Doing M for the World Wide Web

by Jamie Newton

Abstract

M Technology possesses a natural affinity for constructing applications for the World Wide Web. With capabilities such as good string handling, high performance and scalability — and a superior Web-adapted server — M Technology has a very real opportunity to be at the forefront of Internet and intranet technology.

How the Web Works

In essence, Web access is simply a system of requests and responses between a client and a server. A Web server listens for incoming requests from a Web browser, determines the page to be retrieved and delivers the requested page to the browser where it is rendered for display.

The syntax of the request determines what the web server does when it retrieves it. The request is often in the form of a Uniform Resource Locator (URL), which can be thought of as a document's unique address. An example of a URL is:

http://www.company.com/index.htm

This URL specifies a protocol (HTTP), a server address (www.company.com), and a document location (/index.htm). With this information, a browser can locate the specified server on the Web and request retrieval of the document.

The suffix ".htm" indicates that the document contains formatting information specified in the Hypertext Markup Language (HTML). HTML allows text to be "marked up" so that effects such as bold, italic, underlining and other effects may be displayed. It is the browser's responsibility to interpret the HTML and render the page for display.

Other requests may be formulated by entering data into an HTML form. An HTML form is a web page that contains user-interface controls with which the user may interact. When the form is submitted, the entered data is passed to the Web server as part of the URL. For example, a form with a text field for "Social Security Number" may submit a URL such as http://lookup.pl?SSN="1234-567-8910". In this case the document to be retrieved is a CGI script (more on this later), and the string following the '?' is the user-entered data.

Web Page Retrieval

After the Web server receives the request, it attempts to locate and retrieve the requested document. During web page retrieval, the underlying communications protocol, HTTP, takes care of all details related to transmitting the document from the server to the browser. After locating and retrieving a document, the browser scans the page, supplies formatting according to the embedded HTML specifications and then retrieves any images.

Static and Dynamic Web Pages

The HTML pages that are returned to the browser may be classified as static or dynamic. In the case of a static page, a document physically exists on the server containing fixed text and images. In contrast, a dynamic page is an HTML page that is created "on the fly" in response to some user-specified request. Dynamic page creation is possible because instead of locating a document for retrieval, a Web server can be instructed to run a specified program. Such a request was given in the "Social Security" example above. These requests are termed CGI requests (Common Gateway Interface), and programs that service such requests are often called gateway programs.

One familiar example of a dynamic web page created by a CGI request is the web search engine, Alta Vista (http://www.altavista.com). After accessing the Alta Vista web site, the user types keywords into a form, submits the search request, and receives a document containing the results of the search. If the user enters different keywords, then the contents of the result page varies accordingly.

CGI Processing

The scripts that process CGI queries can be as simple or as complex as required. In addition to data collected from the user, information about the server, the client, and the browser is also made available to the script. A CGI query is simply a string of characters (often called the "query string") that contains the originating URL with additional information appended to it. This additional information often contains parameters specified by the user in the form "key = value." For example, the key/value pair SSN="123-56-8910" indicates that the string "123-56-8910" should be associated with the identifier "SSN."

Scripts can use the CGI query to tailor the response to the user, based on the contents of the query string. In most CGI implementations, the method of communicating data to the script is somewhat primitive and has its origins in shell-script programming on UNIX. The value of each key/value pair is associated with a variable name defined by the key. These associations are passed as environmental variables from the server to the invoked script. The environmental variables need to be retrieved by the script before they can be used.

Script Languages

Scripts can be written in any language that is able to retrieve environmental variables, access other software libraries and utilities, and easily manipulate text. A popular language for writing such scripts is PERL (Practical Extraction and Reporting Language). However, PERL has many drawbacks including cryptic syntax, slow execution, and poor database facilities and, because it is freeware, limited support.

Performance

As is well recognized, the evident simplicity of the CGI mechanism has a price. Each time the server receives a request to execute a script, the server must parse the CGI query, enter variables into the environment space, and invoke a process to run the script. On a heavily used server, this overhead is often punitive, resulting in very poor response times. Because of this, server manufacturers such as Microsoft and Netscape have been quick to provide solutions in the form of proprietary server APIs that offer improved performance over traditional CGI. These APIs are sometimes referred to as Binary Gateway Interfaces (BGI).

Microsoft Information Server

By employing the native API of Microsoft Information Server, and using a dynamically loaded link library, a Web server can provide a high-performance alternative to CGI. Communication between the Web server and the M database is via multiple, parallel TCP/IP connections. These are pre-connected to one or more M databases providing a pool of available connections, which can be rapidly accessed under unpredictable web-server load conditions. Because the communication is via the industry standard TCP/IP protocol, the M database or databases may be located anywhere on the network.

