MIT 3107 ADVANCED INTERNET TECHNOLOGIES

Chapter 3

Web Servers

Learning Outcomes

- By the end of this chapter, the learner should be able to:
  - List commonly used Web servers.
  - Describe the functions of a Web Server.
  - Install and configure a Web Server.
  - Access a local and remote Web Server.

Introduction

- Communication between the client and server is essentially a series of requests and responses.
- This is the same for all protocols that are used to access the Internet, be it a Web protocol, FTP or email and other messaging systems.

Introduction

- The actual protocol name that is used for data exchange between the client and Web server is called HTTP (HyperText Transfer Protocol), and is the reason that all Web addresses start with the http://stanza.

Introduction

- HTTP is a mechanism for requesting services of a server and displaying the results of those service requests.

Introduction

- The response to this HTTP request then contains a status code, and then the data that represents the page content, and links to various artifacts such as images and style sheets that are external to the page itself.

Introduction

- It all happens asynchronously—the Web client first is given the page and is expected to request any of the additional items separately.
- The data flow is more or less all downstream towards the client.

Introduction

- There is more data received than submitted.
- However, the HTTP protocol allows a user agent (browser, client, and so on) to send information to the server as well, in a structured manner.

Web Servers

- The term Web server, can refer to either the hardware (the computer) or the software (the computer application) that helps to deliver web content that can be accessed through the Internet.

Web Servers

- A Web server is a specialized software that responds to client requests (typically from a Web browser) by providing resources such as HTML, XHTML documents.
  - E.g. when users enter a URL address such as www.jkuat.ac.ke, into a Web browser, they are requesting a specific document from a Web Server.

Web Servers

- The most common use of web servers is to host websites, but there are other uses such as gaming, data storage, running enterprise applications, handling email, FTP, or other web uses.

