition to use as a starting point, and then a list of changes to apply. If that site definition isn't present, or if it's the wrong version, the site template will fail.

  Site definitions, by contrast, completely describe the sites they create, without dependence on any other site definition. This makes site definitions more attractive to software developers, server administrators, and third-party suppliers.

- **Extent of Modification** A site template can override many aspects of its underlying site definition, but not all. A site template can, for example, modify logos, menus, means of navigation, and default lists and libraries, but it can't incorporate new data types, new file types, new view styles, or new drop-down edit menus.

  The more your site needs to differ from any of the default site definitions, the more likely your approach should be a new site definition.

- **Maintainability** Once you create a site from a template, modifying the template has no effect on that site. This gives you the freedom to change a template, and thereby add, remove, or modify features in future sites. Because templates reside in the configuration database or in a site template gallery, a single command deploys them for an entire server farm.

  Once you've deployed a site definition, there's no safe way to delete or change its features. This is because sites created from the site definition refer to it on an ongoing and active basis, and depend on it remaining constant. If a site definition is no longer adequate, you can only add features or create a new site definition.

- **Performance** Web pages "ghosted" from a site definition run faster than pages that reside in a content database. This is because:

  - The Web server's file system is faster than a WMSDE or SQL Server database.
  - Caching files from the Web server's file system is more efficient than caching them from a content database.
  - The fewer cache hits, the more often ASP.NET must compile any .aspx pages. Each compilation incurs a performance penalty.

  When you create a site solely from a site definition, none of its pages physically resides in the content database, and performance is at peak. Thereafter, the new version of each page you customize (using, for example, a program like FrontPage) resides in the content database and incurs a performance penalty.

  When you create a site from a site template, each page that differs from the underlying site definition resides in the content database, and therefore incurs the performance penalty.

  Initially, a site created from a template will have one or more pages in the content database, and a site created from a site definition will have none. This often results in the templated sites running somewhat slower.

  Note, however, that if you create a site from a site definition and then modify some of its pages, the modified pages then reside in the content database. Depending on the number of unghosted pages, a site created from a site definition could therefore run slower than, at the same speed as, or faster than a site created from a template.

  In practice, the performance difference between sites created from a template and sites created from a site definition usually isn't noticeable. Nevertheless, these factors may be worth considering in a large-scale environment.

For more information about creating and using site definitions, refer to these resources.

- Customizing SharePoint Sites and Portals: Using Templates and Site Definitions by Dino Dato-on and Jinger Zhao (*http://msdn.microsoft.com/library/en-us/odc_SP2003_ta/html/ODC_SPSCustomizingSharePointSites2.asp)*
- Customizing Templates for Microsoft Windows SharePoint Services (*http://msdn.microsoft.com/library/en-us/spptsdk/html/SPPTWSSTemplates.asp*)

In Summary…

This chapter explained four advanced techniques for creating Web Part Pages and sites: built-in SharePoint Web Parts, SharePoint ASP.NET server controls, the Office 2003 Web Parts and Components, and site definitions.

The next chapter begins Part VII, which explains how to write your own Web Parts using Visual Studio .NET.