Web Servers (Apache and IIS)

17.1 Introduction

- A web server responds to client requests (typically from a web browser) by providing resources such as HTML documents
- When users enter a Uniform Resource Locator (URL) address, such as www.deitel.com, into a web browser
  - They're requesting a specific document from a web server
  - The web server maps the URL to a resource on the server (or to a file on the server's network)
    - Returns the requested resource to the client
- A web server and a client communicate using the platform-independent Hypertext Transfer Protocol (HTTP)
  - A protocol for transferring requests and files over the Internet or an intranet

17.2 HTTP Transactions

- URIs (Uniform Resource Identifiers) identify resources on the Internet
- URIs that start with **http://** are called URLs (Uniform Resource Locators)
- Common URLs refer to files, directories or server-side code that performs tasks
  - Such as database lookups, Internet searches and business-application processing
- A URL contains information that directs a browser to the resource that the user wishes to access
- **http://** indicates that the HyperText Transfer Protocol (HTTP) should be used to obtain the resource

17.2 HTTP Transactions (Cont.)

- Next in the URL is the server's fully qualified hostname
  - The name of the web-server computer on which the resource resides
- A hostname is translated into an IP (Internet Protocol) address—a numerical value that uniquely identifies the server
  - An Internet domain name system (DNS) server maintains a database of hostnames and their corresponding IP addresses and performs the translations automatically
- The remainder of the URL after the hostname specifies the resource's location (**/books**) and name on the web server

17.2 HTTP Transactions (Cont.)

- For security reasons the path location is typically a virtual directory
  - The web server translates the virtual directory into a real location on the server, thus hiding the true location of the resource
- When given a web page URL, a web browser uses HTTP to request and display the web page found at that address
- HTTP method get indicates that the client wishes to obtain a resource from the server
  - The remainder of the request provides the path name of the resource (e.g., an HTML5 document) and the protocol's name and version number (HTTP/1.1)

17.2 HTTP Transactions (Cont.)

- Any server that understands HTTP can receive a get request and respond appropriately
- HTTP status code 200 indicates success
- Status code 404 informs the client that the web server could not locate the requested resource
- A complete list of numeric codes indicating the status of an HTTP transaction can be found at
  - http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

17.2 HTTP Transactions (Cont.)

- Next, the server sends one or more HTTP headers, which provide additional information about the data that will be sent
- The Multipurpose Internet Mail Extensions (MIME) standard specifies data formats, which programs can use to interpret data correctly
  - The MIME type text/plain indicates that the sent information is text that can be displayed directly
- The MIME type image/jpeg indicates that the content is a JPEG image
  - When the browser receives this MIME type, it attempts to display the image

17.2 HTTP Transactions (Cont.)

- The header or set of headers is followed by a blank line, which indicates to the client browser that the server is finished sending HTTP headers

