1

THE WAY THE WEB WORKS

A basic introduction to how the WWW works within the context of the Internet with supporting protocols and applications.

This first chapter aims at equipping you with a thorough grounding in what you need to know to start developing applications and pages for the Web.

To begin with there is a brief history lesson on how the Internet and in particular the World Wide Web (WWW) was developed, which explains how various aspects of the technology such as hypertext came about. As time went on, more applications were found that could be handled by the Internet, each one of these slowly becoming standardized in the way they went about communicating across the Web until there were many protocols available for different tasks. The more important protocols are explored here.

To develop applications you need to be able to interact with a server that will show your Web pages and run your applications. To communicate with this machine you need a way of uploading and talking to it. This is shown here along with some issues you may meet along the way.

One of the most important tools you will be using while developing will be the Web browser so some of its useful features, to both a user and developer, are explained. This is approached in a generic way and therefore should be equally applicable to your favorite browser.

1.1 HISTORY

First of all, let's start with a bit of history – how did it all begin and why? As long as information has been collected and stored there has been a need for duplication and, with that, a means for communicating it to other places. This was true when the first books and other documents were being written throughout history. When computers came on the scene it became important that information could be stored and transferred between machines so other people could access the information and maybe add to it.

Computers were initially linked together to store information in universities, defense organizations and government departments. As they did not all have a common standard to communicate by, information could not be passed between dissimilar systems. A university may have had a department where scientists could share their work and papers but it would become a problem to connect to someone else's system and transfer documents or data.

Possibly the biggest driving factor to develop interoperability between systems was the needs of the US Department of Defense. They required a communications architecture for command and control that was reliable, universal and in some way self-healing. The idea here was that if a certain center, or node, was lost in the network (for example by enemy attack) then information would not be disrupted but would find its way around it to its intended destination. In the late 1960s, the Defense Advanced Research Projects Agency (DARPA) worked with both industry and universities, starting a network to test various ideas out in this line. The network was known as ARPANET and was used to develop protocols and techniques that are now used on the Internet.

The standards that were developed in that period were passed on in the 1980s to the US Department of Defense's Defense Communications Agency. The agency became their guardian until finally, with the establishment of a more widely used Internet in the 1990s, they were passed on to the Internet Architecture Board (IAB), an independent organization.

1.1.1 The WWW

The WWW is part of the history of the Internet. Its inventor, Tim Berners-Lee, in 1980 was investigating how computers could store information with random links. In 1989, while working at the European Particle Physics Laboratory, he proposed the idea of a global hypertext space in which any network-accessible information could be referred to by a single 'Universal Document Identifier'. In 1990 this idea was expanded with a program called 'WorlDwidEWeb', a point and click hypertext editor running on a NeXT computer; further developments led to the browser and Web server. Several specifications were developed here including URIs, HyperText Markup Language (HTML) and HyperText Transfer Protocol (HTTP) and were in fact published on the first server to promote wide adoption and discussion.

Over the years 1991 to 1994 the load on the first Web server info.cern.ch increased by a factor of ten every year. A selection of browsers was developed for different types of computer. Initially academia and then industry started taking notice. Under pressure to define future direction, Tim Berners-Lee formed the World Wide Web Consortium (W3C) in 1994, which acted as a neutral forum where companies and organizations could go to discuss and agree on new common computer protocols.

1.2 THE INTERNET AND THE WWW

The Internet works by defining an address for each resource attached to it. In the case of attached computers and some devices, this is the Internet Protocol (IP) address. Without some form of address, no link could be formed between computers, resources or systems. An example of an IP address is 158.162.131.236. The numbers break down to a country, to a specific domain, right down to a machine itself.

Ordinarily, of course, we don't use numbers to find resources and machines – we use a name like mycomputer.co.uk. The name actually will have an associated number but when a name is entered, say, into a Web browser, the computer will use a special system called the Domain Name System (DNS) to look up the associated number which it will then use to get in contact with the desired server. Figure 1-1 shows the basic idea behind IP addressing for one particular network. In this diagram a group of computers are connected to a Local Area Network (LAN), each having a local IP address. These are connected to a gateway or router, which in turn is linked to an Internet Service Provider (ISP), usually with some type of modem. The ISP allows access to the Internet, DNS lookup services and email. The ISP also allocates an IP address for the network, which in turn can be utilized to resolve to individual machines using techniques such as Network Address Translation (NAT).

