Chapter 2

Client/Server Applications — Getting Started

Intended Learning Outcomes

After completing this chapter, you should be able to:

- Explain how Web programs function as client/server applications.
- Identify the content of a file by referring to the file extension.
- Locate files using the Windows addressing scheme.
- Locate files using the Internet (IP) addressing scheme.
- Use the localhost domain to access a standalone Web server.
- Identify the languages we will use in this course.
- Identify the software you will need to complete the hands-on work.
- Install the required software.
- Create, save and open an HTML document that is stored on a local Web server.
- Create, save and run a PHP file that is stored on a local Web server.
- Create, save and run an interactive Web application consisting of an HTML document that includes a form, and a PHP application that processes the form.

Introduction

This chapter will prepare you for the hands-on activities that you will perform as you work through the subsequent chapters of this book. First we will review **client/server** design with a focus on Web-based applications. Next we will look at how files and folders are organized and located, using **local** addresses based on **disk drives**, and **Internet** addresses based on **domain names**. This is important since you will need to be careful

to save files to the correct locations on your disk drive, and then open these files in your Web browser using the correct Internet address.

You will also install the server software that you need to work through the textbook and complete the exercises. This is a very straightforward procedure and you will be able to test that your server is running correctly by running some sample programs.

Once the server has been installed you will have the option to install a text editor, then you will be asked to type in a few small programs, save them, and run them using your Web browser. The code for these programs is provided. The idea is not to learn to develop programs (that comes later), but to simply learn the general process of using an editor to create code, saving your files to the correct location under your Web server, running the server, and then testing that your programs work correctly.

Client/Server Design in Web Applications

In a client/server design, client programs send requests to server programs to perform a task of some kind, just as you might ask someone else to do something for you. The server receives the request from the client and responds appropriately. The server program resides on a networked computer and can respond to hundreds, thousands, or millions of client requests; consider an online shopping site that receives requests from customers all over the world every minute. An important advantage of client/server design is that software updates are applied on the servers without requiring any changes on the client computers.

Client/server applications are delivered world-wide across the **Internet**, and are also used to provide services to users of private networks (known as **intranets**) within an organization or company.

A Web application is a familiar example of an Internet-based client/server design. In the case of Web applications, the client program is a Web browser. The Web browser runs on a user's personal or office computer. Each time the user enters a URL, or clicks on a link on a Web page, or clicks a Submit button after entering data into a Web form, the browser sends the user's request. The request is transmitted across the Internet to the appropriate Web server. The Web server receives and processes the request, then sends back a response for the browser to display to the user. Web servers may be located anywhere on the Internet and can accept requests from any client that has Internet access (Figure 2-1). In order to process each request, the server program may communicate with other programs or access databases or files. This is a very efficient design since it allows the user to obtain all kinds of useful services without installing special software on their local computer. Instead the user uses their Web browser to execute programs that are located on remote servers.

In order to develop Web-based applications we must create files that contain the necessary instructions for our Web pages and programs, and store these files on a Web server. The server can then process the files as needed in response to requests.

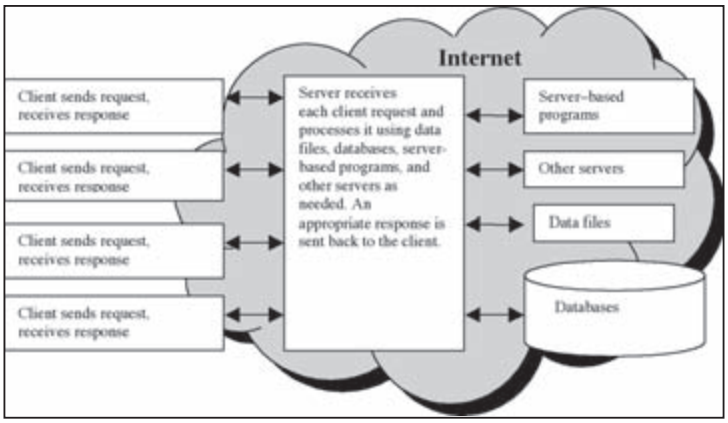

Figure 2-1: Example of client/server design

In this course you will run a Web server on your **local** computer so that you can develop applications without having to connect to a server located on the Internet. In other words, your Web server will actually be located on the same computer as your Web browser, but it will behave exactly like a Web server running on the Internet. This approach allows you to keep all your work on your local computer or USB drive, and saves you the expense of setting up an account with an Internet Service Provider (ISP). Note that, if you prefer, you can also do all the work in this book using an Internet-based Web server; in that case you will need to transfer your files to and from the server in order to edit them.

Even though your Web server will run on your local computer, it will still function as though it were running on the Internet. You will need to be careful to save your files to the correct folder locations so that your Web server can find them, and you will also need to provide the correct Web addresses (URLs) for these files when you wish to view them in your Web browser. Web addresses are quite different from the addresses used in Windows so let's review how files and folders are organized on disks, and then learn how files can be located using the Windows and Internet addressing schemes.

Working with Files and Folders

Files are used to store data, all kinds of data. A file may contain text, images, videos, word-processing documents, programs, etc., but each file may only contain one type of data. The **file extension** usually indicates the format of the data that is stored in the file. This is very useful since the format indicates what type of program is needed to process the file. For example, a file with a **.jpg** extension contains image data stored using the **jpeg** image format, and can be opened by any image-viewing or image-processing program that can read this format. A file with an **.mp3** extension contains an MP3 audio file that can be handled by any MP3 player. A file with a **.zip** extension contains data that has been compressed using the ZIP compression scheme. To create zip files, or extract data from these files, you will need zip utility software.

Text files contain plain text (characters that can be typed on your keyboard). Text files can be viewed and edited using any text editing software. Files that contain plain text are often saved with a .txt extension. Often however, text files are given special extensions to indicate the specific purpose of the text that is stored in the file. In this course you will use a number of different extensions for your text files, as follows:

- **Plain text** files, using a **.txt** or **.dat** extension, will be used to store simple data for use by your programs. For example you might create a file named **scores.txt** that contains a list of student scores.
- **HTML** files, using an **.html** extension, will be used to store the markup instructions for Web pages. Our Web pages will include forms to allow the user to enter information that will be submitted for processing by our programs. For example you might create a file named wages.html that contains a Web page with a form for the user to submit their hours worked and hourly wage.
- **CSS** files, using a **.css** extension, will be used to store style sheet specifications that define how our HTML code is to be rendered on the Web page.
- **PHP** files, using a **.php** extension, will be used to store the source code for PHP programs. Our PHP programs will usually receive information from the user or look up data in files in order to perform useful processing operations and generate output. For example you might create a file named **wages.php** that contains the source code to receive wage information from a Web page, then calculate and display the pay.

Since these are all text files we can **create** and **modify** the content using any text editor. However since these files contain text to serve different purposes, the file extension indicates what software is required to **process** the content of each file. A file with an **.html** extension is usually opened by a **Web browser**, since Web browsers are designed to interpret .html files and display the contents as Web pages. A file with a **.php** extension can only be executed as a set of PHP instructions if it is opened by a **PHP code processor**. A PHP code processor is included with the Web server that you will install as you work through this chapter.

Files are usually organized into **folders** for ease of management, and files and folders are stored on **portable** or **fixed** disks that are accessed through **disk drives** connected to computers. You can access files on disks located in **local** drives that are attached to your personal computer. You can also access files on disks in **remote** disk drives, attached to computers that are connected to your computer through a network such as the Internet. No matter where the drive is located, in order to locate a specific file on a disk, you need to be able to refer to it using some kind of **file addressing scheme**.

