CHAPTER 18

Adding Hyperlinks

In This Chapter

Understand what links are and how they work

Understand the anatomy of a URL

Link to local pages within the same directory

Link to local pages in different directories

Link to pages on other servers

Link to locations on the same page

Link to pages from images

Link to non-Web data

Add a `mailto` link

Hyperlinks are what make your page really click! Without them you couldn't have a site (only a page), and the Web wouldn't be a Web at all. This chapter tells you everything you need to know about links — internal links, relative links, absolute links, FTP links — you name it.

When you complete this chapter, you will be able to create any kind of link you can find on the Web. You can even use an image on the CD-ROM at the back of this book to link from an image on your page to the IDG Books Worldwide page for this book.

Understanding Links

Links are what make the World Wide Web so interesting and so compelling. In the early days of the Web, a lot of people with pages didn't have much to say other than: "This is what I think is cool." What did they give you? A list of their favorite links. At one time, there were more links than pages.

Today, you have an amazing range of places to link. You can create your own *cool links* page, or you can be more discriminating and only link to other places within your own sites and your own strategic alliances.

How do links work? Basically what a link does is tell the browser to load a new URL using the HTTP protocol (or another protocol, but more on that later). It's that simple.

URLs Dissected

Why is it so important to stop and analyze the parts of a URL? It isn't just to slow you down and keep the good stuff for later in the book. You need to understand the parts of a URL if you want to create links. Depending on the type of link you are using, you need to include different parts of the URL. This may be dry, but hang in there. The rest of the chapter assumes you have read this part.

A *uniform resource locator* (URL) has several parts (see Figure 18-1). There is the protocol, say, `http://`. When you loaded a page locally in Chapter 3, what appeared in place of `http://` was `file://`. That is because the file you were opening was a local file; no protocol was necessary.

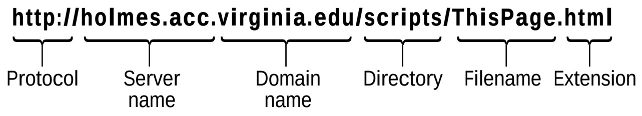

Figure 18-1: Anatomy of a URL

What comes next? The server name. There are two parts to the server name: the machine name and the domain name. If you have a domain with only one server, you've probably never given this any thought. Your server name is something like `www.overtheweb.com`. But if you are part of a large organization, like the University of Virginia, many, many servers exist within the `virginia.edu` domain. Each one has a different name: `holmes.acc.virginia.edu`, `maewest.itc.virginia.edu`, `faraday.clas.virginia.edu`.

Note

In addition to being able to have more than one server associated with one domain, you can have more than one domain reside on one physical server. An ISP may have hundreds of domains hosted from one physical server. How this is configured depends on the Web-server software running.

Following the server name comes the directory location on the server. This directory may be an actual directory, it may be a short name that maps to a longer directory name, or it may be a completely different name. For example, consider the following URL: `http://watt.seas.virginia.edu/~bp/c34.html`. The `~bp` is the directory name. Does this mean there is a single directory directly under the root directory of `watt.seas` with the name `~bp`? No. It means `~bp` maps to some physical directory. This directory could be `h1/users/bp/public_html` or something even longer!

Finally, in the URL, you have the filename `something.html`. In fact, the extension on the file doesn't have to be `.html`. It should be appropriate for whatever type of file it is. If your file is an Active Server Pages (ASP) file, the extension would be `.asp`. If your URL is linking to a specific place on the current page or another page, you also need to include the anchor name of that place. If you don't include a valid anchor name, the page will load at the top. The anchor name begins with a pound sign (#).

Linking Local Pages with Relative File Names

The easiest kind of link to create is a link to a page in the same directory. Say you are currently working on a page you are going to publish as File1.html. You would like to link to another page you are going to publish into the same directory as File2.html. Here is all you need to create the link:

```
<A href="File2.html">Text that links</A>
```

Even though the browser always needs the fully qualified URL, it can figure out the rest of the URL from the current URL. All you need to provide is the file name. The browser assumes any information you leave out matches the information it used to get your current page. So if your visitor comes to your page at `http://this.domain.org/greatstuff/tidbits/File1.html`, his or her browser will turn your link of File2.html into `http://this.domain.org/greatstuff/tidbits/File2.html`.

Qualifying the URL

How does this work? The browser looks to see what page it is already on and then removes only the file name from the existing URL, adds the new file name onto the end of the URL, and requests the page. Pretty smart, eh?

The A element

Without even taking particular note of it, you just learned the `A` element, which you may remember from Chapter 3. When you want to link to a page, or to another location within the same page, all you need is the `A` element with the `href` attribute. In fact, the `A` element has several attributes. The two you use most often are `href` and `name`. The `href` attribute gives the destination of the link; `name` names the anchor you are creating (more on that later).

