Fundamentals

The lives of most inhabitants of industrialized countries, as well as many in unindustrialized countries, have been changed forever by the advent of the World Wide Web. Although this transformation has had some downsides—for example, easier access to pornography and gambling and the ease with which people with destructive ideas can propagate those ideas to others—on balance, the changes have been enormously positive. Many use the Internet and the World Wide Web daily, communicating with friends, relatives, and business associates through e-mail and social networking sites, shopping for virtually anything that can be purchased anywhere, and digging up a limitless variety and amount of information, from movie theater show times, to hotel room prices in cities halfway around the world, to the history and characteristics of the culture of some small and obscure society. In recent years, social networking has been used effectively to organize social and political demonstrations, and even revolutions. Constructing the software and data that provide all of this information requires knowledge of several different technologies, such as markup languages and meta-markup

languages, as well as programming skills in a myriad of different programming languages, some specific to the World Wide Web and some designed for general-purpose computing. This text is meant to provide the required background and a basis for acquiring the knowledge and skills necessary to build the World Wide Web sites that provide both the information users want and the advertising that pays for its presentation.

This chapter lays the groundwork. It begins with introductions to, and some history of, the Internet and the World Wide Web. Then, it discusses the purposes and some of the characteristics of Web browsers and servers. Next, it describes uniform resource locators (URLs), which specify addresses of resources available on the Web. Following this, it introduces Multipurpose Internet Mail Extensions, which define types and file name extensions for files with different kinds of contents. Next, it discusses the Hypertext Transfer Protocol (HTTP), which provides the communication interface for connections between browsers and Web servers. Finally, this chapter gives brief overviews of some of the tools commonly used by Web programmers, including HTML, XML, JavaScript, Flash, Servlets, JSP, JSF, ASP.NET, PHP, Ruby, Rails, and Ajax.

1 A Brief Introduction to the Internet

Many topics discussed in this text are related to the Internet. Therefore, we begin with a quick introduction to the Internet itself.

1.1 Origins

In the 1960s, the U.S. Department of Defense (DoD) became interested in developing a new large-scale computer network. The purposes of this network were communications, program sharing, and remote computer access for researchers working on defense-related contracts. One fundamental requirement was that the network be sufficiently robust so that even if some network nodes were lost to sabotage, war, or some more benign cause, the network would continue to function. The DoD's Advanced Research Projects Agency (ARPA)[1] funded the construction of the first such network, which connected about a dozen ARPA-funded research laboratories and universities. The first node of this network was established at UCLA in 1969.

Because it was funded by ARPA, the network was named ARPAnet. Despite the initial intentions, the primary early use of ARPAnet was simple text-based communications through electronic mail. Because ARPAnet was available only to laboratories and universities that conducted ARPA-funded research, the great

1. ARPA was renamed Defense Advanced Research Projects Agency (DARPA) in 1972.

majority of educational institutions were not connected. As a result, a number of other networks were developed during the late 1970s and early 1980s, with BITNET and CSNET among them. BITNET, which is an acronym for *B*ecause *I*t's *T*ime *Net*work, began at the City University of New York. It was built initially to provide electronic mail and file transfers. CSNET, which is an acronym for *C*omputer *S*cience *Net*work, connected the University of Delaware, Purdue University, the University of Wisconsin, the RAND Corporation, and Bolt, Beranek, and Newman (a research company in Cambridge, Massachusetts). Its initial purpose was to provide electronic mail. For a variety of reasons, neither BITNET nor CSNET became a widely used national network.

A new national network, NSFnet, was created in 1986. It was sponsored, of course, by the National Science Foundation (NSF). NSFnet initially connected the NSF-funded supercomputer centers that were at five universities. Soon after being established, it became available to other academic institutions and research laboratories. By 1990, NSFnet had replaced ARPAnet for most nonmilitary uses, and a wide variety of organizations had established nodes on the new network—by 1992, NSFnet connected more than 1 million computers around the world. In 1995, a small part of NSFnet returned to being a research network. The rest became known as the Internet, although this term was used much earlier for both ARPAnet and NSFnet.

1.2 What Is the Internet?

The Internet is a huge collection of computers connected in a communications network. These computers are of every imaginable size, configuration, and manufacturer. In fact, some of the devices connected to the Internet—such as plotters and printers—are not computers at all. The innovation that allows all of these diverse devices to communicate with each other is a single, low-level protocol named Transmission Control Protocol/Internet Protocol (TCP/IP). TCP/IP became the standard for computer network connections in 1982. It can be used directly to allow a program on one computer to communicate with a program on another computer via the Internet. In most cases, however, a higher-level protocol runs on top of TCP/IP. Nevertheless, it is TCP/IP that provides the low-level interface that allows most computers (and other devices) connected to the Internet to appear exactly the same.[2]

Rather than connecting every computer on the Internet directly to every other computer on the Internet, normally the individual computers in an organization are connected to each other in a local network. One node on this local network is physically connected to the Internet. So, the Internet is actually a network of networks, rather than a network of computers.

Obviously, all devices connected to the Internet must be uniquely identifiable.

2. TCP/IP is not the only communication protocol used by the Internet—User Datagram Protocol/Internet Protocol (UDP/IP) is an alternative that is used in some situations.

1.3 Internet Protocol Addresses

For people, Internet nodes are identified by names; for computers, they are identified by numeric addresses. This relationship exactly parallels the one between a variable name in a program, which is for people, and the variable's numeric memory address, which is for the machine.

The Internet Protocol (IP) address of a machine connected to the Internet is a unique 32-bit number. IP addresses usually are written (and thought of) as four 8-bit numbers, separated by periods. The four parts are separately used by Internet-routing computers to decide where a message must go next to get to its destination.

Organizations are assigned blocks of IPs, which they in turn assign to their machines that need Internet access—which now include virtually all computers. For example, a small organization may be assigned 256 IP addresses, such as `191.57.126.0` to `191.57.126.255`. Very large organizations, such as the Department of Defense, may be assigned 16 million IP addresses, which include IP addresses with one particular first 8-bit number, such as `12.0.0.0` to `12.255.255.255`.

Although people nearly always type domain names into their browsers, the IP works just as well. For example, the IP for United Airlines (`www.ual.com`) is `209.87.113.93`. So, if a browser is pointed at `http://209.87.113.93`, it will be connected to the United Airlines Web site.

In late 1998, a new IP standard, IPv6, was approved, although it still is not widely used. The most significant change was to expand the address size from 32 bits to 128 bits. This is a change that will soon be essential because the number of remaining unused IP addresses is diminishing rapidly.

1.4 Domain Names

Because people have difficulty dealing with and remembering numbers, machines on the Internet also have textual names. These names begin with the name of the host machine, followed by progressively larger enclosing collections of machines, called *domains*. There may be two, three, or more domain names. The first domain name, which appears immediately to the right of the host name, is the domain of which the host is a part. The second domain name gives the domain of which the first domain is a part. The last domain name identifies the type of organization in which the host resides, which is the largest domain in the site's name. For organizations in the United States, `edu` is the extension for educational institutions, `com` specifies a company, `gov` is used for the U.S. government, and `org` is used for many other kinds of organizations. In other countries, the largest domain is often an abbreviation for the country—for example, `se` is used for Sweden, and `kz` is used for Kazakhstan.

Consider this sample address:

```
movies.marxbros.comedy.com
```

Here, `movies` is the hostname and `marxbros` is `movies`'s local domain, which is a part of `comedy`'s domain, which is a part of the `com` domain. The hostname and all of the domain names are together called a *fully qualified domain name*.

