**ملحق 4**

**The Structure of HTML Documents**

**(أعتذر عن عدم توفّر الوقت لديّ لترجمة هذا الملحق)**

* [**HTML Documents**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\HTML\1-%20HTML%20tag.htm)
* [**Attributes:**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\HTML\2-%20Attributes.htm)
* [**URLs and Hyperlinks**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\HTML\3-%20URLs%20and%20Hyperlinks.htm)
* [**The Basic HTML Tags**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\HTML\4-%20The%20Basic%20HTML%20Tags.htm)
* [**Inserting Graphics**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\HTML\5-%20Inserting%20Graphics.htm)
* [**Tables**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\HTML\6-%20Tables.htm)
* [**Forms and Controls**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\HTML\7-%20Forms%20and%20Controls.htm)
* [**Processing Requests on the Server**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\HTML\8-%20Processing%20Requests%20on%20the%20Server.htm)

**<HTML> tag**

To create the most fundamental HTML document, you must start with the <HTML> tag and end with the </HTML> tag. Within these tags should be a HEAD section and a BODY section.

The BODY of the document is the portion that is presented within the browser window. The document’s HEAD, marked with the <HEAD> and </HEAD> tags, is where you normally place the following elements:

\_ The document’s title

\_ Information about the document, such as the META and BASE tags

\_ Scripts The title is the text that appears in the title bar of the browser’s window and is specified with the <TITLE> and </TITLE> tags. META tags don’t display anywhere on the screen but contain useful information regarding the content of the document, such as a description and keywords used by search engines. For example:

<HTML>

<HEAD>

<TITLE>Your Title Goes Here</TITLE>

<META NAME="Keywords"

CONTENT="health, nutrition, weight control, chronic illness">

</HEAD>

<BODY>

Hello, World!

</BODY>

</HTML>

**Attributes:**

Many HTML tags understand special keywords, which are called attributes. The <BODY> tag, which marks the beginning of the document’s body, for instance, recognizes the BACKGROUND attribute, which lets you specify an image to appear in the document’s background. You can also specify the document’s text color and its background color (if there’s no background image) with the TEXT and BGCOLOR attributes, respectively:

<HTML>

<HEAD>

<TITLE>Your Title Goes Here</TITLE>

</HEAD>

<BODY BACKGROUND="paper.jpg" BGCOLOR="yellow" TEXT="black">

<H1>Tiled Background</H1>

<P>The background of this page was created with a small image, which is tiled vertically and horizontally by the browser. If the image can’t display, the page will have a solid yellow background. Either way, the text will be black.</P>

</BODY>

</HTML>

Background images start tiling at the top-left corner and work their way across and then down the screen. Many HTML tags accept attribute parameters that position them precisely on the page.

Unfortunately, not all browsers understand these elements, so the same page may look perfect in Internet Explorer or Netscape and totally misaligned in another browser. The good news is that you don’t have to learn all these attributes; if you’re working with the Visual Studio IDE, the designer will insert them for you.

**URLs and Hyperlinks:**

The key element in a Web page is the hyperlink, a special instruction embedded in the text that causes the browser to load another page. A hyperlink is a string that appears in different formatting from the rest of the text (usually in a different color and underlined); when the mouse pointer is over a hyperlink, the cursor changes, typically into a finger. When you click the mouse button over a hyperlink, the browser requests and displays another document, which could be on the same or another server.

To connect to another computer and request a document, the hyperlink must contain the name of the computer that hosts the document and the name of the document. Just as each computer on the Internet has a unique name, each document on a computer also has a unique name. Thus, each document on the World Wide Web has a unique address, which is called a Uniform Resource Locator (URL). The URL for a document is something like the following:

http://www.someserver.com/docName.htm

Note You will notice that some HTML URLs end in htm and some end in html. They are identical; some older operating systems don’t support long extensions, that’s all.

Every piece of information on the World Wide Web has a unique address and can be accessed via its URL. What the browser does depends on the nature of the item. If it’s a Web page (file extension .html or .htm) or an image (such as a .gif file), the browser displays it. If it’s a sound file (such as a .wav file), the browser plays it back. Today’s browsers can process many types of documents; older versions can’t. When a browser runs into a document it can’t handle, it asks whether the user wants to download and save the file on disk or open it with an application that the user specifies.