Locating Files and Folders on Computers Running a Windows Operating System

Most often when we want to locate a file or folder using a Windows operating system, we open **File Explorer** and point and click our way to the file that we wish to work

with. As a programmer, it is important to know that we can also reference a file by providing its unique **file path** or **address**. Every file and folder on a computer has a unique address that is based on its folder location and disk drive specification. Take a look at Figure 2-2, which displays a screen containing a list of four **folders**.

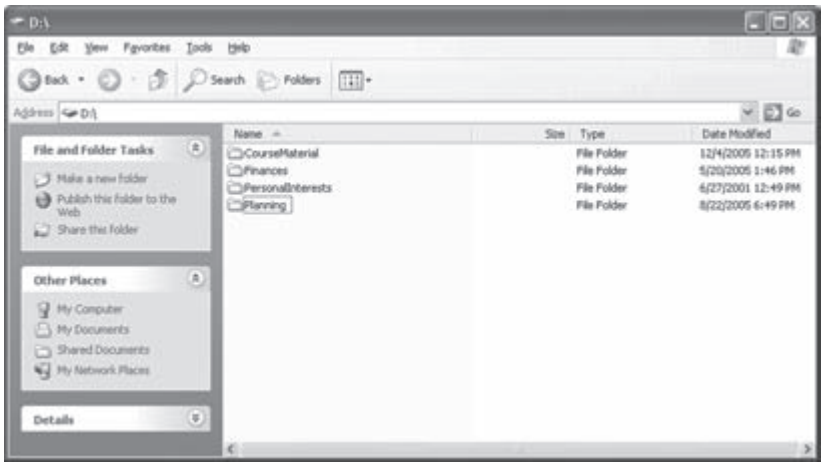

Figure 2-2: Examples of folders on a disk

Note that the address box shows the root address of these folders. They are located on the **D:** drive of the local computer. The address of the folder named **CourseMaterial** is therefore **D:\CourseMaterial**.

Folders can contain any combination of other folders and data files. Let's look inside **CourseMaterial** by double-clicking this folder (Figure 2-3).

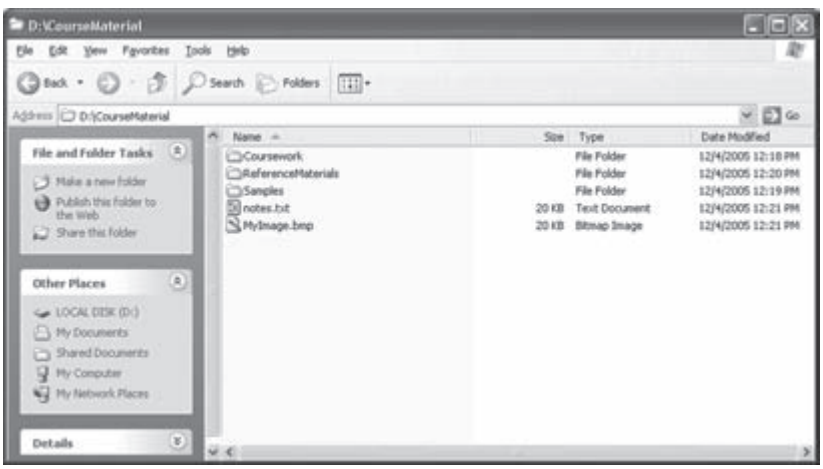

Figure 2-3: Inside the CourseMaterial folder

Notice the address in the address box is now **D:\CourseMaterial**, indicating that we have shifted our location to the CourseMaterial folder. This folder contains three folders (**CourseWork**, **ReferenceMaterials** and **Samples**) and two data files (**notes.txt** and

myImage.bmp). The file extensions indicate the type of data stored in these two files: **notes.txt** contains plain text, while **myImage.bmp** contains a bitmap image. The complete address of the file named **notes.txt** is **D:\CourseMaterial\notes.txt** and the address of the folder named Coursework is **D:\CourseMaterial\Coursework**.

Now let's open the Coursework folder (Figure 2-4).

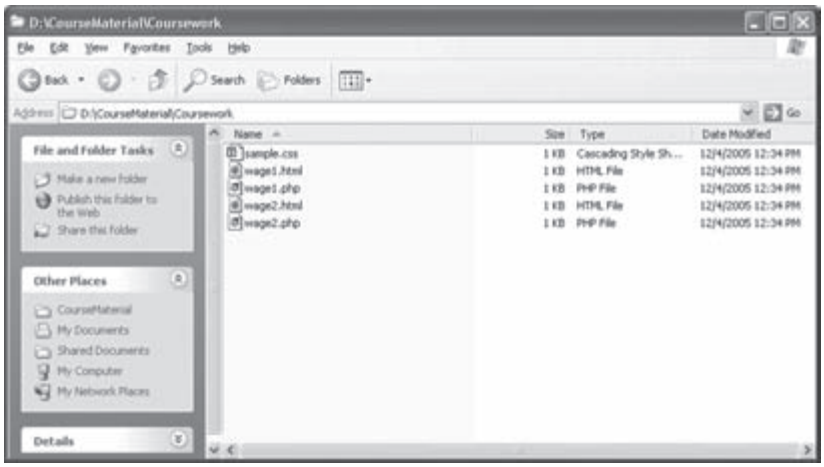

Figure 2-4: Inside the Coursework folder

Note that the address window now displays **D:\CourseMaterial\Coursework**. The Coursework folder contains five files named **sample.css**, **wage1.html**, **wage1.php**, **wage2.html**, and **wage2.php**. You can access any file directly using the file address. For example, instead of navigating to the Coursework folder by clicking through folders as we have done, we can open a file by simply typing the complete file address in the address window. For example, to access the **wage1.html** file we could just type: **D:\CourseMaterial\Coursework\wage1.html**.

Locating Files and Folders on the Internet

The Windows operating system assigns **drive letters** to identify each disk drive attached to your computer. The drive letter forms part of the address of any file stored on one of these disks. However this addressing scheme will not work when we need to locate files on the Internet, which consists of hundreds of thousands of disk drives attached to computers all over the world! An address that begins with **D:** would refer to a different drive on every computer in the network!

Instead all Internet addresses are based on the IP (Internet Protocol) addressing scheme. Each IP address references a specific **folder location** on a specific **disk** attached to a specific **computer** that is performing as a Web server somewhere in the world. Every IP address is unique so it is impossible for a single IP address to refer to two different locations. An example of an IP address is **128.30.52.100** which at the time of writing points to the World Wide Web consortium's Web site.

Imagine typing an IP address every time you needed to connect to a Web server! And how will anyone find your site if you move it to a different IP address? For ease of use and portability, we use **Internet domain names** to represent IP addresses. For example the domain name of the **128.30.52.100** address is **w3.org**, so this domain name can be used in place of the IP address. Try typing **http://www.w3.org**.

The Internet address of a specific file or folder usually combines a domain name with a file and folder path. For example, **http://www.php.net/license/index.php** is the Web address of a file named **index.php** which is located in the **license** folder of the **php.net** domain. The exact folder and disk location of the license folder is determined by the domain name **www.php.net**, which maps to an IP address that refers to a location on a Web server, somewhere in the world. If these files and folders are relocated to a different Web server, the domain name is simply reassigned to point to the IP address of the new server, so the Web addresses do not need to be changed, as long as the files and folder structure remains the same in the new location.

A Web address such as **http://www.php.net/license/index.php** is known as a **URL** (**Uniform Resource Locator**). Note that the separator used in URLs is the forward slash /, whereas the separator in our Windows addresses used the back slash \. That's because the naming convention for Windows file paths derives from the **DOS** operating system which uses the back slash, whereas the naming convention for Internet addresses derives from the **Unix** operating system which uses the forward slash. Fortunately Windows now also allows you to use the Unix forward slash when typing Windows file paths.