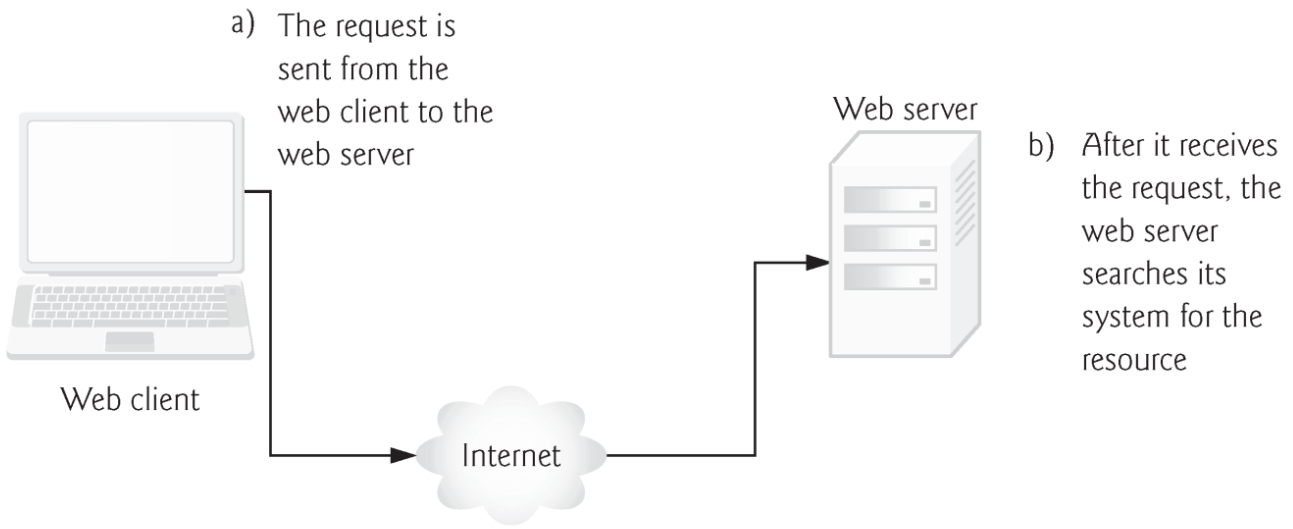

Fig. 17.1 | Client interacting with web server. *Step 1:* The GET request.

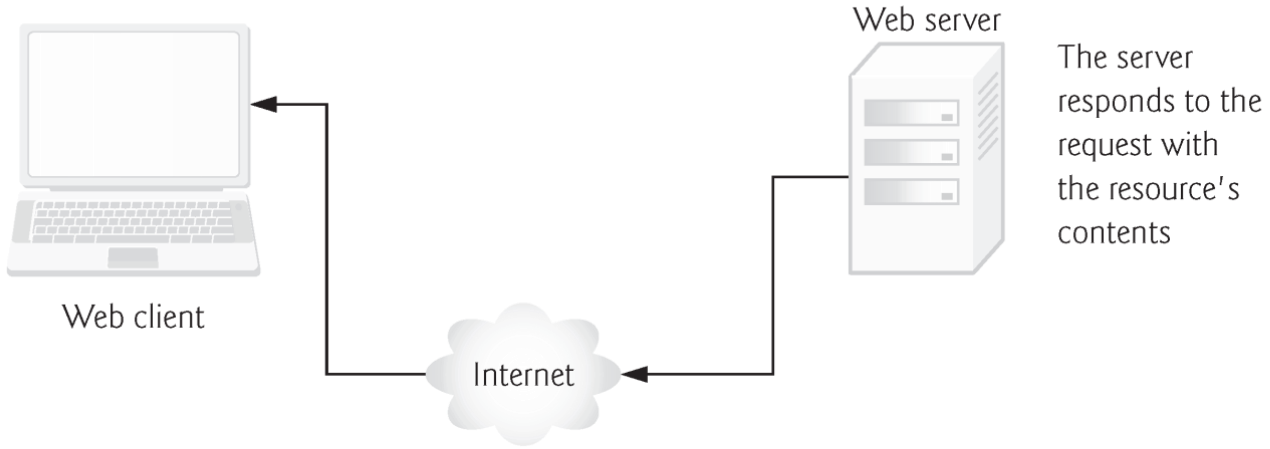

Fig. 17.2 | Client interacting with web server. *Step 2:* The HTTP response.

17.2 HTTP Transactions (Cont.)

- Two most common HTTP request types
  - **get** and **post**
- **get** request typically gets (or retrieves) information from a server
  - Common uses of get requests are to retrieve an HTML document or an image, or search results based on a user-submitted search term
- **post** request typically posts (or sends) data to a server
  - Common uses of post requests are to send information to a server
    - Such as authentication information or data from a form that gathers user input

17.2 HTTP Transactions (Cont.)

- A **get** request appends data to the URL in a query string
- A **?** separates the query string from the rest of the URL in a get request
  - **http://www.google.com/search?q=deitel**
- A name/value pair is passed to the server with the name and the value separated by an equals sign (=)
- If more than one name/value pair is submitted, each pair is separated by an ampersand (&)
  - **https://www.google.com.tw/search?q=deitel&book**
- A get request may be initiated by submitting an HTML form whose method attribute is set to "get", or by typing the URL directly into the browser's address bar

17.2 HTTP Transactions (Cont.)

- An HTTP request often **posts** data to a server-side form handler that processes the data
- A **post** request sends form data as an HTTP message, not as part of the URL
- A **get** request limits the query string to a specific number of characters
- Large amounts of information must be sent using the post method

17.2 HTTP Transactions (Cont.)

- Browsers often cache recently viewed web pages so they can quickly reload the pages
- If there are no changes between the version stored in the cache and the current version on the web, this helps speed up your browsing experience

Software Engineering Observation 17.1

The data sent in a `post` request is not part of the URL, and the user can't see the data by default. However, tools are available that expose this data, so you should not assume that the data is secure just because a `post` request is used.

