Part 1

Getting Started with Web Programming

Chapter 1 **SAS and the Internet 3**

Chapter 2 **Introduction to HTML 15**

Chapter 3 **Creating Static HTML Output 31**

Chapter 1

SAS and the Internet

Introduction 3

TCP/IP and the Internet 5

Markup Languages 7

Deploying Content on the Web Server 8

- Using the Apache Web Server on Windows 10
- Using the Apache Web Server on UNIX/Linux 11
- Using Microsoft Internet Information Server 12

References 12

Introduction

SAS provides a powerful and sophisticated suite of products for Web application development. As is often the case with SAS, there are usually at least three different ways to accomplish the same task with the available Web programming tools. What is more, many of these tools are new even to experienced SAS users. In order to understand how to use these tools and what they do, the user also needs to be familiar with several other SAS products, including the Output Delivery System (ODS), Remote Computing and Remote Data Services, and SAS Integration Technologies. In addition, although it is not essential, it is extremely helpful to have some familiarity with HTML coding, CGI scripts, and Java programming in order to understand how the various SAS components work.

While it may be true that a wrench and a pair of pliers do pretty much the same thing, there are times when one will work and the other won't. Learning to use the tools that SAS has provided is largely a question of figuring out when the wrench won't fit. The goal of this book, therefore, is to introduce the entire SAS Web development tool kit, to explain what each component does, and to suggest when to use specific features and functions. Some familiarity with SAS syntax is assumed, specifically the DATA and PROC steps, but the discussion of Web programming begins with the most basic kinds of information.

Currently there are hundreds if not thousands of books about Web application development, ranging in coverage from the fundamental to the monumental; several of the more useful ones are referenced at the end of each chapter. Nonetheless, it is not easy to find a one-volume introduction to the subject of Web development that manages to combine comprehensiveness with intelligibility.

At the other end of the spectrum, it would be possible to write entire volumes about each of the topics covered in this book. Consequently, compromises had to be made about how much or how little detail needed to be included. The goal of this book is to discuss the design challenges and available solutions, and to attempt to demonstrate by example how the tools available from SAS fit into this conceptual framework. Note too that this is *not* a book about Web usability.[1] Content and page design are assumed. The focus in this book is on how to get the page designers' brainstorms to work in practice.

As used in the SAS documentation and in the rest of this book, SAS AppDev Studio includes the following product components: SAS Integration Technologies, SAS/IntrNet, SAS/CONNECT, SAS/SHARE, webAF, and webEIS. Since these products are not household words (at least not in most households), all of these additional components will be described in their place, as part of the overall conceptual framework that is the SAS Web development suite.

Strictly speaking, however, the new content in SAS AppDev Studio is just the two modules webAF and webEIS; both of these are available only for the Microsoft Windows environment.

- webAF is an *Interactive Development Environment* (IDE) for building Java applications to access your SAS data and present it on the Web.
- webEIS is an *Online Analytical Processing* (OLAP) report builder for the Web; it is a point-and-click application builder for creating documents and publishing them on the Web as Java applets or JavaServer Pages. (Note that webEIS has been deprecated in SAS®9, although it is still supported in SAS AppDev Studio 3.1.)

Don't worry if you don't know what an IDE or an OLAP is; we will get to them in due course. Learning about Web programming is largely a matter of learning to navigate through a haze of jargon. To explain what all these tools do, it is necessary to use a lot of TLAs (three-letter acronyms). There are also quite a few four-letter acronyms, and even some five-letter ones.

[1] A classic one-volume treatment of this topic is by Jakob Nielson, *Designing Web Usability*: *The Practice of Simplicity* (New Riders Press, 2000).

People have different styles of learning, but relatively few people are blessed with the ability to look at a page of documentation and come away with a picture of what the software is supposed to do. Consequently, much of this book consists of examples. The hope is that if you can decode what the documentation is talking about, it will begin to be useful to you, and you can proceed beyond the simple problems in this book. The remainder of this book takes up the challenge of defining these new technologies and illustrating how SAS tools can be used to create distributed information processing systems.

TCP/IP and the Internet

The *protocol* used to communicate among different computers is now almost universally the *Transmission Control Protocol/Internet Protocol* (TCP/IP). In general, a diplomatic protocol is a set of previously agreed-upon rules for negotiation. In order to send data from one computer to another, there must also be an agreed-upon set of rules for how that data should be addressed and formatted. Computers use different sets of protocols to manage this process. Each protocol is designed for a different purpose, depending on how much reliability and control is needed.