The discussion behind how the actual communication takes place deserves a book or course of its own but the basics will be described here. Any communication that occurs on the Internet uses the TCP/IP (Transport Control Protocol/Internet Protocol) suite. This is a set of communication protocols, or conventions, for dialogue between computers and devices. These protocols implement a stack, each layer of which solves problems relating to the transmission of data, as well providing service to higher layers than itself. Higher layers are logically closer to the user, dealing with more abstract data and rely on the lower layers to provide the means in which they can be physically manipulated.

Table 1-1 shows the Operating Systems Interconnection (OSI) reference model, which is a layered abstract description for describing communications and computer network protocols. It is roughly adhered to in the computing and networking industry and is often simplified to produce a model for the Internet as shown in Table 1-2. Note here that the session and presentation layers have been absorbed into the application layer.

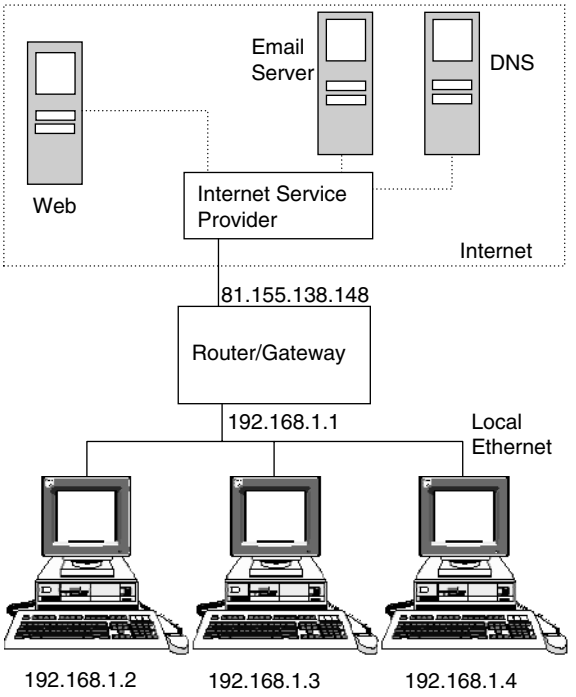


Figure 1-1 *How computers link to the Internet*

Level	Stack layer	Protocol
7	Application	HTTP, SMTP, SNMP, FTP, Telnet, SSH, Scp, NFS, RTSP
6	Presentation	XDR, ASN.1, SMB, AFP
5	Session	TLS, SSH, RPC, NetBIOS, ASP
4	Transport	TCP, UDP, RTP, SCTP, SPX, ATP
3	Network	IP, ICMP, IGMP, X.25, CLNP, ARP, RARP, BGP, OSPF, RIP, IPX, DDP
2	Data Link	Ethernet, Token Ring, PPP, HDLC, Frame Relay, ISDN, ATM, 802.11 Wi-Fi, FDDI
1	Physical	Electrical, radio, laser

Table 1-1 *OSI model*

OSI level	Stack layer	Protocol
7	Application	HTTP, SMTP, SNMP, FTP, Telnet, SSH, Scp, DNS
4	Transport	TCP, UDP, RTP, SCTP, SPX, ATP
3	Network	IP, ICMP, IGMP, X.25, CLNP, ARP, RARP, BGP, OSPF, RIP, IPX, DDP
2	Data Link	Ethernet, Token Ring, PPP, HDLC, Frame Relay, ISDN, ATM, 802.11 Wi-Fi, FDDI
1	Physical	Electrical, radio, laser

Table 1-2 *Internet layered model*

Where the OSI model was theoretical and produced at an earlier stage in the evolution of networks, the Internet model, as shown in Table 1-2, was produced as a practical solution to the engineering problems involved.

The *physical layer* deals with the physical characteristics of communication such as conventions about the medium used for communication (e.g. wires, fiber optic or radio links). Other related details such as connectors, channels, modulation, signal strengths, level synchronization, timing and distances involved are also included here.

The *data link layer* specifies how packets of information are transported over the physical layer, such as how it is framed or set out with special start and stop bit patterns. The *network layer* is concerned with how the packets are transferred over and between networks.