17.3 Multitier Application Architecture

- Web-based applications are often multitier applications that divide functionality into separate tiers
- Although tiers can be located on the same computer, the tiers of web-based applications typically reside on separate computers
- The bottom tier (also called the data tier or the information tier) maintains the application's data
- The middle tier implements business logic
  - Controller logic and presentation logic to control interactions between the application's clients and its data

17.3 Multitier Application Architecture

- Business logic in the middle tier enforces business rules and ensures that data is reliable before the server application updates the database or presents the data to users
  - Business rules dictate how clients can and cannot access application data, and how applications process data
- The top tier, or client tier, is the application's user interface
  - In response to user actions, the client tier interacts with the middle tier to make requests and to retrieve data from the information tier
  - The client tier then displays the data retrieved for the user
  - The client tier never directly interacts with the information tier

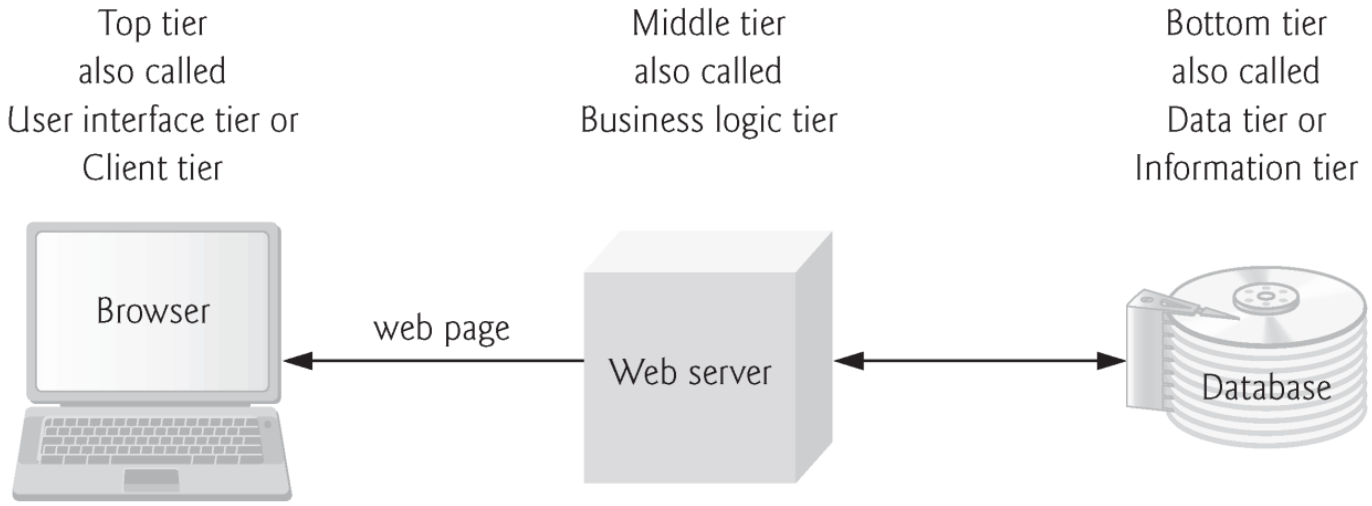

Fig. 17.3 | Three-tier architecture.

17.4 Client-Side Scripting versus Server-Side Scripting

- Client-side scripting can be used to
  - Validate user input, interact with the browser, enhance web pages, and to add client/server communication between a browser and a web server
- Client-side scripting does have limitations, such as browser dependency
  - The browser or scripting host must support the scripting language and capabilities
- Client-side scripts can be viewed by the client by using the browser's source-viewing capability

17.4 Client-Side Scripting versus Server-Side Scripting (Cont.)

- Sensitive information, such as passwords or other personally identifiable data, should not be stored or validated on the client
- Placing large amounts of JavaScript on the client can open web applications to security issues
- Code executed on the server often generate custom responses for clients
- Server-side scripting languages have a wider range of programmatic capabilities than their client-side equivalents

