TRAINING & REFERENCE

murach's Java servlets and JSP

(Chapter 3)

Andrea Steelman
Joel Murach

MIKE MURACH & ASSOCIATES, INC.
1-800-221-5528 • (559) 440-9071 • Fax: (559) 440-0963
murachbooks@murach.com • www.murach.com

Book contents

Introduction xi

Section 1 Introduction to servlet and JSP programming

Chapter 1 An introduction to web programming 3
Chapter 2 How to install and use Tomcat 29
Chapter 3 A crash course in HTML 61

Section 2 The essence of servlet and JSP programming

Chapter 4 How to develop JavaServer Pages 101
Chapter 5 How to develop servlets 141
Chapter 6 How to structure a web application 167
Chapter 7 How to work with sessions and cookies 203
Chapter 8 How to create and use JavaBeans 243
Chapter 9 How to work with custom JSP tags 267

Section 3 The essential database skills

Chapter 10 How to use MySQL to work with a database 307
Chapter 11 How to use Java to work with a database 333

Section 4 Advanced servlet and JSP skills

Chapter 12 How to use JavaMail to send email 375
Chapter 13 How to use SSL to work with a secure connection 397
Chapter 14 How to restrict access to a web resource 417
Chapter 15 How to work with HTTP requests and responses 437
Chapter 16 How to work with XML 465
Chapter 17 An introduction to Enterprise JavaBeans 503

Section 5 The Music Store web site

Chapter 18 An introduction to the Music Store web site 523
Chapter 19 The Download application 545
Chapter 20 The Shopping Cart application 557
Chapter 21 The Admin application 579

Appendixes

Appendix A How to install the software and applications for this book 599
Index 614

3

A crash course in HTML

In a typical web application, HyperText Markup Language is used to provide the user interface. Then, the user can navigate through the HTML pages to view data, and the user can enter data into HTML forms that pass the data to JavaServer Pages or servlets. That's why you need to have a basic set of HTML skills before you can code JSPs or servlets. And that's why this chapter presents a crash course in HTML.

If you already know how to code HTML, of course, you can skip this chapter. And if you want to learn more about HTML after you read this chapter, you can get a book that's dedicated entirely to HTML. Keep in mind, though, that this chapter presents all the HTML that you will need to know as you use this book.

An introduction to HTML 62
Tools for working with HTML 62
An HTML document 64
How to code and view an HTML page 66
How to code an HTML document 66
Where and how to save an HTML document 68
How to view an HTML page 70
More skills for coding HTML documents 72
How to code links to other HTML pages 72
How to code tables 74
Attributes for working with table tags 76
How to include images in an HTML page 78
How to use a style sheet 80
How to code HTML forms 82
How to code a form 82
How to code text boxes, password boxes, and hidden fields 84
How to code buttons 86
How to code check boxes and radio buttons 88
How to code combo boxes and list boxes 90
How to code text areas 92
How to set the tab order of controls 94
Perspective 96

An introduction to HTML

This topic introduces you to HTML and to the editors and IDEs that you can use for coding HTML. Once you understand that, you can learn the specifics of HTML coding.

Tools for working with HTML

When you're coding HTML or JSPs, you can use a general text editor like Notepad or TextPad. In fact, TextPad can be set up so it uses colors to identify the various parts of the HTML syntax, which makes it quite efficient for HTML editing.

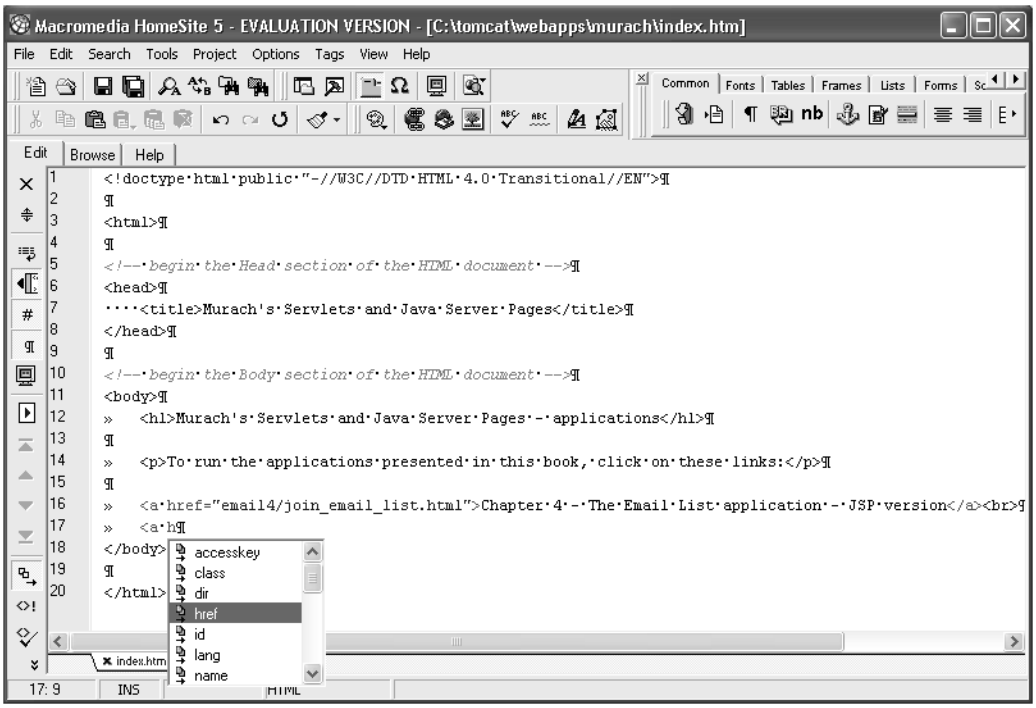
Another alternative, though, is to use an editor that's specifically designed for working with HTML pages and JSPs. One of the most popular of these editors is Macromedia's HomeSite, which is shown in figure 3-1. HomeSite not only uses color, boldfacing, and italics to identify various parts of the HTML syntax, but it also provides tools that can help you enter and edit HTML code. For example, this figure shows a pop-up box that can help you write specific types of code.

If you want to experiment with HomeSite, you'll find a trial copy on the CD that comes with this book. Then, if you decide that you want to continue using it, you can purchase it for a cost of about $85. If cost is an issue, though, you can do all of the HTML and JSP editing that this book requires by using TextPad, which is also available on the CD.

A third alternative for entering and editing HTML code and JSPs is to use an *Integrated Development Environment* (*IDE*) that's designed for developing web applications. Of these, Macromedia's Dreamweaver is the industry leader. It not only provides an editor for HTML and JSPs, but also visual tools for designing and maintaining web pages.

Because an IDE like Dreamweaver presents its own learning problems, we think it's best to start by using TextPad or HomeSite instead of an IDE. Once you master the skills in this book, though, you shouldn't have any trouble mastering an IDE like Dreamweaver. And if you've been using HomeSite, you'll find that it integrates well with Dreamweaver since they're both made by the same company.

Macromedia's HomeSite



Description

- *HyperText Markup Language*, or *HTML*, is used to provide the user interface for web applications.
- To write and edit HTML code and JSPs, you can use a general text editor like TextPad, a text editor that's specifically designed for working with HTML, or an *Integrated Development Environment*, or *IDE*, that's designed for developing web applications.
- Macromedia's HomeSite is a text editor that's specifically designed for working with HTML and JSPs, and it is one of the most popular HTML editors on the market.
- Macromedia's Dreamweaver is the industry-leading IDE that lets you create and manage web sites and Internet applications. It provides visual layout tools and extensive editing support.