Protocols at the *transport layer* can deal with solving problems like reliability; that is, whether or not the data reached the location it was intended to and ensuring that data arrives in the correct order. Transport protocols within the TCP/IP suite also determine which application any given data is intended for. TCP is one such protocol that exists here and is said to be reliable and connection oriented. UDP (User Datagram Protocol) also exists at this level and has some interesting characteristics. For example, because it is usually used for streaming video or audio it is a 'best effort' or 'unreliable' protocol in that it does not verify that packets reach their destination; rather its concern is that they arrive on time.

Finally, the *application layer* is the layer that programs use to interface with in order to communicate across a network with other programs. Data is passed from the program in an application format and encoded into a standard protocol. Some programs are considered to run in this layer, with their associated protocols. Once the data from an application has been encoded into a standard application layer protocol it is then passed down to the next layer of the OSI model.

Checkpoint!

We now know:

- **there was an initial need for information exchange between systems**
- **information exchange was the catalyst for developing networks**
- **networks require common conventions for dialogue (protocols)**
- **each resource in a network needs a unique identifier (IP address)**
- **an IP address can be associated with a name using DNS**
- **a stack of protocols exist, providing services and communication between the various levels in a system.**

1.3 PROTOCOLS AND PROGRAMS

To enable communications to occur between parties, there are a few aspects that have to be considered:

- where the communication takes place (usually a specific *port*, a kind of interface for communication over a network)
- how it takes place
- the rules and conventions involved.

For the Internet, there are many ways of communicating between machines and devices, depending on what is actually required. Often there are many ways to do the same thing, which is important because the particular context that a device is in may limit what resources are available to it.

A specific task usually requires a particular form of communication. Here we look at a few contexts in which we may wish to transfer information between systems and the mechanisms available for the task.

1.3.1 Files

If you want a file to go from one machine to another, how would you do it? You could, for example, simply email it and pick up the email on the other machine. There may be problems with this – what if there was no email client or the file was simply too big? What other possibilities are there? You could use a disk or memory of some kind. Another way

could be to use File Transfer Protocol (FTP) to load it either directly on the machine, or to a server, so it can be downloaded on to the target machine later. Let's say you want to transfer a file called `myIndex.doc`; a typical session may run like this:

```
>ftp mysuperserver.co.uk
Connected to mysuperserver.co.uk.
220 FTP Server ready.
Name (mysuperserver.co.uk:ralphmoseley): ralph101
331 Password required for ralph101.
Password:
230 User ralph101 logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>put myIndex.doc
```

The file will then be transferred. Notice that different files require either binary or ASCII (American Standard Code for Information Interchange, pronounced *ass-key*, a character set) modes, text-based files need to be sent using ASCII. Graphic files, such as JPEG or GIF, and word processor formats, such as MS Word doc, require binary.

When you are logged in you have a range of commands available, mainly based on UNIX commands, such as those shown in Table 1-3.

As you can see, there are lots of commands but there are some that are used more than others:

- `ftp` – start an ftp session
- `ls` – list files
- `get` – download a file from the server
- `put` – upload a file to the server
- `mkdir` – make a directory on the server
- `cd` – change to a new directory on the server
- `close` – close the connection
- `open` – open a new connection
- `bin` – binary mode transfer
- `asc` – ASCII text mode transfer.

These provide an adequate capability to allow a user to get their files from or upload them to a server. When developing Web pages or applications, there is a need for a cycle of local editing and updating of the files on the server. This can be done with an FTP connection; some editors even include the ability to transfer files online!

!	Features	mls	Proxy	size
$	fget	mlsd	put	sndbuf
account	form	mlst	pwd	status
append	ftp	mode	quit	struct
ascii	gate	modtime	quote	sunique
bell	get	more	rate	system
binary	glob	mput	rcvbuf	tenex
bye	hash	msend	recv	throttle
case	help	newer	reget	trace
cd	idle	nlist	remopts	type
cdup	image	nmap	rename	umask
chmod	lcd	ntrans	reset	unset
close	less	open	restart	usage
cr	lpage	page	rhelp	user
debug	lpwd	passive	rmdir	verbose
delete	ls	pdir	rstatus	xferbuf
dir	macdef	pls	runique	?
disconnect	mdelete	pmlsd	send	
edit	mdir	preserve	sendport	
epsv4	mget	sendport	set	
exit	mkdir	progress	site	
		prompt		

Table 1-3 *FTP command set*

Test Yourself!