Often we write URLs without specifying a file name at the end. For example, if we were to type **http://www.php.net/license/** into our Web browser's address box, we would actually receive a file even though we did not include a file name. That's because Web servers are configured to add default file names to URLs if none is provided. For example if a Web server is configured to open a file named index.php by default, then the URL **http://www.php.net/license/** would actually access a file in the **license** folder named **index.php** using the address **http://www.php.net/license/index.php**.

Internet Naming Conventions for Files and Folders

As a Web applications developer you will need to create your own files and folders that will be referenced in URLs. Here are four widely accepted conventions for naming your files and folders for Web applications. These are followed by most Web programmers, and actually they work well as naming conventions for ALL your files and folders!

Use descriptive names that describe the content of a file or folder: For example, let's say you have a page that displays a tax report for the tax year 2016. In that case **tr16.html** is not a meaningful file name, but **tax-report-2016.html** is very descriptive. At the same time be careful not to use **too** simple a name such as just **tax-report.html**; that name may look good right now, but then what will you call the file for next year's tax report? Best to come up with a consistent approach to your file names that will

work over time; remember that once your files are online it's not easy to change their names because other Web sites may have already linked to them using the existing file name. We've all followed links on a Web page only to discover the file no longer exists, which often indicates that the name has been changed.

Avoid very long file names: this may seem to contradict the first rule, but avoid making your file and folder names unnecessarily long. In other words, choose a name like **tax-report-2016.html** rather than **final-version-of-tax-report-for-year-2016.html**. Apart from less typing, this also reduces the risk that your URL might be shortened if it is referenced from another site such as a blog. A blog might automatically reduce the address to something like **http://mywebsite/tax-reports/final-ver...html** where the file name has been shortened. Although the link itself will still work fine, if someone wants to copy the URL and paste it, they won't get the entire URL. Shorter names avoid this problem. So it's a compromise: make your file names descriptive but still keep them as short as possible. Avoid unnecessary words and also avoid connecting words like "the" and "of" in your names.

Use hyphens and never spaces in your file and folder names: We've become used to using spaces in our file and folder names because Window, Mac OS X, and even Linux allow spaces. Even so, spaces are not a good idea for files that will be used on the Web. Web browsers can have trouble with spaces and so can search engines. For this reason, use **hyphens** to separate words in your file and folder names instead of spaces. That's why our tax example uses **tax-report-2016.html** and not **tax report 2016.html**. Similarly use hyphens instead of underscores. For example avoid **tax_report_2016.html**. That's because the underscore is harder to read especially in a link that is already entirely underlined.

Use lower-case and not upper-case letters: Many Web programmers use a mix of upper- and lower-case letters, in fact I did in the first three editions of this book! It's a bad idea because file names are case sensitive on many systems, such as Linux, so **tax-report-2016.html** will be treated as a different file than **Tax-Report-2016.html**. If you and your users know that all files are lower-case only, it greatly reduces the risk of broken links or mis-typed URLs. I can tell you from painful experience that changing all of your file names and links to lower-case is a very time consuming and error-prone activity! Best to get it right from the start!

For more information about file and folder navigation and addresses, refer to Appendix B.

Working with a Local Web Server

You are going to learn the fundamentals of program logic and design by developing Web applications using our own Web server. In order to work simply and securely you will use special software that allows you to run a Web server on your local computer with no need for Internet access. Although the Web server will be installed locally, in all other respects it will perform exactly as an Internet-based server. The Web server

will receive requests from our Web browser, process these requests and respond appropriately (we hope!). Your programs and files will be stored and accessed locally, however, if you were to copy these to a Web server located on the Internet, your applications would perform in exactly the same way across the Web. The required software is easily installed and easy to use.

The IP address of your standalone Web server will be **127.0.0.1** and the domain name will be **localhost**. This is a special non-unique IP address and domain name that allows you to reference your own computer instead of connecting to the **Internet**. So for example, in order to run a program named **my-web1.php** which is in a folder named **samples**, in a folder named **webtech** on your local Web server, you would type the URL:

http://localhost/webtech/samples/my-web1.php

Actually, if you type that URL into your Web browser's address box right now you will get a message that the page cannot be displayed! That's because you are trying to connect to a Web server that is not actually running, which means that the localhost domain is not available. Once you install and run the Web server (later in this chapter) this URL will work.

What Languages Will I Use?

You will develop your Web programs using the following languages:

HTML (Hypertext Markup Language) provides the **markup instructions** that you will use to create and format the Web pages that display the user interface for your Web applications. You will use HTML to display headings, paragraphs, forms, tables, buttons, and images. Since this is a course in logic and design you will not learn everything there is to know about the HTML language. Nevertheless you will learn sufficient HTML to easily extend your skills in subsequent courses or personal research. Everything you learn will be based on current HTML standards.

CSS (Cascading Style Sheets) allows you to assign specific formatting to your HTML markup that will control the way this code will display in your Web browser. This book will teach you just enough CSS to understand the basic concept of a style sheet and apply some minimal formatting to your HTML pages.

PHP (PHP Hypertext Preprocessor) is a **programming language** that you will use to write server-based programs that process user requests by performing calculations, validating input, making decisions, reading data from files or databases, writing output to files or databases, returning results to the user, etc. This course covers sufficient PHP to teach basic programming logic and design as well as many important aspects of software development. This will prepare you for subsequent programming courses in PHP or other current programming languages.

MySQL is a database query language that will be introduced later in the book. MySQL allows you to work with databases, an important component of most online applica-

tions. **MariaDB** is a recent community-developed branch of the MySQL database; MariaDB is included with your Web server.

HTML, CSS, PHP, and MariaDB are all free technologies.

Figure 2-5 shows the same example of a simple client/server application that you reviewed in Chapter 1.

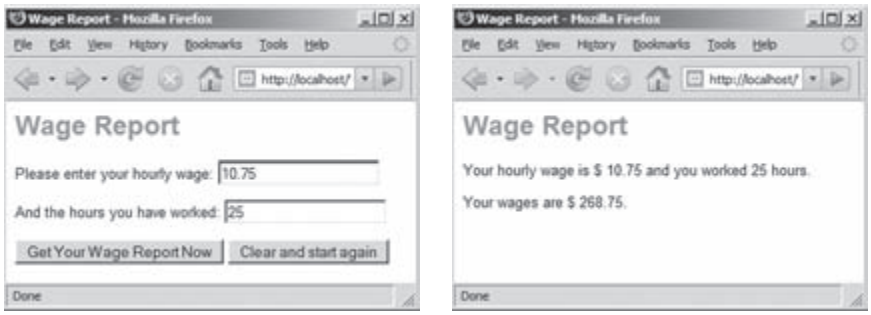

Figure 2-5: Client/Server Example using HTML and PHP

Let's examine this application more carefully. These are two Web pages. The first Web page is generated from HTML code located in an **.html** file on a Web server. The HTML code in this file has been designed to display a Web page that contains a heading, a form with two prompts and two input boxes, and two buttons.

When the user enters the information and presses the "Get Your Wage Report Now" button, the browser sends the user input (10.75 and 25 in this case) back to the server with a request to process a file containing PHP code. The code in the PHP file provides instructions to: (1) receive the data submitted from the first Web page; (2) calculate a wage based on this input; (3) generate a new HTML page to display the results. The second Web page is created by this PHP code and the server returns this to the Web browser for display to the user.

The appearance of the text in both pages is defined by the instructions provided by a CSS stylesheet. We will explore this example in much more detail in the chapters that follow.

What Software Will I Need?

To complete your hands-on activities, you will need the following software:

- A **text editor** to create HTML, CSS, and PHP files.
- A **Web browser** to submit requests to the Web server and display the Web pages that are returned for display.
- A **Web server** that can process requests sent to the **localhost** domain.