In the case of network-based data transmittal, the important concern is that *all* of the data arrive in the correct order. The TCP/IP protocol was developed back in the 1970s by a team of scientists working on the Department of Defense Arpanet (Advanced Research Projects Agency Network) project.[2] The original project had a number of goals, one of which was to assure that command and control messages could still go out and be received in the event of a thermonuclear attack on the United States.

In order to meet this requirement, the researchers created a protocol that would allow messages to be sent as discrete packets of information via any number of possible routes. They would then be reassembled at the receiving end in the correct order. The Internet Protocol, or *IP*, is responsible for forwarding the packets to the specified Internet address; *TCP* is the set of rules for sending and receiving packets over the physical network, and for catching and correcting transmittal errors. (See http://directory.google.com/Top/Computers/Internet/Protocols for a list of available resources on TCP and IP.)

The global network that became known as the Internet consists of a great many loosely connected clusters of networked computers, all using TCP/IP to communicate. The computers in your home or office network can all talk to one another using TCP/IP, and your network can talk to all the other networks in the world using the same mechanism. It should be noted that there are alternative networking protocols, most notably IPX, which runs on Novell networks, and LAN Manager, which was IBM's contribution. These alternative protocols are dwindling in use, however, due to the enormous impact of the Internet and the World Wide Web, which run on TCP/IP. In this book, the focus is on how SAS AppDev Studio makes use of the features of TCP/IP to manage Web communication.

[2] See Katie Hafner and Matthew Lyon, *Where the Wizards Stay Up Late: The Origins of the Internet* (Touchstone Press, 1998).

The existence of the Internet as a shared resource led to an interest in simplifying the user interface. Clearly, a standardized method for access and display was necessary. In 1989, Tim Berners-Lee, a British computer scientist working at CERN, proposed a global project to allow sharing information over this new medium. He developed the first Web server, using the Hypertext Transfer Protocol (HTTP), and the first Web client. He called the combination of these technologies the *World Wide Web (WWW)*. The World Wide Web first became available on the Internet in the summer of 1991. It is the conjunction of the physical network and the World Wide Web user interface that has led to the enormous growth in Internet connectivity and Web development.

Berners-Lee's brilliant contribution was defining how documents could include embedded *links*, or *hypertext*, which would allow users to connect seamlessly to documents on widely distributed computers. The HTTP standard uses the client/server model previously described. A program running on the server continuously listens for client messages; a second program, called a *Web browser*, runs on the client.

The browser has two jobs. First, it can send messages to the server, correctly encoded in HTTP, using TCP/IP to format and address each message. When the user types the address of a Web server in the browser window, the client sends something like the following request over the Internet to the server at that Internet address:

```
GET /index.html HTTP/1.1
```

The HTTP protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take when receiving a request. When the Web server receives a transmission encoded using the HTTP standard, it attempts to respond appropriately. In this example, it responds by sending back the document *index.html* from a specified Web page directory on the server.

In order to find the right server, the local client has to figure out the correct IP address. It does this by sending a preliminary message to a *Domain Name System* (DNS) server with the name of the Web server it has been asked to locate. The DNS server receives this *Uniform Resource Locator* (URL) and replies with a numeric IP address where the Web server can be reached. The browser then inserts this IP address into the header of the outgoing message and transmits it to the requested Web server.

You can also type in an IP address directly as the URL. IP addresses are familiar to most Web users as a set of four three-digit numbers. For example, 66.218.71.81 is the IP address that corresponds to the www.yahoo.com home page. Each of the four fields separated by dots is a number in the range 0 through 255; these are the decimal numbers that can be represented in computer binary language in 8 bits—that is, 2^8 or 256 possible combinations. On most modern computers, 32 bits equals one *word* in storage. Thus four 8-bit numbers were used as the original format of an IP address. As a consequence of the enormous expansion of the Internet, the system is rapidly running out of addresses, and new standards such as IPV6 are currently being advanced to increase the size of possible IP addresses.

The second function the browser provides is the capability to display the received file, using the rules for decoding *Hypertext Markup Language* (HTML) documents. HTML is the set of rules describing the contents of Web files. The development of the HTML standard was what transformed the World Wide Web from an academic curiosity to the ubiquitous entity it is now.