- Access a command line window and try connecting to an FTP site using the commands mentioned above – `ls`, `put`, `get` . . .

1.3.2 Problems with FTP

There are some possible problems when using FTP that you should be aware of. Most machines connected to the Internet have some kind of firewall in place to stop intrusion. A firewall can be part of your operating system, a piece of software you install or included on a piece of hardware on your network. A firewall may be set up to stop access either

outward bound or inward toward your computer, on any of the available ports. This includes the FTP port 21. So, if your computer is having connection problems it's worth checking this; it is usually possible to allow communication on specific ports while others are blocked.

It must also be remembered that normal FTP on port 21 is not very secure and it is possible to intercept and view data that is sent, as well as any passwords. This is because information is sent as clear text without any form of encryption. If the data is sensitive or personal, other methods exist for transfer of information.

Lastly, if you use FTP to transfer files for your Web site, don't forget that any newly uploaded pages may take a while to appear to a browser due to factors such as local caching (a store to help speed up regularly viewed pages) on individual machines and servers.

Checkpoint!

We now know how to:

- **open an FTP connection to another machine**
- **do various operations such as listing the files on the remote server**
- **get single and multiple files from the server**
- **put single and multiple files on the server**
- **close the FTP connection**
- **be aware of problems concerning FTP such as security and firewalls.**

1.3.3 Email

Email existed as long ago as 1965 on time-shared user mainframes and pre-dates the Internet itself. The use was extended to work on networks between many computers. Modern email works by a user writing their message using a mail client (or Mail User Agent, MUA); this program then uses Simple Mail Transfer Protocol (SMTP) to send the message to the local mail transfer agent (MTA), which usually exists on the user's ISP. The MTA then deciphers the email address of the recipient (who the email is to!), which takes the form of myname@myaddress, where myname is the local part and myaddress is the domain name. The MTA uses the DNS to find the appropriate mail exchange server accepting messages for that domain. Once the mail server is found, the message is sent on using SMTP, and from there it is placed using the local name (myname) to find the correct mail box.

To pick up the mail, a user's client retrieves it from the mail box via his MUA, probably using Post Office Protocol (POP3).

1.3.4 Instant Messaging

It's possible to use an Instant Messaging (IM) service to talk to a friend on another computer, so how does this work? The problem here is that the protocols and programs in use are fairly, at this point in time, non-standard in the sense that each has its own approach. One of the more popular is Internet Relay Chat (IRC), which is designed for group communication in channels (group areas where people meet) but also allows one to one communication. IRC is an open protocol that uses TCP and if desired, Secure Sockets Layer (SSL). Users connect with a client application (of which there are many, for each type of operating system), which links to a IRC server. The actual protocol is plain text; that is, it is quite possible to link to an IRC server with Telnet. However, it is easier to use a client as some character encoding can cause problems and the client makes it considerably easier with built-in commands, for example.

An interesting development on IRC were bots, automated clients that perform some duty or service, such as exercising operator privileges (controlling channels and acting, if abuse is detected). There are also bots that perform the opposite to annoy users by sending out unwanted messages to them!

1.3.5 Remote Machine Access

If you need to access a machine from a distance, to maybe run a program or retrieve some data, there are a few ways to do this. One way is through Telnet, which allows you to open a connection to a remote machine and issue commands as if you were actually sat at that machine. It is important to understand that the other end of the connection must be a process (or daemon) that will act as a server. Telnet uses port 23 and is largely now considered a security risk due to vulnerabilities that have manifested themselves over the years. This comes from the degree of flexibility allowed by the Telnet program and also from weaknesses within the protocol. Telnet does not encrypt any communication, so passwords are open to eavesdropping, which can be done fairly easily. It is also possible to hijack (or, take over) a Telnet session due to the lack of authentication between the parties, so it's impossible to know whether parties are who they pertain to be. A better means of remote access is to use SSH (Secure Shell), which is, like Telnet, both a program and a protocol. A typical SSH session runs like this (user typing in bold):

```
Raptor-Computer:~ ralphmoseley$ ssh ralph@192.168.1.4
Password:[type password]
Last login: Sat Mar 12 18:50:54 2005 from 192.168.1.2
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
The Regents of the University of California.  All rights
              reserved.

FreeBSD 4.10-RELEASE (GENERIC) #0: Tue May 25 22:47:12 GMT 2004
```

