Introduction to Internet and Web Applications

Internet & World Wide Web How to Program, 5/e

Modified by Alan Wang

Objectives

- Get familiar with the terms used in web related technologies
- Web development roadmap

Introduction

- The Internet and web programming technologies are designed to be ***portable***, allowing you to design web pages and applications that run across an enormous range of Internet-enabled devices.
- *Client-side programming*: technologies are used to build web pages and applications that are run on the *client* (i.e., in the browser on the user's device).
- *Server-side programming*: the applications that respond to requests from client-side web applications, such as searching the Internet, checking your bank-account balance, ordering a book from Amazon, and ordering concert tickets.

Web Basics

- In its simplest form, a ***web page*** is an HTML (HyperText Markup Language) document (with the extension `.html` or `.htm`) that describes to a web browser the document's content and structure.

Hyperlinks

- HTML documents normally contain hyperlinks, which, when clicked, load a specified web document.
- Both images and text may be hyperlinked.
- When the user clicks a hyperlink, a web request is sent to a web server, which locates the requested web page and sends it back to the user's web browser.
- Similarly, the user can type the *address of a web page* into the browser's *address field* and press *Enter* to view the specified page.

Web Basics (cont.)

- Hyperlinks can reference other web pages, e-mail addresses, files and more.
- If a hyperlink's URL is in the form `mailto:`*emailAddress*, clicking the link loads your default e-mail program and opens a new message window addressed to the specified e-mail address.
- If a hyperlink references a file that the browser is incapable of displaying, the browser prepares to download the file, and generally prompts the user for information about how the file should be stored.

Web Basics (cont.)

URIs and URLs

- *URIs (Uniform Resource Identifiers)* identify resources on the Internet.
- URIs that start with `http://` are called *URLs (Uniform Resource Locators)*.

Parts of a URL

- A URL contains information that directs a browser to the resource that the user wishes to access.
- **Web servers** make such resources available to web clients.
- Popular web servers include Apache's HTTP Server and Microsoft's Internet Information Services (IIS).

Web Basics (cont.)

- Let's examine the components of a URL
  `http://www.deitel.com/books/downloads.html`
- "`http://`" indicates that the HyperText Transfer Protocol (HTTP) should be used to obtain the resource.
- Next in the URL is the server's fully qualified **hostname** (e.g., `www.deitel.com`)—the name of the web-server computer on which the resource resides.
- This computer is referred to as the **host**, because it houses and maintains resources.
- The hostname `www.deitel.com` is translated into an IP (Internet Protocol) address.
- An Internet Domain Name System (DNS) server maintains a database of hostnames and their corresponding IP addresses and performs the translations automatically.

Web Basics (cont.)

- From the client computer, the web browser sends an HTTP request to the server. The request (in its simplest form) is
  - `GET /books/downloads.html HTTP/1.1`
- The word GET is an HTTP method indicating 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`).
- The client's request also contains some required and optional headers.

Web Basics (cont.)

- The server first sends a line of text that indicates the HTTP version, followed by a numeric code and a phrase describing the status of the transaction. For example,
  - `"HTTP/1.1 200 OK"` indicates success, whereas
  - `"HTTP/1.1 404 Not found"` informs the client that the web server could not locate the requested resource.

HTTP Headers

- Next, the server sends one or more HTTP headers, which provide additional information about the data that will be sent.
- In this case, the server is sending an HTML5 text document, so one HTTP header for this example would read:
  - `Content-type: text/html`