The tag that makes HTML documents come alive is the <A> tag, or anchor tag, which inserts hyperlinks in a document. The <A> and </A> tags enclose one or more words that will be highlighted as hyperlinks. In addition, you must specify the URL of the hyperlink’s destination. For example, the URL of the Sybex home page is:

http://www.sybex.com

The URL to jump to is indicated with the HREF attribute of the <A> tag. To display the string "Visit the SYBEX home page" and to use the word SYBEX as the hyperlink, you enter the following in your document:

Visit the <A HREF="http://www.sybex.com">SYBEX</A> home page

This inserts a hyperlink in the document, and each time the user clicks the SYBEX hyperlink, the browser displays the main page at the specified URL.

Note You often need not specify a document name in the hyperlink. Servers are commonly configured to supply the default page, which is known as the home page. The home page is usually the entry to a specific site and contains hyperlinks to other pages making up the site.

To jump directly to a specific page on a Web site, use a hyperlink such as the following:

View a document on <A HREF="http://www.example.com/HTMLTutorial.htm">HTML

programming</A> on this site.

Most hyperlinks on a typical page jump to other documents that reside on the same server. These hyperlinks usually contain a relative reference to another document on that server. For example, to specify a hyperlink to the document Images.htm that resides in the same folder as the current page, use the following tag:

Click <A HREF=".\Images.htm">here</A> to view the images.

**The Basic HTML Tags**

HTML is certainly easy for a Visual Basic programmer to learn and use. The small part of HTML presented here is all you need to build functional Web pages.

Although you can use the visual tools of the IDE, many times it’s actually simpler to open the HTML file and edit it. The following are the really necessary tags for creating no-frills HTML documents, grouped by category.

* [**Headers**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\4-%20The%20Basic%20HTML%20Tags\1-%20Headers.htm)
* [**Paragraph Formatting**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\4-%20The%20Basic%20HTML%20Tags\2-%20Paragraph%20Formatting.htm)
* [**Character Formatting**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\4-%20The%20Basic%20HTML%20Tags\3-%20Character%20Formatting.htm)

**Headers:**

Headers separate sections of a document. Like documents prepared with a word processor, HTML documents can have headers, which are inserted with the <Hn> tag. There are six levels of headers, starting with <H1> (the largest) and ending with <H6> (the smallest). To place a level 1 header in the document, use the tag <H1>:

<H1>Welcome to Our Fabulous Site</H1>

A related tag is the <HR> tag, which displays a horizontal rule and is frequently used to separate sections of a document. The document in Figure 23.1, which demonstrates the HTML tags discussed so far, was produced with the following HTML file:

<HTML>

<HEAD>

<TITLE>

Document title

</TITLE>

</HEAD>

<BODY>

<H1>Sample HTML Document</H1>

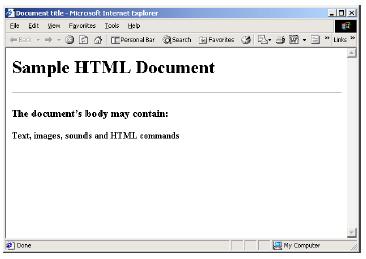
<HR>

<H3>The document’s body may contain:</H3>

<H4>Text, images, sounds and HTML commands</H4>

</BODY>

</HTML>



**Paragraph Formatting**

HTML won’t break lines into paragraphs whenever you insert a carriage return in the text file. The formatting of the paragraphs is determined by the font(s) used in the document and the size of the browser’s window. To force a new paragraph, you must explicitly tell the browser to insert a carriage return with the <P> tag. The <P> tag also causes the browser to insert additional vertical space. To insert a line break without the additional vertical space, use the <BR> tag.

**Character Formatting**

HTML provides tags for formatting words and characters. Table 23.1 shows the basic characterformatting tags. The tags listed in pairs can be used alternately for the same effect; <B>…</B>

produces the same look as <STRONG>…</STRONG> in most browsers.

Table 23.1: The Basic HTML Character-Formatting Tags

TagWha t It Does

<B> or <STRONG> Specifies bold text

<FONT> Specifies text characteristics such as typeface, size, and color

<I> or <EM> Specifies italic text (for emphasis)

<TT> or <CODE> Specifies the "typewriter" attribute, so text is displayed in a monospaced font; used frequently to display computer listings

The <FONT> tag specifies the name, size, and color of the font to be used. The <FONT> tag takes one or more of the following attributes:

SIZE Specifies the size of the text in a relative manner. The value of the SIZE argument is not expressed in points, pixels, or any other absolute unit. Instead, it’s a number in the range 1 (the smallest) through 7 (the largest). The following tag displays the text in the smallest possible size:

<FONT SIZE="1">tiny type</FONT>

The following tag displays text in the largest possible size:

<FONT SIZE="7">HUGE TYPE</FONT>

FACE Specifies the font family. If the specified font does not exist on the client computer, the browser substitutes a similar font. The following tag displays the text between FONT and its matching tag in the Comic Sans MS typeface:

<FONT FACE="Comic Sans MS">Some text</FONT>

COLOR Specifies the color of the text.