Figure 3-1 Tools for working with HTML

An HTML document

Figure 3-2 shows a simple *HTML page* in a browser along with the *HTML document* that contains the HTML code for the page. Within the HTML code, *HTML tags* define the elements of the HTML page. Each of these tags is coded within an opening bracket (<) and a closing bracket (>).

Since HTML tags aren't case sensitive, you can use upper or lowercase letters when you code your tags. Since lowercase is easier to read and type, most programmers use lowercase when coding HTML tags, and that's how the tags are coded in this book.

When coding an HTML document, you can use spaces, indentation, and blank lines to make your code easier to read. In this HTML document, for example, blank lines separate the Head and Body sections, and all tags within the Head and Body sections are indented. As a result, it's easy to tell where these sections begin and end. Similarly, the text after the heading in the body section is broken into two lines to make it easier to read. However, the web browser displays this text as a single line.

An HTML page viewed in a browser

The HTML document for the page

```
<html>

<head>
    <title>Murach's Java Servlets and JSP</title>
</head>

<body>
    <h1>Murach's Java Servlets and JSP applications</h1>
    <p>To run the applications presented in this book,
       click on these links:</p>
</body>

</html>
```

Description

- An *HTML document* is used to define each *HTML page* that's displayed by a web browser.
- Within an HTML document, *HTML tags* define how the page will look when it is displayed. Each of these HTML tags is coded within a set of brackets (< >).
- Since HTML tags aren't case sensitive, you can use upper or lowercase letters for your tags.
- To make your code easier to read, you can use spaces, indentation, and blank lines.

Figure 3-2 An HTML document

How to code and view an HTML page

Now that you have a general idea of how HTML works, you're ready to learn more about coding, saving, and viewing an HTML page. To start, you'll learn about some of the most common HTML tags.

How to code an HTML document

Figure 3-3 shows how to code some of the most common HTML tags. Most of these tags require both an opening tag and a closing tag. Although the closing tags are similar to opening tags, they begin with a slash. For example, you can boldface text like this:

```
<b>This text will be boldfaced</b> but this text won't be.
```

On the other hand, some HTML tags don't require a closing tag. For example, you can insert a line break like this:

```
The text for line 1<br>
The text for line 2
```

To start the HTML document, the code in this figure uses a DocType tag that identifies the type of document that's being sent to the browser:

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//
EN">
```

This tag lets the browser know that this HTML document is an HTML 4.0 document that transitionally conforms to the W3C standard and is in the English language. As a Java programmer, you don't need to completely understand this tag, but you usually begin your web pages with this DocType tag or one that's similar to it. Some HTML editors will automatically insert this tag for you.

After the DocType tag, the opening HTML tag identifies the start of the HTML document, and the closing HTML tag identifies the end of the HTML document. Then, within the HTML tags, the Head and Body tags identify the Head and Body sections of the HTML document. Within the Head tags, the Title tag identifies the title that's displayed in the title bar of the browser. Within the Body tags, the H1 tag identifies a level-1 heading, the H2 tag identifies a level-2 heading, and the P tag identifies a paragraph. Since these tags are coded within the Body tags, they're displayed within the main browser window.

To document your HTML code, you can use *comments* in the same way that you use them in other languages. In this example, comments are used to identify the beginning of the Head and Body sections, but that's just for the purpose of illustration. You can also use comments to "comment out" portions of HTML code that you don't want rendered by the browser. Then, if you want to restore the code, you can just remove the comment's opening and closing tags.

The code for an HTML document that contains comments

```
<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>

<!-- begin the Head section of the HTML document -->
<head>
    <title>Murach's Java Servlets and JSP</title>
</head>

<!-- begin the Body section of the HTML document -->
<body>
    <h1>Murach's Java Servlets and JSP applications</h1>
    <p>Here are links to the applications presented in this book:</p>
</body>

</html>
```

Basic HTML tags

Tag	Description
`<!doctype ... >`	Identifies the type of HTML document. This tag is often inserted automatically by the HTML editor.
`<html> </html>`	Marks the beginning and end of an HTML document.
`<head> </head>`	Marks the beginning and end of the Head section of the HTML document.
`<title> </title>`	Marks the title that is displayed in the title bar of the browser.
`<body> </body>`	Marks the beginning and end of the Body section of the HTML document.
`<h1> </h1>`	Tells the browser to use the default format for a heading-1 paragraph.
`<h2> </h2>`	Tells the browser to use the default format for a heading-2 paragraph.
`<p> </p>`	Tells the browser to use the default format for a standard paragraph.
` `	Inserts a line break.
`<b> </b>`	Marks text as bold.
`<i> </i>`	Marks text as italic.
`<u> </u>`	Marks text as underlined.
`<!-- -->`	Defines a comment that is ignored by the browser.

Description

- The DocType tag lets the browser know what version of HTML is being used, what standards it conforms to, and what language is used.
- Within the Head tags, you code any tags that apply to the entire page such as the Title tag.
- Within the Body tags, you code the text and other components for the body of the HTML page. When working with text, you can use the H1, H2, and P tags to apply the default formatting that's assigned to heading-l, heading-2, and standard paragraphs. You can also use the B, I, and U tags to apply bold, italics, or underlining.
- *Comments* can be used anywhere within an HTML document to document portions of code or to tell the browser not to process portions of code.

Figure 3-3 How to code an HTML document

Where and how to save an HTML document

Figure 3-4 shows where and how to save HTML documents to make them available to a web application. To start, this figure shows the directories where you can store HTML files and shows how to name your HTML files. Then, this figure shows the Save As dialog box for a typical HTML editor.

To make HTML pages available to a web application, you must store them in a directory that's available to the web server. Although this directory varies from one server to another, you can use any subdirectory of the webapps directory when you're working with Tomcat 4.0. For instance, all of the applications that are presented in the first 16 chapters of this book are stored in Tomcat's webapps\murach directory. You can also use the webapps\ROOT directory that's created when you install Tomcat. And you can create your own subdirectories under the webapps or ROOT directory for storing your HTML files.

When you save an HTML file, you must make sure to include an extension of htm or html. In addition, you shouldn't use any spaces in the filename. By convention, most web programmers use underscore characters instead of spaces. However, you can also use a mixture of uppercase and lowercase letters. If you're using a text editor that isn't designed for working with HTML documents, you may need to enclose the filename of the HTML document in quotes. Otherwise, the text editor might not include the extension for the file.

The root directory for the applications in the first 16 chapters of this book

```
c:\tomcat\webapps\murach
```

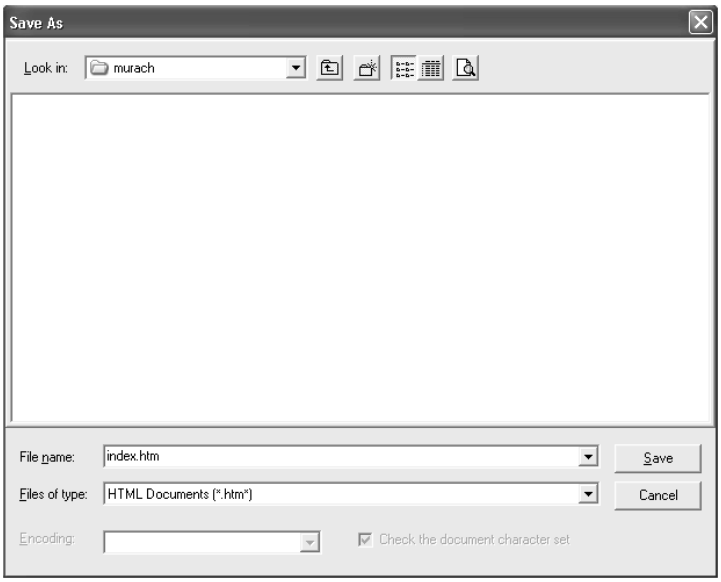
Other directories for saving HTML documents in Tomcat 4.0