Because IP addresses are the addresses used internally by the Internet, the fully qualified domain name of the destination for a message, which is what is given by a browser user, must be converted to an IP address before the message can be transmitted over the Internet to the destination. These conversions are done by software systems called *name servers*, which implement the Domain Name System (DNS). Name servers serve a collection of machines on the Internet and are operated by organizations that are responsible for the part of the Internet to which those machines are connected. All document requests from browsers are routed to the nearest name server. If the name server can convert the fully qualified domain name to an IP address, it does so. If it cannot, the name server sends the fully qualified domain name to another name server for conversion. Like IP addresses, fully qualified domain names must be unique. Figure 1 shows how fully qualified domain names requested by a browser are translated into IPs before they are routed to the appropriate Web server.

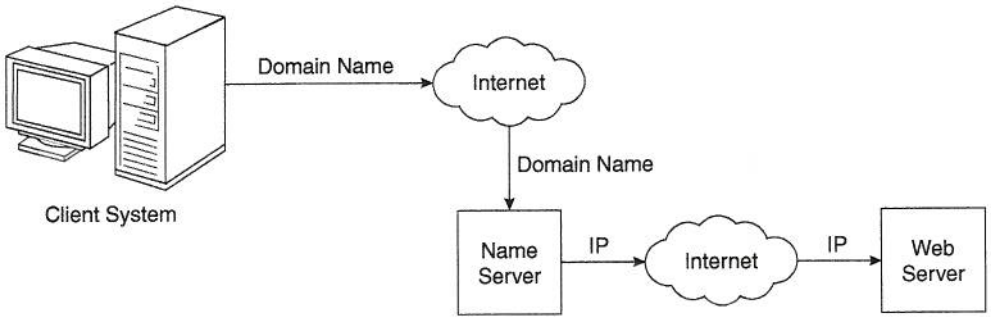

Figure 1 Domain name conversion

One way to determine the IP address of a Web site is by using `telnet` on the fully qualified domain name. This approach is illustrated in Section 7.1.

By the mid-1980s, a collection of different protocols that run on top of TCP/IP had been developed to support a variety of Internet uses. Among these protocols, the most common were `telnet`, which was developed to allow a user on one computer on the Internet to log onto and use another computer on the Internet; File Transfer Protocol (`ftp`), which was developed to transfer files among computers on the Internet; Usenet, which was developed to serve as an electronic bulletin board; and `mailto`, which was developed to allow messages to be sent from the user of one computer on the Internet to other users of other computers on the Internet.

This variety of protocols, each has its own user interface and each is useful only for the purpose for which it was designed, restricted the growth of the Internet. Users were required to learn all the different interfaces to gain all

the advantages of the Internet. Before long, however, a better approach was developed: the World Wide Web.

2 The World Wide Web

This section provides a brief introduction to the evolution of the World Wide Web.

2.1 Origins

In 1989, a small group of people led by Tim Berners-Lee at CERN (Conseil Européen pour la Recherche Nucléaire, or European Organization for Particle Physics) proposed a new protocol for the Internet, as well as a system of document access to use it.[3] The intent of this new system, which the group named the World Wide Web, was to allow scientists around the world to use the Internet to exchange documents describing their work.

The proposed new system was designed to allow a user anywhere on the Internet to search for and retrieve documents from databases on any number of different document-serving computers connected to the Internet. By late 1990, the basic ideas for the new system had been fully developed and implemented on a NeXT computer at CERN. In 1991, the system was ported to other computer platforms and released to the rest of the world.

For the form of its documents, the new system used *hypertext*, which is text with embedded links to text in other documents to allow nonsequential browsing of textual material. The idea of hypertext had been developed earlier and had appeared in Xerox's NoteCards and Apple's HyperCard in the mid-1980s.

From here on, we will refer to the World Wide Web simply as "the Web." The units of information on the Web have been referred to by several different names; among them, the most common are *pages*, *documents*, and *resources*. Perhaps the best of these is *documents*, although that seems to imply only text. *Pages* is widely used, but it is misleading in that Web units of information often have more than one of the kind of pages that make up printed media. There is some merit to calling these units *resources*, because that covers the possibility of nontextual information. This text will use *documents* and *pages* more or less interchangeably, but we prefer *documents* in most situations.

Documents are sometimes just text, usually with embedded links to other documents, but they often also include images, sound recordings, or other kinds of media. When a document contains nontextual information, it is called *hypermedia*.

In an abstract sense, the Web is a vast collection of documents, some of which are connected by links. These documents are accessed by Web browsers, introduced in Section 3, and are provided by Web servers, introduced in Section 4.

3. Although Berners-Lee's college degree (from Oxford) was in physics, his first stint at CERN was as a consulting software engineer. Berners-Lee was born and raised in London.

2.2 Web or Internet?

It is important to understand that the Internet and the Web are not the same thing. The *Internet* is a collection of computers and other devices connected by equipment that allows them to communicate with each other. The *Web* is a collection of software and protocols that has been installed on most, if not all, of the computers on the Internet. Some of these computers run Web servers, which provide documents, but most run Web clients, or browsers, which request documents from servers and display them to users. The Internet was quite useful before the Web was developed, and it is still useful without it. However, most users of the Internet now use it through the Web.

3 Web Browsers

When two computers communicate over some network, in many cases one acts as a client and the other as a server. The client initiates the communication, which is often a request for information stored on the server, which then sends that information back to the client. The Web, as well as many other systems, operates in this client-server configuration.

Documents provided by servers on the Web are requested by *browsers*, which are programs running on client machines. They are called browsers because they allow the user to browse the resources available on servers. The first browsers were text based—they were not capable of displaying graphic information, nor did they have a graphical user interface. This limitation effectively constrained the growth of the Web. In early 1993, things changed with the release of Mosaic, the first browser with a graphical user interface. Mosaic was developed at the National Center for Supercomputer Applications (NCSA) at the University of Illinois. Mosaic's interface provided convenient access to the Web for users who were neither scientists nor software developers. The first release of Mosaic ran on UNIX systems using the X Window system. By late 1993, versions of Mosaic for Apple Macintosh and Microsoft Windows systems had been released. Finally, users of the computers connected to the Internet around the world had a powerful way to access anything on the Web anywhere in the world. The result of this power and convenience was an explosive growth in Web usage.

A browser is a client on the Web because it initiates the communication with a server, which waits for a request from the client before doing anything. In the simplest case, a browser requests a static document from a server. The server locates the document among its servable documents and sends it to the browser, which displays it for the user. However, more complicated situations are common. For example, the server may provide a document that requests input from the user through the browser. After the user supplies the requested input, it is transmitted from the browser to the server, which may use the input to perform some computation and then return a new document to the browser to inform the user of the results of the computation. Sometimes a browser directly requests the execution of a program stored on the server. The output of the program is then returned to the browser.

Although the Web supports a variety of protocols, the most common one is the Hypertext Transfer Protocol (HTTP). HTTP provides a standard form of communication between browsers and Web servers. Section 7 presents an introduction to HTTP.

The most commonly used browsers are Microsoft Internet Explorer (IE), which runs only on PCs that use one of the Microsoft Windows operating systems,[4] Firefox, and Chrome. The latter two are available in versions for several different computing platforms, including Windows, Mac OS, and Linux. Several other browsers are available, including Opera and Apple's Safari. However, because the great majority of browsers now in use are either IE, Firefox, or Chrome, in this text we focus on them.

4 Web Servers

Web servers are programs that provide documents to requesting browsers. Servers are slave programs: They act only when requests are made to them by browsers running on other computers on the Internet.