Tip The XHTML specification recommends, but doesn’t yet require, using external cascading style sheets instead of most formatting tags. (In technical language, tags such as <FONT> are deprecated.) At some point, browsers might not recognize formatting within your HTML document, but you can keep using it during this transition period.

**Inserting Graphics**

Graphics play an important role in Web page design. Almost every page on the World Wide Web uses graphics, and some pages contain hardly any text. Graphics are not inserted in the HTML document directly. The document itself contains special tags that reference the image to be inserted by the browser when the page is opened. Because of this, graphics files are downloaded separately and placed on the page by the browser.

On the Web, where every byte counts and downloads must be fast, images must contain as much information in as few bytes as possible. Despite the large number of graphics formats available today, two formats have dominated the Web:

\_ JPEG (Joint Photographic Experts Group)

\_ GIF (Graphics Interchange Format)

These formats are used because they compress graphics files to a manageable size. JPEG files can be compressed a good deal (albeit with some loss of detail), but they maintain a good image quality overall. The problems become evident when the compressed image is enlarged, but the graphics on Web pages are meant to be viewed in the context of the Web page to which they belong. The GIF file format is an old one, and it supports only 256-color images, but it has a few really handy features. It’s the only format that supports transparency, and its compression ratio is even better than JPEG without losing detail.

To insert an image at the current location in the document, use the <IMG> tag with the SRC attribute, which specifies the image to be displayed. Figure 23.2 shows a page with a simple graphic, centered across the page.



Here is the HTML source code that produced the page in Figure:

<HTML>

<HEAD>

<TITLE>Graphics on Web pages</TITLE>

</HEAD>

<BODY>

<CENTER>

<H1>Placing an Image on a Web page</H1>

<IMG SRC="earth1.jpg" ALT="Earth photo"><BR>

Our small planet, centered on the page

</CENTER>

</BODY>

</HTML>

The <IMG> tag has the following syntax:

<IMG SRC="picture.jpg" ALT="alt text">

The <IMG> tag recognizes additional attributes, but you must include the SRC attribute, which is the location of an image file on the server or any URL on the Web. When you use the following attributes with the <IMG> tag, the browser can manipulate the image in several ways:

ALIGN aligns the image to the left, right, center, top, bottom, or middle of the screen.

WIDTH and HEIGHT specify the width and height of the image.

BORDER adds a border to the image, which is visible only if the image is a hyperlink.

VSPACE and HSPACE clear space around the image vertically or horizontally. The empty space is specified in pixels.

ALT includes a text message to be displayed if the user has turned off graphics.

If you want to change the size of an image, you can specify the size with the WIDTH and HEIGHT attributes, and the browser will size the existing file to the new values. For instance, to create a straight vertical line two pixels wide, simply use a square image two pixels on each side, and set the <IMG> tag’s WIDTH and HEIGHT properties:

<IMG WIDTH="2" HEIGHT="200" SRC="picture.jpg">

Your image will stretch 200 pixels high. You can also distort bitmaps with the WIDTH and HEIGHT attributes.

The BORDER attribute specifies the width of the border to appear around an image. Borders two pixels wide automatically surround any image used as a hyperlink. You may want to eliminate this automatic border with the BORDER="none" attribute.

One aspect affecting the appearance of images, especially when they are surrounded by text, is the amount of space between the image and surrounding text. Space can be cleared horizontally and vertically with the HSPACE and VSPACE attributes. Simply specify the amount of space in pixels, for example, HSPACE="10" or VSPACE="20".

The ALT attribute displays alternative text for users whose browsers don’t display images (to speed up the loading of the page or perhaps to use special software to accommodate a disability).

The attribute ALT="Company Logo" tells the user that the image is not displayed in the browser. In addition, if the image takes a long time to download, the message "Company Logo" is displayed in the image’s space on the page. If for some reason your images are not transmitted or don’t show up, the user can still navigate your Web site and get the picture, so to speak.