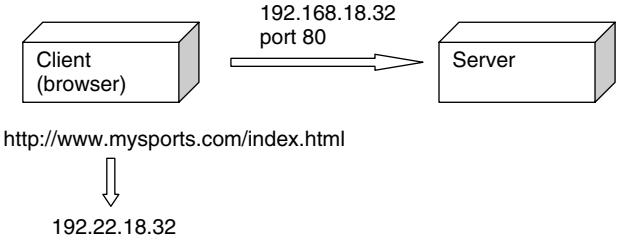
```
Welcome to FreeBSD!

aphid# ls
.cshrc   .login          .rnd            server.csr
.history .mysql_history  mbox            server.key
.klogin  .profile        server.crt
aphid# logout
Connection to 192.168.1.4 closed.
```

1.3.6 **Web Pages**

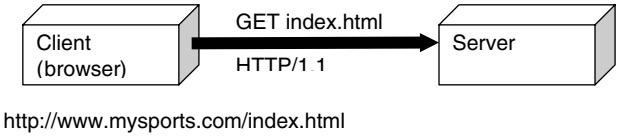
The main protocol used for communication between a browser and a Web server is HTTP. This protocol was designed to enable documents to be transferred but can be used with other types of data too. For Web documents the HTTP protocol works by sending commands over a TCP connection.

To understand how information gets passed from machine to machine we need to know how such systems can connect to each other. Generally, the main model used for the Web is client–server.

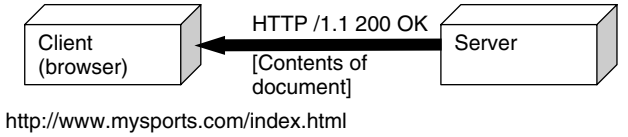
The first stage is the user typing a URL in the browser address window.



In the next stage the URL is converted to an IP address, which is then used to make a connection to the server at that location via port 80, the one used for HTTP and the Web.



Once the connection is established, the client application extracts the file name that is required from the URL and sends the request down the established connection. When received, the server looks up the request.

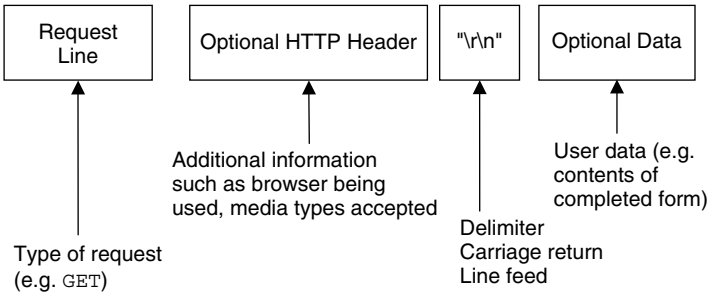


If all is well the HTTP message is sent saying that the page was found, followed by the page itself. When the page has been sent the connection is dropped.

The following HTTP command may originate from a browser to request a Web page from a server:

```
GET /index.html HTTP/1.0
```

This command, which is a text-based command, as above, has several fields:



In fact, it is quite easy to pretend to be a Web browser! Using a Telnet program it is possible to connect to a Web server and ask for the initial page, usually index.html or index.htm. The session may run something like this:

```
$ telnet aphid.dynalias.net 80
Trying 81.155.138.148...
Connected to 81.155.138.148.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 200 OK
Date: Fri, 04 Mar 2005 20:02:01 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.6 mod_ssl/2.8.16
              OpenSSL/0.9.7d
```

```
Last-Modified: Mon, 06 Sep 2004 12:43:33 GMT
ETag: "4c9003-a71-413c5b75"
Accept-Ranges: bytes
Content-Length: 2673
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<!--Web page here -->
</HTML>
Connection closed by foreign host.
```

The initial command opens the connection to the server. The first thing the computer does is translate the name of the server to an IP address by using DNS. It then can link to the server and form a temporary connection for requests. Once established, commands can be written just as a browser does. Here there is a request for a Web page, `index.html`, using protocol HTTP 1.0. The server then responds by saying the command was understood, followed by some other identification data and then by the Web page itself. Finally, the connection is closed as the request has been dealt with.