Markup Languages

Standard Generalized Markup Language (SGML) is the standard for organizing the elements of a document. SGML was developed and proposed by the ISO in 1986. This system uses *markup tags* enclosed in angle brackets (<>) to identify and delimit the various parts of a document (header, body, paragraph, and so forth). Although SGML itself is too large and cumbersome to have wide appeal, various subsets of the standard, including HTML and *Extensible Markup Language* (XML) have become tremendously important for international e-commerce. Note that markup languages such as HTML are not programming languages. In form, encoded texts are more like the familiar word processing documents, containing instruction as to how the information contained is to be formatted and displayed. The markup language is simply a set of rules for encoding the text.

XML uses customized tags to provide for verifiable transmittal of data between applications and between organizations. In contrast to HTML, XML was designed to support only the information content of the message; in HTML, this content is combined with the presentation and formatting of the data as well. Most recently, *Extensible Hypertext Markup Language* (XHTML) has been proposed as a way to combine the validation features of XML with HTML formatting capabilities.

Finally, *Dynamic HTML* (DHTML) refers to Web content that can change each time it is viewed. It is important to note that the term is frequently used to refer to two quite different things. The first meaning, which is the one that is used in this book, is simply Web content that can change each time it is viewed.

The second use refers to competing proposals from Microsoft and Netscape to the World Wide Web Consortium (W3C) for various extensions to HTML that allow a Web page to react to user input without sending requests to the Web server. The current position of the W3C on DHTML is as follows:

> "Dynamic HTML" is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated. The W3C has received several submissions from members companies on the way in which the object model of HTML documents should be exposed to scripts. These submissions do not propose any new HTML tags or style sheet technology. The W3C DOM WG is working hard to make sure interoperable and scripting-language neutral solutions are agreed upon.
> (See "Why the Document Object Model?" http://www.w3.org/DOM/)

Interested users can find more information on DHTML and DOM in the references at the end of this chapter.

There are two main strategies for managing dynamic Web page content:

- *Client-side* DHTML uses JavaScript, cascading style sheets (CSS) or Java applets.
- *Server-side* content can be distributed using Common Gateway Interface (CGI) scripts, PHP: Hypertext Preprocessor, Java servlets, or JavaServer Pages (JSP).

In addition, Microsoft has developed a parallel set of technologies, including JavaScript and *Visual Basic Scripting Edition* (VBScript) which can be used to create *Active Server Pages* (ASP) on the server.[3] These latter all require some version of the Windows operating system, and are explicitly integrated with Microsoft *Internet Information Services* (IIS) and SQL Server, the Microsoft relational database management system.

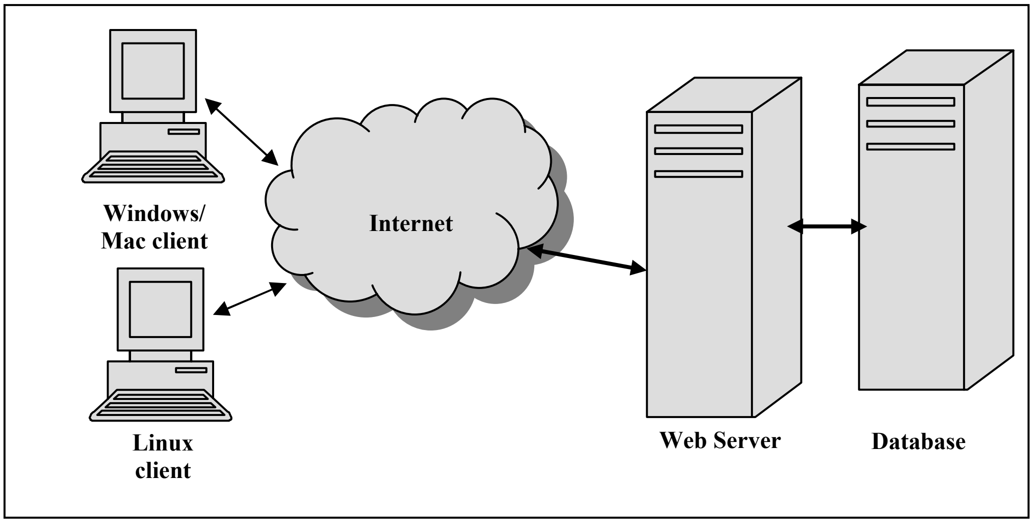
Since the introduction of SAS 5 in the 1980s, SAS software is and has been largely platform-independent. The SAS AppDev Studio toolkit is available only for Windows, however. Nonetheless, Web pages developed with SAS AppDev Studio can be deployed equally well in the UNIX and mainframe server environments as on the Windows platform, although there are necessarily some differences in implementation. The examples in this book show how to use the tools available for the Linux environment as well as for the Windows XP and Windows 2000 platforms.