**Tables**

Tables are invaluable tools for organizing and presenting data in a grid or matrix. Tables are used in an HTML document for the same reasons they are used in any other document. There is, however, one more reason for using tables with an HTML document: to align the elements on a page. A table’s cell may contain text, hyperlinks, or images, and you can use the cell to align these elements on the page in ways that are simply impossible with straight HTML or even other tables. You can ven use tables without borders, so your audience doesn’t see how you accomplished your amazing (for HTML) feats of graphic design.

**The Basic Table Tags**

* [**TABLE tag**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\6-%20Tables\1-%20TABLE%20tag.htm)
* [**Aligning Cell Contents**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\6-%20Tables\2-%20Aligning%20Cell%20Contents.htm)
* [**Table Width and Alignment**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\6-%20Tables\3-%20Table%20Width%20and%20Alignment.htm)
* [**Multiple Row and Multiple Column Cells**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\6-%20Tables\4-%20Multiple%20Row%20and%20Multiple%20Column%20Cells.htm)

**TABLE tag**

Every table begins with the <TABLE> tag and ends with the </TABLE> tag. The attributes of the <TABLE> tag allow you to specify whether the table has borders, the width of borders, the distance between cells, and the proximity of cell contents to the edge of the cell. You can specify the width and height of the table either in pixels or as a percentage of total screen size.

Within the <TABLE> tags, each table row is marked by the <TR> tag. Each row’s cells are marked by the <TD> tag. Here’s the structure of a simple table. If you create an HTML file with the following lines and open it with your browser, you will see the items arranged as a table without any lines around them.

<HTML>

<TABLE>

<TR>

<TD> Row 1, Column 1 </TD>

<TD> Row 1, Column 2 </TD>

<TD> Row 1, Column 3 </TD>

</TR>

<TR>

<TD> Row 2, Column 1 </TD>

<TD> Row 2, Column 2 </TD>

<TD> Row 2, Column 3 </TD>

</TR>

<TR>

<TD> Row 3, Column 1 </TD>

<TD> Row 3, Column 2 </TD>

<TD> Row 3, Column 3 </TD>

</TR>

</TABLE>

</HTML>

**Aligning Cell Contents**

The ALIGN and VALIGN attributes specify the alignment of the cell’s contents. The ALIGN attribute is used for horizontal alignment and can have the value LEFT, CENTER, or RIGHT.

The VALIGN attribute specifies the vertical alignment of the text, and it can have the value TOP, MIDDLE, or BOTTOM. The default alignment is LEFT (horizontal alignment) and MIDDLE (vertical alignment).

A great deal of control over the alignment, spacing, and placement of cell contents within tables translates directly into excellent formatting capability for documents that would not ordinarily be built as tables. In fact, in HTML there are some effects you just can’t get (in a practical way) without the effective use of tables.

**Table Width and Alignment**

All the examples we have looked at so far use the default table width, which is determined by the entries of the individual cells. If a column contains a very long entry, the browser will wrap its contents to make sure that all columns are visible. However, it is possible to specify the width of the entire table with the WIDTH attribute of the <TABLE> tag.

The WIDTH attribute can be a value that specifies the table’s width in pixels or as a percentage of the window’s width. The table defined as <TABLE WIDTH="50%"> occupies one-half of the window’s width.

The table defined as <TABLE WIDTH="200"> will be 200 pixels wide, regardless of its contents and/or the window’s size. If the window is less than 200 pixels wide, part of the table will be invisible.

To display the part of the table that’s outside the window, you’ll have to use the horizontal scroll bar.

The <TABLE> tag can also take the ALIGN attribute, but instead of aligning the table contents, this aligns the table itself. If you don’t include an ALIGN attribute, the table will be leftaligned in the browser’s window.

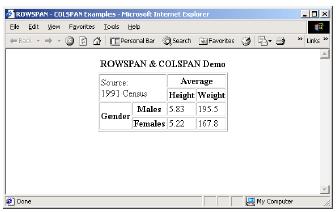
Tip This is true of most elements; paragraphs (<P>) and images (<IMG>) can also use ALIGN, and if it’s not present they default to aligning left.

**Multiple Row and Multiple Column Cells**

Quite often, tables don’t contain identically sized rows and columns. Some rows may contain fewer and wider cells than the others, and some columns may span multiple rows. The figures in this section contain tables with peculiar formatting.

Figure 23.3 shows a table with cells that span multiple columns and rows. These cells use the ROWSPAN and COLSPAN attributes, which let you create really elaborate tables. Either or both can appear in a <TD> tag, and they merge the current cell with one or more of its adjacent cells on the same row (in the case of the COLSPAN attribute) or column (in the case of the ROWSPAN attribute). The number of adjacent cells to be merged is the value of the COLSPAN and ROWSPAN attributes; COLSPAN="2" means that the current cell covers two columns.