Lots of things can go wrong when asking for a Web page in this way! You may write the command in carefully but the page doesn't exist, in which case you will receive an error code back instead of a Web page. Some of the more common ones are:

- 404 = Not found
- 401 = Unauthorized
- 500 = Internal server error
- 501 = Not implemented.

There are quite a few more too, such as errors involving redirection services. A browser will also send other information along with the request, identifying its various properties and capabilities but not the user.

Test Yourself!

- Using a command line window as above, type the telnet command to talk to a Web server on port 80 e.g.:

  `telnet www.yahoo.com 80`

  `GET /index.html HTTP/1.1` [press enter twice]
  If you type this correctly you could get a few responses:

The correct page
A redirection page
Can't find it page

All will be in HTML, as you would expect! The connection will then close (it's not a persistent connection, remember).
You also try this with some terminal programs such as PuTTY.

1.4 SECURE CONNECTIONS

Most of the protocols that have been looked at are not secure; they use plain text to transfer data and could be viewed or tampered with at some stage. There are some protocols that make it more secure to send and receive data, which include special versions of FTP as SFTP/FTPS and HTTP as HTTPS. Other ways of actually executing and logging into a machine exist too, similar to Telnet. These include SSH and various virtual networking tools that allow a user to create a secure tunnel through to the host machine, almost as if the connection were local. Table 1-4 shows the various protocols, associated ports (which can be changed) and whether security is available with that particular communication mode.

SSH allows communications to take place in a secure manner over port 22, with encryption, minimizing the risk of interception or tampering. The SSH program is similar to Telnet in that it allows a user to log in and execute commands on a remote computer. It is usually included with UNIX (and its variants) as a program. A server process/daemon usually also exists to accept incoming requests. Other operating systems have similar programs available that are freeware or shareware, such as WinSCP.

To transfer files in a similar way to FTP there exists a version that uses a secure method of communication, SFTP. It is important to note that there are several protocols with this name: the one referred to here is Secure (Shell) File Transfer Protocol (another one is Simple File Transfer Protocol). SFTP can refer to the protocol or the application program involved, depending on the context it is used in. It is not simply an FTP connection run through a secure shell but a completely new protocol.

Yet another version of FTP exists as FTPS, which utilizes Secure Sockets Layer (SSL)/ Transport Layer Security (TLS) to secure any connections made.

For shopping, banking and financial transactions on the Web another protocol was called for, besides the insecure and easy to intercept HTTP. This is HTTPS and it provides authentication and encryption for electronic commerce by encrypting communications using a version of SSL or TLS. You will know when you encounter it as the URL is `https://`

Communication	Application	Protocol	Port	Security
Web page	Browser	HTTP	80	—
Web page	Browser	HTTPS	443	Secure
Files[Binary/Text]	FTP	FTP	21	—
Files[Binary/Text]	SFTP	SFTP	22	Secure
Files[Binary/Text]	FTP	FTPS	990	Secure
Commands	Telnet	Telnet	23	—
Commands	SSH	SSH	22	Secure
Instant Messaging	IRC	IRC	194	Not usually secure

Table 1-4 *Transfer of various media and appropriate attributes*

rather than `http://`. HTTPS provides a measure of security while the data, such as credit card details, is in transit to the server. However, once at the server and possibly stored in a database, the data may still be open to attack prior to any further transmission to a credit card processor, for example.

Checkpoint!

We now know:

- **the importance of using the correct level of security required by a situation**
- **the choices available to provide a secure alternative**
- **SFTP and FTPS can be used in place of FTP**
- **SSH can be used in place of Telnet**
- **the SSH TCP port is 22.**

1.5 APPLICATIONS AND DEVELOPMENT TOOLS

For example, you can use FTP from a console window in Windows or a UNIX-based system and it will allow you a basic level of ability to move, copy or manipulate files. It is also possible to use a far more complex program that enhances capabilities in some sense. This may all be presented in a graphical manner rather than text driven so, for example, you can drag and drop files from a local folder into the remote server. In the case of FTP it can

be useful to have the ability to transfer groups of files and directories without typing, as it also allows a better interface with your working environment.

There are lots of resources on the Web for developers, some of which are freeware, shareware or proprietary. The resources you can find cover most things you will ever need; from fully featured Integrated Development Environments (IDEs), through to simple editors and graphic tools. What will you need to start developing applications? To some extent it depends on whether you take a minimalist approach or whether you prefer environments and tools that automate tasks for you.