17.5 Accessing Web Servers

- To request documents from web servers, users must know the hostnames on which the web server software resides
- Users can request documents from local web servers or remote web servers
- Local web servers can be accessed through your computer's name or through the name localhost
  - A hostname that references the local machine and normally translates to the IP address 127.0.0.1
  - Also known as the loopback address

17.6 Apache, MySQL and PHP Installation

- The Apache HTTP Server, maintained by the Apache Software Foundation, is currently the most popular web server
- It's open source software that runs on UNIX, Linux, Mac OS X, Windows and numerous other platforms
- MySQL is the most popular open-source database management system
  - It, too, runs on Linux, Mac OS X and Windows
- The Apache HTTP Server, MySQL database server and PHP can each be downloaded and installed separately
  - This also requires additional configuration on your part

17.6 Apache, MySQL and PHP Installation

- For simplicity, we'll use the XAMPP integrated installer provided by the Apache Friends website
  - http://www.apachefriends.org
- Go to https://www.apachefriends.org/index.html
- Choose the installer for your platform
- Carefully follow the provided installation instructions and be sure to read the entire installation page for your platform
- If you'd prefer to use PHP with Microsoft's IIS Express and SQL Server Express, you can use their Web Platform Installer to set up and configure PHP
  - https://www.meersworld.net/2019/02/how-to-install-php-on-iis-in-windows-10.html

17.6.2 Running XAMP

- Windows
  - Go to your c:\xampp folder (or the folder in which you installed XAMPP) and double click xampp_start.exe
  - If you need to stop the servers (e.g., so you can shut down your computer), use xampp_stop.exe in the same folder
- Mac OS X
  - Go to your Applications folder (or the folder in which you installed XAMPP), then open the XAMPP folder and run XAMP Control.app
  - Click the Start buttons in the control panel to start the servers
  - If you need to stop the servers (e.g., so you can shut down your computer), you can stop them by clicking the Stop buttons

17.6.2 Running XAMP

- Linux
  - Open a shell and enter the command /opt/lampp/lampp start
  - If you need to stop the servers (e.g., so you can shut down your computer), open a shell and enter the command /opt/lampp/lampp stop

17.6.3 Testing Your Setup

- You've started the servers, you can open any web browser on your computer and enter the address http://localhost/ to confirm that the web server is up and running
- You're now ready to go!

Fig. 17.4 | default XAMPP webpage displayed on Windows.

17.6.4 Running the Examples Using Apache HTTP Server

- Now that the Apache HTTP Server is running on your computer, you can copy the book's examples into XAMPP's **htdocs** folder
- Assuming you copy the entire examples folder into the **htdocs** folder, you can run the examples in Chapters 2–16 and 19 with URLs of the form
  - http://localhost/examples/chapter/figure/filename
  - http://localhost/sample_code/ch02/links.html
  - http://localhost/sample_code/ch09/fig09_06/RollDice.html

17.7 Microsoft IIS Express and WebMatrix

- Microsoft Internet Information Services (IIS) is a web server that is included with several versions of Windows
- Installing IIS on a machine allows that computer to serve documents
- You can install it in a bundle with Microsoft's WebMatrix
  - A free development tool for building PHP and ASP.NET web apps

17.7.1 Installing and Running IIS Express

- If you simply want to test your web pages on IIS Express, you can install it from
  - http://www.microsoft.com/web/gallery/
  - We recommend using the default installation options
- Once you've installed IIS Express you can learn more about using it at
  - http://learn.iis.net/page.aspx/860/iis-express/

17.7.2 Installing and Running WebMatrix (No support now)

- You can install the WebMatrix from
  - https://www.microsoft.com/web/webmatrix/wmx3features.aspx
- Again, we recommend using the default installation options
- You can run WebMatrix by opening the Start menu and selecting All Programs > Microsoft WebMatrix > Microsoft WebMatrix
- This will also start IIS Express

17.7.3 Running the Client-Side Examples Using IIS Express

- To execute IIS Express, open a Command Prompt window and change directories to the IIS Express folder
- On 32-bit Windows versions, use the command
  - cd "c:\Program Files\IIS Express"
- On 64-bit Windows versions, use the command
  - cd "c:\Program Files (x86)\IIS Express"
- You can now run your examples with URLs of the form
  - http://localhost:8080/chapter/figure/filename
  - http://localhost:8080/sample_code/ch07/fig07_11/analysis.html

