21

Web Servers

OBJECTIVES

In this chapter you will learn:

- To understand a web server's functionality.
- To introduce Apache HTTP Server.
- To set up virtual directories from which content can be served.
- To test whether you set up the virtual directory properly.

- Introduction
- HTTP Transactions
- Multitier Application Architecture
- Client-Side Scripting versus Server-Side Scripting
- Accessing Web Servers
- Apache HTTP Server
- Requesting Documents
- Web Resources

Introduction

- **A web server responds to client requests (typically from a web browser) by providing resources such as XHTML documents. When users enter a Uniform Resource Locator (URL) address, such as `www.deitel.com`, into a web browser, they are 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) and 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.**

HTTP Transactions

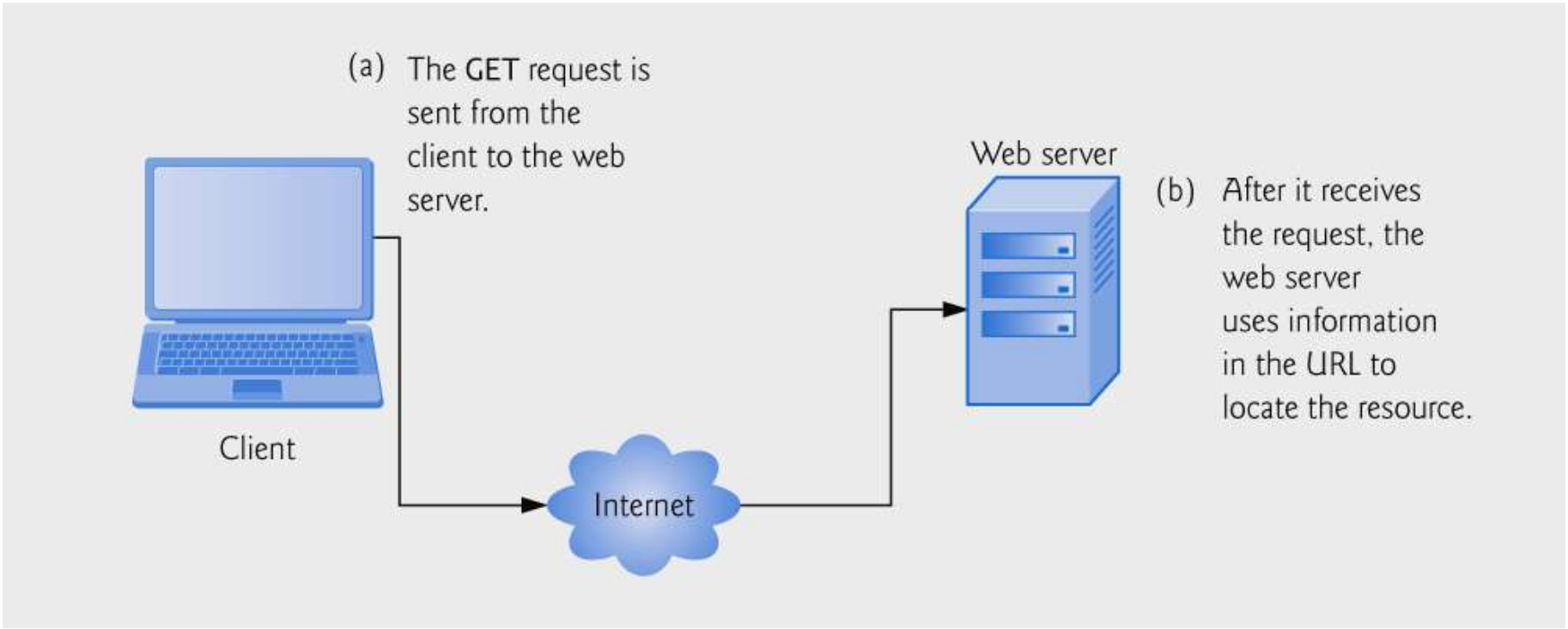
- **The HTTP protocol allows clients and servers to interact and exchange information in a uniform and reliable manner.**
- **HTTP uses URIs (Uniform Resource Identifiers) to identify data on the Internet.**
- **URIs that specify document locations are called URLs (Uniform Resource Locators). Common URLs refer to files, directories or objects that perform complex tasks, such as database lookups and Internet searches.**
- **A URL contains information that directs a browser to the resource that the user wishes to access.**
- **`http://` indicates that the resource is to be obtained using the HTTP protocol.**

HTTP Transactions (Cont.)