You can make do with a simple text editor that probably already exists on your computer, your browser and an FTP client if you want to upload your files to a server. You can get editors that are specific to the languages you are using or download plug-ins to assist you. These exist for HTML, JavaScript, PHP, Perl and most other Web languages. There are syntax checkers, program coloring aids and even editors with built-in mini-servers to test code.

Development environments and editors will also automate tasks such as uploading pages via FTP or the batch conversion of files from one type to another.

All these various tools can be bought (as you usually would) from shops or online. There are also freeware or shareware versions where you pay a small fee or nothing at all. You may encounter the term open source while looking for suitable software, these are projects that are freely available to the general public and are usually ongoing, developed and supported by a volunteer community. The code behind such projects is usually available to view and alter. End users have the right to change such code and redistribute the software. Open source licenses may have some restrictions applied though, such as the requirement to preserve an author's name and copyright statement in the code.

Freeware can be distributed under a different kind of license where, although the code isn't available to view or alter, you do not pay a charge for it. It may possibly have a restriction such that it may not be sold on or used by government agencies or armed forces.

Shareware is usually software that is distributed ahead of payment, which is set to some point in the future so you have time to try it out.

Resources!

- **Open up your browser and, using a search engine, have a look around for various types of utility that may help you. You have various choices for types of license for software:**
  - **freeware is free to use and download**

- **shareware to try and then pay later**
- **off-the-shelf to buy.**

- **You may find FTP utilities and more secure software that use SFTP or FTPS.**
- **Freeware examples include: WinSCP, PuTTY . . .**
- **There are also lots of editors available for every kind of language and platform you are developing on.**
- **You may need tools later to convert files, such as graphics, between different formats.**

1.6 THE WEB BROWSER

Possibly one of the main tools you will be using while developing Web applications is the Web browser. This connects, as we have seen, over the Internet to a Web server that answers requests from selections of hypertext documents. The HTTP protocol is used in this dialog. Pages are located through the use of URLs – Uniform Resource Locators – which usually begin with `http://`, although most browsers will also support `ftp` and `https`.

Most browsers share common features in that they will display graphics and support many media features. However, this doesn't have to always be the case; there are browsers that allow the Web to be visited using text only, which is very fast.

The format of incoming Web pages is HTML which is interpreted by the browser and displayed as the instructions describe. In addition to HTML most browsers support other types of file such as JPEG, GIF and PNG. More can be supported using plug-ins – units of software that can be added into the browser.

There have always been problems with standards and issues over compatibility, ever since browsers came about on various machines and platforms. When a new browser was made available there were things it did slightly differently, which led to that difference making a required change in HTML. In this way there was an evolution of the language over time, which allowed new browser features to be used, such as displaying images, changing colors or fonts. This led to several strands of the language developing and it therefore being non-standard. Standards were introduced to bring about the ability for pages to look the same no matter what browser they were loaded into. There are now many modern browsers that will work within the standard versions of HTML and XHTML (Extensible HyperText Markup Language).

Browsers have expanded their capabilities beyond simple HTML rendering and often have support for IRC, newsgroups and email.

1.6.1 Choices

So, if you are going to develop applications, or even just Web pages, which browser do you choose? The best way to answer this is to ask yourself what you need and, probably more importantly, what your visitors are going to be using.

You could develop using a browser that you find you are comfortable with, then check on a few popular browsers. What features do browsers generally have? Obviously, their main function is to show text and graphics and possibly other media. Other features exist though that you should try and familiarize yourself with. For example, you should know that browsers can also open local files, usually under 'files' on the menu. A useful function of most browsers is the ability to look at a Web page's source code, which is usually under 'view' on the file menu. This can be useful when developing HTML, JavaScript or even to just check that the page you expect to load is the one displayed.

Most browsers include a section in 'preferences' or 'tools' where you can customize to some extent or set up special options. It's possible, for example, on most to set your home page (the place where the browser initially will go to, or visit when you click on the home button).

Browsers also use a cache, a local memory area, to speed up the loading of regularly visited pages. This is done by simply storing a copy in the client computer, which it will show when the page is requested again. You can usually adjust the size of this buffer, along with some other options such as how often a page in the cache is updated. Another option is to clear the cache, a very useful function at times! For example, if you are developing Web pages, what happens if a page you are working on gets cached? It may not update when you refresh (ask it to reload) the current page view. You could turn caching off or put a cache size of 0. As well as these methods there are ways of turning off caching through the Web pages themselves, which we will look at later.