NOTE: If you were working with a "live" Web server that was actually located on the Internet, you would also need an FTP client application to transfer files between your local computer and your Web server (FTP stands for File Transfer Protocol). Since we are using

a standalone Web server that is located on our local computer we won't need an FTP application. However, the textbook Web site provides a handout that explains how to set up an FTP server with your standalone Web server, and how to use an FTP client application to connect to an FTP server.

Read the following sections carefully and follow the instructions to install the necessary software for your computer. Once you have the software installed you will be ready to create and test some sample applications.

Installing a Text Editor

You will need to install a text editor to create and modify your HTML and PHP. You can use any text editor and there are many free editors available for Windows, Macintosh, and Linux. Some text editors contain special features that you will find useful as a programmer, for example code indentation, line numbering, and search and replace functions. Additionally a text editor that recognizes HTML and PHP will automatically apply different colors to special words, tags, and other syntactical elements of your code, which makes it easier to identify typing errors.

If you are using Windows, consider **Notepad++** (**http://notepad-plus-plus.org/**) which is designed for programmers and provides lots of useful features. Note: do not confuse **Notepad++** with **Notepad**, a very simple and not very useful editor that comes with the Windows operating system. You will want to use Notepad++.

Macintosh users might consider **Textwrangler**, a highly regarded text editor that can be found at: **http://www.textwrangler.com**.

Linux users might like **Kate** or **Bluefish**.

You might also wish to try either of two recently introduced editors: **Atom** (**http://atom.io**) and **Brackets** (**http://brackets.io**). Either can be installed on Windows, Macintosh or Linux.

In all cases installation is simple and for most users the default installation settings will be fine. Take some time to get used to the basic operations to use your editor – you will have a chance to do this later in the chapter when you create some simple applications. Don't try and learn everything — you can explore the full functionality of your editor as you gain experience.

IMPORTANT NOTE: do not use a word-processor (such as MS Word) to create and edit your code. You need to save your code as plain text files.

Installing One or More Web Browsers

You need a Web browser to view the examples and your own programs. Any major browser should be fine and you will already have at least one browser already installed on your computer. Professional developers like to use two or three browsers so that

they can test their Web sites more thoroughly. You are encouraged to work with multiple browsers for this reason but this is **not** required for the material in this textbook.

If you want to install additional browsers, **Mozilla Firefox** is an excellent and freely available browser that is available for Windows, Mac or Linux (**http://www.mozilla.com/firefox**). **Apple Safari** is another great browser available for Macintosh and Windows (**http://www.apple.com/safari**). And you might also consider **Google Chrome** (**http://www.google.com/chrome**) for Windows, Mac or Linux. In all cases, installation is simple and for most users the default installation settings will be fine.

Installing Your Web Server

You must also install a Web server that will process your PHP code and deliver Web pages to your browser. We will use a free standalone distribution of open source software that includes the Apache Web server. This distribution, named **xampp**, has been been compiled by the Apache Friends project. Versions of xampp are available for Windows, Macintosh, and Linux. The textbook Web site contains complete instructions to download and install this software, along with a custom **webtech** folder that contains the sample and coursework files you will use with this textbook. Instructions and download files can be found at:

http://www.mikeokane.com/textbooks/WebTech/support.php

The installation documents also explain how to **run** and **stop** your Web server, and how to test your Web server to be sure that it is working correctly. The material that follows assumes that you have successfully installed and tested your Web server, that you have at least one Web browser, and that you have a text editor to create and edit your program code.

Using Your Web Server

Now let's learn how to use the server to run and test our Web applications! First start the Web server if you have not already done so. Now open any Web browser and type the following URL in the address window:

http://localhost/webtech

Your installation is configured so that this URL will open a Web page (Figure 2-6) that will help you to use this textbook. If this page displays then your Web server is running successfully. As you will see, the page provides links that make it easy for you to view and work with the samples and coursework files that are discussed in each chapter. Note that your Web server must be running in order for you to view this page (or to view any page with a URL that begins **http://localhost**). If you were to stop your Web server, this URL would no longer work and your browser would report a connection error.

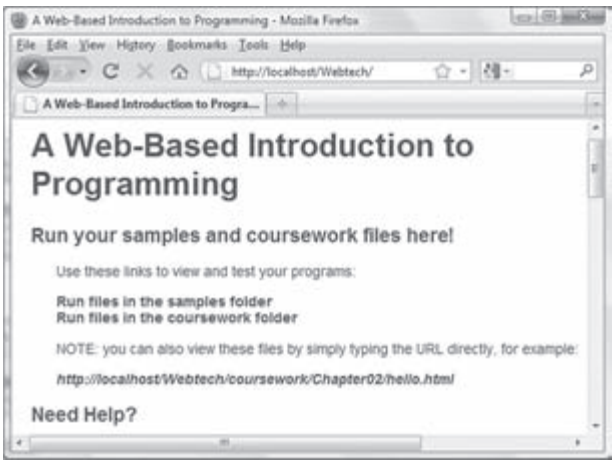

Figure 2-6: http://localhost/webtech

Let's run a simple program from the samples folder. Click the "Run files in the samples folder" link, and then click welcome.html from the list of files that appear. This should bring up a welcome page with an interactive form for you to complete (Figure 2-7).

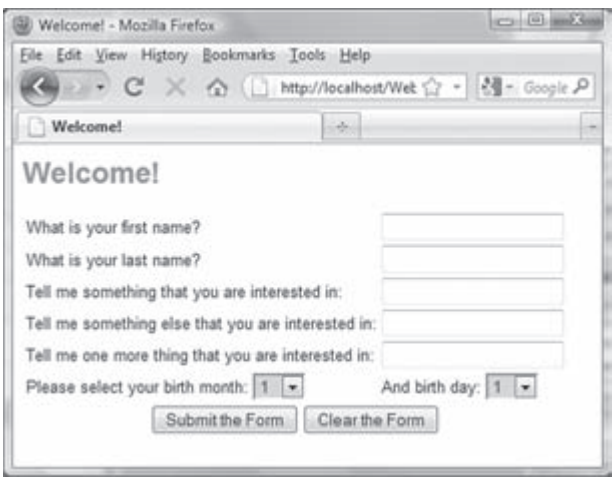

Figure 2-7: http://localhost/webtech/samples/welcome.html

Try completing the form and the press the "Submit the Form" button. If a second welcome screen appears with a response to your submission, then everything is installed correctly and working fine.

Using URLs with Your Web Server

Every file on the Web has a unique URL, or Web address. The **URL** consists of a domain name, followed by the folders and a file name that indicate the location of the

particular file on the server. Our local Web server has the domain name **localhost** and this domain name points to the **htdocs** folder which is in the folder that contains your installation).

Earlier you opened the file **welcome.html** from your **samples** folder. Open the file again and this time notice the URL in your browser's address window:

http://localhost/webtech/samples/welcome.html

All of the URLs of files located on your Web server will begin **http://localhost** because **localhost** is the domain name of your local server. But how does the Web server know where to find the **welcome.html** file in order to send the contents of the file to your Web browser?

The answer is that, by default, the Web server looks in the **htdocs** folder of your Web server installation in response to any URL that begins **http://localhost**. In other words the URL **http://localhost** is associated with the **htdocs** file folder on your drive, and in case you're wondering, the name **htdocs** is a shortened version of "hypertext documents". Use **File Explorer** (in Windows) or **Finder** (on a Macintosh) to locate the **htdocs** folder in your Web server installation.

So if the URL is **http://localhost/webtech/samples/welcome.html** then the Web server will process the file named welcome.html that is located in the folder **htdocs/webtech/samples**. Use **Windows Explorer** or **Finder** to find this file on your disk — can you find it?

Be sure that you understand this. Each URL that begins **http://localhost/webtech** refers to a file on your Web server that is located under the **htdocs/webtech** folder and if you were to change the contents of any file inside this folder and then save your changes, the new version of the file would be displayed if you were to type the URL of the file in your Web browser.