```
c:\tomcat\webapps\yourDocumentRoot
c:\tomcat\webapps\yourDocumentRoot\yourSubdirectory
c:\tomcat\webapps\ROOT
c:\tomcat\webapps\ROOT\yourSubdirectory
```

Typical names for HTML documents

```
index.htm
index.html
join_email_list.htm
join_email_list.html
```

A dialog box for saving an HTML document



Description

- To make an HTML document available to a user, you must save it in a directory that's available to the web server. If you're starting to work on a new system, you need to find out what directories are available for your web pages.
- For Tomcat 4.0, you must save your HTML files in a subdirectory of Tomcat's webapps directory.
- All HTML files that are presented in the first 16 chapters of this book are stored in the webapps\murach directory or one of its subdirectories.
- When naming an HTML file, don't use spaces. Use underscores instead. In addition, you must be sure that the file is saved with an extension of htm or html.
- In the Save As dialog box, you can make sure that the filename is saved with the right extension by entering the filename and extension within quotation marks.

Figure 3-4 Where and how to save an HTML document

How to view an HTML page

Once you've coded an HTML document and saved it in a directory that's available to the web server, you can view the HTML page as shown in figure 3-5. To do that, you start your web browser and enter an HTTP URL as shown in this figure.

If, for example, you have saved an HTML file named index.htm in the webapps\murach directory, you can view that file by entering

```
http://localhost:8080/murach/index.htm
```

Or, since most web servers will automatically look for an index file when you enter a directory, you can get the same file by entering

```
http://localhost:8080/murach
```

Finally, if you want to view a web page named join_email_list.htm that's stored in the murach\email directory, you can enter the name of the file like this:

```
http://localhost:8080/murach/email/join_email_list.htm
```

While you're coding a web page, it's common to have the web page open in a browser. Then, when you make a change to the web page, you can easily view the change in the browser. To do that, use your HTML editor to save the change, switch to the browser, and click the Refresh (for the Explorer) or Reload button (for Netscape).

Incidentally, you can also view an HTML page without going through the web server by entering the path and file name in the Address box of your browser. This is illustrated by the third URL in this figure. And if you double-click on the name of an HTML file in your Windows Explorer, your web browser will open and the page will be displayed.

An HTML page viewed in a browser

How Tomcat maps directories to HTTP calls

Directory	URL
c:\tomcat\webapps\murach	http://localhost:8080/murach/
c:\tomcat \webapps\murach\email	http://localhost:8080/murach/email
c:\tomcat \webapps\root	http://localhost:8080/
c:\tomcat \webapps\root\email	http://localhost:8080/email

A URL that requests an HTML page from a local server

```
http://localhost:8080/murach/email/join_email_list.htm
```

A URL that requests an HTML page from an Internet server

```
http://www.murach.com/email/join_email_list.htm
```

A URL that requests an HTML page without going through a server

```
c:\tomcat\webapps\murach\email\join_email_list.htm
```

Description

- To use HTTP to view a local HTML page, start the Tomcat server and enter a URL that calls the page.
- To use HTTP to view a web page that's on the Internet, connect to the Internet and enter the URL for the page.
- If you modify the HTML for a page and save your changes, you can view the new results in the browser by clicking the Refresh button (for the Internet Explorer) or the Reload button (for Netscape).
- You can also view an HTML page without going through the web server by entering its path and filename or by double-clicking on it in the Windows Explorer. However, this won't work correctly for JSPs, which need to go through the web server.

Figure 3-5 How to view an HTML page

More skills for coding HTML documents

Now that you know how to code, save, and view a simple HTML document, you're ready to learn more of the skills for working with HTML documents. In particular, you're going to learn the HTML tags that are used in most commercial web pages. These, of course, are the tags that Java programmers need to know and understand.

How to code links to other HTML pages

Any web site or web application is a series of web pages that are connected by *links*. To code a link to another web page, you can use an Anchor, or A tag, as shown in figure 3-6. Within the opening tag, you code an *attribute* that contains a *value* that identifies the linked page. Between the opening and closing tags, you code the text that describes the link. When viewed in a browser, this text is usually underlined. Then, the user can click on the link to jump to the page that's specified by the link.

To code an attribute, you code the name of the attribute, the equals sign, and the value, with no intervening spaces. However, if you code two or more attributes within a single tag, you separate the attributes with spaces. Although it's generally considered a good coding practice to include quotation marks around the attribute's value, that's not required. As a result, you may come across HTML code that doesn't include the quotation marks.

The first group of examples in this figure shows how to use a *relative link*. Here, the first link specifies a web page named join_email_list.htm that's in the same directory as the current web page. Then, the second link specifies the join_email_list.htm page that's stored in the email subdirectory of the current directory. If necessary, you can code a link that navigates down several directories like this:

```
books/java/ch01/toc.htm
```

The second group of examples shows how to navigate up the directory hierarchy. To do that, you can start your link with a slash or two periods (..). The first two examples in this group use two periods to navigate up one or two directory levels. Since these examples don't include the name of a web page, they select the index page for that directory. The third and forth examples in this group show how to use a slash to navigate to the directory that's defined as the root directory by your web server.

The third group of examples shows how to use an *absolute link* to specify a web page. Here, the first link shows how to specify a web page named join_email_list.htm that's stored in the email subdirectory of the root HTML directory for the www.murach.com web site. The second link specifies the same web page, but it uses an IP address instead of a URL. Although you rarely need to use IP addresses, you do need to use them for sites that haven't yet been assigned their domain names.

Two Anchor tags viewed in a browser

The Email List application 1
The Email List application 2

Examples of Anchor tags

Anchor tags with relative URLs that are based on the current directory

```
<a href="join_email_list.htm">The Email List application 1</a><br>
<a href="email/join_email_list.htm">The Email List application 2</a><br>
```

Anchor tags with relative URLs that navigate up the directory structure

```
<a href="../">Go back one directory level</a>
<a href="../../">Go back two directory levels</a>
<a href="/">Go to the root level for HTML files</a>
<a href="/murach">Go to the murach subdirectory of the root level</a>
```

Anchor tags with absolute URLs

```
<a href="http://www.murach.com/email">An Internet address</a>
<a href="http://64.71.179.86/email">An IP address</a>
```

The Anchor tag

Tag	Description
`<a> </a>`	Defines a link to another URL. When the user clicks on the text that's displayed by the tag, the browser requests the page that is identified by the Href attribute of the tag.

One attribute of the Anchor tag

Attribute	Description
`href`	Specifies the URL for the link.

How to code attributes for tags

- Within the starting tag, code a space and the attribute name. Then, if the attribute requires a value, code the equals sign followed by the value between quotation marks with no intervening spaces.
- If more than one attribute is required, separate the attributes with spaces.

Description

- A tag can have one or more *attributes*, and most attributes require values. Although it's considered a good coding practice to code values within quotation marks, the quotation marks aren't required.
- The Href attribute of the Anchor tag is used to specify the URL that identifies the HTML page that the browser should request when the user clicks on the text for this tag.
- When you code a *relative URL* in the Href attribute, you base it on the current directory, which is the one for the current HTML page. To go to the root directory for web pages, you can code a slash. To go up one level from the current directory, you can code two periods and a slash. And so on.
- When you code an *absolute URL*, you code the complete URL. To do that, you can code the name of the host or the IP address for the host.

Figure 3-6 How to code links to other HTML pages

How to code tables

If you do any serious work with HTML, you need to use one or more *tables* to present data in *rows* and *columns* as shown in figure 3-7. To start, you use the Table tag to identify the start and end of the table. Within the Table tag, you use the Table Row tag, or TR tag, to specify a new row, and you use the Table Data tag, or TD tag, to specify a new column. In this figure, for example, the table contains three rows and two columns.

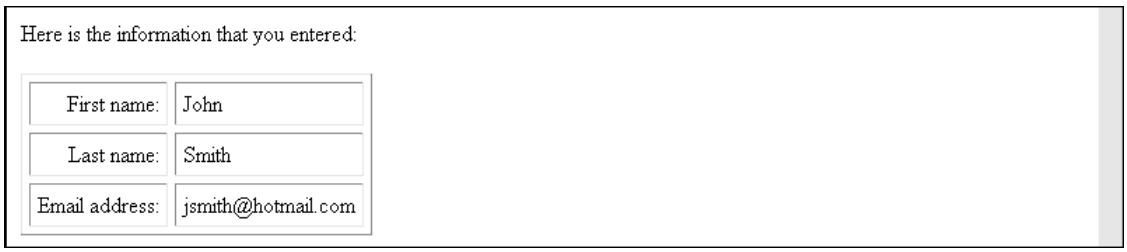
The intersection of a row and column is known as a *cell*. Typically, each cell of a table stores text as shown in this figure. However, cells can store any type of data including links to other pages, images, controls, and even other tables. In fact, it's common for a web page to contain one or more tables nested within other tables. As a Java programmer, though, you will probably only have to work with one table at a time.

The HTML code for a table