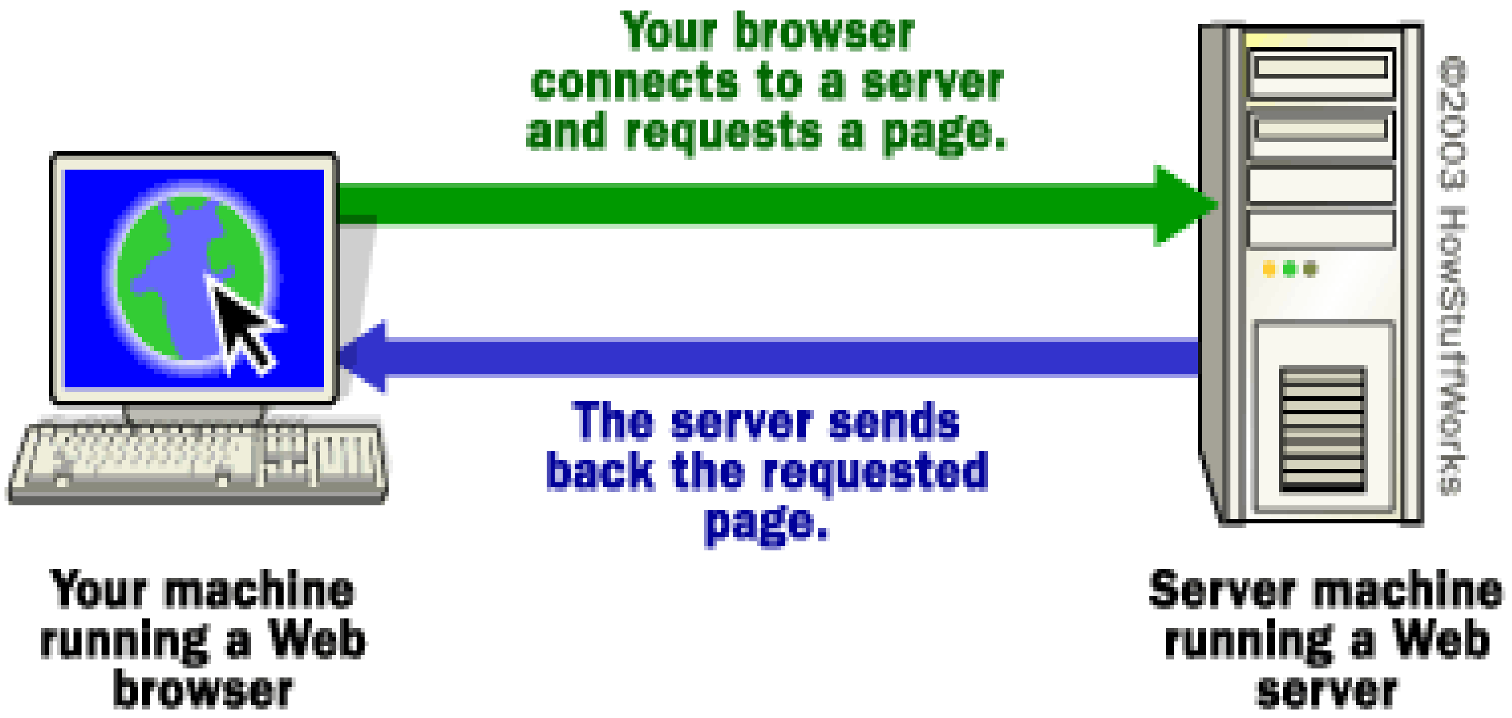
Web Services

- Web services refer to a set of loosely coupled software components that exchange information with each other using universal Web communication standards and languages.

Web Services

- They can exchange information between two different systems regardless of the operating systems or programming languages on which the systems are based.

Web Server Communication


Your browser connects to a server and requests a page.
The server sends back the requested page.
Your machine running a Web browser
Server machine running a Web server
©2003 HowStuffWorks

Web Servers

•Every Web server has an IP address and possibly a domain name.

•For example, if you enter the URL *http://www.pcwebopedia.com/index.html* in your browser, this sends a request to the Web server whose domain name is *pcwebopedia.com*.

Web Servers

- The server then fetches the page named index.html and sends it to your browser.
- The returned content may either be static (comes from an existing file) or Dynamic (dynamically generated by some other program/script called by the Web server).

Web Servers

- 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.
- During this interaction, the Web server and the client communicate using the platform-independent HTTP, a protocol for transferring requests and files over the Internet between Web servers and Web browsers.

Web Servers

- Examples:
  - Apache Web Server.
  - Microsoft Internet Information Services (MIIS).
  - Sun Java Web Server from Sun Microsystems
  - Zeus Web Server from Zeus Technology

HTTP Request Types

- There are two most common HTTP request types (also called request methods):
  - Get.
  - Post

HTTP Request Types

- Technically, the difference is that a GET request contains data that is part of the URL, whereas a POST request sends the data as part of an independent message.

HTTP Request Types: Get

- A get request typically gets (or retrieves) information from a server.
- Common uses of a get request are to:
  - Retrieve an HTML document or an image.
  - Fetch search results based on a user-submitted search term.

HTTP Request Types: Post

- A 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
  - Send data from a form that gathers user input.

HTTP Request Types

- An HTTP request often posts data to a server-side form handler that processes the data.
- E.g. when a user performs a search or participates in a Web-based survey, the Web server receives the information specified in the HTML or XHTML form as part of the request.

HTTP Request Types

- Get requests and post requests can both be used to send form data to a Web server, yet each type sends the information differently.

HTTP Request Types: Get

- A get requests sends information to the server as part of the URL (E.g. www.search-ending.com/search?name=value), where:
  - Search is the name of a server-side form handler,
  - Name is the name of a variable in an HTML or XHTML form and value is the value assigned to that variable.

HTTP Request Types: Get

- Notice the ? In the preceding URL.
  - 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 equal sign (=).If more than one name/value pair is submitted, each pair is separated by an ampersand (&).