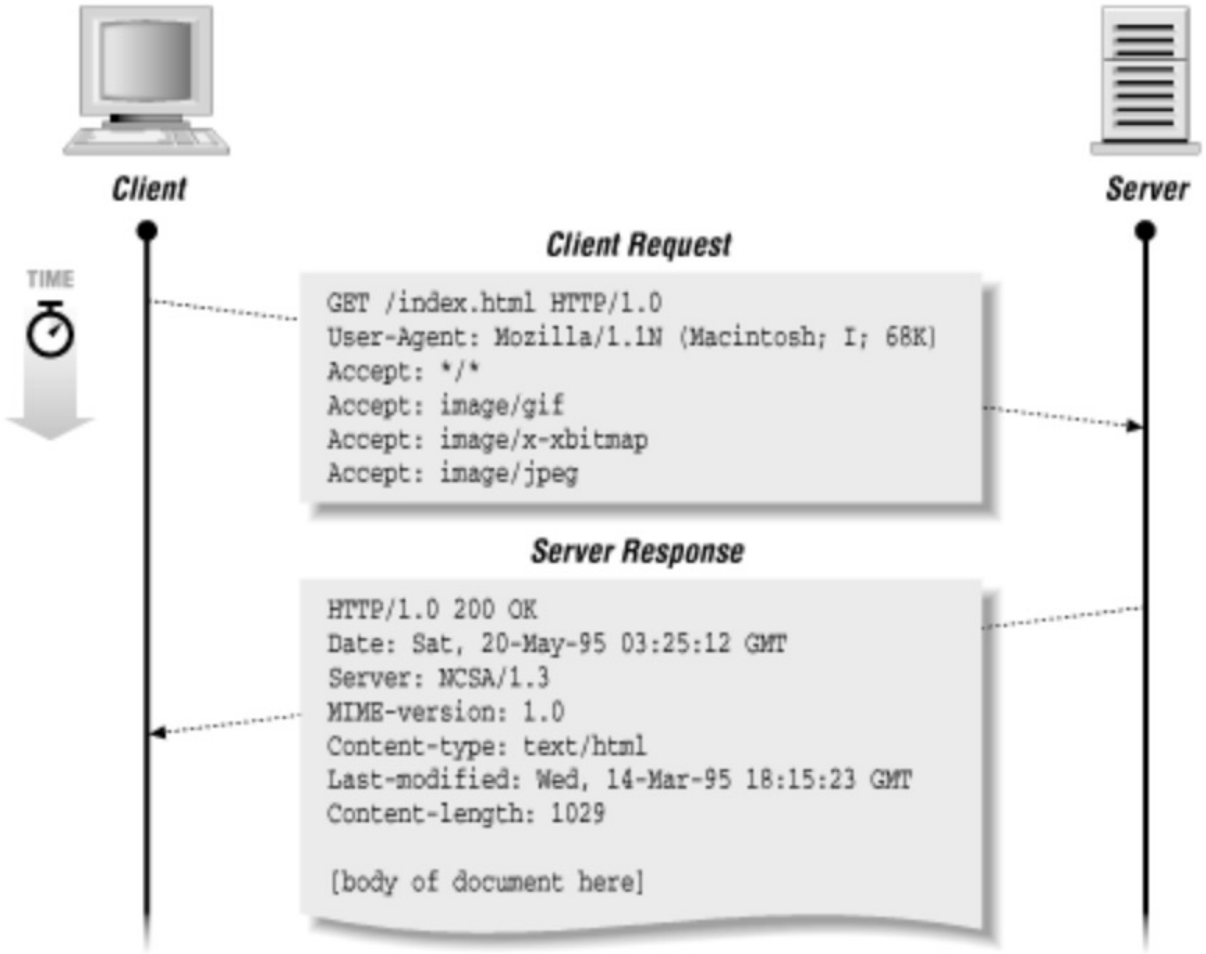
Web Basics (cont.)

- "text/html" specifies the Multipurpose Internet Mail Extensions (MIME) type of the content that the server is transmitting to the browser.
- The MIME standard specifies data formats, which programs can use to interpret data correctly.
- For example, the MIME type `text/plain` indicates that the sent information is text that can be displayed directly.
- Similarly, 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.

Web Basics (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.
- Finally, the server sends the contents of the requested document (`downloads.html`).
- The client-side browser then renders (or displays) the document, which may involve additional HTTP requests to obtain associated CSS and images.

An Example of HTTP Request and Response



Web Basics (cont.)

HTTP GET and POST Requests

- The two most common HTTP request types (also known as request methods) are GET and POST.
- **A GET request typically gets (or retrieves) information from a server**, such as an HTML document, an image or search results based on a user-submitted search term.
- **A POST request typically posts (or sends) data to a server.**
- Both request types can send data from the client to a server (check out: Compare GET vs. POST, https://www.w3schools.com/tags/ref_httpmethods.asp)

Web Basics (cont.)

How a GET request works

- A `GET` request appends data to the URL, e.g., `www.google.com/search?q=deitel`.
- In this case `search` is the name of Google's server-side form handler, `q` is the name of a variable in Google's search form and `deitel` is the search term.
- The ? in the preceding URL separates the query string from the rest of the URL in a 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 (&).
- The server uses data passed in a query string to retrieve an appropriate resource from the server.
- The server then sends a response to the client. A `GET` request may be initiated by submitting an HTML 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.

Web Basics (cont.)

How a POST request works:

- A `POST` request sends form data as part of the HTTP message, not as part of the URL.
- A `GET` request typically limits the query string (i.e., everything to the right of the ?) to a specific number of characters, so it's often necessary to send large amounts of information using the `post` method.
- The `POST` method is also sometimes preferred because it hides the submitted data from the user by embedding it in an HTTP message.
- If a form submits several hidden input values along with user-submitted data, the `POST` method might generate a URL like `www.searchengine.com/search`.
- The form data still reaches the server and is processed in a similar fashion to a `GET` request, but the user does not see the exact information sent.