```
<p>Here is the information that you entered:</p>

<table cellspacing="5" cellpadding="5" border="1">
  <tr>
    <td align="right">First name:</td>
    <td>John</td>
  </tr>
  <tr>
    <td align="right">Last name:</td>
    <td>Smith</td>
  </tr>
  <tr>
    <td align="right">Email address:</td>
    <td>jsmith@hotmail.com</td>
  </tr>
</table>
```

The table displayed in a browser



The tags for working with tables

Tag	Description
`<table> </table>`	Marks the start and end of the table.
`<tr> </tr>`	Marks the start and end of each row.
`<td> </td>`	Marks the start and end of each cell within a row.

Description

- A *table* consists of *rows* and *columns*. The intersection of a row and column creates a *cell* that can hold data.
- Although this figure shows a single table that stores text within each cell, it's common to store other types of data within a cell such as images, links, or even another table.
- The attributes in figure 3-8 can be used with the Table, TR, and TD tags to control the formatting of the table.

Figure 3-7 How to code tables

Attributes for working with table tags

When coding the HTML for a table, you usually need to use some of the attributes that are summarized in figure 3-8. In the last figure, for example, you can see the use of the Border, CellSpacing, and CellPadding attributes of the Table tag. They make the table border visible and add some spacing and padding between the cells of the table. That example also uses the Align attribute of the TD tag to align the text in the first column with the right edge of that column.

When you're creating and modifying tables, you can set the Border attribute to 1 so you can see the cells within the table. This makes it easier to organize the rows and columns of the table and to set the attributes of the table. Then, when you've got the table the way you want it, you can set the Border attribute to 0 to hide the borders of the table.

Although this figure doesn't present all of the attributes for working with tables, it does present the most useful ones. If you experiment with them, you should be able to figure out how they work. In addition, a good HTML editor will provide pop-up lists of other attributes that you can use when you're working with tables.

When working with a table, you can use *pixels* to specify most height and width measurements. Pixels are the tiny dots on your computer's monitor that display the text and images that you see. Today, many computers display a screen that's 800 pixels wide and 600 pixels tall. However, some computers run at higher resolutions such as 1024 pixels wide by 728 pixels tall. When coding a table, then, it's a good practice to make the total width of your table less than the total number of pixels for your target resolution. That usually means coding tables where all cell spacing and width measurements add up to be less than 780 (800 pixels wide minus 20 pixels for the vertical scroll bar).

Attributes of the Table tag

Attribute	Description
border	Specifies the visual border of the table. To turn the border off, specify a value of 0. To specify the width of the border in pixels, specify a value of 1 or greater.
cellspacing	Specifies the number of pixels between cells.
cellpadding	Specifies the number of pixels between the contents of a cell and the edge of the cell.
width	Specifies the width of the table. To specify the width in pixels, use a number such as 300. To specify a percent of the browser's display space, use a number followed by the percent sign such as 60%.
height	Specifies the height of the table in pixels or as a percentage of the browser's display space. This works like the Width attribute.

Attributes of the TR tag

Attribute	Description
valign	Specifies the vertical alignment of the contents of the row. Acceptable values include Top, Bottom, and Middle.

Attributes of the TD tag

Attribute	Description
align	Specifies the horizontal alignment of the contents of the cell. Acceptable values include Left, Right, and Center.
colspan	Specifies the number of columns that the cell will span.
rowspan	Specifies the number of rows that the cell will span.
height	Specifies the height of the cell in pixels.
width	Specifies the width of the cell in pixels.
valign	Specifies the vertical alignment of the contents of the row. Acceptable values include Top, Bottom, and Middle and will override any settings in the TR tag.

Description

- Although there are other attributes for working with tables, these are the ones that are commonly used.

Figure 3-8 Attributes for working with table tags

How to include images in an HTML page

Although text and links are an important part of any web site, most web pages include one or more images. Figure 3-9 shows how to use the Image (or IMG) tag to display an image. Unlike most of the tags presented so far, the IMG tag doesn't have a closing tag. As a result, you just need to code the tag and its attributes.

The example in this figure shows how to code an IMG tag and its three required attributes. Here, you must code the Source (SRC) attribute to specify the image that you want to include. In this case, the SRC attribute specifies the filename of an image that's stored in the images directory. If you have any trouble specifying the file for the image, please refer to figure 3-6 because many of the skills that apply to the Href attribute of the Anchor tag also apply to the SRC attribute of the IMG tag.

After the SRC attribute, you can code the Width and Height attributes. The values for these attributes must specify the height and width of the image in pixels. If you use one of your HTML editor's tools to insert an image, all three of these attributes may be automatically specified for you. Otherwise, you may need to open the image in an image editor to determine the number of pixels for the width and height.

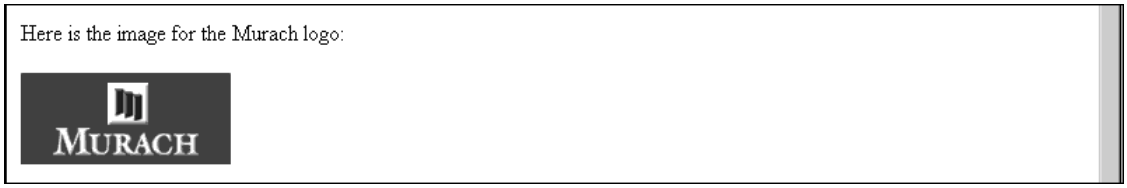
The last five attributes described in this figure aren't required, but they're commonly used. For example, the Alt attribute is commonly used to specify text that's displayed if an image can't be loaded. This text is usually displayed when a web browser takes a long time to load all of the images on a page, but it can also be displayed for text-only browsers. If you experiment with the attributes described in this list, you shouldn't have any trouble using them.

When you include images in a web page, you can use a *Graphic Interchange Format* file, or *GIF* file, or you can use a *Joint Photographic Experts Group* file, or *JPEG* file. Typically, a web designer will use imaging software such as Adobe's Photoshop to create and maintain these files for a web site and will save these files in a directory named images or graphics. In this book, for example, all images used by the applications for the first 16 chapters are stored in the murach\images directory. Although GIF files are stored with a GIF extension, JPEG files can be stored with either a JPEG or JPG extension.

HTML code that includes an image