Deploying Content on the Web Server

So far, the term "Web server" has been loosely used to mean two quite different things. Strictly speaking, a Web server is not a computer; it is a computer program. Still, most people use the term "server" to refer both to the software and to the hardware on which it runs.

The usual model for a *three-tier* Web computing environment looks something like the following figure:

Figure 1.1 Typical Web Client/Server Configuration



In this design, the client computers are connected to the Internet (via TCP/IP), which in turn is connected to the computer on which the Web server software is running. In addition, the Web server can talk to a database server running on a third computer system.

[3] ASP is an older technology from Microsoft; the current platform for delivering active server content is called ASP.NET 2.0; see http://www.asp.net/ for more information.

While this is a common model, in principle all three programs—the client, the Web server, and the database server—could be on two computers, or even all on one. In the previous figure, the data server is located on a different hardware platform from the Web server. In many installations, the two servers are both on the same system. In either case, the SAS/SHARE, SAS/CONNECT or SAS Integration Technologies products must be licensed in order to use SAS as the back-end database service; see Chapter 4, "Remote Access to SAS Data" for more detail on how to implement this connection.

The neat trick about TCP/IP is that it does not know or care where the destination IP address is actually located. The client computer is perfectly happy talking to a Web server that happens to physically reside on the same machine. When you install SAS AppDev Studio on a Windows client, the installation routine offers to install a Web server for you so that you can test your Web pages. The server that currently comes bundled with SAS AppDev Studio is from the Apache Software Foundation (http://www.apache.org).

The Apache HTTP server has been the most popular Web server on the Internet since 1996, with about 70% of the installed server base of 75 million sites (as of December 2005, according to http://www.netcraft.com/survey). Several other Web server programs are available, in particular Microsoft IIS with about a 20% market share,[4] along with various others, none of which has more than 3% penetration.

As a historical note, the Apache HTTP server software was originally developed as a voluntary, part-time project:

> In February of 1995, the most popular server software on the Web was the public domain HTTP daemon developed by Rob McCool at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign. However, development of that httpd had stalled after Rob left NCSA in mid-1994, and many webmasters had developed their own extensions and bug fixes that were in need of a common distribution. A small group of these webmasters, contacted via private e-mail, gathered together for the purpose of coordinating their changes (in the form of "patches").
> ("How Apache Came to Be," http://httpd.apache.org/ABOUT_APACHE.html)

Consequently, the project became known as "a patchy" server.

The two main reasons why Apache is so dominant are (1) it works, and (2) it's free. In addition, because there are over 50 million Apache installations worldwide, there is a great deal of free support available from user groups and other resources. The big drawback with Apache (and corresponding advantage for IIS) is that since the former is open source, if it doesn't work you have to figure it out yourself. There is no service number to call when things go wrong (but no service fees, either!).

If you are working in an environment where you have access to a remote Web server, you need to find out from your system administrator whether it is (1) an Apache server on Windows, (2) Microsoft IIS, or (3) an Apache server on UNIX (or Linux). The examples that follow focus on Apache, but IIS is conceptually similar. All Web server programs serve HTML pages from a specific directory; the main difference is just the name of the folder where the pages are located.

[4] A recent Microsoft press release indicates that in June 2005 ASP and ASP.NET represented about 44% of the content on Fortune 1000 corporate application servers. This is probably a result of the fact that larger companies are less likely to reply on free software. (http://www.port80software.com/about/press/060105)

Using the Apache Web Server on Windows

Apache was originally written to run on various flavors of the UNIX operating system. While earlier versions of Apache, notably 1.3, came with warnings that the Windows performance was inferior to the UNIX versions, the newest releases (starting with version 2.0) are said to work reliably with current versions of the Windows operating system. You can download a copy of the binary executables for most operating systems from the Apache Web site at http://httpd.apache.org/, or if you are installing SAS AppDev Studio on a client system, the setup routine will ask you if you want to install a copy locally.

