Part I Getting Started with Web Programming

Chapter 1 SAS® and the Internet

Introduction

SAS® AppDev Studio™ is a powerful and sophisticated suite of tools for Web application development. As with most of the SAS System, there are usually at least three different ways to accomplish the same task. What is more, many of the Web programming tools are new even to experienced SAS users. In order to understand specifically what AppDev Studio does the user also needs to be familiar with several other SAS products, including the Output Delivery System (ODS), SAS/SHARE and SAS/CONNECT. In addition, although it is not absolutely essential, it is extremely helpful to have some familiarity with HTML coding, CGI scripts, and Java servlets and applets 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. It is hoped that most users will find something of value here.

Note too that this is *not* a book about Web usability.[1] Content and page design are assumed. The focus in this volume is on how to get the page designers' brainstorms to work in practice.

That being said, what is SAS AppDev Studio? The little volume from SAS Publishing, *Getting Started with AppDev Studio*, says that it includes "all of the software products that you need for developing client/server and *n*-tiered Web applications on your Windows PC." That's terrific if you know what an "*n*-tiered Web application" is.

More generally, there are currently two different bundles that U.S. customers can license:

- The AppDev Studio System Edition bundle includes SAS Integration Technologies, SAS/IntrNet, SAS/CONNECT, SAS/SHARE, webAF, and webEIS. These are the SAS product components for Web application developers who already have some SAS software licensed and want to add AppDev Studio.
- The AppDev Studio Standard Edition bundle adds the base SAS, SAS/GRAPH, SAS/FSP, SAS/AF, and SAS/EIS components to the System Edition. This package contains everything necessary for developers who are starting from scratch with SAS.

Since these products are not household names (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.

As used in the SAS documentation and in the rest of this book, AppDev Studio encompasses all of their Web application building tools. Strictly speaking, however, the new content in AppDev Studio is just the two modules webAF and webEIS; both of these are currently available only for the Microsoft Windows environment.

- webAF is an IDE (Interactive Development Environment) for building Java applications to access your SAS data and present them on the Web.
- webEIS is an OLAP (Online Analytical Processing) 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 Java Server pages.

Don't worry if you don't know what an IDE or an OLAP is; we will get to them in due course. 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; learning about Web programming is largely a matter of learning to navigate through a haze of jargon. People have different styles of learning, but there seem to be relatively few people 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.

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

TCP/IP and the Internet

The *protocol* used to communicate among different computers is now almost universally TCP/IP, or the *Transmission Control Protocol/Internet Protocol.* 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 transmission, the important concern is that *all* of the data arrive in the correct order. The TCP/IP protocol was developed back in the 1980s by a team of scientists working on the Department of Defenses 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, and 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 AppDev Studio makes use of the features of TCP/IP to manage Web communication.

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 Hyper Text Transfer Protocol (HTTP) protocol, and the first Web client, which together he called 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 seamlessly connect to documents on widely distributed computers. The HTTP standard he developed uses the client/server model described above. A program running on the server continuously listens for client messages; a second program, called a *Web browser* runs on the client.

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

The browser has two jobs. First, it can send messages to the server, correctly encoded in HTTP, using TCP/IP to format and address the 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 *DNS* (Domain Name Server) with the name of the Web server it has been asked to locate. The DNS Server receives this *URL* (Uniform Resource Locator) 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 numbers of up to three-digits each. 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 0-255; 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 Version 6 of Internet Protocol (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 HTML documents. HTML (Hyper Text Markup Language) 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

SGML (Standard Generalized Markup Language*)* is the standard for organizing the elements of a document 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 XML (Extensible Markup Language) 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 transmission 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 is combined with the presentation and formatting of the data as well. Most recently *XHTML* (Extensible Hypertext Markup Language) has been proposed as a way to combine the validation features of XML with HTML formatting capabilities.

Finally, *DHTML* (Dynamic HTML) refers to Web content that can change each time it is viewed. It is important to note that the term DHTML 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?" at 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, CSS (Cascading Style Sheets) or Java applets.
- Server-side content can be distributed using CGI (Common Gateway Interface), Java servlets, or JSP (JavaServer Pages).

