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 4 SAS and XML 57

Chapter 1

SAS and the Internet

Introduction 3
SAS Web Technologies 4
TCP/IP and the Internet 5
Using PROC HTTP 7
Markup Languages 7
Deploying Content on the Web Server 9
Using the Apache Web Server 10
Using Microsoft Internet Information Services 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 them 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, the SAS Open Metadata Architecture (OMA), and the SAS Intelligence Platform. In addition, although it is not essential, it is extremely helpful to have some familiarity with HTML 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 majority of the components in the SAS Web development toolkit, 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 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 this 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. Content and page design are assumed. The focus in this book is on how to get the page designers' brainstorms to work in practice.

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.

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.

SAS Web Technologies

This book covers many (but not all) of the components of three of the Web technologies available in SAS 9.2:

1. SAS/IntrNet - create and deploy Web-enabled applications using SAS and the Common Gateway Interface (CGI)

    - *Application Dispatcher* – run SAS applications and display the result in a browser window (Chapter 7).
    - *htmSQL* – a Common Gateway Interface (CGI) program that allows SQL processing from a Web page (Chapter 8).

2. SAS Business Intelligence (BI) Server and Enterprise BI Server software suites

    - *SAS Information Delivery Portal* – manage Web portal content; only available in EBI (Chapter 10).
    - *SAS Web Report Studio* – IDE to create Web-enabled reports (Chapter 11).
    - *SAS Information Maps and OLAP Cubes* – create information maps from SAS or relational databases or OLAP cubes used to support SAS Web reports (Chapter 12).

- *SAS Stored Processes* – enable client applications to execute SAS programs stored centrally on a server (Chapter 13).

3. Web applications

- SAS AppDev Studio – a set of plug-ins for the open source Eclipse IDE to develop customized JavaServer Pages (Chapter 15).
- SAS BI Web Services – enable client applications to send XML requests and parameters in a SOAP envelope (Chapter 16).
- Using SAS with JavaScript and AJAX (Chapter 17).

As noted, each of these products is described in the sections that follow. For more information, including a list of the components that are not covered in this book, see the references at the end of this chapter. The specific products included in each bundle may change; you should always discuss your specific needs with your SAS Software representative to get the most recent information.

The remainder of this chapter explains some of the details of setting up, configuring, and using Web server software.

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. The original project was intended simply to allow researchers to communicate among various universities and academic institutions, but it quickly became obvious that this system might also be used to assure that command and control messages could still go out and be received in the event of an attack on the United States.

The ARPANET researchers created a protocol that would allow messages to be sent as discrete packets of information via any number of possible routes. Each packet contains the address of the sender and the intended recipient. The packets are then reassembled at the receiving end in the correct order. IP addresses are familiar to most Web users as a set of four three-digit numbers. For example, 98.137.149.56 is one of the IP addresses that correspond to the www.yahoo.com home page.[1] 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 many modern computers, 32 bits equal 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.

A *domain name* is just a human-readable label that maps to one or more numeric IP addresses. Domain name lookup and translation to an IP address is provided by networked computers called *Domain Name System* (DNS) servers. Each TCP/IP configuration includes the IP addresses of one or more domain name servers to provide this capability.

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. The global network that has become 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.

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 (European Organization for Nuclear Research), 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.[2]

Berners-Lee's brilliant contribution was defining how documents could include embedded *links*, or *hypertext*, that would allow users to connect seamlessly to documents on widely distributed computers. The HTTP standard uses a *client/server model*. 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 and receive messages to and from 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.

There are eight methods defined in the HTTP protocol; not all are supported on all Web servers. In fact, most servers limit anonymous access to GET, POST, and OPTIONS.

- DELETE – delete a file from the server.
- GET – request a Web document, optionally with parameters to be passed, as from a form.
- HEAD – request only the HTTP response header.
- OPTIONS – returns the methods supported by the server.
- PATCH – apply modifications to a file; for security reasons, this usually requires special permission.
- POST – submit data to be processed from a form.
- PUT – upload a file to the server; usually requires special permission.

- TRACE – echo back the received request.

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 server with the name of the Web server that it has been asked to locate. The server receives this *Uniform Resource Locator* (URL) and replies with the 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.

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 that describe 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 that it is now.