Anchor `<A>`

Start Tag:	Required
Content:	Inline elements but no nesting of `A` elements
End Tag:	Required
Attributes:	`id`, `class`, `lang`, `dir`, `title`, `style`: previously defined
	`href`: destination for a link
	`hreflang`: language of the `href` attribute
	`type`: content type of the link; most commonly "text/html"
	`rel`: relationship in sequence of this `hrefrev`: relationship in sequence of this `href`

`charset`: character encoding of the link

`shape`, `coords`: for imagemaps, see Chapter 39

`target`: identifies which frame in a frameset gets loaded with the contents of the `hreftabindex`, `accesskey`: for accessibility

`name`: name of an anchor

Linking to Pages in Other Directories

Of course, you can't spend your life linking only to other files within the same directory on the same server. Sometimes you simply must link to pages in other directories. When you do this, you can go one of two directions: up or down. Up means you will be going to a directory further up the directory structure or at least down a different leg of the directory structure than you are in currently. Down means you will go to a subdirectory of the directory you are in currently. The following helps make this clear:

```
Root
history
American              European
revolution   20thcent  WWI  WWII
```

If you are working on a page in the `history/american/revolution` directory and you want to link to a page in the `20thcent` directory, you would take advantage of double dots (`..`) to go up a directory. This means if you are working on a page, the ultimate URL of which will be `http://this.domain.org/history/American/revolution/index.html`, and you want to link to a page in the `20thcent` directory, you would use the following `href` value:

```
<A href="../20thcent/index.html">Learn about the history of the
20th Century</A>
```

Be sure not to put a slash (/) before the double dots or it won't work. How would you point to a file called `index.html` in the `WWI` directory? Don't look ahead.

```
<A href="../../European/WWI/index.html">Learn about World War I
in Europe</A>
```

You can put as many sets of double dots as you need. However, at some point, it might be clearer just to use an absolute reference, which you learn about next.

Linking to External Pages

Relative references were covered in the last section. That is to say, all links were relative to the current page. When you link to pages outside your current server, you need to use absolute references. Absolute references do not depend on the location of the page that is linking to them. An *absolute reference* is simply a fully qualified URL. When you link to an external page, you use the URL as you would type it in the location window of your browser, if you wanted to open that page.

```
<A href= "http://my.domain.org/history/American/revolution/
index.html">Learn about the American Revolution</A>
```

In most browsers today, you don't actually need to type the protocol information (`http://`). If you simply type `my.domain.org/history/American/revolution`, you normally get the right page on your screen. Why? First, the browser assumes you are using the http protocol. Second, `index.html` (along with `default.html` or `home.html`) is one of the file names a server will serve if you don't tell it which page in a directory you want. Servers can be configured to serve any page as the default page; they frequently have a list and go down the list looking for the page defined as the default.

Tip

You can actually give your Web server a bit more information about your links to help them load faster. By putting a / at the end of a link, you tell the Web server the name is a *directory name*, and not a *filename*. This saves the Web server the trouble of looking for a file with that name. This will not work if your link is to a file name (`something.html`). In fact, it will result in a broken link.

In your absolute reference, however, you always need to give the protocol and the file name. So the following won't work:

```
<A href= "my.domain.org/history/American/revolution">Learn
about the American Revolution</A>
```

Because typing a URL incorrectly is so easy and then you end up with a broken link on your page, it is *highly recommended* you do bring up the page to which you want to link in your browser, copy the URL from the location window, and paste it into your HTML.

Linking to Locations on the Same Page

What if you have a long page, such as a FAQ, which by all the chunking rules you know you shouldn't break up into more than one page, perhaps because people will want to print it? How can you link to the middle of a page? You create a named anchor. A *named anchor* is simply a point on the page to which you can link directly.

```
<A name="Q21">Question 21</A>
```

The previous code creates a named anchor at Question 21. There are no rules about names, except they can't have spaces in them, so make up something you'll remember. Now if you want to link from somewhere else in the same page to Question 21, you use the following code:

```
<A href="#Q21">See also Question 21.</A>
```

If you want to link from somewhere else in another page in the same directory, you use the following code:

```
<A href="faq.html#Q21">See also Question 21 in the FAQ.</A>
```

You can, of course, link to a named anchor from pages in other directories or from pages on other servers. You should be able to figure out how to do this by now.

Cross-Reference

Chapter 22 explains frames and targets in detail.

Tip

Does capitalization matter when you are creating links? It depends. If you are publishing to an NT server, no. If you are publishing to a UNIX server, yes. It definitely matters when you are linking to pages on other sites, because you don't know on what kind of server they reside. The easiest way to keep your own links straight on your own site is to pick a system and stick with it. Use all lowercase filenames and you shouldn't have a problem.