As another example let's look at a file in your **coursework** folder. Type the URL **http://localhost/webtech** and choose the **coursework** folder. Now click **chapter02** and choose **hello.html**. This page displays a short introductory message about the kind of work you will do for each chapter (Figure 2-8).

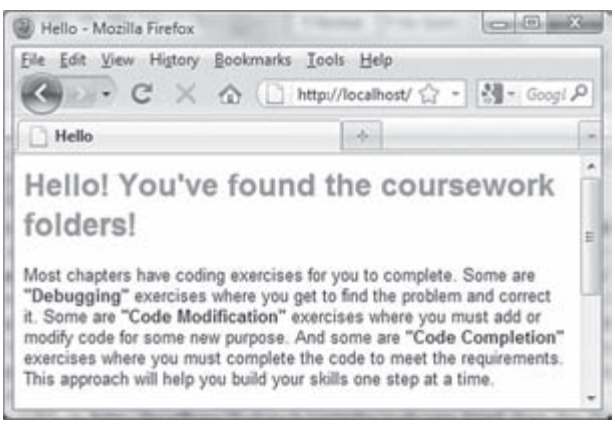

Figure 2-8: http://localhost/webtech/coursework/chapter02/hello.html

Note that the URL for this file is:

http://localhost/webtech/coursework/chapter02/hello.html

Can you use **File Explorer** (in Windows) or **Finder** (on a Mac) to locate this file on your disk?

We will keep all our work files in **two** folders under the **htdocs\webtech** folder. The **samples** folder will contain all of the sample files referenced in the textbook. Try opening some of these programs in your browser now. You will notice that many files are listed in pairs with the same name but two different extensions (**.html** and **.php**). In these cases, click the **.html** files rather than the **.php** files to see what they do (the **.html** files display Web pages with forms that are used to "drive" the PHP programs). The **coursework** folder contains sub-folders for each chapter, and each chapter folder contains the files for your code exercises. If you were to try opening these files you will find that many of them do not work correctly or generate errors — that's because they contain code that you will complete yourself as you work through the chapters.

To summarize, always remember to first **start** your Web server **before** you attempt to run your programs, and always **stop** the Web server and then **exit the Control Panel** once you have completed your work. The URL to your programs will always begin with **http://localhost/** and this should be followed by the names of any subfolders, followed by the name of the file that you wish to open. To avoid typing the complete URLs, you can just type **http://localhost/webtech** (Figure 2-6) and then click through the links to open the file you want to view.

Always Use URLs to Run Your Web Applications!

As you probably know, in **Windows** you can often use **File Explorer** to not only find a file but also to open the file (on a Macintosh you can use **Finder**). That's because your operating system associates the file **extension** of a file with a default application that can handle that file type. For example, files with **.doc** and **.docx** extensions are usually associated with the **MS Word** application, which is why MS Word runs and opens the file when you click a file with one of these extensions. Your computer usually associates **.htm** and **.html** extensions with your Web browser, so if you click a file with one of these extensions the file will be displayed by your Web browser. **The page will display whether or not your Web server is running because you have opened it directly from your file system.**

This is a problem because if you don't use your Web server to open your Web applications, they will not be processed by the Web server. Your **.html** pages will still display OK because these files do not require any special processing. But if you try to submit a form or if you click a **.php** file you will have trouble because .php files **must** be processed by the Web server before they can be displayed correctly by the browser. Since we are working with a combination of .html and .php files, whenever you want to run your applications, you **must always first** run the Web server and **then view your .html**

and .php files by connecting to the Web server, using a URL that begins with the **localhost** domain.

To understand this better, let's try running a Web application without using a URL, just to see what happens.

Use **Windows File Explorer** (or **Finder** on a Macintosh) to navigate to the **samples** folder on your drive. Double-click the file named **add-two-numbers.html**. A Web browser will probably start up, open the file and display the Web page (Figure 2-9). This page looks fine, but notice the file path that is displayed in the browser's address window, which in Windows will be something like:

file://F:/xampp/htdocs/webtech/samples/add-two-numbers.html

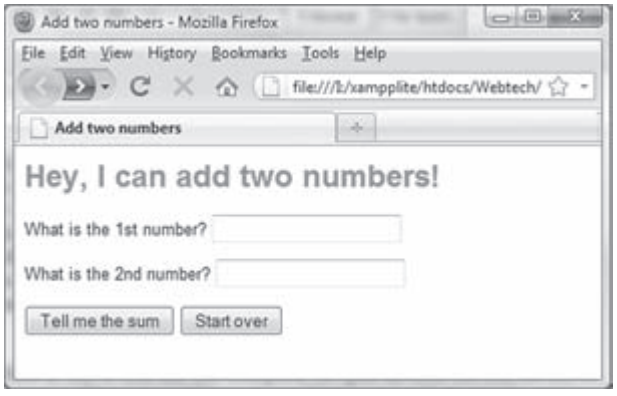

Figure 2-9: Opening add-two-numbers.html using a Windows file path instead of a URL

Because we opened the file in Windows, we see the Windows file path in the address window that begins **file://** rather than a URL that begins **http://localhost**. That tells you that you are not connecting to the file through your Web server, which means you cannot process the form. If you type in two numbers and click the "Tell me the Sum" button, you will see something unexpected (Figure 2-10).

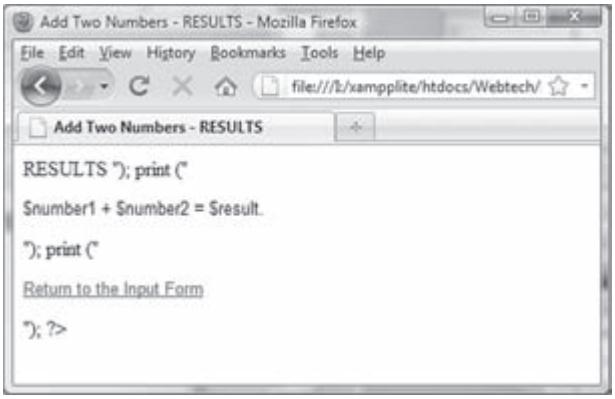

Figure 2-10: Result when opening add-two-numbers.html without a URL

This doesn't make much sense! What is happening is that when the "Tell me the Sum" button was clicked, the Web browser opened a file named add-**two-numbers.php** which contains the PHP instructions to process the form. However the browser was simply opening the file and displaying the contents. The browser was not submitting a request to your Web server to open and process the file. Your Web server includes the PHP processor which is necessary to execute the PHP instructions in this file. To access the Web server from your Web browser you **must** use URLs that begin with the domain name **localhost**.

So the correct way to view your .html and .php files is to always first run the Web server (if it is not already running), and then provide the URL to open the appropriate file. For example to correctly run the addTwoNumbers application, use the URL:

http://localhost/webtech/samples/add-two-numbers.html

Go ahead and do this to see that the application now works as expected.

Where to Save Your Work Files

You will create and edit your HTML and PHP work files in the chapter folders under the **xampp\htdocs\webtech\coursework** folder. These files will produce small Web applications. You will use a text editor to develop the code for these files and then save them. It is important that you save these files in the correct location so that the Web server can find them when you type in the URL. Once you are ready to test your applications, be sure that the Web server is running and then use your Web browser to run your programs. The URL for your files will be:

http://localhost/webtech/coursework/chapterXX/yyy

where chapterXX is a chapter number (for example **chapter02**) and yyy is the name of a specific file, for example **my-first.html**. To avoid typing the entire URL, you can just type **http://localhost/webtech** and then click the links on that page to obtain the appropriate folder and file.