17.7.4 Running the PHP Examples Using IIS Express

- The easiest way to test Chapter 19's PHP examples is to use WebMatrix to enable PHP for the ch19 folder in the book's examples
- To do so, perform the following steps
  - Run WebMatrix by opening the Start menu and selecting All Programs > Microsoft WebMatrix > Microsoft WebMatrix
  - In the Quick Start - Microsoft WebMatrix window, select Site From Folder
  - Locate and select the ch19 folder in the Select Folder window, then click the Select Folder button
- This opens the ch19 folder as a website in WebMatrix ()

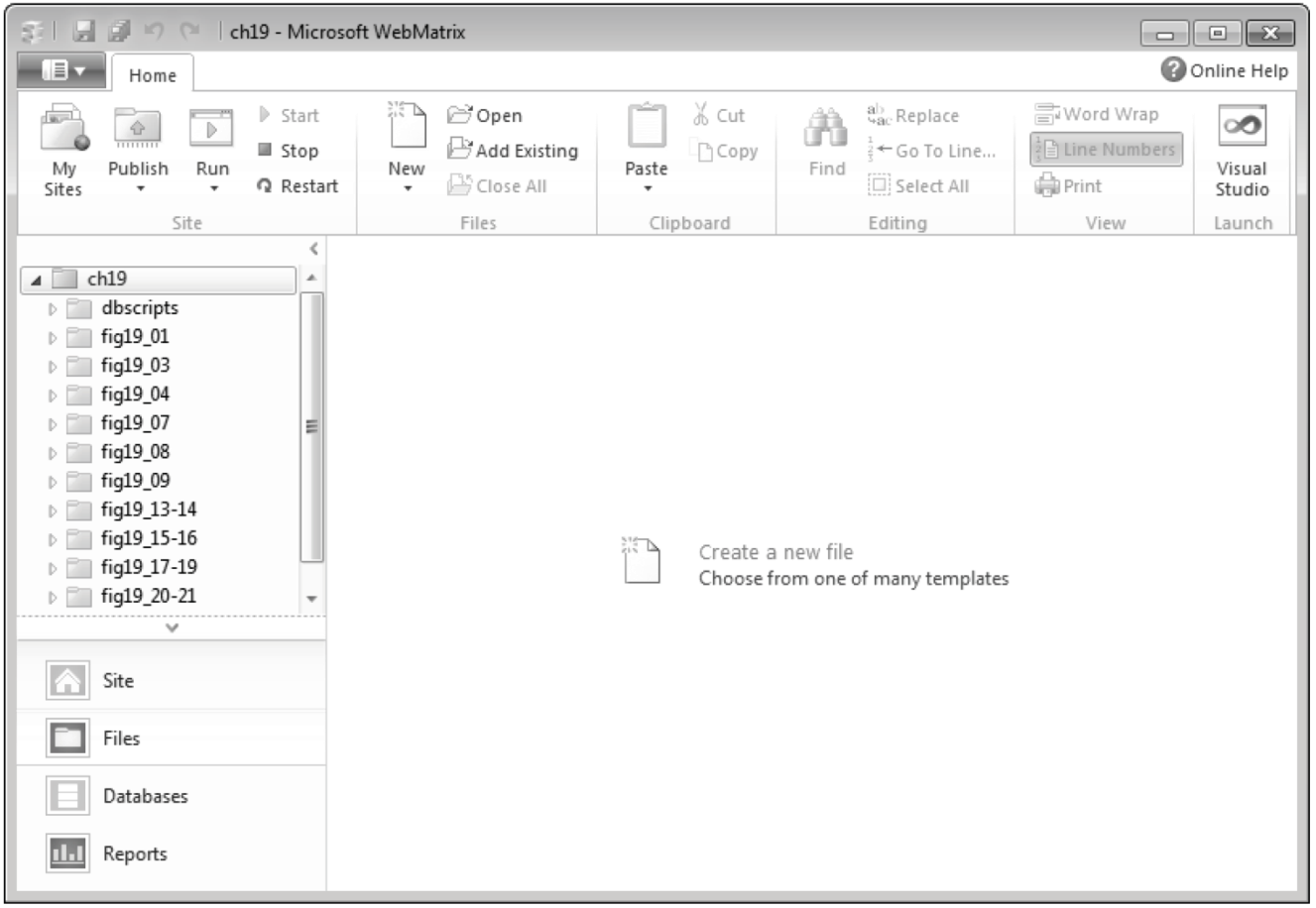

Fig. 17.5 | The ch19 examples folder in WebMatrix.