The most commonly used Web servers are Apache, which has been implemented for a variety of computer platforms, and Microsoft's Internet Information Server (IIS), which runs under Windows operating systems. As of October 2011, there were over 150 million active Web hosts in operation,[5] about 65 percent of which were Apache, about 16 percent of which were IIS, and the remainder of which were spread thinly over several others. (The third-place server was `nginx` (pronounced "engine-x"), a product produced in Russia, with about 8 percent.)[6]

4.1 Web Server Operation

Although having clients and servers is a natural consequence of information distribution, this configuration offers some additional benefits for the Web. On the one hand, serving information does not take a great deal of time. On the other hand, displaying information on client screens is time consuming. Because Web servers need not be involved in this display process, they can handle many clients. So, it is both a natural and efficient division of labor to have a small number of servers provide documents to a large number of clients.

Web browsers initiate network communications with servers by sending them URLs (discussed in Section 5). A URL can specify one of two different things: the address of a data file stored on the server that is to be sent to the client, or a program stored on the server that the client wants executed and the output of the program returned to the client.

---

4. Actually, versions 4 and 5 of IE (IE4 and IE5) were also available for Macintosh computers, and IE4 was available for UNIX systems. However, later versions are available for Windows platforms only.

5. There were more than 500 million sites on line.

6. These statistics are from `http://www.netcraft.com`.

All the communications between a Web client and a Web server use the standard Web protocol, Hypertext Transfer Protocol (HTTP), which is discussed in Section 7.[7]

When a Web server begins execution, it informs the operating system under which it is running that it is now ready to accept incoming network connections through a specific port on the machine. While in this running state, the server runs as a background process in the operating system environment. A Web client, or browser, opens a network connection to a Web server, sends information requests and possibly data to the server, receives information from the server, and closes the connection. Of course, other machines exist between browsers and servers on the network—specifically, network routers and domain name servers. This section, however, focuses on just one part of Web communication: the server.

Simply put, the primary task of a Web server is to monitor a communications port on its host machine, accept HTTP commands through that port, and perform the operations specified by the commands. All HTTP commands include a URL, which includes the specification of a host server machine. When the URL is received, it is translated into either a file name (in which case the file is returned to the requesting client) or a program name (in which case the program is run and its output is sent to the requesting client). This process sounds pretty simple, but, as is the case in many other simple-sounding processes, there are a large number of complicating details.

All current Web servers have a common ancestry: the first two servers, developed at CERN in Europe and NCSA at the University of Illinois. Currently, the most common server configuration is Apache running on some version of UNIX.

4.2 General Server Characteristics

Most of the available servers share common characteristics, regardless of their origin or the platform on which they run. This section provides brief descriptions of some of these characteristics.

The file structure of a Web server has two separate directories. The root of one of these is called the *document root*. The file hierarchy that grows from the document root stores the Web documents to which the server has direct access and normally serves to clients. The root of the other directory is called the *server root*. This directory, along with its descendant directories, stores the server and its support software.

The files stored directly in the document root are those available to clients through top-level URLs. Typically, clients do not access the document root directly in URLs; rather, the server maps requested URLs to the document root, whose location is not known to clients. For example, suppose that the site name is `www.tunias.com` (not a real site—at least, not yet), which we will assume to be a UNIX-based system. Suppose further that the document root is named `topdocs` and is stored in the `/admin/web` directory, making its address `/admin/web/topdocs`. A request for a file from a client with the URL `http://www.tunias.com/petunias.html` will cause the server to search for the file with the file

7. Some of these communications use HTTPS, the secure version of HTTP.

path `/admin/web/topdocs/petunias.html`. Likewise, the URL `http://www.tunias.com/bulbs/tulips.html` will cause the server to search for the file with the address `/admin/web/topdocs/bulbs/tulips.html`.

Many servers allow part of the servable document collection to be stored outside the directory at the document root. The secondary areas from which documents can be served are called *virtual document trees*. For example, the original configuration of a server might have the server store all its servable documents from the primary system disk on the server machine. Later, the collection of servable documents might outgrow that disk, in which case part of the collection could be stored on a secondary disk. This secondary disk might reside on the server machine or on some other machine on a local area network. To support this arrangement, the server is configured to direct-request URLs with a particular file path to a storage area separate from the document-root directory. Sometimes files with different types of content, such as images, are stored outside the document root.

Early servers provided few services other than the basic process of returning requested files or the output of programs whose execution had been requested. The list of additional services has grown steadily over the years. Contemporary servers are large and complex systems that provide a wide variety of client services. Many servers can support more than one site on a computer, potentially reducing the cost of each site and making their maintenance more convenient. Such secondary hosts are called *virtual hosts*.

Some servers can serve documents that are in the document root of other machines on the Web; in this case, they are called *proxy servers*.

Although Web servers were originally designed to support only the HTTP protocol, many now support `ftp`, `gopher`, `news`, and `mailto`. In addition, nearly all Web servers can interact with database systems through server-side scripts.

4.3 Apache

Apache began as the NCSA server, `httpd`, with some added features. The name *Apache* has nothing to do with the Native American tribe of the same name. Rather, it came from the nature of its first version, which was *a patchy* version of the `httpd` server. As seen in the usage statistics given at the beginning of this section, Apache is the most widely used Web server. The primary reasons are as follows: It is both fast and reliable. Furthermore, it is open-source software, which means that it is free and is managed by a large team of volunteers, a process that efficiently and effectively maintains the system. Finally, it is one of the best available servers for Unix-based systems, which are the most popular for Web servers.

Apache is capable of providing a long list of services beyond the basic process of serving documents to clients. When Apache begins execution, it reads its configuration information from a file and sets its parameters to operate accordingly. A new copy of Apache includes default configuration information for a "typical" operation. The site manager modifies this configuration information to fit his or her particular needs and tastes.

For historical reasons, there are three configuration files in an Apache server: `httpd.conf`, `srm.conf`, and `access.conf`. Only one of these, `httpd.conf`, actually stores the directives that control an Apache server's behavior. The other two point to `httpd.conf`, which is the file that contains the list of directives that specify the server's operation. These directives are described at `http://httpd.apache.org/docs/2.2/mod/quickreference.html`.

4.4 IIS

Although Apache has been ported to the Windows platforms, it is not the most popular server on those systems. Because the Microsoft IIS server is supplied as part of Windows—and because it is a reasonably good server—most Windows-based Web servers use IIS. Apache and IIS provide similar varieties of services.

From the point of view of the site manager, the most important difference between Apache and IIS is that Apache is controlled by a configuration file that is edited by the manager to change Apache's behavior. With IIS, server behavior is modified by changes made through a window-based management program, named the IIS snap-in, which controls both IIS and `ftp`. This program allows the site manager to set parameters for the server.

Under Windows XP and Vista, the IIS snap-in is accessed by going to *Control Panel*, *Administrative Tools*, and *IIS Manager*. Clicking this last selection takes you to a window that allows starting, stopping, or pausing IIS. This same window allows IIS parameters to be changed when the server has been stopped.

5 Uniform Resource Locators

Uniform (or universal)[8] resource locators (URLs) are used to identify documents (resources) on the Internet. There are many different kinds of resources, identified by different forms of URLs.

5.1 URL Formats

All URLs have the same general format:

scheme:object-address

The scheme is often a communications protocol. Common schemes include `http`, `ftp`, `gopher`, `telnet`, `file`, `mailto`, and `news`. Different schemes use object addresses that have different forms. Our interest here is in the HTTP protocol, which supports the Web. This protocol is used to request and send

8. Fortunately, resource addresses are usually referred to as URLs, so whether it is *uniform* or *universal* is usually irrelevant.