```
<p>Here is the image for the Murach logo:</p>
<img src="../images/murachlogo.gif" width="150" height="65">
```

The HTML code displayed in a browser



The Image tag

Tag	Description
<img>	Specifies how to place a GIF or JPEG image within an HTML page.

Common attributes of the Image tag

Attribute	Description
src	Specifies the relative or absolute URL for the GIF or JPEG file.
height	Specifies the height of the image in pixels.
width	Specifies the width of the image in pixels.
alt	Specifies the text that's displayed when the image can't be displayed.
border	Specifies the width of the border in pixels with 0 specifying no border at all.
hspace	Specifies the horizontal space in pixels. This space is added to the left and right of the image.
vspace	Specifies the vertical space in pixels. This space is added to the top and bottom of the image.
align	Specifies the alignment of the image on the page. Acceptable values include Left, Right, Top, Bottom, and Middle.

Other examples

```
<img src="../../images/java.jpg" width="175" height="243">
<img src="http://www.murach.com/images/murachlogo.gif" width="150" height="65">
```

Description

- The two types of image formats that are supported by most web browsers are the *Graphic Interchange Format* (*GIF*) and the *Joint Photographic Experts Group* (*JPEG*). JPEG files, which have a JPEG or JPG extension, are typically used for photographs and scans, while GIF files are typically used for other types of images.
- GIF, JPG, and JPEG files are typically stored in a separate directory named images or graphics. All of the GIF, JPG, and JPEG files used by the applications for the first 16 chapters of this book are stored in the webapps\murach\images directory.
- If you use an HTML editor to insert the image tag into your page, the HTML editor usually sets the SRC, Height, and Width attributes automatically.

Figure 3-9 How to include images in an HTML page

How to use a style sheet

Although you can code formatting attributes such as fonts, colors, margins, and alignment within a web page, most commercial web sights store this information in a file known as a *style sheet*. That way, all of the web pages within a site will have a uniform look that can be quickly and easily modified if necessary. Figure 3-10 shows how you can code a style sheet and link it to your web pages.

To start, this figure shows the code for a style sheet that's stored in a file named murach.css in the webapps\murach\styles directory. Since this style sheet is stored in its own file and must be linked to web pages, it's known as an *external style sheet* or a *linked style sheet*. The styles in this style sheet set the font, color, indentation, and alignment for the P, A, H1, H2, and H3 tags. A style sheet like this is normally developed by the web designer.

Next, this figure shows how to link a web page to an external style sheet. To do that, you can code the Link tag and its attributes in the Head section of the HTML document. Here, the Rel attribute should always be set to "stylesheet", and the Href attribute works like it does for the Anchor tag in figure 3-6. Web programmers use this type of tag so the styles developed by the web designer are used in the web pages.

An external style sheet represents the highest level of a series of *cascading style sheets*. To override the styles in an external style sheet, you can code a style sheet within an opening and closing Style tag in the Head section of an HTML document. Similarly, you can override a Style tag coded in the Head section by coding a Style attribute within a tag. Most programmers, though, don't need to override or modify styles. Instead, they use the external style sheet developed by the web designer.

The code for a style sheet

```
p {font-family: Arial, sans-serif; font-size: 12px}
a:hover {text-decoration : underline; color : #CC0000}

h1 { font-family: Arial, sans-serif; font-size: 16px; color: #003366;
     vertical-align: top; margin-top: 10px; margin-bottom: 0px
   }
h2 { font-family: Arial, sans-serif; font-size: 16px; color: #003366}
h3 { font-family: Arial, sans-serif; font-size: 14px; color: #003366}
```

The code in the HTML document that links to a style sheet

```
<head>
    <title>Murach's Servlets and Java Server Pages</title>
    <link rel="stylesheet" href="../styles/murach.css">
</head>
```

An HTML page after the style sheet above has been applied to it

Murach's Java Servlets and JSP applications

To run the applications presented in this book, click on these links:

Chapter 4 - The Email List application - JSP version
Chapter 5 - The Email List application - servlet version
Chapter 6 - The Email List application - MVC version

The Link tag

Tag	Description
`<link>`	Specifies the external style sheet.

Attributes of the Link tag

Attribute	Description
`href`	Specifies the location of the style sheet.
`rel`	Specifies the type of link. To specify a style sheet, supply a value of “stylesheet”.

Description

- A *style sheet* can be used to define the font *styles* that are applied to the text of an HTML page. To identify the location of the style sheet that should be used for a page, you use the Link tag within the Head tags of the page.
- This type of style sheet can be referred to as an *external style sheet*, or *linked style* sheet, and it is actually the top style sheet in a series of *cascading style sheets*. An external style sheet is typically stored in its own directory, and css is used as the extension for its file name.
- The style sheet for the first 16 chapters of this book is named murach.css, and it is stored in the webapps\murach\styles directory.
- As a Java web programmer, you shouldn’t have to create style sheets of your own, but you should know how to use them if they are available to you.

Figure 3-10 How to use a style sheet

How to code HTML forms

This topic shows how to code an HTML *form* that contains one or more *controls* such as text boxes and buttons. In this topic, you'll learn how to code 11 types of controls to gather data. In the next chapter, you'll learn how to process the data that's gathered by these controls.

How to code a form

Figure 3-11 shows how to code a form that contains three controls: two text boxes and a button. To code a form, you begin by coding the opening and closing Form tags. Within the opening Form tag, you must code an Action attribute that specifies the servlet or JSP that will be called when the user clicks on the Submit button. In addition, you can code a Method attribute that specifies the HTTP method that will be used. Between the two Form tags, you code the controls for the form.

This example shows a form that contains two text boxes and a Submit button. When the user clicks on the Submit button, the data that's in the text boxes will be passed to the JSP named entry.jsp that's specified by the Action attribute of the form. Although a form can have many controls, it should always contain at least one control that executes the Action attribute of the form. Typically, this control is a Submit button like the one shown in this figure.

You can use the Input tag to code several types of controls. This figure shows how to use the Input tag to code a text box and a button, but you'll learn more about coding these controls in the next two figures.

When coding controls, you can use the Name attribute to specify a name that you can use in your Java code to access the parameter that's passed from the HTML form to a servlet or JSP. To access the data in the two text boxes in this figure, for example, you can use firstName to access the data that's been entered in the first text box, and you can use lastName to access the data in the second text box.

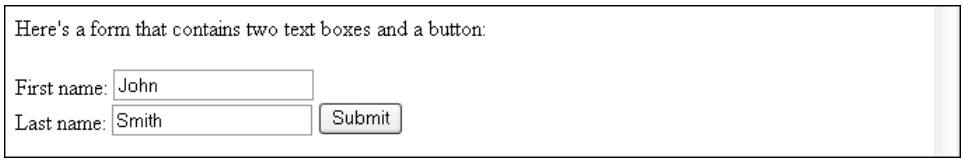
In contrast, you can use the Value attribute to specify a value for a control. This works differently depending on the control. If, for example, you're working with a button, the Value attribute specifies the text that's displayed on a button. But if you're working with a text box, the Value attribute specifies the default text that's displayed in the box.

The HTML code for a form