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

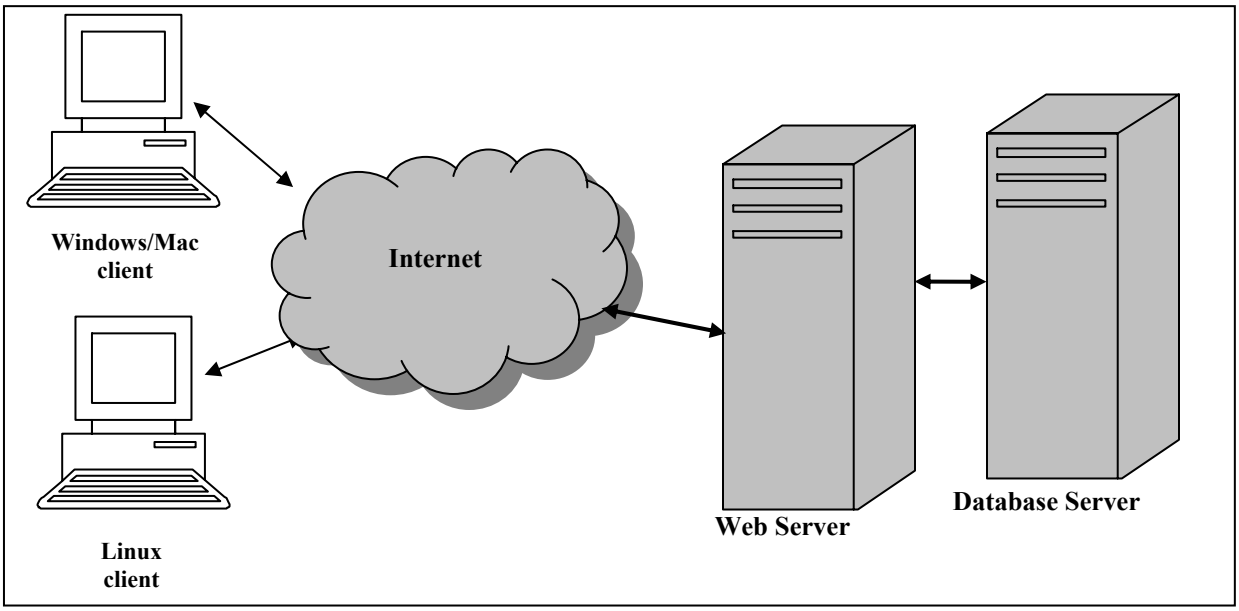
Since the introduction of Version 5 in the 1980s, the SAS System is and has been largely platform-independent. However, AppDev Studio is only available for Windows. Nonetheless, Web pages developed with AppDev Studio can be deployed equally well in the UNIX and mainframe server environments or on the 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 and Sun Solaris environments 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 "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 Figure 1.1.

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.

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 Figure 1.1 the database 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," 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 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 AppDev Studio is from the Apache Software Foundation (http://www.apache.org).

The Apache server has been the most popular Web server on the Internet since 1996, currently with nearly 60% of the installed server base. There are several other server programs available—in particular Microsoft IIS (Internet Information Server*)* with about a 30% market share, and iPlanet from AOL/Netscape, which represents less than 5% as of June 2002 (http://www.netcraft.com/survey).

As a historical note, the Apache project 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 the HTTP daemon 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"). (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 nearly 20 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 it is open source software, 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 from the users’ perspective is just the name of the directory where the pages are located.

Using the Apache Web Server on Windows

Apache was originally written to run on various flavors of the UNIX operating system. Since 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 NT/2000/XP system, you can define Apache as a service, and start and stop it from the **Start** 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 Service** and select **Install Service**. You can start and stop the service from the same menu.

In order to display HTML pages, they first must 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 AppDev Studio setup routine will install the Apache server in **`C:\Program Files\Apache Group\Apache`**. The executable file is called **`apache.exe`**. Within the Apache directory, the default root directory for HTML documents is called **`htdocs`**. Default file locations for Apache are specified in the configuration directory **`C:\Program Files\Apache Group\Apache\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.