Another set of options in your preferences on a browser relate to cookies. Cookies are small pieces of information that identify you to an application when you (re)visit the site. This may be to set up preferences so that when you do visit again, it knows what you like and will change the screen and greet you. This is not generally taken as foolproof for security; you will usually be asked for a password to verify if it is appropriate. For various reasons you may not want to be identified or have your habits tracked, so you can usually switch off the acceptance of cookies or set it so you're asked when one is presented.

Other options relate to the auto-filling of forms – do you want your details automatically added when a site gives you a form to fill out? Another useful option allows you to select where files that you want to download (such as MP3 or programs) will go.

Both cached pages and cookies can be cleared from memory, either individually or all together.

Test Yourself!

- Explore the various options of your favorite browser.
- How big is the cache?
- How do you clear all cookies and the cache?
- How do you stop cookies being accepted?
- Change the home page.
- Where do files go to that you download, the desktop or . . .?
- Load a page from a Web site, or one of your own, and look at the source code through the browser – is it what you expect?

Checkpoint!

We now know:

- **a browser contains quite a few interesting features beyond simple site visiting and viewing**
- **a home page can be changed and set to whatever you want**
- **a cookie is a small piece of information usually to identify a visitor the next time a site is visited**
- **a cache is a memory area, or buffer, that is used to store Web pages that are frequently visited**
- **how to change and view where downloaded files go.**

1.7 CHAPTER SUMMARY

- **The Internet was developed as the need to communicate information between remote locations became evident.**
- **The WWW is a large subset of the Internet and was developed as a way of connecting information that could be universally accessed.**
- **The main model for communication on the WWW is client–server based.**
- **HTTP was developed as a protocol to work for document transfer and is essentially request/response in nature.**
- **Other protocols exist for different uses, such as the transfer of files between machines.**
- **Many useful command line and applications exist that utilize the various protocols to help with file transfer and remote access.**

- **Some problems do exist with certain protocols, which can be overcome.**
- **Security should always be considered when transferring or accessing information over the Internet.**

Chapter Quiz

- See if you can find out a little more about the origins of the Internet by using search engines such as Google. You can also use the useful Web addresses at the end of this chapter.
- Other models of communication exist than just client–server as in the case of using a browser. Can you think of any software you use that may use a different kind of approach?
- Is the computer you use for development protected by a firewall and, if so, how is it configured to allow various communications through the ports, such as FTP?

Key Words and Phrases

FTP File Transfer Protocol, a standard for transferring files between computers

FTPS File Transfer Protocol over SSL, another version of FTP running over SSL/TLS for security

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol, the main method of transferring information on the WWW, using a request/response mechanism

Internet When written with a capital 'I', this refers to the publicly available system of interconnected networks that communicate standardized protocols such as IP

IP Internet Protocol, one of the main protocols used on the Internet

Port An interface for communicating with a computer program over a network

Protocol A standard, or set of rules and conventions that enables communication between two systems. A protocol can exist in hardware terms as well as software

SFTP Secure Shell (SSH) File Transfer Protocol, a more secure protocol for transfer of files

SMTP Simple Mail Transfer Protocol, the standard text-based method for transferring email

SSH Secure Shell, both a protocol and a command line program for connecting to remote computers

SSL Secure Sockets Layer, a secure protocol with encryption

TCP Transmission Control Protocol, a connection-oriented protocol with reliability, working at the transport layer level

TLS Transport Layer Security, the successor to SSL with few differences between SSL 3.0 and TLS 1.0

UDP User Datagram Protocol, a fairly minimal message-oriented transport layer protocol, providing no guarantee for message delivery

URI Uniform Resource Indicator, an Internet protocol element that consists of a string of characters that indicate a name or address referring to a resource

URL Uniform Resource Locator, a standardized address for some resource on the Internet or elsewhere; it is a type of URI

WWW Refers to the World Wide Web, an information space and large subset of the Internet, where items (known as resources) are accessible via links

Useful Web Addresses

`http://www.w3.org/`

`http://www.isoc.org/internet/history/`

`http://en.wikipedia.org/wiki/Internet`