The Importance of Frequent Backups

Always keep a recent backup of your **webtech** folder on a separate disk (for example on your hard drive at home if you are using a portable disk as your primary workplace), or on a portable disk if your hard drive is your primary workplace. It is good practice to back up your work every time you make major changes, or at least once a week. If your drive crashes, you can reinstall the Web server on a new drive and then copy the backup of your **webtech** folder into the **htdocs** folder of your new installation. Take your backups seriously — there is nothing worse than losing hours, days, or weeks of hard work.

Creating an HTML Document

In order to get started, you will first create a simple HTML document, store it on your server and then send a request to open the document from your client Web browser. Don't be concerned about understanding this document right now — you will learn about HTML in Chapter 4.

Open your text editor. Type in the text for **my-first.html**, but write your name instead of "YOUR NAME", write today's date instead of "TODAY'S DATE", and write something about yourself to replace the words "WRITE ABOUT YOURSELF HERE":

```
<!-- Author:  YOUR NAME
   Date:      TODAY'S DATE
   File:      my-first.html
   Purpose:   HTML Practice
-->
<html>
<head>
   <title>HTML Example</title>
</head>
<body>

   <h1>My Web Page</h1>

   <p>Hi! My name is <strong>YOUR NAME</strong>. Let me tell you a
   little bit about myself ... </p>

   <p>WRITE ABOUT YOURSELF HERE</p>

</body>
</html>
```

Code Example: my-first.html

Choose **Save As** from the **File** menu and save the file as follows (your **Save in** address will reflect the actual location of your **xampp** folder):

Save in: **xampp\htdocs\webtech\coursework\chapter02**
File name: **my-first.html**
Save as Type: **HTML**

The file has now been stored on the Web server. You can now submit a request to view this file from your Web browser. Be sure your server is running. Now type the following URL in your browser's address box:

http://localhost/webtech/coursework/chapter02/my-first.html

When the browser submits this request, the Web server receives the URL and locates the file (my-first.html). Since the file has an .html extension, the server simply sends the file contents back to the Web browser for display. Web browsers are designed to read HTML documents and treat any HTML tags as formatting instructions. We will

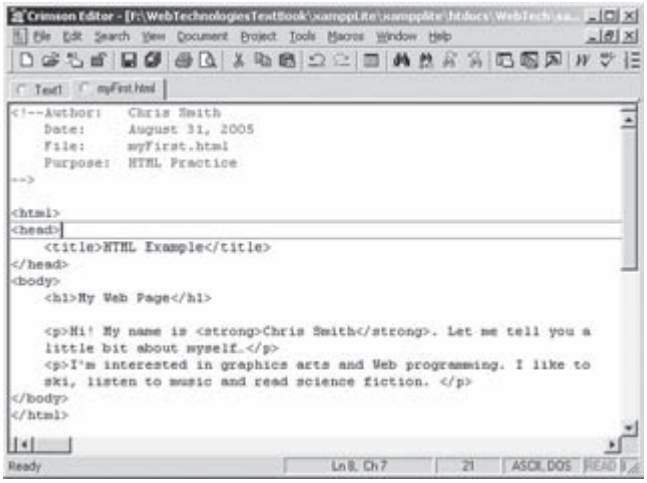

Figure 2-11: Using a text editor

learn more about this in the next chapter. Your HTML page should look similar to the screenshot in Figure 2-12.

If the link to **my-first.html** does not work either your file name is different or the file is not in the correct location, or the Web server is not running.

If you wish to make changes to your document, simply edit your code in your text editor. Be sure to save your changes before viewing the file again and be sure to **refresh** the page in your Web browser otherwise your browser may continue to display the previous version.

Congratulations! You have just created a simple Web page, stored it on your local server, then accessed the page from a client (your Web browser)!

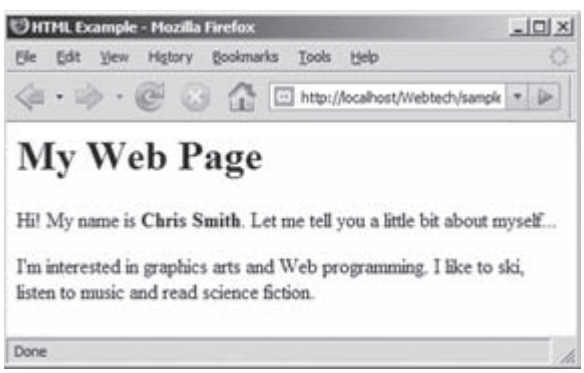

Figure 2-12: my-first.html screenshot

Creating a PHP program

Now let's create a simple PHP program. Don't be concerned about understanding this document right now — you will learn about PHP in Chapter 5. Type the code listing for **myFirst.php** in your preferred text editor exactly as written except type your name instead of "YOUR NAME" and today's date instead of "TODAY'S DATE". Note that you can copy and paste code from **my-first.html** to save some time:

```
<!-- Author: YOUR NAME
   Date:     TODAY'S DATE
   File:     my-first.php
   Purpose:  PHP Practice
-->
<html>
<head>
   <title>First PHP Example</title>
</head>
<body>

   <h1>Circle Calculation</h1>

   <?php
      $radius = 15.75;
      $area = pi() * pow ($radius, 2);
      $circumference = 2 * pi() * $radius;

      print("<p>A circle with a radius of $radius has an area of
         $area and a circumference of $circumference.</p>");

      print("<p>That's all that I have been designed to tell
        you!</p>");

   ?>
</body>
</html>
```

Code Example: my-first.php

Choose **Save As** from the **File** menu and save the file as follows:

Save in: **xampp\htdocs\webtech\coursework\chapter02**
File name: **my-first.php**
Save as Type: **PHP**

Now view this in your browser by typing the following URL:

http://localhost/webtech/coursework/chapter02/my-first.php

When the browser submits this request, the Web server receives the URL and locates the file (my-first.php). Since the file has a .php extension, the server runs a PHP processor to execute any PHP code in the file and assemble a new HTML document. Once the PHP has been completely processed, the newly created HTML document is sent

back to the Web browser for display. Your page should look something like the screenshot in Figure 2-13. We will learn more about this process in later chapters.

The text may wrap differently depending on the size of your browser window. If the link to **my-first.php** does not work as expected, you may have used the wrong file name, or saved the files in the wrong location. Or you may have forgotten to start your Web server. You may receive an error message like this:

```
Parse error: parse error, unexpected T_VARIABLE in
F:\xampp\htdocs\webtech\coursework\my-first.php on line 15
```

That means you have a syntax error in your PHP code. Programming languages such as PHP require a very precise syntax. There is a good chance that you may mistype

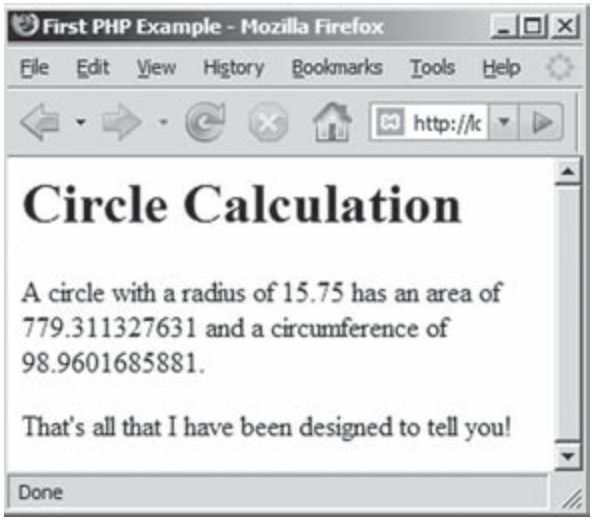

Figure 2-13: myFirst.php screenshot

something and your page may display differently (for example one or more numbers may not display as expected). Compare your code carefully with the example and see if you can find the errors. Once again remember to save your changes and remember to refresh the browser window to view your revised program.

Congratulations! You have just created a simple PHP program, stored it on your local Web server, and accessed it from a client (your Web browser)!