Hypertext Markup Language (HTML) documents. In the case of HTTP, the form of the object address of a URL is as follows:

//fully-qualified-domain-name/path-to-document

Another scheme of interest to us is `file`. The `file` protocol means that the document resides on the machine running the browser. This approach is useful for testing documents to be made available on the Web without making them visible to any other browser. When `file` is the protocol, the fully qualified domain name is omitted, making the form of such URLs as follows:

`file:`*//path-to-document*

Because the focus of this text is on HTML documents, the remainder of the discussion of URLs is limited to the HTTP protocol.

The host name is the name of the server computer that stores the document (or provides access to it, although it is stored on some other computer). Messages to a host machine must be directed to the appropriate process running on the host for handling. Such processes are identified by their associated port numbers. The default port number of Web server processes is 80. If a server has been configured to use some other port number, it is necessary to attach that port number to the host name in the URL. For example, if the Web server is configured to use port 800, the host name must have `:800` attached.

URLs can never have embedded spaces.[9] Also, there is a collection of special characters, including semicolons, colons, and ampersands (`&`), that cannot appear in a URL. To include a space or one of the disallowed special characters, the character must be coded as a percent sign (`%`) followed by the two-digit hexadecimal ASCII code for the character. For example, if `San Jose` is a domain name, it must be typed as `San%20Jose` (`20` is the hexadecimal ASCII code for a space). All of the details characterizing URLs can be found at `http://www.w3.org/Addressing/URL/URI_Overview.html`.

5.2 URL Paths

The path to the document for the HTTP protocol is similar to a path to a file or directory in the file system of an operating system and is given by a sequence of directory names and a file name, all separated by whatever separator character the operating system uses. For UNIX servers, the path is specified with forward slashes; for Windows servers, it is specified with backward slashes. Most browsers allow the user to specify the separators incorrectly—for example, using forward slashes in a path to a document file on a Windows server, as in the following:

```
http://www.gumboco.com/files/f99/storefront.html
```

The path in a URL can differ from a path to a file because a URL need not include all directories on the path. A path that includes all directories along the way is called a *complete path*. In most cases, the path to the document is relative

9. Actually, some browsers incorrectly accept spaces in URLs, although doing so is nonstandard behavior.

to some base path that is specified in the configuration files of the server. Such paths are called *partial paths*. For example, if the server's configuration specifies that the root directory for files it can serve is `files/f99`, the previous URL is specified as follows:

```
http://www.gumboco.com/storefront.html
```

If the specified document is a directory rather than a single document, the directory's name is followed immediately by a slash, as in the following:

```
http://www.gumboco.com/departments/
```

Sometimes a directory is specified (with the trailing slash) but its name is not given, as in the following example:

```
http://www.gumboco.com/
```

The server then searches at the top level of the directory in which servable documents are normally stored for something it recognizes as a home page. By convention, this page is often a file named `index.html`. The home page usually includes links that allow the user to find the other related servable files on the server.

If the directory does not have a file that the server recognizes as being a home page, a directory listing is constructed and returned to the browser.

6 Multipurpose Internet Mail Extensions

A browser needs some way of determining the format of a document it receives from a Web server. Without knowing the form of the document, the browser would not be able to render it, because different document formats require different rendering software. The forms of these documents are specified with Multipurpose Internet Mail Extensions (MIME).

6.1 Type Specifications

MIME was developed to specify the format of different kinds of documents to be sent via Internet mail. These documents could contain various kinds of text, video data, or sound data. Because the Web has needs similar to those of Internet mail, MIME was adopted as the way to specify document types transmitted over the Web. A Web server attaches a MIME format specification to the beginning of the document that it is about to provide to a browser. When the browser receives the document from a Web server, it uses the included MIME format specification to determine what to do with the document. If the content is text, for example, the MIME code tells the browser that it is text and also indicates the particular kind of text it is. If the content is sound, the MIME code tells the browser that it is sound and then gives the particular representation of sound so the browser can choose a program to which it has access to produce the transmitted sound.

MIME specifications have the following form:

type/subtype

The most common MIME types are `text`, `image`, and `video`. The most common text subtypes are `plain` and `html`. Some common image subtypes are `gif` and `jpeg`. Some common video subtypes are `mpeg` and `quicktime`. A list of MIME specifications is stored in the configuration files of every Web server. When we say *document type*, we mean both the type and subtype of the document.

Servers determine the type of a document by using the file name's extension as the key into a table of types. For example, the extension `.html` tells the server that it should attach `text/html` to the document before sending it to the requesting browser.[10]

Browsers also maintain a conversion table for looking up the type of a document by its file name extension. However, this table is used only when the server does not specify a MIME type, which may be the case with some older servers. In all other cases, the browser gets the document type from the MIME header provided by the server.

6.2 Experimental Document Types

Experimental subtypes are sometimes used. The name of an experimental subtype begins with `x-`, as in `video/x-msvideo`. Any Web provider can add an experimental subtype by having its name added to the list of MIME specifications stored in the Web provider's server. For example, a Web provider might have a handcrafted database whose contents he or she wants to make available to others through the Web. Of course, this raises the issue of how the browser can display the database. As might be expected, the Web provider must supply a program that the browser can call when it needs to display the contents of the database. These programs either are external to the browser, in which case they are called *helper applications*, or are code modules that are inserted into the browser, in which case they are called *plug-ins*.

Every browser has a set of MIME specifications (file types) it can handle. All can deal with `text/plain` (unformatted text) and `text/html` (HTML files), among others. Sometimes a particular browser cannot handle a specific document type, even though the type is widely used. These cases are handled in the same way as the experimental types described previously. The browser determines the helper application or plug-in it needs by examining the browser configuration file, which provides an association between file types and their required helpers or plug-ins. If the browser does not have an application or a plug-in that it needs to render a document, an error message is displayed.

A browser can indicate to the server the document types it prefers to receive, as discussed in Section 7.

10. This is not necessarily correct. HTML documents also use the `.html` file extension, but, strictly speaking, should use a different MIME type.

7 The Hypertext Transfer Protocol

All Web communications transactions use the same protocol: the Hypertext Transfer Protocol (HTTP). The current version of HTTP is 1.1, formally defined as RFC 2616, which was approved in June 1999. RFC 2616 is available at the Web site for the World Wide Web Consortium (W3C), `http://www.w3.org`. This section provides a brief introduction to HTTP.

HTTP consists of two phases: the request and the response. Each HTTP communication (request or response) between a browser and a Web server consists of two parts: a header and a body. The header contains information about the communication; the body contains the data of the communication if there is any.

7.1 The Request Phase

The general form of an HTTP request is as follows:

1. HTTP method Domain part of the URL HTTP version
2. Header fields
3. Blank line
4. Message body

The following is an example of the first line of an HTTP request:

```
GET /storefront.html HTTP/1.1
```

Only a few request methods are defined by HTTP, and even a smaller number of these are typically used. Table 1 lists the most commonly used methods.

Table 1 HTTP request methods

Method	Description
`GET`	Returns the contents of the specified document
`HEAD`	Returns the header information for the specified document
`POST`	Executes the specified document, using the enclosed data
`PUT`	Replaces the specified document with the enclosed data
`DELETE`	Deletes the specified document

Among the methods given in Table 1, `GET` and `POST` are the most frequently used. `POST` was originally designed for tasks such as posting a news article to a newsgroup. Its most common use now is to send form data from a browser to a server, along with a request to execute a program on the server that will process the data.