```
<p>Here's a form that contains two text boxes and a button:</p>

<form action="entry.jsp" method="post">
  <p>
  First name: <input type="text" name="firstName"><br>
  Last name: <input type="text" name="lastName">
  <input type="submit" value="Submit">
  </p>
</form>
```

The form displayed in a browser after the user enters data



Tags for working with a simple form

Tag	Description
`<form> </form>`	Defines the start and end of the form.
`<input>`	Defines the input type. You'll learn more about this tag in the next two figures.

Attributes of the Form tag

Attribute	Description
`action`	The Action attribute specifies the URL of the servlet or JSP that will be called when the user clicks on the Submit button. You'll learn more about specifying this attribute in the next two figures.
`method`	The Method attribute specifies the HTTP method that the browser will use for the HTTP request. The default method is the Get method, but the Post method is also commonly used. You'll learn more about these methods in the next chapter.

Common control attributes

Attribute	Description
`name`	The name of the control. When writing Java code, you can use this attribute to refer to the control.
`value`	The default value of the control. This varies depending on the type of control. For a text box, this attribute sets the default text that's displayed in the box. For a button, this attribute sets the text that's displayed on the button.

Description

- A *form* contains one or more *controls* such as text boxes, buttons, check boxes, and list boxes.
- A form should always contain at least one control like a Submit button that activates the Action attribute of the form. Then, any data in the controls is passed to the servlet or JSP that is identified by the URL in the Action attribute.

Figure 3-11 How to code a form

How to code text boxes, password boxes, and hidden fields

Figure 3-12 shows how to use the Input tag to code three types of *text boxes*. You can use a *standard text box* to accept text input from a user. You can use a *password box* to let a user enter a password that is displayed as one asterisk for each character that's entered. And you can use a *hidden field* to store text that you don't want to display on the HTML page.

To create a text box, you set the Type attribute to Text, Password, or Hidden. Then, you code the Name attribute so you'll be able to access the text that's stored in the text box from your Java code.

If you want the text box to contain a default value, you can code the Value attribute. Although all three of the text boxes shown in this figure have Value attributes, you often don't need to code them for standard text boxes and password boxes. Then, the user can enter values for these text boxes. In contrast, since a user can't enter text into a hidden field, you commonly code a Value attribute for a hidden field.

When coding text boxes, you can use the Size attribute to control the width of the text box. To set this attribute, you should specify the approximate number of characters that you want to display. However, since the Size attribute is based upon the average width of the character for the font that's used, it isn't exact. As a result, to be sure that a text box is wide enough to hold all of the characters, you can add a few extra characters. If, for example, you want a text box to be able to display 40 characters, you can set the size attribute to 45.

You can also use the MaxLength attribute to specify the maximum number of characters that can be entered into a text box. This can be helpful if you create a database that can only store a fixed number of characters for certain fields. If, for example, the FirstName field in the database accepts a maximum of 20 characters, you can set the MaxLength attribute of its textbox to 20. That way, the user won't be able to enter more than 20 characters for this field.

The code for three types of text controls

```
Username: <input type="text" name="username" value="jsmith"><br>
Password: <input type="password" name="password" value="opensesame"><br>
          <input type="hidden" name="productCode" value="jr01"><br>
```

The text controls displayed in a browser

Attributes of these text controls

Attribute	Description
`type`	Specifies the type of input control. Acceptable types for text boxes are Text, Password, and Hidden.
`name`	Specifies the name of the control. This is the name that is used to refer to the data in the control from a servlet or JSP.
`value`	Specifies the value of data in the control.
`size`	The width of the text control field in characters based on the average character width of the font.
`maxlength`	The maximum number of characters that can be entered into the text box.

Description

- The Type attribute identifies the type of *text box* to be used. A value of Text creates a *standard text box*. A value of Password creates a *password box* that displays asterisks instead of text. And a value of Hidden creates a *hidden field* that can store text but isn't shown by the browser.
- For a standard text box or a password box, you can use the Value attribute to provide a default value. For a hidden field, you always use the Value attribute to supply a value that can be used by a servlet or JSP.
- Since the Size attribute specifies an approximate number of characters, you may want to make a text box slightly larger than necessary to make sure that all characters will fit within the box.

Figure 3-12 How to code text boxes, password boxes, and hidden fields

How to code buttons

Figure 3-13 shows how to use the Input tag to code three types of *buttons*. A *submit button* executes the Action attribute that's specified in the Form tag. A *reset button* resets all controls on the current form to the default values that are set by their Value attributes. And a *JavaScript button* executes the JavaScript method that's specified by its OnClick attribute.

To create a button, you set the Type attribute of the Input tag to Submit, Reset, or Button. Then, you can code a Value attribute that contains the text that's displayed on the button. For submit and reset buttons, that's all you need to do.

For a JavaScript button, though, you also need to code an OnClick attribute that specifies the JavaScript method that should be called. In chapter 6, you'll learn how to code some simple JavaScript methods for data validation so don't be bothered if you don't know JavaScript right now. In this example, the JavaScript button calls a JavaScript method named validate that accepts the current form as an argument. To specify the current form, this code uses the *this* keyword and the *form* keyword.

When coding a web page, you often to need to have two or more buttons per page that link to different servlets or JSPs. However, each form can only contain one submit button. As a result, you often need to code more than one form per page. In some cases, that means that you code a form that only contains one button. For instance, the second example in this figure actually shows two forms where the first form contains only the Continue Shopping button and the second form contains only the Checkout button.

The code for three types of buttons

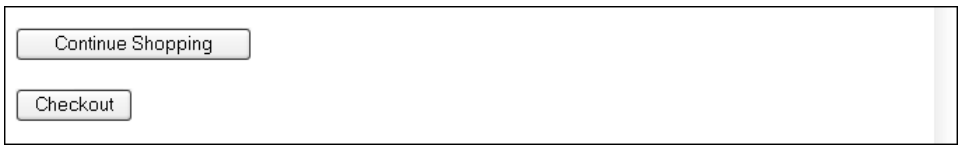
```
<input type="submit" value="Submit">
<input type="reset" value="Reset">
<input type="button" value="Validate" onClick="validate(this.form)">
```

The buttons displayed in a browser

The code for two submit buttons on the same page

```
<form action="/murach/cart/index.jsp" method="post">
  <input type="submit" value="Continue Shopping">
</form>
<form action="/murach/servlet/cart.CustomerServlet" method="post">
  <input type="submit" value="Checkout">
</form>
```

The buttons displayed in a browser



Attributes of these buttons

Attribute	Description
type	Specifies the type of input control. Acceptable types are Submit, Reset, and Button.
onclick	Specifies the JavaScript method that the button will execute when the user clicks the button.

Description

- The Type attribute identifies the type of *button* to be used.
- A Type attribute of Submit creates a *submit button* that activates the Action attribute of the form when it's clicked.
- A Type attribute of Reset creates a *reset button* that resets all controls on the form to their default values when it's clicked.
- A Type attribute of Button creates a *JavaScript button*. When this type of button is clicked, the JavaScript method that's specified by the OnClick attribute of the button is executed.
- To pass the current form to a JavaScript method in the OnClick attribute, you can use the *this* keyword and the *form* keyword. You'll learn more about JavaScript in chapter 6.

Figure 3-13 How to code buttons

How to code check boxes and radio buttons

Figure 3-14 shows how to use the Input tag to code *check boxes* and *radio buttons*. Although check boxes work independently of other check boxes, radio buttons can be set up so the user can select only one radio button from a group of radio buttons. In this figure, for example, you can select only one of the three radio buttons. However, you can select or deselect any combination of check boxes.

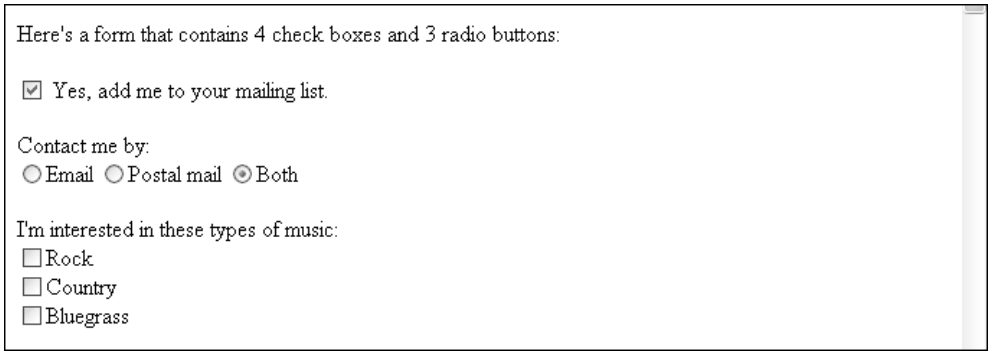
To create a check box, you set the Type attribute of the Input tag to Checkbox. Then, you can set the Name attribute for the check box so you can access the value from your Java code. If you want the check box to be checked by default, you can also code the Checked attribute. Unlike the Type and Name attributes, you don't need to supply a value for the Checked attribute.

To create a radio button, you can set the Type attribute of the Input tag to Radio. Then, you can set the Name attribute just as you do for other controls. However, you can specify the same name for all of the radio buttons in a group. In this figure, for example, all three radio buttons are named contactVia. That way, the user will only be able to select one of these radio buttons at a time. When coding radio buttons, you typically supply a value for each radio button. Later, when you write your Java code, you can access the value that's coded for the selected radio button.

The code for four checkboxes and three radio buttons