The table in Figure 23.3 was created with the HTML lines shown in Listing 23.1. The only thing I’ve done differently here is add the COLSPAN attribute in the appropriate <TD> tags to force some cells of the first row to span two columns, and I’ve added the ROWSPAN attribute to force some cells in the first column to span multiple rows. (The <TH> tag is simply a special table cell that indicates a table heading.) Other than that, the new table is as simple as those in the previous examples.



Listing 23.1: The ROWSPAN and COLSPAN Attributes

<HTML>

<HEAD>

<TITLE>ROWSPAN - COLSPAN Examples</TITLE>

</HEAD>

<BODY>

<TABLE BORDER="1" ALIGN="CENTER">

<CAPTION><B>ROWSPAN & COLSPAN Demo</B></CAPTION>

<TR>

<TD COLSPAN="2" ROWSPAN="2">Source:<BR>1991 Census</TD>

<TH COLSPAN="2">Average</TH>

</TR>

<TR>

<TH>Height</TH>

<TH>Weight</TH>

</TR>

<TR>

<TH ROWSPAN="2">Gender</TH>

<TH>Males</TH>

<TD>5.83</TD>

<TD>195.5</TD>

</TR>

<TR>

<TH>Females</TH>

<TD>5.22</TD>

<TD>167.8</TD>

</TR>

</TABLE>

</BODY>

</HTML>

**Forms and Controls**

As you know already, HTML pages contain controls that let the user enter information, similar to the usual Windows controls: text boxes, option buttons, and so on. The areas on the HTML page where these controls appear are called forms (or Web forms), and the controls themselves are called intrinsic controls. HTML provides special tags for placing intrinsic controls on a form.

* [**FORM tag**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\7-%20Forms%20and%20Controls\1-%20FORM%20tag.htm)
* [**The Text Control**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\7-%20Forms%20and%20Controls\2-%20The%20Text%20Control.htm)
* [**The TextArea Control**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\7-%20Forms%20and%20Controls\3-%20The%20TextArea%20Control.htm)
* [**The CheckBox Control**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\7-%20Forms%20and%20Controls\4-%20The%20CheckBox%20Control.htm)
* [**The RadioButton Control**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\7-%20Forms%20and%20Controls\5-%20The%20RadioButton%20Control.htm)
* [**The Multiple Selection Control**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\7-%20Forms%20and%20Controls\6-%20The%20Multiple%20Selection%20Control.htm)
* [**The Command Button Control**](file:///C:\Users\Administrator\Desktop\vb.net\g-%20%20الملاحق\4-%20HTML\7-%20Forms%20and%20Controls\7-%20The%20Command%20Button%20Control.htm)

**FORM tag**

Before placing a control on the page, you must create a form with the <FORM> tag. Its syntax is:

<FORM NAME="name" ACTION="action" METHOD="method">

</FORM>

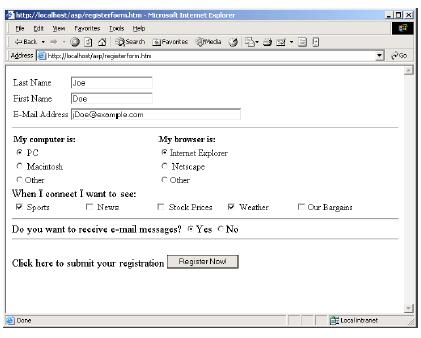
All the controls must appear between these two tags. The NAME attribute is the name of the form and is used when a page contains multiple forms. The ACTION attribute is the name of an application on the server that will be called to process the information. The METHOD attribute specifies how the controls’ values will be transmitted to the server. All the information needed by the browser to contact an application in the server is contained in the <FORM> tag. But more on this later, in the section "Passing Parameters to the Server."

HTML provides support for the following intrinsic controls. Figure 23.4 shows a Web page with a form that contains most of HTML’s intrinsic controls. You are going to see the HTML code that produced the page seen in the figure in the section "Processing Requests on the Server," later in this chapter.

**The Text Control**

The Text control is a box in which visitors can enter a single line of text (such as name, address, and so on). To insert a Text control on a form, use the following tag:

<INPUT TYPE="text" NAME="Publisher" VALUE="Sybex">



The VALUE attribute specifies the initial value. After the visitor changes this entry, VALUE holds the new string. To edit the contents of a Text control, the visitor can use the common editing keys (Home, Del, Insert, and so on), but the text can’t be formatted.