Following the first line of an HTTP communication is any number of header fields, most of which are optional. The format of a header field is the field name

followed by a colon and the value of the field. There are four categories of header fields:

1. *General:* For general information, such as the date
2. *Request:* Included in request headers
3. *Response:* For response headers
4. *Entity:* Used in both request and response headers

One common request field is the `Accept` field, which specifies a preference of the browser for the MIME type of the requested document. More than one `Accept` field can be specified if the browser is willing to accept documents in more than one format. For example; we might have any of the following:

```
Accept: text/plain
Accept: text/html
Accept: image/gif
```

A wildcard character, the asterisk (*), can be used to specify that part of a MIME type can be anything. For example, if any kind of text is acceptable, the `Accept` field could be as follows:

```
Accept: text/*
```

The `Host: host name` request field gives the name of the host. The `Host` field is required for HTTP 1.1. The `If-Modified-Since: date` request field specifies that the requested file should be sent only if it has been modified since the given date.

If the request has a body, the length of that body must be given with a `Content-length` field, which gives the length of the response body in bytes. `POST` method requests require this field because they send data to the server.

The header of a request must be followed by a blank line, which is used to separate the header from the body of the request. Requests that use the `GET`, `HEAD`, and `DELETE` methods do not have bodies. In these cases, the blank line signals the end of the request.

A browser is not necessary to communicate with a Web server; `telnet` can be used instead. Consider the following command, given at the command line of any widely used operating system:

```
>telnet blanca.uccs.edu http
```

This command creates a connection to the `http` port on the `blanca.uccs.edu` server. The server responds with the following:[11]

```
Trying 128.198.162.60 ...
Connected to blanca
Escape character is '^]'.
```

11. Notice that this `telnet` request returns the IP of the server.

The connection to the server is now complete, and HTTP commands such as the following can be given:

```
GET /~user1/respond.html HTTP/1.1
Host: blanca.uccs.edu
```

The header of the response to this request is given in Section 7.2.

7.2 The Response Phase

The general form of an HTTP response is as follows:

1. Status line
2. Response header fields
3. Blank line
4. Response body

The status line includes the HTTP version used, a three-digit status code for the response, and a short textual explanation of the status code. For example, most responses begin with the following:

```
HTTP/1.1  200  OK
```

The status codes begin with `1`, `2`, `3`, `4`, or `5`. The general meanings of the five categories specified by these first digits are shown in Table 2.

Table 2 First digits of HTTP status codes

First Digit	Category
1	Informational
2	Success
3	Redirection
4	Client error
5	Server error

One of the more common status codes is one users never want to see: `404 Not Found`, which means the requested file could not be found. Of course, `200 OK` is what users want to see, because it means that the request was handled without error. The `500` code means that the server has encountered a problem and was not able to fulfill the request.

After the status line, the server sends a response header, which can contain several lines of information about the response, each in the form of a field. The only essential field of the header is `Content-type`.

The following is the response header for the request given near the end of Section 7.1:

```
HTTP/1.1 200 OK
Date: Sat, 25 July 2009 22:15:11 GMT
Server: Apache/2.2.3 (CentOS)
Last-modified: Tues, 18 May 2004 16:38:38 GMT
ETag: "1b48098-16c-3dab592dc9f80"
Accept-ranges: bytes
Content-length: 364
Connection: close
Content-type: text/html, charset=UTF-8
```

The response header must be followed by a blank line, as is the case for request headers. The response data follows the blank line. In the preceding example, the response body would be the HTML file, `respond.html`.

In HTTP versions prior to 1.1, when a server finished sending a response to the client, the communications connection was closed. However, the default operation of HTTP 1.1 is that the connection is kept open for a time so that the client can make several requests over a short span of time without needing to reestablish the communications connection with the server. This change led to significant increases in the efficiency of the Web.

8 Security

It does not take a great deal of contemplation to realize that the Internet and the Web are fertile grounds for security problems. On the Web server side, anyone on the planet with a computer, a browser, and an Internet connection can request the execution of software on any server computer. He or she can also access data and databases stored on the server computer. On the browser end, the problem is similar: Any server to which the browser points can download software to be executed on the browser host machine. Such software can access parts of the memory and memory devices attached to that machine that are not related to the needs of the original browser request. In effect, on both ends, it is like allowing any number of total strangers into your house and trying to prevent them from leaving anything in the house, taking anything from the house, or altering anything in the house. The larger and more complex the design of the house, the more difficult it will be to prevent any of those activities. The same is true for Web servers and browsers: The more complex they are, the more difficult it is to prevent security breaches. Today's browsers and Web servers are indeed large and complex software systems, so security is a significant problem in Web applications.

The subject of Internet and Web security is extensive and complicated, so much so that numerous books on the topic have been written. Therefore, this one section can give no more than a brief sketch of some of the subtopics of security.

One aspect of Web security is the matter of getting one's data from the browser to the server and having the server deliver data back to the browser without anyone or any device intercepting or corrupting those data along the way. Consider a simple case of transmitting a credit card number to a company from which a purchase is being made. The security issues for this transaction are as follows:

1. *Privacy*—it must not be possible for the credit card number to be stolen on its way to the company's server.
2. *Integrity*—it must not be possible for the credit card number to be modified on its way to the company's server.
3. *Authentication*—it must be possible for both the purchaser and the seller to be certain of each other's identity.
4. *Nonrepudiation*—it must be possible to prove legally that the message was actually sent and received.

The basic tool to support privacy and integrity is encryption. Data to be transmitted is converted into a different form, or encrypted, such that someone (or some computer) who is not supposed to access the data cannot decrypt it. So, if data is intercepted while en route between Internet nodes, the interceptor cannot use the data because he or she cannot decrypt it. Both encryption and decryption are done with a key and a process (applying the key to the data). Encryption was developed long before the Internet existed. Julius Caesar crudely encrypted the messages he sent to his field generals while at war. Until the middle 1970s, the same key was used for both encryption and decryption, so the initial problem was how to transmit the key from the sender to the receiver.

This problem was solved in 1976 by Whitfield Diffie and Martin Hellman of Stanford University, who developed *public-key encryption*, a process in which a public key and a private key are used, respectively, to encrypt and decrypt messages. A communicator—say, Joe—has an inversely related pair of keys, one public and one private. The public key can be distributed to all organizations that might send Joe messages. All of them can use the public key to encrypt messages to Joe, who can decrypt the messages with his matching private key. This arrangement works because the private key need never be transmitted and also because it is virtually impossible to decrypt the private key from its corresponding public key. The technical wording for this situation is that it is "computationally infeasible" to determine the private key from its public key.

The most widely used public-key algorithm is named RSA, developed in 1977 by three MIT professors—Ron Rivest, Adi Shamir, and Leonard Adleman—the first letters of whose last names were used to name the algorithm. Most large companies now use RSA for e-commerce.

Another, completely different security problem for the Web is the intentional and malicious destruction of data on computers attached to the Internet. The number of different ways this can be done has increased steadily over the life span of the Web. The sheer number of such attacks has also grown rapidly. There is now a continuous stream of new and increasingly devious denial-of-service (DoS) attacks, viruses, and worms being discovered, which have caused billions of dollars

of damage, primarily to businesses that use the Web heavily. Of course, huge damage also has been done to home computer systems through Web intrusions.

DoS attacks can be created simply by flooding a Web server with requests, overwhelming its ability to operate effectively. Most DoS attacks are conducted with the use of networks of virally infected "zombie" computers, whose owners are unaware of their sinister use. So, DoS and viruses are often related.

Viruses are programs that often arrive in a system in attachments to e-mail messages or attached to free downloaded programs. Then they attach to other programs. When executed, they replicate and can overwrite memory and attached memory devices, destroying programs and data alike. Two viruses that were extensively destructive appeared in 2000 and 2001: the ILOVEYOU virus and the CodeRed virus, respectively.

Worms damage memory, like viruses, but spread on their own, rather than being attached to other files. Perhaps the most famous worm so far has been the Blaster worm, spawned in 2003.