HTTP Request Types: Get

- The server uses data passed in a query string to retrieve an appropriate source from the server.
- The server then sends a response to the client.

HTTP Request Types: Get

- A get request may be initiated by submitting an HTML form whose method attribute is set to “get”.

HTTP Request Types: Get

- Suppose a search engine’s Web page contains a form element with the attributes method=“get” and action = “search”, and that includes a text field named “SearchTerm”.

HTTP Request Types: Get

- A search for the term “jkuat” might send a get request for the URL www.searchengine.com/search?SearchTerm=jkuat.

HTTP Request Types: Get

- The server would then perform a search for the value jkuat and return a Web page containing the search results to the user

HTTP Request Types: Post

- A post request is specified in an HTML or XHTML form by the method “post”.
- The post method sends form data as an HTTP message, not as part of the URL.

HTTP Request Types: Post

- Because a get request limits the query string (i.e. everything to the right of the ?) to 2048 characters, it is often necessary to send large pieces of information using the post method.

HTTP Request Types: Post

- The *post* method is also sometimes preferred because it hides the submitted data from the user by embedding it in an HTTP message.

HTTP Request Types: Post

- If a form submits several hidden input values along with user-submitted data, the post method might generate the URL www.searchengine.com/search.

HTTP Request Types: Post

- 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 Request Types: Post

- The data sent in a *post request* is not part of the URL and cannot easily be seen by the user.
- Forms that contain many fields are submitted most often by a post request.
- Sensitive form fields, such as *passwords*, usually are sent using this request type.

HTTP Request Types: Post

- Browsers often cache (save on disk) Web pages so they can quickly reload the pages.
- If there are no changes between the last version stored in the cache and the current version on the Web, this helps speed up browsing experience.

HTTP Request Types: Post

- The browser first asks the server if the document has changed or expired since the date the file was cached.
- If not, the browser loads the document from the cache.

HTTP Request Types: Post

- Thus the browser minimizes the amount of data that must be downloaded for a user to view a Web page.
- Browsers typically do not cache the server’s response to a post request because the next post might not return the same results.

HTTP Request Types: Post

- E.g. in a survey, many users could visit the same Web page and respond to a question.
- The survey results could then be displayed for the user.

HTTP Request Types: Post

- Each new response changes the overall results of the survey.
- When you use a Web-based search engine, the browser normally supplies the information you specify in an HTML form to the search engine with a get request.
- The search engine performs the search, then returns results to you as a Web page.

HTTP Request Types: Post

- Such pages are often cached by the browser in case you perform the same search again.
- As with *post requests, get requests* can supply parameters as part of the request to the Web server.

System Architecture

- A Web server is part of a multi-tier application, sometimes referred to as an n-tier application.
- Multi-tier application divide functionality into separate tiers (i.e. logical groupings of functionality).
- Tiers can be located on the same computer or on separate computers.

Scripting Languages

- A scripting language is a form of programming language that is usually interpreted rather than compiled.
- Conventional programs are converted permanently into executable files before they are run.

Scripting Languages

- In contrast, programs in scripting language are interpreted one command at a time.
- Scripting languages are often written to facilitate enhanced features of Web sites.

Scripting Languages

- These features are processed on the server but the script in a specific page runs on the user's browser.

Scripting Languages

- In most cases, it is easier to write the code in a scripting language than in a compiled language.
- However, scripting languages are slower because the instructions are not handled solely by the basic instruction processor.

Scripting Languages

- Scripting languages allow rapid development and can communicate easily with programs written in other languages.
- Scripting languages can be used to create specialized GUIs (graphical user interfaces) and forms that enhance the convenience of search engines, Web-based e-mail and e-commerce.

Scripting Languages

- Many Web sites require that the user's browser be set to run scripts to take advantage of all the features of the site.
- In some cases, Web sites are practically useless unless the user's computer is set to run programs locally in a scripting language.