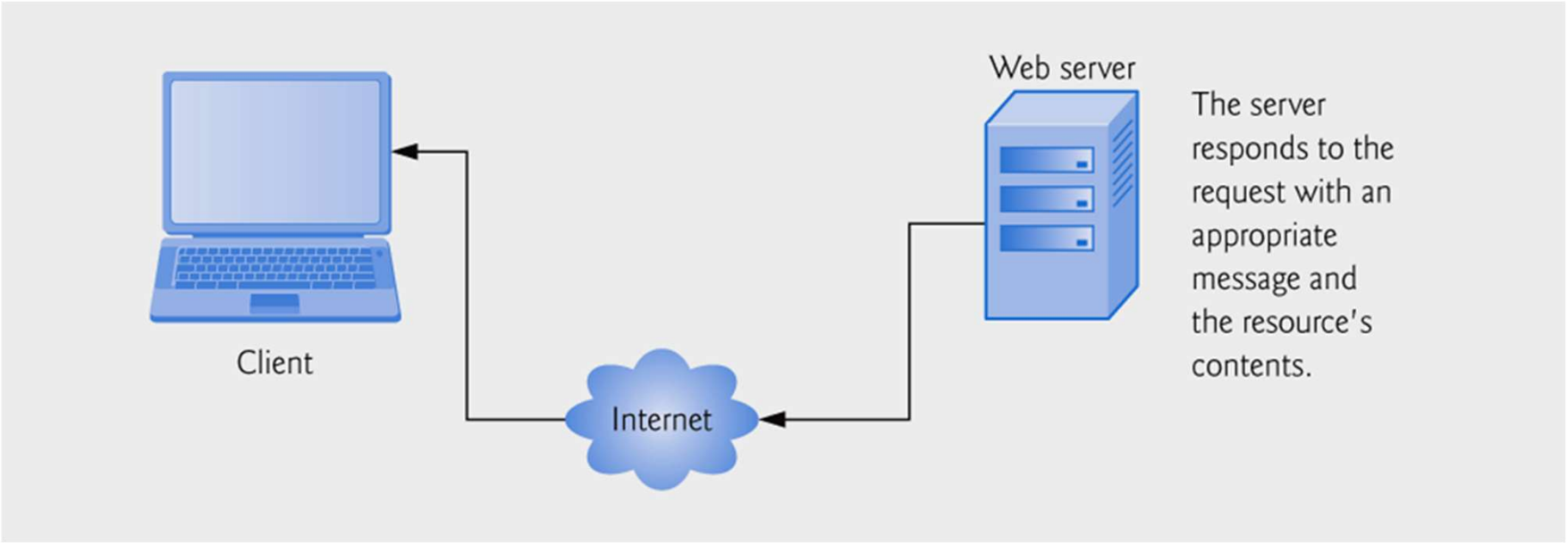
- **Fully qualified hostname**
  - **the name of the server on which the resource resides, called the host**
- **A hostname is translated into an IP address—a unique numerical value which identifies the server much as a telephone number uniquely defines a particular phone line**
  - **Translation is performed by a domain name system (DNS) server—a computer that maintains a database of hostnames and their corresponding IP addresses—and the process is called a DNS lookup**
- **The remainder of the URL after the hostname specifies both the name of the requested resource and its path, or location, on the web server**
- **For security reasons the path normally specifies the location of a virtual directory. The server translates the virtual directory into a real location on the server (or on another computer on the server's network), thus hiding the true location of the resource**
- **Some resources are created dynamically and do not reside anywhere on the server**

HTTP Transactions (Cont.)