Creating an Interactive HTML and PHP Program

That last example displays information concerning a circle with a radius of **15.75**. We could improve the utility of this application by allowing the user to enter **any** radius. Next we will create a new version of this application that consists of two documents.

The first (named **circle.html**) will be an HTML document that contains a form so that the user can submit a radius and submit this for processing. The second document (named **circle.php**) will contain a PHP program that receives the radius and calculates and displays the circumference and area of the circle. Here is the code for **circle.html**:

```
<!-- Author: YOUR NAME
   Date:     TODAY'S DATE
   File:     circle.html
   Purpose:  PHP Practice
-->
<html>
<head>
   <title>Circle Calculation</title>
</head>
<body>

   <h1>Circle Calculation</h1>

   <form action="circle.php" method="post">
      <p>What is the radius of the circle?
      <input type="text" size="20" name="radius"></p>
      <p><input type="submit" value="Tell me the area and
         circumference"></p>
   </form>
</body>
</html>
```

Code Example: circle.html

Choose Save As from the File menu and save the file as follows (your Save in address will reflect the actual location of your xampp folder):

Save in: **xampp\htdocs\webtech\coursework\chapter02**
File name: **circle.html**
Save as Type: **HTML**

The file has now been stored on the Web server. You can now submit a request to view this file from your Web browser. Just type the following URL in your browser's address box:

http://localhost/webtech/coursework/chapter02/circle.html

If you do this the document will display in your browser. Assuming that you typed everything correctly, you will see that it contains a Web page with a form (Figure 2-14).

If you enter a radius into the text box and click the "Tell me the area and circumference" button, you will get an error message, similar to that shown in Figure 2-15.

That's because the form on this Web page is designed to send the radius to a program named **circle.php** in order for it to be processed. The problem is that we haven't created the **circle.php** program yet! So let's do that right now!

Here is the code for **circle.php**:

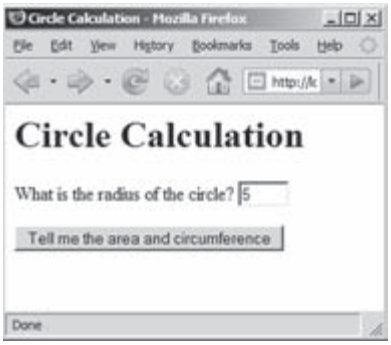

Figure 2-14: circle.html screenshot

```
<!-- Author: YOUR NAME
   Date:      TODAY'S DATE
```

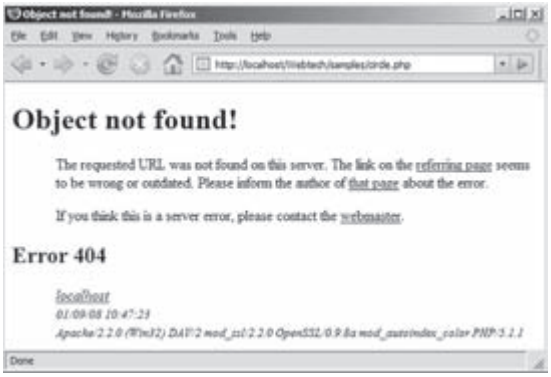

Figure 2-15: Connecting to a page that does not exist

```
   File:      circle.php
   Purpose:   PHP Practice
-->
<html>
<head>
   <title> Circle Calculation</title>
</head>
<body>

   <h1>Circle Calculation</h1>

   <?php
      $radius = $_POST['radius'];
      $area = pi() * pow ($radius, 2);
      $circumference = 2 * pi() * $radius;
```

```
    print("<p>A circle with a radius of $radius has an area of
      $area and a circumference of $circumference.</p>");
  ?>
  <p><a href="circle.html">Calculate another circle?</a></p>
</body>
</html>
```

Code Example: circle.php

Choose **Save As** from the **File** menu and save the file as follows:

Save in: **xampp\htdocs\webtech\coursework\chapter02**
File name: **circle.php**
Save as Type: **PHP**

Now you should be able to use your form correctly. Open your Web document again:

http://localhost/webtech/coursework/chapter02/circle.html

Type a radius into the text box and click the "Tell me the area and circumference" button. This time you should see a new page that displays the area and circumference of a circle with the radius that you submitted (Figure 2-16).

Note that you can click the "Calculate another circle?" link to return to the first page.

Congratulations! You have now created a simple Web application that: provides an input form, processes the input, and displays the result.

Do not be concerned about how these documents actually work at this point. The purpose of the current exercise is to give you practice using a text editor to type HTML and PHP code, saving your documents to the correct location on your disk, making cor-

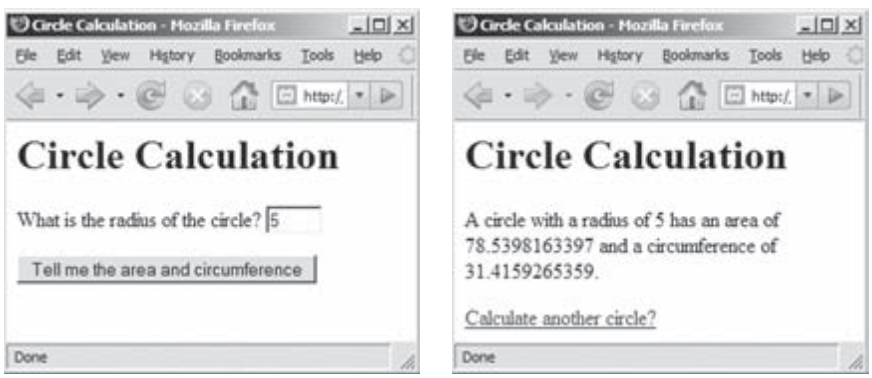

Figure 2-16: circle.html and circle.php screenshots

rections as needed, and viewing your applications in a Web browser using the correct URL. In the following chapters you will learn how to design and create Web applications that include HTML pages with forms and PHP programs that process these forms.

Summary

Client/Server applications are designed so that client-based software such as Web browsers can submit requests to server-based programs such as Web servers for processing. This makes it easy for large numbers of client programs to make use of common services. Another advantage is that new software installations and updates to existing software are performed on the server without requiring changes to the client computers. Client/server applications are used world-wide across the **Internet**, and are also used to deliver services on private **intranets**.

Web-based applications are developed by creating the required code and storing this on a Web server. Clients can submit requests to the server to process these files and return the results. When a server receives a request for an **.html** file, it simply locates and opens the file and returns the document for display by the Web browser. When a server receives a request for a **.php** file, it first processes the PHP instructions in the file and then returns the new HTML document that is generated. The Web server must be running in order for a client to submit requests.

Files usually include file extensions to indicate the data format that is stored in the file. Files can only be processed by programs designed to work with the file format. Text files can be opened by any text editing program, and sometimes contain text that requires specialized processing. In this course we will work with text files using various file extensions. Files with a **.txt** extension will be used to store data such as student scores or wage information. Files with an **.html** extension will contain HTML markup code for display in a Web browser. Files with a **.php** extension will contain PHP code that is processed by a PHP processor running on a Web server.

Under the Windows addressing scheme, file/folder locations are based on a drive letter to represent a specific disk drive, for example:

C:\xampp\htdocs\webtech\coursework\chapter02\circle.html

Under the Internet addressing scheme, file/folder locations are based on an Internet domain name, for example:

http://localhost/webtech/coursework/chapter02/circle.html

The domain name is mapped to an IP address. In this case the IP address for the **localhost** domain is **127.0.0.1**

In this chapter you learned the basic steps to install, start and stop a local Web server. You also learned to create and modify HTML and PHP code using a text editor, and then view the results in a Web browser.