Common Scripting Languages

- JavaScript
- JScript
- Perl
- PHP
- Python
- VBScript

Client-Side Scripting and Server-Side Scripting

- Web service is a kind of client / server process
- Programming for providing Web service can also be divided into
  - Client-side programming: to define the operation to be performed on the client's machine
  - Server-side programming: to define the operation to be performed on the server

Client-Side Scripting and Server-Side Scripting

- Client-sides scripting:
  - Validates user input,
  - Accesses the browser and
  - Enhances Web pages with dynamic HTML.
- Client-Side validation reduces the number of requests that need to be passed to the server.

Client-Side Scripting and Server-Side Scripting

- Interactivity allows users to make decisions, click buttons, play games thereby making the Web-site experience more interesting.
- Client-Side scripts can access the browser, use features specific to the browser and manipulate browser documents.

Client-Side Scripting and Server-Side Scripting

- Client-Side scripting does not have limitations, such as browser dependency; the browser or scripting host must support the scripting language.
- Another issue is that the Client-Side scripts are viewable to the client (e.g. by using the View menu's source option in the browser).

Client-Side Scripting and Server-Side Scripting

- Some Web developers to not advocate this because users potentially can view proprietary scripting code.
- Sensitive information, such as passwords or other personally-identifiable data, should not be stored or validated on the client.

Client-Side Scripting and Server-Side Scripting

- To conserve server resources and minimize Internet traffic and delays, perform as much processing as possible on the client side.
- Programmers have greater flexibility when using Server-Side scripts.
- Scripts executed on the server often generate custom responses for clients.

Client-Side Scripting and Server-Side Scripting

- E.g. a client might connect to an airline’s Web server and request a list of all flights from Nairobi to Eldoret between two dates.
- The server queries the database, dynamically generates HTML content containing the flight list and sends the HTML document to the client.

Client-Side Scripting and Server-Side Scripting

- This technology allows clients to obtain the most current flight information from the database by connecting to an airline’s Web server.
- Server-Side scripting languages have a wider range of programmatic capabilities than their Client-Side equivalents.

Client-Side Scripting and Server-Side Scripting

- E.g. Server-Side scripts often can access the server's file directory structure, whereas client-side scripts cannot access the client's directories.
- Server-Side scripts also have access to server side software that extends server functionality.

Client-Side Scripting and Server-Side Scripting

- Microsoft Web servers use ISAPI (Internet Server Application Program Interface) Extensions and Apache Web Servers use modules.
- Components and modules range from programming language support to counting the number of Web page hits.

Accessing Web Servers

- To request documents from Web servers, users must know the machine names (called host names) on which the Web serves software resides.
- Users can request documents from local Web Servers (i.e. ones residing on user’s machines) or remote Web Servers (i.e. ones residing on different machines).

Accessing Web Servers: Local

- Local Web servers can be accessed in two ways:
  - Through the machine name or
  - Through localhost (a host name that references the local machine).

Accessing Web Servers: Remote

- A remote Web server referenced by a domain name (e.g. yahoo) and an Internet Protocol (IP) address also can serve documents.
- A domain name represents a group of hosts on the Internet.
- It combines with a hostname (e.g. www) and a top-level domain (TLD), such as com, org or edu, to form a fully qualified hostname.

Accessing Web Servers: Remote

- This provides a user friendly way to identify a site on the Internet.
- In a fully qualified hostname, the TLD often describes the types of organizations that owns the domain name.
- com usually refers to a commercial business, *org* to a non-profit organization.

Accessing Web Servers

- Each fully qualified host name is assigned a unique address called IP address, which is used by computers to locate computers on the Internet.
- A Domain Name System (DNS) server , a computer that maintains a database of host names and their corresponding IP addresses, translates the fully qualified host name to an IP address.