HTTP Versions

- HTTP 1/2/3 (not required content)

https://youtu.be/a-sBfyiXysI

Web Basics (cont.)

Browser Caching

- Browsers often cache (save on disk) recently viewed web pages for quick reloading.
- If there are no changes between the version stored in the cache and the current version on the web, this speeds up your browsing experience.
- An HTTP response can indicate the length of time for which the content remains "fresh."
- If this amount of time has not been reached, the browser can avoid another request to the server. If not, the browser loads the document from the cache.
- Similarly, there's also the "not modified" HTTP response, indicating that the file content has not changed since it was last requested (which is information that's send in the request).
- Browsers typically do not cache the server's response to a POST request, because the next POST might not return the same result.
- Clear your browser cache if the browser fails to sync with the server

Multitier Application Architecture

- Web applications are often multitier applications (sometimes referred to as *n*-tier applications) that divide functionality into separate tiers (i.e., logical groupings of functionality).
- Figure 1.10 presents the basic structure of a three-tier web-based application.

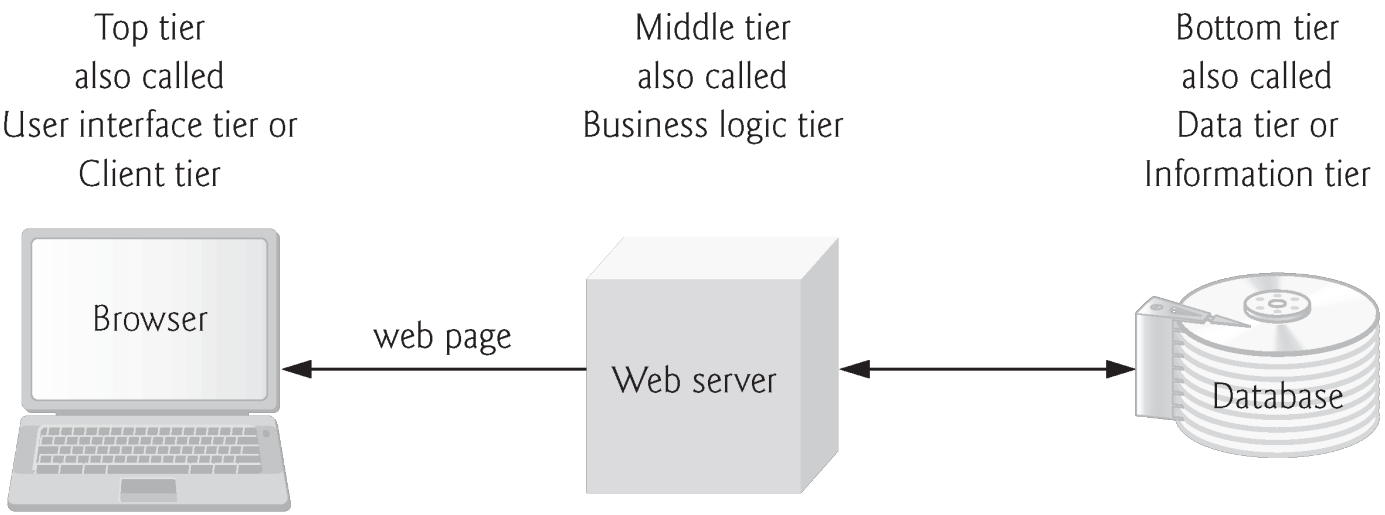
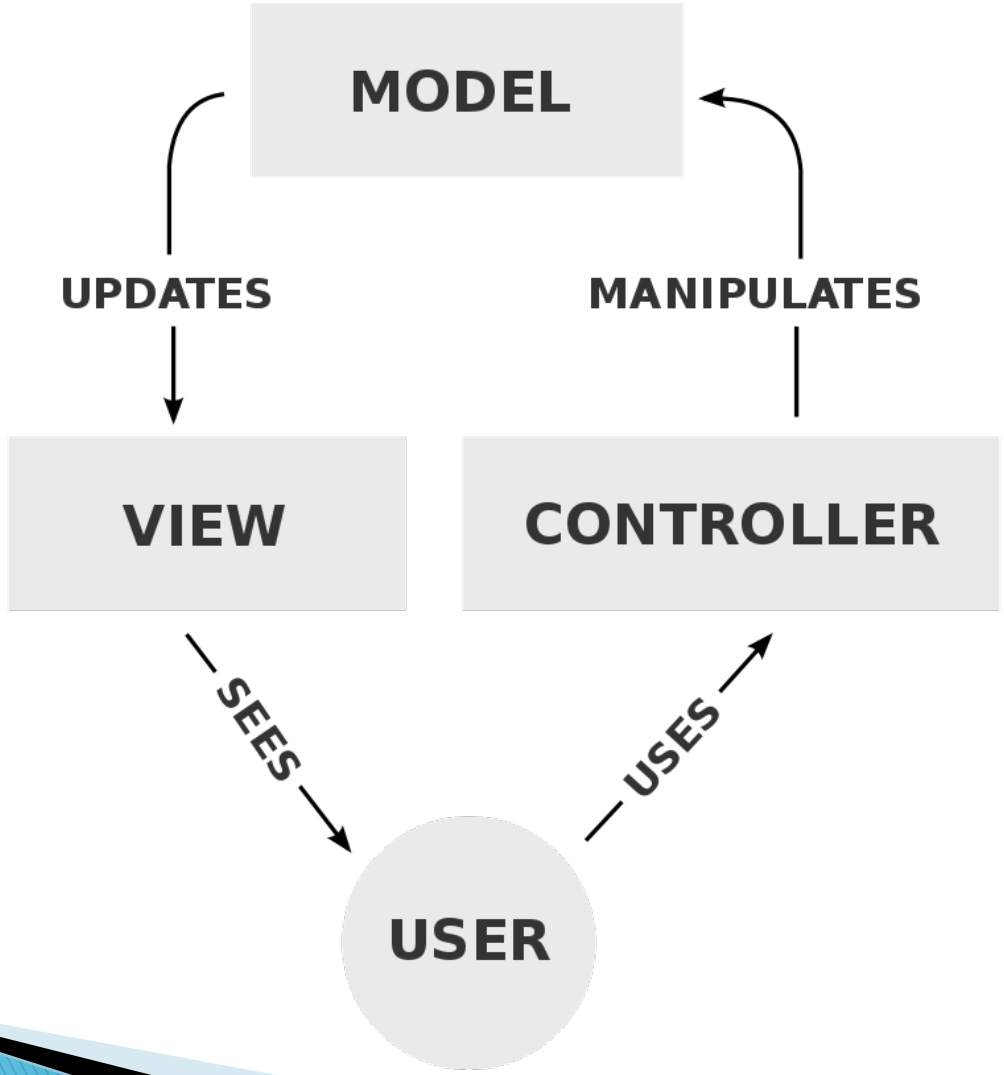