17.7.4 Running the PHP Examples Using IIS Express

- To enable PHP, perform the following steps
  - Click the Site option in the bottom-left corner of the window
  - Click Settings and ensure that Enable PHP is checked ()
  - The first time you do this, WebMatrix will ask you for permission to install PHP
  - You must do this to test the PHP examples

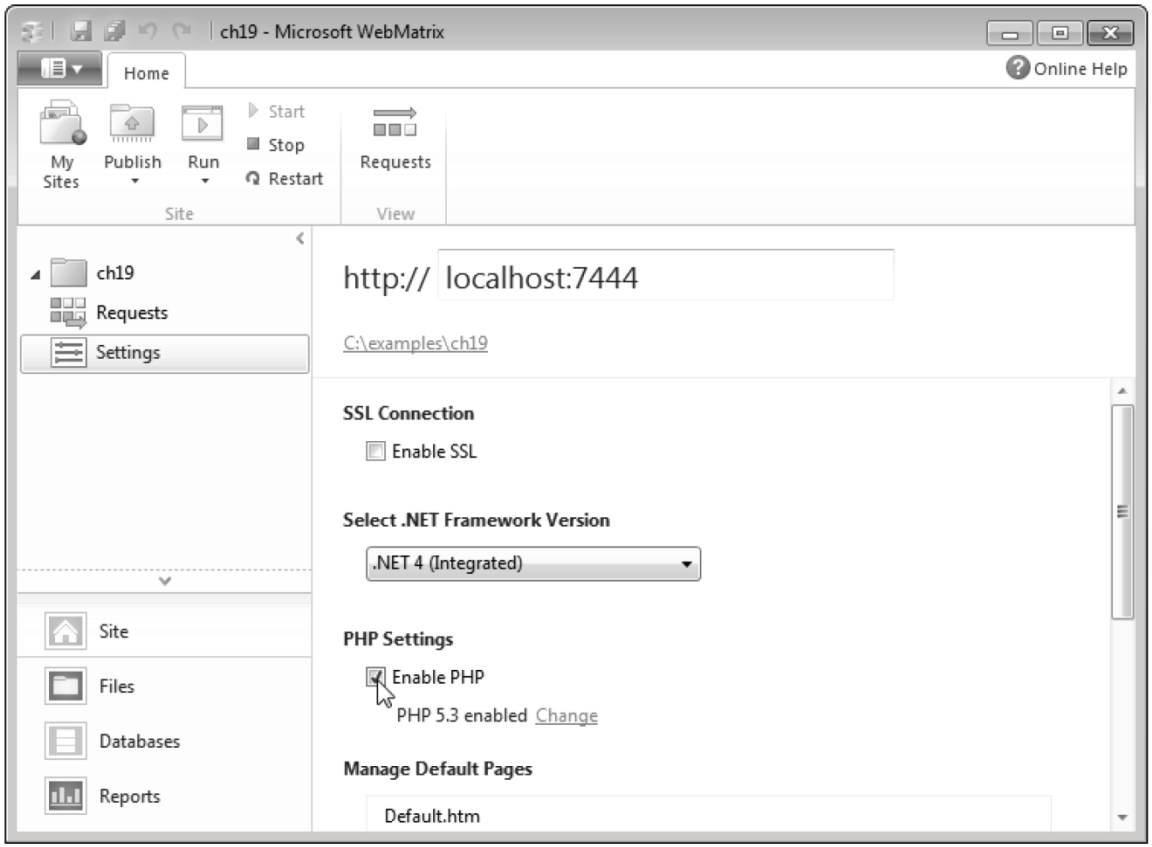

Fig. 17.6 | Enabling PHP for the ch19 examples folder in WebMatrix.