DoS, virus, and worm attacks are created by malicious people referred to as *hackers*. The incentive for these people apparently is simply the feeling of pride and accomplishment they derive from being able to cause huge amounts of damage by outwitting the designers of Web software systems.

Protection against viruses and worms is provided by antivirus software, which must be updated frequently so that it can detect and protect against the continuous stream of new viruses and worms.

9 The Web Programmer's Toolbox

This section provides an overview of the most common tools used in Web programming—some are programming languages, some are not. The tools discussed are HTML, a markup language, along with a few high-level markup document-editing systems; XML, a meta-markup language; JavaScript, PHP, and Ruby, which are programming languages; JSF, ASP.NET, and Rails, which are development frameworks for Web-based systems; Flash, a technology for creating and displaying graphics and animation in HTML documents; and Ajax, a Web technology that uses JavaScript and XML.

Web programs and scripts are divided into two categories—client side and server side—according to where they are interpreted or executed. HTML and XML are client-side languages; PHP and Ruby are server-side languages; JavaScript is most often a client-side language, although it can be used for both.

We begin with the most basic tool: HTML.

9.1 Overview of HTML

At the onset, it is important to realize that HTML is not a programming language—it cannot be used to describe computations. Its purpose is to describe the general form and layout of documents to be displayed by a browser.

The word *markup* comes from the publishing world, where it is used to describe what production people do with a manuscript to specify to a printer how the text, graphics, and other elements should appear in printed form. HTML is not the first markup language used with computers. TeX and LaTeX are older markup languages for use with text; they are now used primarily to specify how mathematical expressions and formulas should appear in print.

An HTML document is a mixture of content and controls. The controls are specified by the tags of HTML. The name of a tag specifies the category of its content. Most HTML tags consist of a pair of syntactic markers that are used to delimit particular kinds of content. The pair of tags and their content together are called an *element*. For example, a paragraph element specifies that its content, which appears between its opening tag, `<p>`, and its closing tag, `</p>`, is a paragraph. A browser has a default style (font, font style, font size, and so forth) for paragraphs, which is used to display the content of a paragraph element.

Some tags include attribute specifications that provide some additional information for the browser. In the following example, the `src` attribute specifies the location of the `img` tag's image content:

```
<img src = "redhead.jpg"/>
```

In this case, the image document stored in the file `redhead.jpg` is to be displayed at the position in the document in which the tag appears.

9.2 Tools for Creating HTML Documents

HTML documents can be created with a general-purpose text editor. There are two kinds of tools that can simplify this task: HTML editors and what-you-see-is-what-you-get (WYSIWYG, pronounced *wizzy-wig*) HTML editors.

HTML editors provide shortcuts for producing repetitious tags such as those used to create the rows of a table. They also may provide a spell-checker and a syntax-checker, and they may color code the HTML in the display to make it easier to read and edit.

A more powerful tool for creating HTML documents is a WYSIWYG HTML editor. Using a WYSIWYG HTML editor, the writer can see the formatted document that the HTML describes while he or she is writing the HTML code. WYSIWYG HTML editors are very useful for beginners who want to create simple documents without learning HTML and for users who want to prototype the appearance of a document. Still, these editors sometimes produce poor-quality HTML. In some cases, they create proprietary tags that some browsers will not recognize.

Two examples of WYSIWYG HTML editors are Microsoft FrontPage and Adobe Dreamweaver. Both allow the user to create HTML-described documents without requiring the user to know HTML. They cannot handle all of the tags of HTML, but they are very useful for creating many of the common features of documents. Between the two, FrontPage is by far the most widely

used. Information on Dreamweaver is available at `http://www.adobe.com/`; information on FrontPage is available at `http://www.microsoft.com/frontpage/`.

9.3 Plug-ins and Filters

Two different kinds of converters can be used to create HTML documents. *Plug-ins*[12] are programs that can be integrated with a word processor. Plug-ins add new capabilities to the word processor, such as toolbar buttons and menu elements that provide convenient ways to insert HTML into the document being created or edited. The plug-in makes the word processor appear to be an HTML editor that provides WYSIWYG HTML document development. The end result of this process is an HTML document. The plug-in also makes available all the tools that are inherent in the word processor during HTML document creation, such as a spell-checker and a thesaurus.

A second kind of converter is a *filter*, which converts an existing document in some form, such as LaTeX or Microsoft Word, to HTML. Filters are never part of the editor or word processor that created the document—an advantage because the filter can then be platform independent. For example, a WordPerfect user working on a Macintosh computer can use a filter running on a UNIX platform to produce HTML documents with the same content on that machine. The disadvantage of filters is that creating HTML documents with a filter is a two-step process: First you create the document, and then you use a filter to convert it to HTML.

Neither plugs-ins nor filters produce HTML documents that, when displayed by browsers, have the identical appearance of that produced by the word processor.

The two advantages of both plug-ins and filters, however, are that existing documents produced with word processors can be easily converted to HTML and that users can use a word processor with which they are familiar to produce HTML documents. This obviates the need to learn to format text by using HTML directly. For example, once you learn to create tables with your word processor, it is easier to use that process than to learn to define tables directly in HTML.

The HTML output produced by either filters or plug-ins often must be modified, usually with a simple text editor, to perfect the appearance of the displayed document in the browser. Because this new HTML file cannot be converted to its original form (regardless of how it was created), you will have two different source files for a document, inevitably leading to version problems during maintenance of the document. This is clearly a disadvantage of using converters.

12. The word *plug-in* applies to many different software systems that can be added to or embedded in other software systems. For example, many different plug-ins can be added to Web browsers.

9.4 Overview of XML

HTML is defined with the use of the Standard Generalized Markup Language (SGML), which is a language for defining markup languages. (Such languages are called meta-markup languages.) XML (eXtensible Markup Language) is a simplified version of SGML, designed to allow users to easily create markup languages that fit their own needs. Whereas HTML users must use the predefined set of tags and attributes, when a user creates his or her own markup language with XML, the set of tags and attributes is designed for the application at hand. For example, if a group of users wants a markup language to describe data about weather phenomena, that language could have tags for cloud forms, thunderstorms, and low-pressure centers. The content of these tags would be restricted to relevant data. If such data is described with HTML, cloud forms could be put in generic tags, but then they could not be distinguished from thunderstorm elements, which would also be in the same generic tags.

Whereas HTML describes the overall layout and gives some presentation hints for general information, XML-based markup languages describe data and its meaning through their individualized tags and attributes. XML does not specify any presentation details.

The great advantage of XML is that application programs can be written to use the meanings of the tags in the given markup language to find specific kinds of data and process it accordingly. The syntax rules of XML, along with the syntax rules for a specific XML-based markup language, allow documents to be validated before any application attempts to process their data. This means that all documents that use a specific markup language can be checked to determine whether they are in the standard form for such documents. Such an approach greatly simplifies the development of application programs that process the data in XML documents.

9.5 Overview of JavaScript

JavaScript is a client-side scripting language whose primary uses in Web programming are to validate form data, to build Ajax-enabled HTML documents, and to create dynamic HTML documents.

The name *JavaScript* is misleading because the relationship between Java and JavaScript is tenuous, except for some of the syntax. One of the most important differences between JavaScript and most common programming languages is that JavaScript is dynamically typed. This design is virtually the opposite of that of strongly typed languages such as C++ and Java.

JavaScript "programs" are usually embedded in HTML documents,[13] which are downloaded from a Web server when they are requested by browsers. The JavaScript code in an HTML document is interpreted by an interpreter embedded in the browser on the client.

13. We quote the word *programs* to indicate that these are not programs in the general sense of the self-contained collections of C++ or C code we normally call programs.