If the Web page has been copied to the root **htdocs** directory, and the name of the page is example.htm, the URL for this Web page is http://server-name/example.htm. At most sites, users do not have write access to the **htdocs** directory on the corporate Web server. In this case the user has to contact the system administrator for a directory structure with the correct access permissions. The URL would thus contain an alias to this directory, for example http://*server-name/~username*/example.htm

If you get the message that "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. As noted above, the directory **`C:/Program Files/Apache Group/Apache/htdocs`** is the default Apache document root directory. Unless you tell it otherwise Apache will try to serve web pages from this directory. You do not need to (nor should you) try to specify "htdocs" in the URL.

Using the Apache Web Server on UNIX/Linux

In a Windows environment as long as you have permission to write to the public documents directory, you can just map a drive to this directory and drag and drop your Web pages there. On a UNIX server transferring documents is slightly more complex. 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 *FTP* (File Transfer Protocol) or *sftp* (Secure File Transfer) 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 *Secure File Transfer* using *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 a command 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 **`sftp` *`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`**.

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 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 the Microsoft Internet Information Server

The Microsoft Windows 2000 and Windows XP Professional editions come with a bundled commercial Web server called IIS v5.0.[3] This product is available only for the Windows operating system (it does not run under UNIX), and for sites with Windows 2000 servers, it is an ideal choice. The Web server can be installed in a few minutes from the operating system installation disk, or by going to **Control Panel → Add or Remove Programs → Add/Remove Windows Components**.

The default installation directory for IIS is **`c:\InetPub`**; the directory **`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 directory, 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 next 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.

[3] Windows 95/98 and NT 4.0 included a product called *Personal Web Server (PWS),* but this is not available in the newer releases; see http://www.serverwatch.com/stypes/servers/article.php/15907_1434261 for a comparison of IIS and PWS. Do not confuse PWS with *Personal Web Manager (PWM)*, a Microsoft site management utility which does come with Windows 2000 and XP.

References

SAS

SAS Institute Inc. 2001. *Getting Started with AppDev Studio, Second Edition,* Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2001. *SAS Web Tools: Overview of SAS Web Technology* (Course Notes), Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2001. *SAS Web Tools: Static and Dynamic Solutions Using SAS/IntrNet Software* (Course Notes), Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2001. *SAS Web Tools: Advanced Dynamic Solutions Using SAS/IntrNet Software* (Course Notes), Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2001. *SAS Web Tools: SAS/IntrNet Administration* (Course Notes), Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2001. *Solutions@Work: SAS/IntrNet Software Examples, Multi-User Version,* Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2001. *SAS Web Tools: Developing JavaServer Pages and Servlets Using webAF Software* (Course Notes), Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2001. *SAS Web Tools: Accessing MDDB Data Using webEIS Software* (Course Notes), Cary, NC: SAS Institute Inc.

Web Programming

As of December 2002 there were 1910 volumes on the topic of Web programming on www.amazon.com. The following are the works cited in the text, plus a couple of useful additions.

Hafner, K., and M. Lyon. 1998. *Where the Wizards Stay Up Late: The Origins of the Internet*. Touchstone Press.

Leiden, C., M. Wilensky, and J. Landry. 2000. *TCP/IP for Dummies*. 4th ed. New York: John Wiley & Sons.

Nielson, Jakob. 2000. *Designing Web Usability: The Practice of Simplicity*. Indianapolis, IN: New Riders Publishing.

Torvalds, L., and David Diamond. 2001 *Just for Fun: The Story of an Accidental Revolutionary*. New York: HarperBusiness.

Links

Internet Protocols - http://directory.google.com/Top/Computers/Internet/Protocols

Document Object Model - http://www.w3.org/DOM/

Web Server Survey - http://www.netcraft.com/survey

Apache HTTP Server Project - http://httpd.apache.org/ABOUT_APACHE.html

Apache on Windows - http://httpd.apache.org/docs/windows.html

IIS and PWS - http://www.serverwatch.com/stypes/servers/