Using PROC HTTP

In SAS 9.2, a new procedure was introduced for sending and receiving HTPP requests, called (logically enough) PROC HTTP. We will see more examples of how to use this useful little utility in Chapter 16, "SAS BI Web Services." For now, the following example illustrates the basic functionality of the procedure. Note that in the preceding section, an HTTP GET request was sent to the server, specifying the relative path to an HTML file to be returned by the server. You can see the workings of this in the following example:

Example 1.1 Download HTML Source Code

```
filename in  "input.txt";
filename out "output.txt";

data _null_;
   file in;
   input;
   put infile_;
   datalines4;
GET /index.html HTTP/1.1

;;;;

proc http in=in out=out url="http://<server-name>";
run;
```

The first DATA step creates a small text file that contains the HTTP request. Note that there is a blank line following the GET request; this is a requirement of the HTTP protocol. The requested file is `index.html`. The HTTP procedure transmits the contents of this file to the server and places the resulting HTML into an output text file. Used in this way, the procedure is equivalent to opening the Web site in a browser window and selecting **View Page Source**. As we shall see later on, it can also be used for much more interesting tasks. It is included at this point simply to illustrate that the HTTP protocol is in fact nothing more than a way to send and receive files from a Web server.

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 the *Extensible Markup Language* (XML) have become tremendously important for international e-commerce.

The body responsible for setting standards and for advancing the state of the art is the *World Wide Web Consortium* (W3C). The W3C's most recent recommendation, HTML 4.01, was finalized in 2001. In 2004 work began on HTML 5, which is intended to eliminate the need for "rich Internet" technologies such as Adobe Flash and Microsoft Silverlight. As of 2011, the details of the HTML 5 standard were still in development and are not yet in final recommendation form (if ever).

Note that markup languages such as HTML are not themselves programming languages. Encoded texts are more like word processing documents, with embedded instructions as to how the information contained is to be formatted and displayed. A markup language is simply a set of rules for encoding the text. Chapter 2 contains a brief introduction to using HTML, explaining some of the more common elements of the language.

XML (see Chapter 4) 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. The *Extensible Hypertext Markup Language* (XHTML) has been proposed as a way to combine the validation features of XML with HTML formatting capabilities, but this has not yet replaced HTML 4 (nor is it likely to).

Finally, *Dynamic HTML* (DHTML) refers to Web content that can change each time it is viewed. The 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 the Document Object Model 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), and/or Java applets.
- *Server-side* content can be distributed using the Microsoft ASP.NET framework, PHP, Java servlets, or Ruby on Rails (Rails or RoR), to name some of the more common approaches.

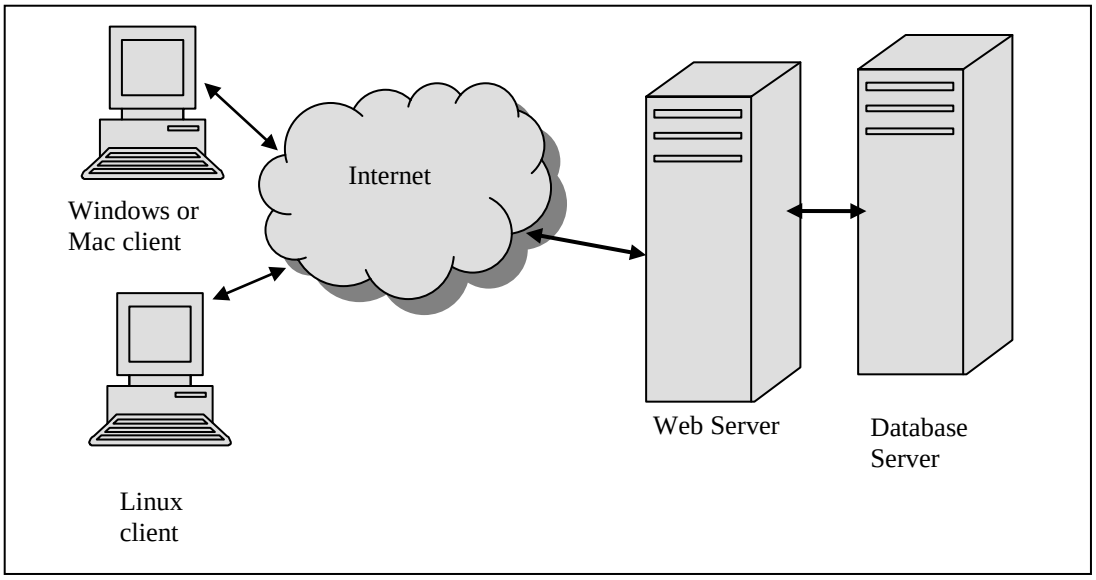
ASP .NET is available only for the Windows platform. Since the introduction of SAS 5 in the 1980s, SAS software is and has been largely platform-independent. SAS/IntrNet uses the Common Gateway Interface, an older but still viable technology, while SAS BI is largely written in Java. The examples in this book show how to use the tools available for the UNIX environment as well as for Windows.

There are two big advantages to creating dynamic content on the server, as opposed to using JavaScript on the client. The first is that different browsers (Internet Explorer, Firefox, Safari, et al.) each handle scripts differently and the same page might appear differently depending on the browser. The second is that the full power of SAS and Java can be used to construct HTML output; the browser need only perform the job of displaying the pages.