To control the size and contents of the control, use the SIZE and MAXLENGTH attributes.

The SIZE attribute specifies the size of the control on the form, in number of characters, and the MAXLENGTH attribute specifies the maximum number of characters the user can type in the control.

A variation of the Text control is the Password control, which looks identical but doesn’t display the characters as they are typed. Instead, it displays asterisks, and it is used to enter passwords.

**The TextArea Control**

The TextArea control is similar to the Text control, but it allows the entry of multiple lines of text.

All the usual navigation and editing keys work with the TextArea control. To place a TextArea control on a form, use the <TEXTAREA> tag:

<TEXTAREA NAME="Comments" ROWS="10" COLS="30">The best editor I’ve ever used!</TEXTAREA>

Because the TextArea control allows you to specify multiple lines of initial text, it’s not inserted with the usual <INPUT> tag, but with a pair of <TEXTAREA> tags. The ROWS and COLS attributes specify the dimensions of the control on the page, in number of characters. Unlike the rest of an HTML document, white space in the content between the two <TEXTAREA> tags is preserved when the text is displayed on the control; line breaks you insert, for instance, will appear in the browser. Even if you include HTML tags in the initial text, they will appear as text on the control.

**The CheckBox Control**

The CheckBox control is a little square with an optional checkmark, which acts as a toggle. Every time the visitor clicks it, it changes state. It is used to present a list of options, from which the user can select one or more. To insert a CheckBox control on a form, use the <INPUT> tag:

<INPUT TYPE="checkbox" NAME="Check1">

To initially check a CheckBox control, specify the CHECKED attribute in the corresponding <INPUT> tag. The control’s value can be ON and OFF (or 1 and 0), indicating whether it’s checked or cleared, respectively.

**The RadioButton Control**

The RadioButton control is round and contains a dot in the center. RadioButton controls are used to present lists of options, similar to the CheckBox controls, but only one of a set can be selected at a time. Each time a new option is checked by the visitor, the previously selected one is cleared. To insert a RadioButton control on a form use the following:

<INPUT TYPE="radio" NAME="Radio1">

Whereas each CheckBox control has a different name, a group of RadioButtons all have the same name. This is how the browser knows which RadioButton controls belong to the same group and that only one of them can be checked at a time. To specify the control that will be initially checked in the group, use the CHECKED attribute. The following lines insert a group of four RadioButton controls on a form:

<INPUT TYPE="radio" NAME="Level">Beginner<BR>

<INPUT TYPE="radio" NAME="Level">Intermediate<BR>

<INPUT TYPE="radio" NAME="Level" CHECKED="checked">Advanced<BR>

<INPUT TYPE="radio" NAME="Level">Expert<BR>

**The Multiple Selection Control**

The Multiple Selection control is basically a list of options. The visitor can select none, one, or multiple items in the list. The list is delimited with a pair of <SELECT> tags. Each item in the list is inserted with a separate <OPTION>. To place a Multiple Selection List on the form, add the following lines:

<SELECT NAME="MemoryOptions" SIZE="3" MULTIPLE="multiple">

<OPTION VALUE="16">16 MB</OPTION>

<OPTION VALUE="32">32 MB</OPTION>

<OPTION VALUE="64">64 MB</OPTION>

<OPTION VALUE="128">128 MB</OPTION>

<OPTION VALUE="256">256 MB</OPTION>

</SELECT>

The SIZE attribute specifies how many lines will be visible. If you omit it, the list will be reduced to a single line, and the visitor must use the up and down arrow keys to scroll through the available options. If the list contains more lines, a vertical scroll bar is automatically attached to help the visitor locate the desired item. The MULTIPLE attribute specifies that the visitor can select multiple items in the list by clicking their names while holding down the Shift or Ctrl key. If you omit the MULTIPLE attribute, each time an item is selected, the previously selected one is cleared.

The <OPTION> tag has a VALUE attribute that represents the value of the selected item. If the user selects the 64 MB option in the earlier list, the value 64 is transmitted to the server. Finally, to initially select one or more options, specify the SELECTED attribute:

<OPTION SELECTED="selected" VALUE="128"> 128 MB</OPTION>

**The Command Button Control**

Clicking a Command button triggers certain actions. Without VBScript, Command buttons can trigger only two actions:

\_ Submit the data entered on the controls to the server.

\_ Reset all control values on the form to their original values.

With VBScript, Command buttons can trigger any actions you can program in your pages. You can place three types of buttons on a form: Submit, Reset, and General.