One of the most important applications of JavaScript is to create and modify documents dynamically. JavaScript defines an object hierarchy that matches a hierarchical model of an HTML document. Elements of an HTML document are accessed through these objects, providing the basis for dynamic documents.

9.6 Overview of Flash

There are two components of Flash: the authoring environment, which is a development framework, and the player. Developers use the authoring environment to create static graphics, animated graphics, text, sound, and interactivity to be part of stand-alone HTML documents or to be part of other HTML documents. These documents are served by Web servers to browsers, which use the Flash player plug-in to display the documents. Much of this development is done by clicking buttons, choosing menu items, and dragging and dropping graphics.

Flash makes animation very easy. For example, for motion animation, the developer needs only to supply the beginning and ending positions of the figure to be animated—Flash builds the intervening figures. The interactivity of a Flash application is implemented with ActionScript, a dialect of JavaScript.

Flash is now the leading technology for delivering graphics and animation on the Web. It has been estimated that nearly 99 percent of the world's computers used to access the Internet have a version of the Flash player installed as a plug-in in their browsers.

9.7 Overview of PHP

PHP is a server-side scripting language specifically designed for Web applications. PHP code is embedded in HTML documents, as is the case with JavaScript. With PHP, however, the code is interpreted on the server before the HTML document is delivered to the requesting client. A requested document that includes PHP code is preprocessed to interpret the PHP code and insert its output into the HTML document. The browser never sees the embedded PHP code and is not aware that a requested document originally included such code.

PHP is similar to JavaScript, both in terms of its syntactic appearance and in terms of the dynamic nature of its strings and arrays. Both JavaScript and PHP use dynamic data typing, meaning that the type of a variable is controlled by the most recent assignment to it. PHP's arrays are a combination of dynamic arrays and hashes (associative arrays). The language includes a large number of predefined functions for manipulating arrays.

PHP allows simple access to HTML form data, so form processing is easy with PHP. PHP also provides support for many different database management systems. This versatility makes it an excellent language for building programs that need Web access to databases.

9.8 Overview of Ajax

Ajax, shorthand for *A*synchronous *J*a*v*aScript + *X*ML, had been around for a few years in the early 2000s, but did not acquire its catchy name until 2005.[14] The idea of Ajax is relatively simple, but it results in a different way of viewing and building Web interactions. This new approach produces an enriched Web experience for those using a certain category of Web interactions.

In a traditional (as opposed to Ajax) Web interaction, the user sends messages to the server either by clicking a link or by clicking a form's *Submit* button. After the link has been clicked or the form has been submitted, the client waits until the server responds with a new document. The entire browser display is then replaced by that of the new document. Complicated documents take a significant amount of time to be transmitted from the server to the client and more time to be rendered by the browser. In Web applications that require frequent interactions with the client and remain active for a significant amount of time, the delay in receiving and rendering a complete response document can be disruptive to the user.

In an Ajax Web application, there are two variations from the traditional Web interaction. First, the communication from the browser to the server is asynchronous; that is, the browser need not wait for the server to respond. Instead, the browser user can continue whatever he or she was doing while the server finds and transmits the requested document and the browser renders the new document. Second, the document provided by the server usually is only a relatively small part of the displayed document, and therefore it takes less time to be transmitted and rendered. These two changes can result in much faster interactions between the browser and the server.

The *x* in *Ajax*, from *XML*, is there because in many cases the data supplied by the server is in the form of an XML document, which provides the new data to be placed in the displayed document. However, in some cases the data is plain text, which may even be JavaScript code. It can also be HTML.

The goal of Ajax is to have Web-based applications become closer to desktop (client-resident) applications, in terms of the speed of interactions and the quality of the user experience. Wouldn't we all like our Web-based applications to be as responsive as our word processors?

9.9 Overview of Servlets, JavaServer Pages, and JavaServer Faces

There are many computational tasks in a Web interaction that must occur on the server, such as processing order forms and accessing server-resident databases. A Java class called a *servlet* can be used for these applications. A servlet is a compiled Java class, an object of which is executed on the server system when

14. Ajax was named by Jesse James Garrett, who has on numerous occasions stated that Ajax is shorthand, not an acronym. Thus, we spell it Ajax, not AJAX.

requested by the HTML document being displayed by the browser. A servlet produces an HTML document as a response, some parts of which are static and are generated by simple output statements, while other parts are created dynamically when the servlet is called.

When an HTTP request is received by a Web server, the Web server examines the request. If a servlet must be called, the Web server passes the request to the servlet processor, called a *servlet container*. The servlet container determines which servlet must be executed, makes sure that it is loaded, and calls it. As the servlet handles the request, it generates an HTML document as its response, which is returned to the server through the response object parameter.

Java can also be used as a server-side scripting language. An HTML document with embedded Java code is one form of JavaServer Pages (JSP). Built on top of servlets, JSP provides alternative ways of constructing dynamic Web documents. JSP takes an opposite approach to that of servlets: Instead of embedding HTML in Java code that provides dynamic documents, code of some form is embedded in HTML documents to provide the dynamic parts of a document. These different forms of code make up the different approaches used by JSP. The basic capabilities of servlets and JSP are the same.

When requested by a browser, a JSP document is processed by a software system called a *JSP container*. Some JSP containers compile the document when the document is loaded on the server; others compile it only when requested. The compilation process translates a JSP document into a servlet and then compiles the servlet. So, JSP is actually a simplified approach to writing servlets.

JavaServer Faces (JSF) adds another layer to the JSP technology. The most important contribution of JSF is an event-driven user interface model for Web applications. Client-generated events can be handled by server-side code with JSF.

9.10 Overview of Active Server Pages .NET

Active Server Pages .NET (ASP.NET) is a Microsoft framework for building server-side dynamic documents. ASP.NET documents are supported by programming code executed on the Web server. As we saw in Section 9.9, JSF uses Java to describe the dynamic generation of HTML documents, as well as to describe computations associated with user interactions with documents. ASP.NET provides an alternative to JSF, with two major differences: First, ASP.NET allows the server-side programming code to be written in any of the .NET languages.[15] Second, in ASP.NET all programming code is compiled, which allows it to execute much faster than interpreted code.

Every ASP.NET document is compiled into a class. From a programmer's point of view, developing dynamic Web documents (and the supporting code) in ASP.NET is similar to developing non-Web applications. Both involve defining classes based on library classes, implementing interfaces from a library, and calling methods defined in library classes. An application class uses, and interacts with,

15. In most cases, it is done in either Visual BASIC .NET or C#.

existing classes. In ASP.NET, this is exactly the same for Web applications: Web documents are designed by designing classes.

9.11 Overview of Ruby

Ruby is an object-oriented interpreted scripting language designed by Yukihiro Matsumoto (a.k.a. Matz) in the early 1990s and released in 1996. Since then, it has continually evolved and its level of usage has grown. The original motivation for Ruby was dissatisfaction of its designer with the earlier languages Perl and Python.

The primary characterizing feature of Ruby is that it is a pure object-oriented language, just as is Smalltalk. Every data value is an object and all operations are via method calls. The operators in Ruby are only syntactic mechanisms to specify method calls for the corresponding operations. Because they are methods, many of the operators can be redefined by user programs. All classes, whether predefined or user defined, can have subclasses.

Both classes and objects in Ruby are dynamic in the sense that methods can be dynamically added to either. This means that classes and objects can have different sets of methods at different times during execution. So, different instantiations of the same class can behave differently.

The syntax of Ruby is related to that of Eiffel and Ada. There is no need to declare variables, because dynamic typing is used. In fact, all variables are references and do not have types, although the objects they reference do.

Our interest in Ruby is based on Ruby's use with the Web development framework Rails (see Section 9.12). Rails was designed for use with Ruby, and it is Ruby's primary use in Web programming.