When you use the Web server, the URLs of your documents will always begin:

http://localhost/

This URL points to the **xampp/htdocs** folder which is located wherever you installed the Web server on your disk. Be sure that the Web server is running before using this

URL. Always be sure to use the Internet address when you attempt to view your .html and .php files. Do not use the Windows address, otherwise your .php programs will not be processed.

Don't be concerned about the actual content of the HTML and PHP code yet. The purpose of these exercises is to learn the procedures. You are now ready to learn how to design and code your own Web-based, client/server programs using HTML and PHP.

Chapter 2 Review Questions

1. Consider the following address: **D:\CourseMaterial\Coursework\wage1.html** Which statement is true?
   a. The file is stored in a folder named wage1.html
   b. The file is stored in a folder named Coursework
   c. The file is stored in a folder named CourseMaterial
   d. The file is stored in a folder named D:
   e. This address does not specify a file

2. Consider the following address: **D:\CourseMaterial\Coursework\wage1.html** What is the name of the file?
   a. wage1.html
   b. Coursework
   c. CourseMaterial
   d. D:
   e. This address does not specify a file

3. Consider the following address: **D:\CourseMaterial\Coursework\wage1.html** Where is the CourseMaterial folder located?
   a. Inside wage1.html
   b. Inside the Coursework folder
   c. On the D: drive
   d. It is not possible to tell from this address
   e. The address is incorrect

4. What type of address is this? D:\CourseMaterial\Coursework\wage1.html
   a. Internet address
   b. Microsoft Windows address

5. What is wrong with the following URL?
   http:\\www.w3.org\markup\guide\style.html
   a. Internet addresses must use the forward slash / as a separator
   b. The file name is in the wrong location
   c. The domain name is the wrong location
   d. The file name is missing
   e. The drive letter is missing

6. What is wrong with the following URL?
   http://www.w3.org/style.html/markup/guide/
   a. Internet addresses must use the back slash \ as a separator
   b. The file name is in the wrong location
   c. The domain name is the wrong location
   d. The file name is missing
   e. The drive letter is missing

7. Which component of a client/server application processes a PHP file?
   a. Client
   b. Server

8. What does HTML stand for?
   a. Highly Technical Markup Language
   b. Host Translated Markup Language
   c. Hypertext Markup Language
   d. Hands-on Technical Markup Language
   e. Hyper Transitional Markup Language

9. What is HTML?
   a. A markup language used to provide formatting instructions for text
   b. A programming language used to process input, perform calculations and other operations, and generate output
   c. An addressing scheme for URLs
   d. A name for a button on a user interface
   e. A form used to validate user input

10. What is PHP?
    a. A markup language used to provide formatting instructions for text
    b. A programming language used to process input, perform calculations and other operations, and generate output
    c. An addressing scheme for URLs
    d. A name for a button on a user interface
    e. A form used to validate user input

11. What is the domain name of the Internet address that you will use to access Web pages delivered by your standalone server?
    a. localhost
    b. www.w3.org
    c. webtech
    d. samples
    e. welcome.html

12. Where should you save the html and php files that you create for this course?
    a. Anywhere on your disk is fine
    b. In the correct Chapter folder under xampp\htdocs\webtech\coursework\
    c. In the correct Chapter folder under xampp\webtech\htdocs\coursework\
    d. In the correct Chapter folder under xampp\webtech\coursework\
    e. In the correct Chapter folder under xampp\coursework\

13. Which folder is divided into chapters?
    a. samples folder
    b. coursework folder

14. Which one of the following files can be found in your samples folder?
    a. getting-started.html
    b. getting-started.php
    c. course-web-site.html
    d. quote-generator.html
    e. quote-generator.php

15. In your samples folder there is a file named temp-converter1.php. What should you do to run this program and find out what it does?
    a. Open the file using File Explorer (Windows) or Finder (Macintosh)
    b. Run your local Web server and then type the url **http://localhost/webtech/samples/temp-converter1.php** in your browser window.
    c. Run your local Web server and type http://localhost/webtech/ in your browser window, then click on samples and then click on **temp-converter1.php.**
    d. Either of the last two procedures will work but not the first one

16. In your samples folder, what happens if you run quote-generator.php in your browser window?
    a. The browser displays the same quote every time you run it.
    b. The browser displays a different quote each time you run it.
    c. The browser displays a different presidential quote each time you run it.
    d. The browser asks you to input a quote.

17. Which of the following file types does NOT appear in the samples folder?
    a. .bmp
    b. .html
    c. .jpg
    d. .php
    e. .txt

18. What is the correct URL for your my-first.php document?
    a. http://localhost/webtech/coursework/chapter02/my-first.php
    b. http://webtech/coursework/chapter02/my-first.php
    c. http://localhost/coursework/chapter02/my-first.php
    d. http://webtech/chapter02/my-first.php
    e. http://localhost/my-first.php

19. In the samples folder, if you open welcome.html and submit the information but leave the first name and last name boxes blank, what does the resulting Web page display (among other things)?
    a. ERROR — INPUT IS MISSING!
    b. ERROR — YOU MUST ENTER YOUR FIRST NAME AND LAST NAME!
    c. Welcome!
    d. Welcome Whoever You Are!
    e. Welcome! You must enter your first name and last name!

20. In the samples folder, what is the difference between wage1.html and wage2.html?
    a. wage1.html allows you to input your hours worked and hourly wage but wage2.html does not
    b. wage2.html allows you to input your hours worked and hourly wage but wage1.html does not
    c. wage1.html allows you to input your name but wage2.html does not wage2.html allows you to input your name but wage1.html does not
    d. There is no difference between wage1.html and wage2.html

Chapter 2 Code Exercises

The exercises for this chapter are simply intended to ensure that (a) you have your software installed and working correctly, and (b) you are comfortable with the process of creating, editing and running your Web applications.

1. First be sure that you have installed your Web server. Now run the XAMPP Control Panel and start your server. If you have any problems when you do this, first review the steps to install and run your Web server before you assume there is a problem with your installation. Refer to the **Installation Problems** guide on the textbook Web site at:

   http://www.mikeokane.com/textbooks/webtech/support.php

2. With your Web server started, open a Web browser and type the URL:

   http://localhost/webtech/samples/art-gallery.html

   You should see a Web page that displays a "Welcome to the Art Gallery" heading and allows you to choose an artist from a drop down list. If you do not see this and receive an error message instead, first be sure that you typed the URL correctly.

If you still receive an error, then (a) your Webtech folder was not included in your installation (use **File Explorer (Windows) or Finder (Macintosh)** to ensure that this folder is located in your **xampp/htdocs** folder), or (b) your server is not running (the server may not be running if you also get an error message when you type the URL **http://localhost** in your browser address window), or (c) your server was not installed correctly (this may be the case if you received an error message when you started the server in the XAMPP Control Panel). Make a careful note of all error messages and anything else that might be useful, then refer to the **Installation Problems** guide on the textbook Web site for help.

3. Assuming that the artGallery page is displayed, select an artist and then click the "Show me an Artwork" button. An artwork should now be displayed along with some additional information. If you do not see this, check the URL in your Web browser's address window. If the URL begins with **file://** then you are not using the right URL and you are not connecting to the Web server. Go back and use the correct URL. Remember that, to connect to your Web server, your URLs must **always** begin **http://** and in this case the URL should be:

   http://localhost/webtech/samples/artGallery.html

4. If the art-gallery application performed successfully then you are ready to work. To become familiar with the procedure that we will follow throughout this textbook, use your text editor as directed in this chapter to create **my-first.html**, **my-first.php**, **circle.html**, and **circle.php** and save these files in the **chapter02** folder of your coursework folder if you have not already done this. Test each of these by running your Web server, opening a Web browser, and typing the appropriate URLs. You may need to fix some of your code but eventually each program should run as described in the chapter. **Do not bypass this exercise**. You need to be comfortable creating and running your applications and using your Web server in order to work through this book.