Link to Pages from Images

One feature you see on many pages is linking from images. Sometimes you can click an image and hyperlink to another page or to another place on the same page. Linking from an image is simple. An image is on the CD-ROM in the back of this book that you can use for this. This image is called `HTML4.gif`. To place this image on your page and link from that image to the IDG Books Worldwide page for this book, include the following code:

```
<A href="http://164.109.153.102/product.asp?isbn=0764534734"><IMG
src="images/HTML4.gif"></A>
```

You just used the `IMG` element. The `IMG` element is covered in depth in Chapter 19. For now, just know you need to publish the image file from the CD to your images directory as binary (or raw data) for this to work.

How does this work? The image behaves exactly the same as text would as the content of an `A` element. On older browsers, you might see a blue box around the image to indicate it links. But you'll learn how to turn that off when you get to Chapter 26 about style sheets.

Linking to Non-Web Data

What if you want to link to a file for your visitors to download? You can do this using the same `A` element.

```
<A href="http://www.yourserver.com/pub/contract.doc"> Download
the latest version of our contract.</A>
```

This works as long as it is in the pub directory and it is a file called `contract.doc`. When a visitor clicks this link, the browser asks what to do with this file. Once the browser gets it downloaded, the visitor can open it with any program that will open a `.doc` file. It's a good idea to give visitors a choice of ways to get at any file you want to download and also to give them a choice of formats including `.rtf` (rich text format). Check with your systems administrator to make sure permissions are right for this to work.

Tip Better yet, create the document and save it as a `.pdf` file (portable document format) using Adobe Acrobat. Then visitors can see the document with all its formatting right on the screen and print it to a printer without needing to have the right word processor installed.

The BASE Element

What if you want to have many links to a different directory on the same server or you have to move your page to another directory, breaking all your links? HTML helps you out here by offering the `BASE` element. The `BASE` element goes into the `HEAD` element. It is easy to use. It tells any visiting browsers they should act as if this page is located in the `BASE` URL location for purposes of all relative references. An example helps:

You have 20 links on your page to other pages in your site. Your current page is called `history/American/20thCentury/WWII/Pacific/thispage.html`. Your links are mostly to pages in the `history/European/WWII/Japan` directory. So most of your links look something like this:

```
<A href="../../../../European/WWII/Japan/page1.html"> Click
here.</A>
```

Now, for some reason, you have to move this page to the history/Asian directory. Does this mean you must go through and recode all your links? No. You can simply add the following BASE element to your page:

```
<BASE
url="http://this.domain.org/history/American/20thCentury/WWII/
Pacific/">
```

Now all your relative links originate from the correct directory. This solves this immediate problem, but what you might want to do in the first place is to make all your links relative to the history directory. Then your BASE element would look like this:

```
<BASE url="http://this.domain.org/history/">
```

And your link would look like this:

```
<A href="European/WWII/Japan/page1.html"> Click here.</A>
```

Using the BASE element can make maintenance of your site much easier in the long run.

Adding a mailto Link

Have you ever clicked someone's e-mail address at the bottom of a page and had a mail window open? This is done with a mailto link. If you do use a mailto link, you should also give your e-mail address so people with nasty old browsers, or the stand-alone version of Navigator, can still send you e-mail. Here's how it works:

```
Contact me at<A href="mailto:info@yourdomain.org">
info@yourdomain.org.</A>
```

Where info@yourdomain.org is your e-mail address.

Bonus: Create a Link Without Leaving Your Page

Isn't it risky to put links on your pages to other sites? Doesn't this mean you could be sending your visitors away, never to return? It is, indeed, risky, but you don't want to have a site without any links to the rest of the world. There is a way around this quandary. When you link to other locations, open up new windows! This means your own site is still there — it just ceases to be the active window.

```
<A href="http://other.site.com/relevant/index.html"
target="_blank">A strategic partner of ours</A>
```

The *target* attribute is set to _blank, which is a reserved target name. This attribute tells the browser to open a new, blank window the URL specified by the `href` attribute. You learn more about targets and reserved target names in Chapter 22, which is all about frames.

From Here

Cross-Reference

Jump to Chapter 22 to learn how to use frames in your Web page.

Proceed to Chapter 19 and start inserting graphics and other objects in your Web pages.

Summary

In this chapter, you learned how to create all kinds of links: links to pages in the same directory on the same server, links to pages in different directories on the same servers, and links to pages on different servers. You learned how to link images to pages. You also learned how to create a `mailto` link and how to open a new page when you link to an external page, so the visitors don't leave your page altogether.

✦ ✦ ✦