- When given a URL, a web browser performs a simple HTTP transaction to retrieve 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 XHTML document) and the protocol's name and version number (`HTTP/1.1`).
- 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 Messages (w3schools.com)



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



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

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 XHTML document or an image, or to fetch 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.
  - An HTTP request often posts data to a server-side form handler that processes the data.
  - A `get` request sends information to the server as part of the URL in a query string. A `?` separates the query string from the rest of the URL in a get request. 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 (&).
  - A `get` request may be initiated by submitting an XHTML form whose `method` attribute is set to `"get"`, or by typing the URL (possibly containing a query string) directly into the browser's address bar
  - A `post` request is specified in an XHTML form by the `method "post"`. The `post` method 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 (2083 in IE; more in other browsers).
  - Large pieces of information must be sent using the `post` method.

Software Engineering Observation

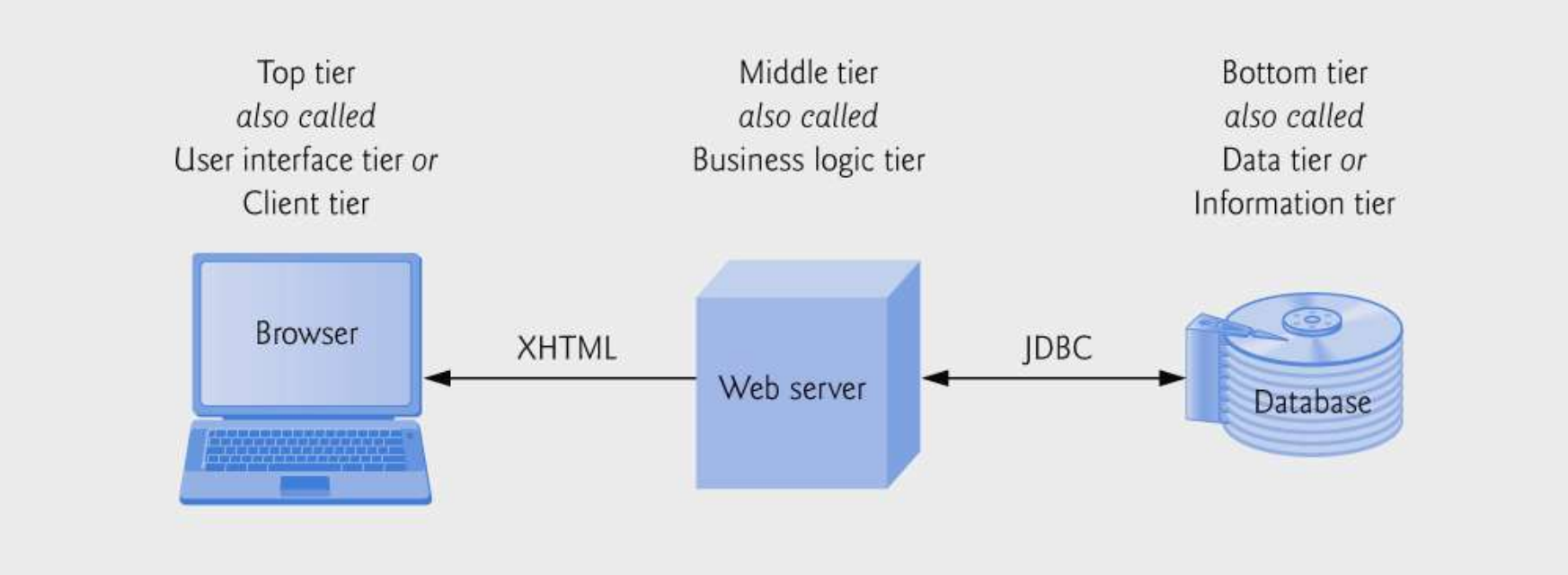
The data sent in a `post` request is not part of the URL and the user can't see the data by default. However there are tools available that expose this data, so you should not assume that the data is secure just because a `post` request is used.

HTTP Transactions (Cont.)

- **Browsers often cache 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.**

Multitier Application Architecture

- **Web-based applications are 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.**
- **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.**



Three-tier architecture.