Fig. 1.10 | Three-tier architecture.

Multitier Application Architecture (cont.)

- The bottom tier (also called the data tier or the information tier) maintains the application's data, typically in a relational database management system (RDBMS).
- The middle tier acts as an intermediary between data in the information tier and the application's clients.
  - Controller logic processes client requests (such as requests to view a product catalog) and retrieves data from the database.
  - Business logic enforces business rules and ensures that data is reliable before the application updates a database or presents data to users.
  - Presentation logic then processes data from the information tier and presents the content to the client.

Model–View–Controller (MVC)



Model-View-Controller (MVC)

- Model: Represents application data and business rules that govern data accessing and updates
- View: Renders the user interface
- Controller: Interprets user actions and events and maps them into actions in the model or the view

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, and to add client/server communication between a browser and a web server.
- It uses computing resources of the client
- Client-side scripting limitations:
  - browser dependency
  - Restricted from arbitrarily accessing the local hardware and file system for security reasons.
  - Source code can be viewed by the client.
  - Sensitive information should not be on the client side.
  - Operations on the client can open web applications to security issues.

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

- Programmers have more flexibility with server-side scripts, which often generate custom responses for clients.
- Server-side scripting languages have a wider range of programmatic capabilities than their client-side equivalents.
- Limitations:
  - Consume resources on the server.
  - Less efficient than client-side script.
  - Increases Internet traffic.

World Wide Web Consortium (W3C)

- In October 1994, Tim Berners-Lee founded an organization—the World Wide Web Consortium (W3C)—devoted to developing nonproprietary, interoperable technologies for the World Wide Web.
- One of the W3C's primary goals is to make the web universally *accessible*—regardless of disability, language or culture.
- The W3C is also a standards organization.
- Web technologies standardized by the W3C are called Recommendations.
- Current and forthcoming W3C Recommendations include the HyperText Markup Language 5 (HTML5), Cascading Style Sheets 3 (CSS3) and the Extensible Markup Language (XML).

2022 Web Developer Roadmap

- Source:

https://roadmap.sh/