Ruby is culturally interesting because it is the first programming language designed in Japan that has achieved relatively widespread use outside that country.

9.12 Overview of Rails

Rails is a development framework for Web-based applications that access databases. A framework is a system in which much of the more-or-less standard software parts are furnished by the framework, so they need not be written by the applications developer. ASP.NET and JSF are also development frameworks for Web-based applications. Rails, whose more official name is Ruby on Rails, was developed by David Heinemeier Hansson in the early 2000s and was released to the public in July 2004. Since then, it has rapidly gained widespread interest and usage. Rails is based on the Model–View–Controller (MVC) architecture for applications, which clearly separates applications into three parts: presentation, data model, and program logic.

Rails applications are tightly bound to relational databases. Many Web applications are closely integrated with database access, so the Rails relational database framework is a widely applicable architecture.

Rails can be, and often is, used in conjunction with Ajax. Rails uses the JavaScript framework prototype to support Ajax and interactions with the JavaScript model of the document being displayed by the browser. Rails also provides other support for developing Ajax, including producing visual effects.

Rails was designed to be used with Ruby and makes use of the strengths of that language. Furthermore, Rails is written in Ruby.

Summary

The Internet began in the late 1960s as the ARPAnet, which was eventually replaced by NSFnet for nonmilitary users. NSFnet later became known as the Internet. There are now many millions of computers around the world that are connected to the Internet. Although much of the network control equipment is different and many kinds of computers are connected, all of these connections are made through the TCP/IP protocol, making them all appear, at least at the lowest level, the same to the network.

Two kinds of addresses are used on the Internet: IP addresses, which are four-part numbers, for computers; and fully qualified domain names, which are words separated by periods, for people. Fully qualified domain names are translated to IP addresses by name servers running DNS. A number of different information interchange protocols have been created, including `telnet`, `ftp`, and `mailto`.

The Web began in the late 1980s at CERN as a means for physicists to share the results of their work efficiently with colleagues at other locations. The fundamental idea of the Web is to transfer hypertext documents among computers by means of the HTTP protocol on the Internet.

Browsers request HTML documents from Web servers and display them for users. Web servers find and send requested documents to browsers. URLs are used to address all documents on the Internet; the specific protocol to be used is the first field of the URL. URLs also include the fully qualified domain name and a file path to the specific document on the server. The type of a document that is delivered by a Web server appears as a MIME specification in the first line of the document. Web sites can create their own experimental MIME types, provided that they also furnish a program that allows the browser to present the document's contents to the user.

HTTP is the standard protocol for Web communications. HTTP requests are sent over the Internet from browsers to Web servers; HTTP responses are sent from Web servers to browsers to fulfill those requests. The most commonly used HTTP requests are `GET` and `POST`.

Web programmers use several languages to create the documents that servers can provide to browsers. The most basic of these is HTML, the standard markup language for describing how Web documents should be presented by browsers. Tools that can be used without specific knowledge of HTML are available to create HTML documents. A plug-in is a program that can be integrated with a word processor to make it possible to use the word processor to create HTML. A filter converts a document written in some other format to HTML. XML is

a meta-markup language that provides a standard way to define new markup languages.

JavaScript is a client-side scripting language that can be embedded in an HTML document to describe simple computations. JavaScript code is interpreted by the browser on the client machine; it provides access to the elements of an HTML document, as well as the ability to change those elements dynamically.

Flash is a framework for building animation into HTML documents. A browser must have a Flash player plug-in to be able to display the movies created with the Flash framework.

Ajax is an approach to building Web applications in which partial document requests are handled asynchronously. Ajax can significantly increase the speed of user interactions, so it is most useful for building systems that have frequent interactions.

PHP is the server-side equivalent of JavaScript. It is an interpreted language whose code is embedded in HTML documents. PHP is used primarily for form processing and database access from browsers.

Servlets are server-side Java programs that are used for form processing, database access, or building dynamic documents. JSP documents, which are translated into servlets, are an alternative approach to building these applications. JSF is a development framework for specifying forms and their processing in JSP documents.

ASP.NET is a Web development framework. The code used in ASP.NET documents, which is executed on the server, can be written in any .NET programming language.

Ruby is a relatively recent object-oriented scripting language that is introduced here primarily because of its use in Rails, a Web applications framework. Rails provides a significant part of the code required to build Web applications that access databases, allowing the developer to spend his or her time on the specifics of the application without dealing with the drudgery of the housekeeping details.

Review Questions

1 What was one of the fundamental requirements for the new national computer network proposed by the DoD in the 1960s?

2 What protocol is used by all computer connections to the Internet?

3 What is the form of an IP address?

4 Describe a fully qualified domain name.

5 What is the task of a DNS name server?

6 What is the purpose of `telnet`?

7 In the first proposal for the Web, what form of information was to be interchanged?

8 What is hypertext?

9 What category of browser, introduced in 1993, led to a huge expansion of Web usage?

10 In what common situation is the document returned by a Web server created after the request is received?

11 What is the document root of a Web server?

12 What is a virtual document tree?

13 What is the server root of a Web server?

14 What is a virtual host?

15 What is a proxy server?

16 What does the `file` protocol specify?

17 How do partial paths to documents work in Web servers?

18 When a browser requests a directory without giving its name, what is the name of the file that is normally returned by the Web server?

19 What is the purpose of a MIME type specification in a request–response transaction between a browser and a server?

20 With what must a Web server furnish the browser when it returns a document with an experimental MIME type?

21 Describe the purposes of the five most commonly used HTTP methods.

22 What is the purpose of the `Accept` field in an HTTP request?

23 What response header field is most often required?

24 Prior to HTTP 1.1, how long were connections between browsers and servers normally maintained?

25 What important capability is lacking in a markup language?

26 What problem is addressed by using a public-key approach to encryption?

27 Is it practically possible to compute the private key associated with a given public key?

28 What is the difference between a virus and a worm?

29 What appears to motivate a hacker to create and disseminate a virus?

30 What is a plug-in?

31 What is a filter HTML converter?

32 Why must code generated by a filter often be modified manually before use?

33 What is the great advantage of XML over HTML for describing data?

34 How many different tags are predefined in an XML-based markup language?

35 What is the relationship between Java and JavaScript?

36 What are the most common applications of JavaScript?

37 Where is JavaScript most often interpreted, on the server or on the browser?

38 What is the primary use of Flash?

39 Where are Flash movies interpreted, on the server or on the browser?

40 Where are servlets executed, on the server or on the browser?

41 In what language are servlets written?

42 In what way are JSP documents the opposite of servlets?

43 What is the purpose of JSF?

44 What is the purpose of ASP.NET?

45 In what language is the code in an ASP.NET document usually written?

46 Where is PHP code interpreted, on the server or on the browser?

47 In what ways is PHP similar to JavaScript?

48 In what ways is Ruby more object oriented than Java?

49 In what country was Ruby developed?

50 What is the purpose of Rails?

51 For what particular kind of Web application was Rails designed?

52 Which programming languages are used in Ajax applications?

53 In what fundamental way does an Ajax Web application differ from a traditional Web application?

Exercises

1 For the following products, to what brand do you have access, what is its version number, and what is the latest available version?

   a. Browser

   b. Web server

   c. JavaScript

   d. PHP

   e. Servlets

f. ASP.NET
g. Ruby
h. Rails

2 Search the Web for information on the history of the following technologies, and write a brief overview of those histories:
a. TCP/IP
b. SGML
c. HTML
d. ARPAnet
e. BITnet
f. XML
g. JavaScript
h. Flash
i. Servlets
j. JSP
k. JSF
l. Rails
m. Ajax