Since SAS AppDev Studio is a Windows-only product, the version of the Apache Web server supplied is intended for Windows NT, 2000, and XP. SAS has automated the installation and configuration of the server program so that it is fairly simple to operate in the Windows environment. On a Windows system, you can define Apache as a service, and start and stop it from the **Start ▶ Programs** menu. This is the recommended approach, since that way the Web server will start automatically when you reboot your computer. Just go to **Programs ▶ Apache Web server ▶ Apache as a Servic**e and select **Install Service**. You can start and stop the service from the same menu.

In order to display HTML content, the pages must first be copied to a specific directory on the Web server system. Under Windows, it is possible to copy HTML documents to the server by mapping a network drive (Z: for example) using Windows Network Neighborhood, and then just dragging or dropping the HTML files to this drive. (As we shall see, this task is somewhat more complex on a UNIX or Linux system.) Even if the Web server is on the same PC you are using, it is still a good idea to map a drive to the directory where you want to display your Web pages.

Unless otherwise requested, the SAS AppDev Studio setup routine will install the Apache 2.0 server in **`C:\Program Files\Apache Group\Apache2`**. Within the Apache folder, the executable file **`apache.exe`** is located in the subdirectory **`bin`**. The default root directory for HTML documents is **`htdocs`**. Default file locations for Apache are specified in the configuration directory **`conf\httpd.conf`**. Unless you know what you are doing, you should not try to change these yourself. On a corporate Web server, you almost certainly will not have permission to edit this file; if you have installed Apache on your local workstation, you probably do not need to change the defaults anyway.

As noted above, the folder **`C:/Program Files/Apache Group/Apache2/htdocs`** is the default Apache document root directory. Unless you tell it otherwise, Apache will try to serve Web pages from this folder. You do not need to (nor should you) try to specify "htdocs" in the URL. If the name of the Web page is **`example.htm`**, located in the **`htdocs`** folder, the URL for this Web page would be http://<server-name>/example.htm.

At most sites, users do not have write access to the **`htdocs`** folder on the Web server. In this case, the user has to contact the system administrator for a directory structure with the correct access and permissions. The URL would thus contain an alias to this folder—for example http://<server-name>/~username/example.htm. If you get the nasty message "The page you are looking for is currently unavailable," the HTML file is most likely not in the right directory. This would be a good time to get help from someone who has tried this before on your Web server.

If your Web server is on a different network, including one on the other side of the world, it is still possible to map a drive locally using *WebDAV* (Web-based Distributed Authoring and Versioning). If your server is configured to support Web DAV, you can set up a client PC to access a Web directly using HTTP. Although it is technically possible to do this using Web Folders in Microsoft Office or Internet Explorer, most people prefer to use a dedicated client application such as NetDrive or WebDrive; see the resources listed at the end of this chapter for information about these products. Using WebDAV greatly simplifies the process of managing Web content. Check with your system administrator to find out whether WebDAV is supported in your environment.

Using the Apache Web Server on UNIX/Linux

In a Windows environment, as long as you have permission to write to the public documents folder, you can just map a drive to this folder and drag and drop your Web pages there. On UNIX systems, transferring documents is slightly more complex. The default directory paths are specified in the file **`conf/httpd.conf`** under the server root directory. As noted above, these values must be supplied at installation time by the system administrator. If you are the system administrator and you installed Apache yourself, change to the Apache directory and try typing the following command:

```
grep DocumentRoot conf/httpd.conf
```

This should list out the line in the file containing the value assigned to this parameter. One common location for sites running Apache 2 is **`/usr/local/Apache2/htdocs`**. Earlier versions of Apache such as 1.3 used **`/var/www/html`**, and the configuration files were in **`/etc/apache2/conf`**.

The UNIX system administrator has to set up a password and some space on the Web server for each user. Ask what directory you should use for your Web pages, and whether you should use File Transfer Protocol (FTP) or Secure File Transfer Protocol (SFTP) to transfer them. FTP is the TCP/IP protocol for copying files from one computer to another. You can use it for copying files from one UNIX system to another, or from a Windows system to a UNIX server. FTP is a relatively old protocol. Unfortunately it has one major security problem. When you type in your user name and password, they are sent over the network unencrypted. This is bad enough when connecting to an FTP server on your LAN or Intranet, but it is a real no-no when sending a file over the Internet to a remote server. Anyone can find out your password just by monitoring the network traffic. Consequently most sites are now requiring SFTP. This is easy to set up, but again, you need to talk to your system administrator about what you need to do at your specific installation.