The most important button is Submit. It transmits the contents of all the controls on the form to the server (the values will be processed by an application whose URL is specified in the ACTION attribute of the <FORM> tag). The Reset button resets the values of the other controls on the form to their initial values. The Reset button doesn’t submit any values to the server. Most forms contain Submit and Reset buttons, which are inserted like this:

<INPUT TYPE="submit" VALUE="Send Data">

<INPUT TYPE="reset" VALUE="Reset Values">

The VALUE attribute specifies the string that will appear on the button—its caption. The Submit button reads the name of the application that must be contacted on the server (the <FORM> tag’s ACTION attribute), appends the values of the controls to this URL, and transmits it to the server.

**Processing Requests on the Server**

The RegisterForm.htm page, previously shown in Figure 23.4, contains several of the controls you can place on a Web page to request information from the user. The FORM section of the page is defined with the following tag:

<FORM ACTION="ASP/Register.asp" METHOD="GET">

The data collected on this page will be transmitted to the application Register.asp on the same server, and they will be processed there.

What you must keep in mind for now is that the browser will automatically submit the controls’ values to the server. All you have to do is specify the URL of the program to intercept them on the server in the <FORM> tag. The URL used in this example begins with the ASP folder. With no protocol, domain, or other parent folder specified, the ASP folder has to be in the same location as the current document, and the browser remembers where that is. The URL of this example is equivalent to http://www.example.com/ASP/Register.asp (where the first part of the address is the location of the HTML page with the form). The data will be transmitted to the server when the Submit button (Register Now) at the bottom of the form is clicked.

The rest of the code is trivial. It uses the <INPUT> tag to display the various controls and most of the controls are grouped into tables for alignment purposes. You can open the RegisterForm.htm file, from this chapter’s folder on the CD, to see the statements for creating the page shown in Figure 23.4.

Listing 23.2 shows how the various inputs are constructed; I’ve listed the tags for the intrinsic controls only, omitting the table and text-formatting tags in the interest of conserving space.

Listing 23.2: Key Elements from the RegisterForm.htm Page

<FORM ACTION="ASP/Register.asp" METHOD="GET">

Last Name

<INPUT TYPE="text" SIZE="20" MAXLENGTH="20" NAME="LName">

First Name

<INPUT TYPE="text" SIZE="20" MAXLENGTH="20" NAME="FName">

E-Mail Address

<INPUT TYPE="text" SIZE="46" MAXLENGTH="256" NAME="EMail">

My computer is:

<INPUT TYPE="radio" CHECKED="checked" NAME="Hardware" VALUE="PC">PC<BR>

<INPUT TYPE="radio" NAME="Hardware" VALUE="Mac">Macintosh<BR>

<INPUT TYPE="radio" NAME="Hardware" VALUE="OtherHardware">Other<BR>

My browser is:

<INPUT TYPE="radio" CHECKED="checked" NAME="Browser" VALUE="IE">Internet

Explorer

<INPUT TYPE="radio" NAME="Browser" VALUE="Netscape">Netscape

<INPUT TYPE="radio" NAME="Browser" VALUE="OtherBrowser">Other

When I connect I want to see:

<INPUT TYPE="checkbox" NAME="Sports" VALUE="ON">Sports

<INPUT TYPE="checkbox" NAME="News" VALUE="ON">News

<INPUT TYPE="checkbox" NAME="Stock" VALUE="ON">Stock Prices

<INPUT TYPE="checkbox" NAME="Weather" VALUE="ON">Weather

<INPUT TYPE="checkbox" NAME="Bargains" VALUE="ON">Our Bargains

Do you want to receive e-mail messages?

<INPUT TYPE="radio" CHECKED="checked" NAME="Mail" VALUE="YES">Yes

<INPUT TYPE="radio" NAME="Mail" value="NO">No

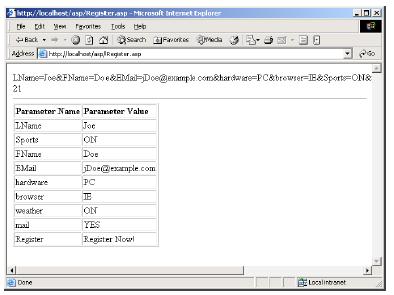
Click here to submit your registration

<INPUT TYPE="submit" NAME="Register" VALUE="Register Now!">

</FORM>

If you click the Register Now button, the browser displays a warning, indicating that it couldn’t find the Register.asp application. This page can’t be tested without a Web server running that program (otherwise, you’d get a result similar to that shown in Figure 23.5). You can view its contents like any other page, but it can’t contact the server unless it’s opened on a Web server.