Accessing Web Servers

- The translation operation is called a DNS lookup.
- The IP address 127.0.0.1 always refers to the local Web server (localhost) running on the computer from where it is accessed.

Microsoft Internet Information Services (MIIS)

- Is an enterprise-level Web server that is included in several versions of Windows.
- Installing IIS on a machine allows that computer to serve documents.
- For instructions on how to install IIS go to www.deitel.com/books/iw3HTP3/index.html

Apache Web Server

- This is maintained by the Apache Software Foundation.
- Is currently the most popular Web Server because of its:
  - Stability.
  - Efficiency.
  - Portability.
  - Security and Small size.

Apache Web Server

- It is an open-source product (i.e. software that can be freely obtained and customized) that runs on Unix, Linux, Mac OS X, Windows and numerous other platforms.
- Obtain the Apache Web Server for a variety of platforms from http.apache.org/download.cgi.

Security Issues

- When it comes to Web applications, security means different things to different people, depending on the type of interaction they have with the application itself:
  - The customer/user of the Web site
  - Server administrator and
  - Web site developer

The customer/user of the Web site

- A customer will probably define security in terms of trust i.e. customers want to feel as if the confidential information they provide is being entered via a mechanism (such as a Web form) that is both fault-tolerant and immune from unauthorized access.

The customer/user of the Web site

- Moreover, customers want to feel that after they've entered their information (their order), the process governing the transaction of processing that order is also secure and fault-tolerant.

The server administrator

- The server administrator (the person responsible for maintaining the Web server on which the Web site resides) also has his/her own unique perspective on security.

The server administrator

- Although the server administrator might have concerns about the actual code used to drive the Web site, she/he'll probably be more concerned with securing the server itself, as well as protecting Web server against external, unauthorized physical access and environmental issues.

The server administrator

- More than likely, you will be using a third-party service to host your PHP- and MySQL-enabled Web sites, so you can leave the server administration issues to the server administrators.

The server administrator

- Still, you should be aware of the special security issues these third party service providers must address.
- Also, you should be aware of how you can take advantage of Web server security in your own programming.

The Web site developer

- More than likely, you fall into the role of Web site developer.
- And, in many ways, your role in the larger security equation is the most complex one of all.

The Web site developer

- As the developer, you’ll often find yourself on the ground floor of the security issue.

The Web site developer

- However, rather than view this as a negative, you should relish the opportunity you have as a developer to write secure code that takes advantage of server security and to write code that's smart enough to remain secure (and, critically, communicate a feeling of security) when there are system errors.

The Web site developer

- To be sure, it’s a big responsibility, but one that you should and can effectively address with some general awareness of the larger security picture as well as effective planning.

Security issues

- These different types of security roles will overlap (with the possible exception of the customers, who probably won’t care about the underlying code of the Web site they’re using).

Security issues

- From direct experience, security is an enormously complex issue and is by no means a one-person job.
- So although you might not be an expert in all aspects of the "big security picture," you can and should be at least aware of the big picture.

Security issues

- Depending on the type of Web site you are developing, there are some common security threats you need to be aware of.

Common Security Threats

– Loss of data.

– Exposing confidential data.

– Having data accidentally modified or changed.

– Malicious external attacks.

Loss of data

- You might say to yourself, “Well, if I lose my customer’s data, that’s better than having it stolen, right?”
- You could say that, but it’s definitely not a philosophy to bet the bank on.

Loss of data

- Lost data can have significant security ramifications because such data might be required to verify (for example) the existence of a previous order or verify credentials of an individual requesting access to sensitive information.
- Protecting against data loss is often a data backup issue

Exposing confidential data

- Being the one responsible for exposing confidential/sensitive data is a very, very bad position to be in — sort of the electronic equivalent of being caught with your pants down in a very crowded space.

Exposing confidential data

- Not only can it rattle your confidence in your programming skills, it can also, depending on the severity of the issue, cost you your job.