```
<input type="checkbox" name="addEmail" checked>
Yes, add me to your mailing list.<br>
<br>
Contact me by:<br>
<input type="radio" name="contactVia" value="Email">Email
<input type="radio" name="contactVia" value="Postal Mail">Postal mail
<input type="radio" name="contactVia" value="Both" checked>Both<br>
<br>
I'm interested in these types of music:<br>
<input type="checkbox" name="rock">Rock<br>
<input type="checkbox" name="country">Country<br>
<input type="checkbox" name="bluegrass">Bluegrass<br>
```

The check boxes and radio buttons when displayed in a browser



Attributes of these buttons

Attribute	Description
type	Specifies the type of control. A value of Checkbox creates a check box while a value of Radio creates a radio button.
checked	Selects the control. When several radio buttons share the same name, only one radio button can be selected at a time.

Description

- You can use *check boxes* to allow the user to supply a true/false value.
- You can use *radio buttons* to allow a user to select one option from a group of options. To create a group of radio buttons, use the same name for all of the radio buttons.
- If you don't group radio buttons, more than one can be on at the same time.

Figure 3-14 How to code check boxes and radio buttons

How to code combo boxes and list boxes

Figure 3-15 shows how to code *combo boxes* and *list boxes*. You can use a combo box to allow the user to select one option from a drop-down list, and you can use a list box to allow the user to select one or more options. In this figure, for example, the combo box lets you select one country, and the list box lets you select more than one country.

To code a combo or list box, you must use the Select tag and two or more Option tags. To start, you code an opening and closing Select tag. Within the opening Select tag, you must code the Name attribute. Within the two Select tags, you can code two or more Option tags. These tags supply the options that are available for the box. Within each Option tag, you must code a Value attribute. After the Option tag, you must supply the text that will be displayed in the list. This text is often similar to the text of the Value attribute.

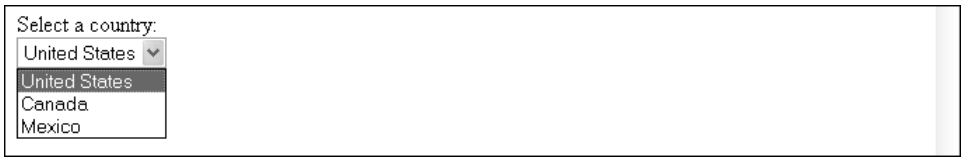
The Multiple attribute of the Select tag allows you to create a list box. If you don't code this attribute, the Select tag will produce a combo box that lets the user select one option. If you do code this attribute, the Select tag will produce a list box that lets the user select multiple options. In fact, the only difference between the combo box and list box in this figure is that the Multiple attribute is supplied for the list box.

If you want to select a default option for a combo or list box, you can code the Selected attribute within the Option tag. Since a combo box only allows one option to be selected, you should only code one Selected attribute per combo box. However, for a list box, you can code the Selected attribute for one or more options.

The code for a combo box

```
Select a country:<br>
<select name="country">
  <option value="USA" selected>United States
  <option value="Canada">Canada
  <option value="Mexico">Mexico
</select>
```

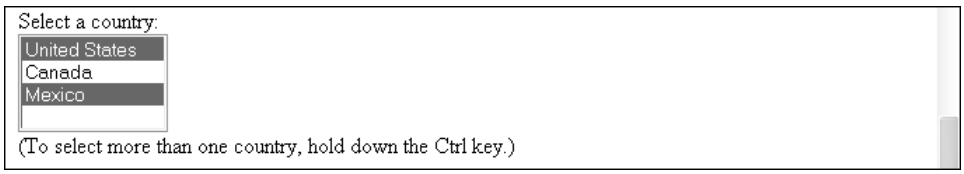
The combo box displayed in a browser



How to convert a combo box to a list box

```
<select name="country" multiple>
```

The combo box displayed as a list box



Attributes of the Select tag

Attribute	Description
`multiple`	Converts a combo box to a list box.

Attributes of the Option tag

Attribute	Description
`selected`	Selects the option.

Description

- A *combo box* provides a drop-down list that lets the user select a single option. A *list box* provides a list of options and lets the user select one or more of these options.
- To select adjacent options from a list box, the user can click the top option, hold down the Shift key, and click the bottom option.
- To select non-adjacent options from a list box, the user can click one option, hold down the Ctrl key, and click other options.

Figure 3-15 How to code combo boxes and list boxes

How to code text areas

Figure 3-16 shows how to code a *text area*. Although a text area is similar to a text box, a text area can display multiple lines of text. By default, a text area wraps each line of text to the next line and provides a vertical scroll bar that you can use to scroll up and down through the text.

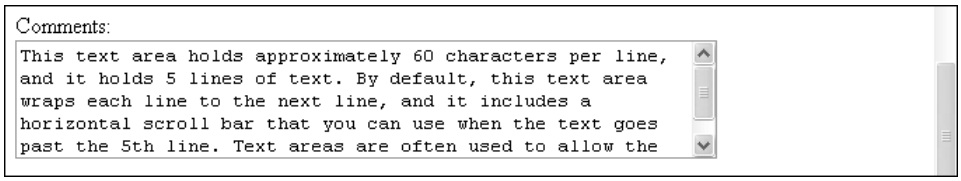
To code a text area, you can begin by coding the opening and closing TextArea tags. Within the opening TextArea tag, you can code a Name attribute so you can access the control through Java. You can also code the Rows attribute to specify the number of visible rows, and you can code the Cols attribute to specify the width of the text area. Here, the Cols attribute works like the Width attribute of a text box.

Within the opening and closing TextArea tags, you can code any default text that you want to appear in the text area. By default, a text area automatically wraps text to the next line.

The code for a text area