To see how this form works with the Register application, copy the RegisterForm.htm file from the CD to the root folder of your Web server. The default root folder is C:\InetPub\wwwroot. Even better, create a subfolder under the root folder of the Web server and store the file there. For the example of this section, I have used the ASP folder. Then create a new file with the statements of Listing 23.3, name it Register.asp, and copy it into the same folder as the HTML file (or copy the Register.asp file from the CD to the ASP folder).



Listing 23.3: Reading the Parameter Values Passed to the Server

<HTML>

<BODY>

<%

Response.Write Request.QueryString

Response.Write "<HR>"

Response.Write "<TABLE BORDER=’1’ RULES=’ALL’>"

Response.Write "<TR><TD><B>Parameter

Name</B></TD><TD><B>Parameter

Value</B></TD></TD>"

Set Params = Request.QueryString

For Each PValue in Params

Response.Write "<TR><TD>" & PValue & "</TD><TD>" & Params(PValue) & "</TD></TR>"

Next

Response.Write "</TABLE>"

%>

</BODY>

</HTML>

Let me go through this script quickly. The tags <% and %> delimit the script, which is executed on the server. Anything that appears outside these two tags must be straight HTML code, which is sent to the client as is. The first two lines, as well as the last two, are sent to the client without any further processing. The lines between the <% and %> tags are considered statements and are executed on the server. The output they produce replaces them in the output. For example, the following code is a script that can be embedded anywhere on a page:

The date on the server is <% = Date %>.

When executed, this statement will send a string like "The date on the server is 3/21/2001."

The ASP script must prepare a new page to send to the client. Therefore, it must emit HTML code. The Response object represents the information you want to send to the client and you use the Write method of the Response object to send HTML code to the client. The Request object represents the request made by the client, and the QueryString property of the Request object holds the data passed to the server along with the request.

Now start Internet Explorer and open the RegisterForm.htm file by entering the following URL in the browser’s Address bar:

http://127.0.0.1/ASP/RegisterForm.htm

Enter some values on the controls and click the Register Now button. The browser will request another page with a URL like the following:

http://127.0.0.1/ASP/Register.asp?LName=Doe&FName=Joe&EMail=JoeDoe@example.com&

Hardware=PC&Browser=Netscape&Sports=ON&Stock=ON&Mail=YES&Register=Register+Now%21

This is a long URL, but notice that it starts with the address of the ASP file that will process this page on the server. Following the question mark is a long string with the values you have entered on the controls. Each value is submitted to the server as a pair of a name and a value, and each name/value pair is delimited with the & symbol. This long string will be stored in the Request.QueryString property, and you can access this property from within the script’s code. The expression Request.QueryString will return the string with the parameters following the name of the script. You can also access individual parameters by name. The value of the LName TextBox is returned by the expression Request.QueryString("LName").

The RegisterForm.htm page passes the parameters with the GET method. An alternative method of passing the parameters is the POST method (this is specified in the FORM tag). If you use the POST method, the parameter values are not displayed on the address bar and you can retrieve their values on the server through the Form property of the Request object. Change the <FORM> tag in the HTML page from

<FORM ACTION="ASP/Register.asp" METHOD="GET">

to <FORM ACTION="ASP/Register.asp" METHOD="POST">

and then change the script as follows:

<%

Response.Write Request.Form

Response.Write "<HR>"

Response.Write "<BODY>"

Response.Write "<TABLE BORDER=’1’ RULES=’ALL’>"

Response.Write "<TR><TD><B>Parameter Name</B></TD>

Response.Write "<TD><B>Parameter Value</B></TD></TR>"

Set Params = Request.Form

For Each PValue in Params

Response.Write "<TR><TD>" & PValue & "</TD><TD>" & Params(PValue) & "</TR></TR>"

Next

Response.Write "</TABLE>"

%>

The changes are trivial: just replace all instances of the QueryString property with the Form property. The two methods of passing parameter values to the server are equivalent, but the POST method allows you to pass longer strings and doesn’t append the parameter values to the URL.

This is how ASP processes the requests. It’s also how ASP.NET processes the requests, but ASP.NET makes it look as if you’re working with a Windows form. As you will see shortly, it pretends that the LName control is a TextBox that exposes the Text property, and it allows you to access the contents of the LName control with the expression LName.Text. It generates a lot of code behind the scenes, but fortunately you don’t have to see it. You work as if you’re writing code for a Windows application, but what you get is an application that works with the browser.