Having data accidentally modified or changed

- Yet another security threat you need to be aware of in your application development is the accidental modification of data.
- Although having critical data erroneously modified doesn't qualify as suffering from a loss or exposure of data, it can be worse than simply losing the data.

Having data accidentally modified or changed

- Imagine that you've developed a MySQL database that tracks results of a critical drug trial for a new medicine that is coming to market.

Having data accidentally modified or changed

- Within this database, you store various attributes of the trial participants, including their gender, age, reactions to the drug at various stages of the trial, and so on.

Having data accidentally modified or changed

- Now, imagine that one day — because of a problem with the PHP code you've written — a database query is executed that updates the age of all the male patients in the database to the same value.

Having data accidentally modified or changed

- Perhaps this wouldn’t be a big problem if the database contained only a few records.
- But what if the database contained several thousand records?
- Correcting a large modification of data would require an equally large amount of time and therefore risk the successful progression of the drug trial.

Malicious external attacks

- A malicious outside attack is when an unauthorized intruder uses a worm, virus, or another piece of malicious code in an attempt to gain access to your system and (as is usually the case) snoop around/steal/modify/delete your confidential information.

Malicious external attacks

- Many of the issues involved with this security threat will fall to the server administrator because he or she must ensure that the Web servers are current with software patches as well as physically protected.

Malicious external attacks

- However, a large part of addressing this security threat also falls to you, the developer i.e. poorly written code can allow an easy path for intruders.

Repudiation

- Repudiation occurs when a party involved in a transaction denies having taken part.
- E-commerce examples might include a person ordering goods off a Web site and then denying having authorized the charge on his credit card, or a person agreeing to something in email and then claiming that somebody else forged the email.

Repudiation

- Ideally, financial transactions should provide the peace of mind of nonrepudiation to both parties.
- Neither party could deny their part in a transaction, or, more precisely, both parties could conclusively prove the actions of the other to a third party, such as a court.

Balancing Usability, Performance, Cost, and Security

- By its very nature, the Web is risky.
- It is designed to allow numerous anonymous users to request services from your machines.
- Most of those requests will be perfectly legitimate requests for Web pages, but connecting your machines to the Internet will allow people to attempt other types of connections.

Balancing Usability, Performance, Cost, and Security

- Although it can be tempting to assume that the highest possible level of security is appropriate, this is rarely the case.
- If you wanted to be really secure, you would keep all your computers turned off, disconnected from all networks, in a locked safe.

Balancing Usability, Performance, Cost, and Security

- In order to make your computers available and usable, some relaxation of security is required.
- There is a trade-off to be made between security, usability, cost, and performance.

Balancing Usability, Performance, Cost, and Security

- Making a service more secure can reduce usability by, for instance, limiting what people can do or requiring them to identify themselves.
- Increasing security can also reduce the level of performance of your machines.

Balancing Usability, Performance, Cost, and Security

- Running software to make your system more secure, such as encryption, intrusion detection systems, virus scanners, and extensive logging uses resources.
- It takes a lot more processing power to provide an encrypted session, such as an Secure Socket Layer (SSL) connection to a Web site, than to provide a normal one.

Balancing Usability, Performance, Cost, and Security

- These performance losses can be countered by spending more money on faster machines or hardware specifically designed for encryption.
- You can view performance, usability, cost, and security as competing goals.

Balancing Usability, Performance, Cost, and Security

- You need to examine the trade-offs required and make sensible decisions to come up with a compromise.

Balancing Usability, Performance, Cost, and Security

- Depending on the value of your information, your budget, how many visitors you expect to serve, and what obstacles you think legitimate users will be willing to put up with, you can come up with a compromise position.

Summary

- Considerations in choosing a Web server include how well it works with the operating system and other servers, its ability to handle server-side programming, security characteristics, and publishing, search engine, and site building tools that may come with it.

End of Chapter 3