Client-Side Scripting versus Server-Side Scripting

- **Client-side scripting can be used to validate user input, to interact with the browser, to enhance web pages by manipulating the DOM of a page.**
- **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.**
- **Sensitive information, such as passwords or other personally identifiable data, should not be stored or validated on the client.**

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

- Placing large amounts of JavaScript on the client can open web applications to attack and other 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. For example, server-side scripts often can access the server's file directory structure, whereas client-side scripts cannot access the client's directories.
- Properly configured server-side scripts are not visible to the client; only XHTML and any client-side scripts are visible to the client.

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).**

Apache HTTP Server

- **The Apache HTTP Server, maintained by the Apache Software Foundation. It is open source software that runs on UNIX, Linux, Mac OS X, Windows and numerous other platforms.**
- **Mac OS X and many versions of Linux come preinstalled with Apache.**
- **All documents that will be requested from an Apache HTTP Server must be either in the default directory or in a directory for which an Apache HTTP Server alias is configured. An alias in Apache is a pointer to an existing directory that resides on the local machine or on the network.**
- **The `httpd.conf` file contains all the information that the Apache HTTP Server needs to run correctly and serve web documents. An introductory comment at the top of the `httpd.conf` file explains how the file is organized. After this comment, the configuration information starts with the most important, global settings.**

Good Programming Practice

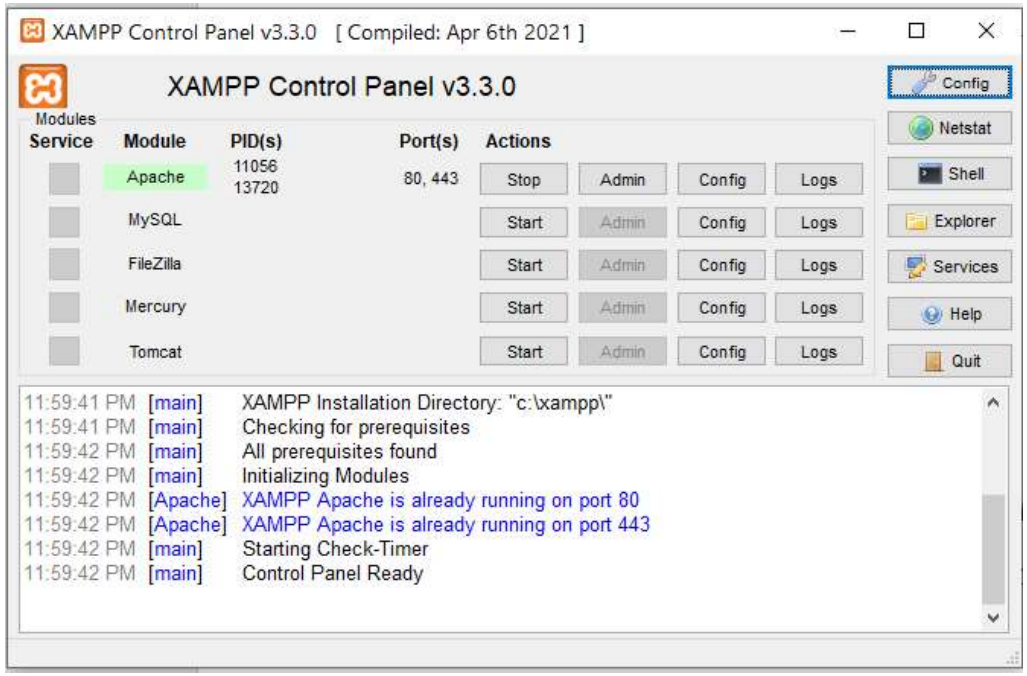
Place a small comment near any changes you make to the Apache `httpd.conf` file.

Note: A file cannot be copied directly to a virtual directory, because a virtual directory is only a name referring to a physical local directory.

Example 1: Simple PHP application

Step 1: install XAMPP follow the following link: https://sourceforge.net/projects/xampp/files/

Step 2: Run XAMPP Control Panel and start Apache server



Step 3: copy **clientForm.html** and **serverForm.php** to directory ***c:\xampp\htdocs***

Step 4: open Internet browser and type in ***http://localhost/clientForm.htm***