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 an *n-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 that may be located on a third computer system. For SAS processing, a common strategy is to split the services on two separate machines, a *Metadata server* (to manage the stored information) and an *Application server* (to run SAS programs). 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 another physical machine.

Currently there are two main types of Web servers, the open-source Apache HTTP server and Microsoft Internet Information Services. The Apache server has been the most popular Web server on the Internet since 1996, with more than half of the installed server base (> 200 million sites as of May 2011, according to http://www.netcraft.com/survey). Microsoft IIS has about a 22% market share, along with various others, none of which has more than 7% penetration. (A 2007 survey indicated that various versions of Microsoft IIS represented more than half 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/101106.asp.)

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," at 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 100 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.

Using the Apache Web Server

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 including Windows 7.

You can download a copy of the executables for most operating systems from the Apache Web site at http://httpd.apache.org/download.cgi. Binary versions of Apache 2.2 (the most recent release) are available for Windows 32-bit machines, RedHat (as RPM files), and Solaris, among others. Most Linux distributions come with Apache pre-installed. Since SAS is available only for Windows, Solaris, RedHat, and SUSE Linux, as well as IBM z/OS, you will want to use one of these. (See http://support.sas.com/resources/sysreq/hosts/ for more information on supported operating systems.)

In order to display HTML content, the pages must first be copied to a specific document root 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) 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 that you are using, it is still a good idea to map a drive to the directory where you want to display your Web pages.

Different Apache distributions use different folders for the document root directory. For example, on RedHat, the default is `/var/www/html`, while, on Windows, it is `C:\Program Files\Apache Software Foundation\Apache2.2\htdocs`. Unless you tell it otherwise (in the `apache2.conf` configuration file), Apache will try to serve Web pages from this folder. You do not need to (nor should you) try to specify the document root in the URL. If the name of the Web page is `example.htm`, located in the document root 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 document root 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 that 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 on Windows systems, most people prefer to use a dedicated client application such as *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.

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 an `apache2.conf` file under the server root directory. As noted above, these values must be supplied at installation time by the system administrator.

The UNIX system administrator has to set up a password and some space on the Web server for each user. Ask which 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 a 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 that 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 that 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 that 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 that 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 that you want to display.

Using Microsoft Internet Information Services

Microsoft IIS does not run under UNIX, but for sites with Windows servers, it is an ideal choice. On Windows XP Professional, the IIS Web server can be installed by going to **Start ▶ Control Panel ▶ Add or Remove Programs ▶ Add/Remove Windows Components**. (This version is limited to 10 users.) For Windows Vista and Windows 7, go to **Start ▶ Control Panel ▶ Programs and Features ▶ Turn Windows Features on and off** and check the box for **Internet Information Services.** This will install all of the IIS tools as well as the Web server. (You may want to uncheck the **FTP** box, for the security reasons noted earlier.)

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 Documentation

SAS/IntrNet 9.2 -
http://support.sas.com/documentation/cdl/en/intrnet/59545/HTML/default/main_contents.htm

SAS Integration Technologies -
http://support.sas.com/documentation/cdl/en/itechov/60309/HTML/default/a003260793.htm

SAS BI Server - http://support.sas.com/documentation/onlinedoc/biserver/

SAS AppDev Studio 3.4 - http://support.sas.com/rnd/appdev/

Web Programming

As of the winter of 2011, there were over 11,000 volumes on the topic "Web programming" at www.amazon.com. The following are some useful guides:

- Berners-Lee, Tim. 1999. *Weaving the Web: the Original Design and Ultimate Destiny of the World Wide Web*. San Francisco, CA: Harper.
- 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: Simon & Schuster.
- Krug, Steve. 2005. *Don't Make Me Think: A Common Sense Approach to Web Usability*. 2nd ed. Berkeley, CA: New Riders Press.
- Nielsen, Jakob. 2000. *Designing Web Usability*: *The Practice of Simplicity*. Indianapolis, IN: New Riders Press.
- Sebesta, Robert W. 2010. *Programming the World Wide Web, 6th ed*. New York, NY: Addison-Wesley.
- 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 (DOM) – http://www.w3.org/DOM/
- Internet Protocols – http://directory.google.com/Top/Computers/Internet/Protocols
- Microsoft Internet Information Services (IIS) – http://www.iis.net/
- South River Technologies WebDrive – http://www.webdrive.com/products/webdrive/
- WebDAV Resources – http://www.webdav.org/
- Web Server Survey – http://www.netcraft.com/archives/category/web-server-survey/
- Working with Distributed Authoring and Versioning (DAV) and Web Folders – http://support.microsoft.com/kb/q221600/

[1] In order to find out the IP address for a Web site, you can use the `ping` utility, available both in Windows and UNIX. This value was determined by typing "ping www.yahoo.com."
[2] For an insider perspective on the invention of the Web, see Tim Berners-Lee, *Weaving the Web.*