In either case, the syntax is easy, if you just follow these steps:

1. On a Windows client, open an MS-DOS window. On a UNIX system, open a terminal window. In either case, you should have a prompt character after which you can type commands.
2. Open a connection to the remote system by typing **`ftp`** ***`host-name`*** or **`ftp`** ***`host-name`***, where ***`host-name`*** is the name or IP address of the remote computer.
3. You will be prompted for your user name and password. Use the ones you got from your system administrator.
4. You may need to change to your directory. Type **`cd`** ***`name`***, where ***`name`*** is the path to the directory where you want to put your HTML pages.
5. To transfer the files, type **`put`** ***`name`***, where ***`name`*** is the name of the HTML document you want to display.
6. Type **`quit`**.

Note that a variety of GUI-based clients support FTP and SFTP; using these allows you to drag and drop files to server directories, assuming your user ID has the proper permissions to do so.

You should now be able to open a Web browser on your PC and type in the URL of the document you have just copied. Just as with the Windows version, this URL will consist of the name of the server (which you found out from the system administrator), followed by any specific directory locations (like **`sasweb`**), followed by the name of the HTML page you want to display.

Using Microsoft Internet Information Server

The Microsoft Windows Server and Windows XP Professional editions come with IIS included. This product does not run under UNIX, but for sites with Windows servers, it is an ideal choice. The Web server can be installed in a few minutes from the operating system installation CD, or by going to **Start ▶ Control Panel ▶ Add or Remove Programs ▶ Add/Remove Windows Components**.

The default installation directory for IIS is **`C:\InetPub`**; the folder **`wwwroot`** is the root directory for serving Web pages. Once IIS has been installed on the server, you can get detailed instructions on use by opening http://<server-name>/iishelp, where <server-name> is the host name for your Web server.

Whether your server is running Apache or IIS, locally on your PC or in Australia, the idea is the same. There will be one specific directory on the server where you want to copy your HTML documents. The URL to this directory will consist of the server name, possibly followed by the path to your folder, followed by the name of your HTML document.

Now that you know how to deploy Web pages, it is time to start creating a few. The following chapter is a short introduction to using HTML to create Web pages. If you are familiar with HTML, you may want to go directly to Chapter 3, “Creating Static HTML Output,” which covers several options for creating static Web pages with SAS.

References

SAS Publications

URL references are current as of the date of publication.

- SAS Institute Inc. 2001. *Getting Started with AppDev Studio*. 2nd ed. Cary, NC: SAS Institute Inc.
- SAS Institute Inc. 2001. *SAS Web Tools: Overview of SAS Web Technology* (Course Notes). http://support.sas.com/training/us/crs/wovr.html

Web Programming

As of December 2005, there were 2200 volumes on the topic “Web programming” at http://www.amazon.com. The following are the works cited in the text.

- Cooper, Alan. 2004. *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How To Restore The Sanity*, 2nd ed. Indianapolis, IN: Sams.
- Hafner, Katie and Matthew Lyon. 1998. *Where the Wizards Stay Up Late: The Origins of the Internet*. New York, NY: Touchstone Press.
- Krug, Steve. 2005. *Don't Make Me Think: A Common Sense Approach to Web Usability.* 2nd ed. Indianapolis, IN: New Riders Press.

- Nielson, Jakob. 2000. *Designing Web Usability*: *The Practice of Simplicity*. Indianapolis, IN: New Riders Press.
- Torvalds, Linus and David Diamond. 2001. *Just for Fun: The Story of an Accidental Revolutionary*. New York, NY: HarperBusiness.

Links

- Apache HTTP Server Project – http://httpd.apache.org/ABOUT_APACHE.html
- Document Object Model – http://www.w3.org/DOM/
- Internet Protocols – http://directory.google.com/Top/Computers/Internet/Protocols
- Microsoft Internet Information Services (IIS) – http://www.microsoft.com/WindowsServer2003/iis/
- Novell NetDrive 4.1 – http://www.novell.com/documentation/ifolder21/netdrive/data/a2iii88.html
- South River Technologies WebDrive – http://www.webdrive.com/products/webdrive/
- WebDAV – http://www.webdav.org/
- Web Server Survey – http://www.netcraft.com/survey
- Working with Distributed Authoring and Versioning (DAV) and Web Folders – http://support.microsoft.com/kb/q221600/