```
Comments:<br>
<textarea name="comment" rows="5" cols="60"></textarea>
```

The text area displayed in a browser



Attributes of the TextArea tag

Attribute	Description
rows	Specifies the number of visible lines in the text area. If the number of lines in the text box exceeds this setting, the text area will display a vertical scroll bar.
cols	Specifies the width of the text area based on the average character width of the font that's being used.

Description

- A *text area* is similar to a text box, but it can display multiple lines of text.
- To specify a default value for a text area, you can code the text within the opening and closing TextArea tags.

Figure 3-16 How to code text areas

How to set the tab order of controls

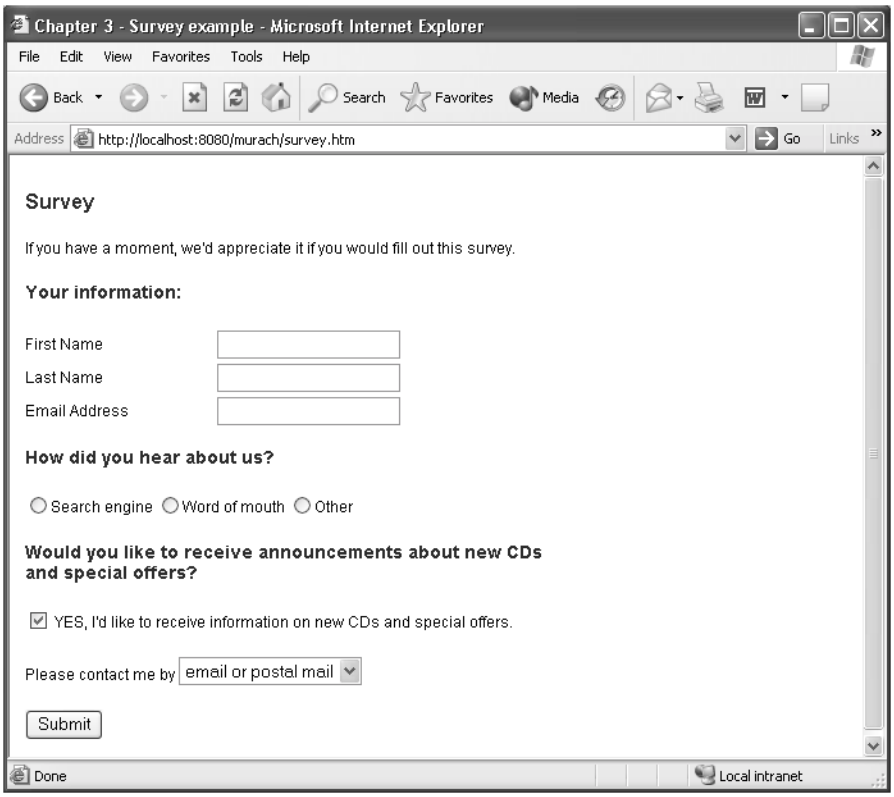
Figure 3-17 shows an HTML page that takes many of the skills described in this chapter and puts them all together. This page uses an external style sheet, tables, and a form that contains text boxes, radio buttons, a check box, a combo box, and a submit button. In addition, this page shows how to control where the focus goes when you press the Tab key. This is referred to as the *tab order* of the controls.

To change the default tab order, which is the sequence in which the controls are coded in the HTML document, you use the TabIndex attribute to assign a number to each control. In this figure, for example, the TabIndex attribute has been set so the focus moves from the text boxes to the first radio button and then to the submit button. In other words, it skips the check box and the combo box. In this case, modifying the default tab order isn't that useful, but it can be when the default tab order doesn't work the way you want it to.

When you code the TabIndex attribute, you should know that it doesn't work on some older versions of Netscape. But that's usually okay because the tab order isn't critical to the operation of the form.

Since the HTML page presented here summarizes many of the skills presented in this chapter, you may want to view the complete code for this page. To do that, you can open the survey.htm file that's stored in Tomcat's webapps\murach directory. Then, you can read through its code and experiment with it. Once you understand how this code works, you're ready to begin coding HTML pages of your own.

An HTML page for a survey



Code that controls the tab order of the survey page

```
<input type="text" name="firstName" size="20" tabindex="1">
<input type="text" name="lastName" size="20" tabindex="2">
<input type="text" name="emailAddress" size="20" tabindex="3">
<input type=radio name="heardFrom" value="Search Engine" tabindex="4">
<input type=submit value="Submit" tabindex="5">
```

Description

- The *tab order* determines the sequence in which the controls on a form will receive the focus when the Tab key is pressed. By default, the tab order is the same as the sequence in which the controls are coded in the HTML document.
- To modify the default tab order, you can use the TabIndex attribute of any visible control. Then, if you omit the TabIndex attribute for some controls, they will be added to the tab order after the controls that have TabIndex attributes.
- The TabIndex attribute only works for the first radio button in a group. And some older versions of Netscape ignore the TabIndex attribute.
- To view the complete code for this page, use a text or HTML editor to open the survey.htm file that's stored in Tomcat's webapps\murach directory.

Figure 3-17 How to set the tab order of controls

Perspective

The goal of this chapter has been to show you how to code HTML documents that get input from the user. Since web designers typically code HTML documents and Java web programmers typically write servlets and JSPs that use the data that's gathered, this may be all the HTML that you need to know as a Java programmer. Remember, though, that there's a lot more to HTML than what this chapter presents.

In the next chapter, you'll learn how to code and test JSPs that process the data that's sent from the forms on HTML documents. Once you master that, you will have a better understanding of how the controls need to be coded so their values can be used by JSPs and servlets.

Summary

- *HyperText Markup Language*, or *HTML*, is the language that a web browser converts into the user interface for a web application.
- An *HTML document* consists of the HTML for one *HTML page*. This document contains the *HTML tags* that define the elements of the page.
- To make HTML documents available to a web application, you must save them in files with htm or html extensions in a directory that's available to the web server.
- To view an HTML page, you enter a URL in the address box of your web browser. These URLs are directly related to the directories on the web server that stores the HTML files.
- To move from one web page to another, you can code *links* within the HTML. These can be *relative links* that are based on the current directory or *absolute links*.
- *Tables* that consist of *rows* and *columns* are used for the design of many web pages. The *cells* within the table can be used to store links, images, controls, and even other tables.
- An HTML page can include images in *Graphic Interchange Format* (*GIF*) or *Joint Photographic Experts Group* (*JPEG*) format.
- To insure consistent formatting, the formatting for the web pages of an application can be stored in an *external style sheet*, or just *style sheet*. Then, each web page can refer to that style sheet.
- An *HTML form* contains one or more controls like text boxes, check boxes, radio buttons, combo boxes, and list boxes. It should also contain a submit button that calls a JSP or servlet when the user clicks it. If necessary, one HTML document can contain two or more forms.
- The *tab order* of the controls on a form determines where the focus goes when the user presses the Tab key.

Terms

Hypertext Markup Language (HTML)
Integrated Development Environment (IDE)
HTML page
HTML document
HTML tag
comment
link
attribute
value
relative link
absolute link
table
row
column
cell
pixel
Graphic Interchange Format (GIF)
Joint Photographic Experts Group (JPEG)
style sheet
external style sheet
linked style sheet
cascading style sheet
form
control
text box
standard text box
password box
hidden field
button
submit button
reset button
JavaScript button
check box
radio button
combo box
list box
text area
tab order

Objectives

1. Code the HTML for an HTML document using any of the tags and attributes presented in this chapter.
2. Save your HTML documents in a directory that is appropriate for your web server.
3. Use your browser to view your own HTML pages or any other HTML pages on a local or Internet server.
4. Describe the use of the Head, Title, and Body tags that are used for HTML documents.
5. Describe the use of the Anchor (A), Table, and Image (IMG) tags.
6. Describe the use of check boxes, radio buttons, combo boxes, list boxes, and text areas.
7. Explain how the directories and HTML files on a web server directory are related to the URLs that refer to these files.
8. Explain how a submit button in an HTML form is used to pass control to a JSP or a servlet.
9. Explain how a standard text box is used to pass data to a JSP or servlet.

Exercise 3-1 Modify the survey page

This exercise steps you the process of modifying the survey page shown in figure 3-17. After each step, you can save and view the page to make sure your modifications are working correctly.

1. Use your editor to open the survey web page that's stored in the webapps\murach directory. Then, edit the survey page so it displays the image for the Murach logo at the top of the page.
2. Add another row to the table that provides for the date of birth.
3. Add an Advertising option button before the Other option button, and turn the Search engine button on as the default for the group.
4. Add a text area for comments before the Submit button.