A Web server can employ what is called a daemon process, which runs in the M database environment. The daemon spawns multiple server jobs, one per TCP/IP connection, which are available to instantly respond to requests. Requests arriving from the Web browser via the Web server are parsed and made available to the M environment in the local array called %. The invoked M program can access these request parameters and produce the HTML response. Writing to the current device generates the response. This means that pre-existing programs that write data directly to the screen may be ported to output their data to a Web browser with little or no modification (especially if terminal escape sequences have been abstracted from the implementation and kept in a global).

In order to create a page with multimedia-rich content including video, sound and ActiveX controls, all that is required is that the references to the files containing such data are incorporated into the generated HTML. The Web server takes care of making the data available to the browser.

Programming HTML need not be an onerous task. There are many tools on the market that provide a WYSIWYG ("what-you-see-is-what-you-get") method of producing web pages. However, HTML is so simple that a rudimentary web page can be put together with just a few lines of code. A good source of information on HTML is the Internet itself. For example, the *Beginners' Guide to HTML* from NCSA can be found at:

http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html

The example below shows how to use the HTML form method of formulating CGI requests that the Web server understands.

```
<HTML>
<HEAD><TITLE>Test HTML Form</Title></HEAD>
<BODY>
<H1> Testing M server Calls from a Form</H1>
<HR>
<FORM ACTION="bin/myApp.web?" METHOD=POST>
<INPUT TYPE=HIDDEN NAME="EP" VALUE="REG">
Enter your first name: <INPUT NAME="firstname" VALUE="John" SIZE=50>
<P>
Enter your last name: <INPUT NAME="lastname" VALUE="Doe" SIZE=50>
<INPUT TYPE=SUBMIT VALUE="Register">
</FORM>
</BODY>
</HTML>
```

In the following discussion, numbers in parentheses refer to the line numbers in the example above. Note that actual HTML scripts do not contain imbedded line numbers.

HTML forms allow users to enter data into text fields and, via the mouse, select choices with radio buttons, check boxes, and list boxes. It is important to note that data is not processed until the form is submitted for processing by the CGI script; this is usually accomplished by using a button with a type of submit. When the button is pressed, user information is encoded and passed to the CGI script for processing.

In this example, the lines (6-12) define the HTML form. The action to be taken when the form is submitted is defined by line (6). In this case, the configuration file is located in the bin subdirectory below the root of the server. The file is called myApp.web, and the method for invocation is POST.

On line (7) there is a hidden field. The value of this hidden field is an identifier that specifies the entry-point of the M routine that is executed on the server after the form is submitted.

Line (8) specifies a text-entry field. The form displays the text "Enter your first name:" and provides a text-input box for the user to enter a value. The value statement specifies a default value to associate with the variable named "firstname." Line (10) is a similar entry allowing the user to enter his or her last name.

Line (11) specifies the "Register" button. The button's type is SUBMIT, meaning that when pressed, any data gathered from the user is formatted and sent to the web server. In this case, when the form is submitted, the variables firstname and lastname have the values specified by the user (or the defaults) associated with them. Additionally, the value of the hidden field, "EP," is passed to the server.

Processing on the Server

The previous section discussed how to gather user input in an HTML form and described how this data is submitted to the web server. This section explains how to write M routines to access the information passed from the browser and to format the HTML reply on a web page. A rudimentary understanding of HTML and M scripting is again assumed.

In the previous example, values for the variables firstname and lastname were gathered from the user. Additionally, a hidden field called "EP" with the value "REG" was defined in the HTML form.

When the CGI query containing these details reaches the server, the server formats the data and passes it to the M server for processing. The server performs a lookup on the entry-point (EP) field and determines the M entry point to be called. The M server then calls the specified M entry point and waits for a reply from the M server.

In addition to the parameters specified by the user or in hidden fields on forms, a number of other variables are also made available to the M server. These variables contain information about the server or the current connection. Some of these variables are browser-specific. The variables provide information about the Web server and the client, such as the browser and HTML version that are supported. The client variables are dynamic and change on each connection request. Server variables, on the other hand, are static, at least for the life of the connection between the Web server and the M server.

Of course, the processing can be as complex as needed: another form can be constructed and passed to the browser, or a graphical web page may be displayed.

An important point is that graphics, AVI files, Java applets, and other non-text web components do not have to be stored in the M database. They naturally reside on the web server in known locations. All that is required to include those components in the web page is to specify their locations by including their URLs in the HTML produced by the M script.

It is not necessary to construct HTML from scratch. For example, a web page can be designed and constructed with an HTML authoring tool. The web page could be designed using a template with placeholders where M data is to be inserted. By placing the template in a known location on the M server, the file can be accessed from M, data from the database can be inserted into the correct place-holders, and the resulting HTML document can be sent back to the browser, all in a fraction of a second.

As can be seen from this discussion, the task of making your data available to the World Wide Web with M technology is extremely simple. Even better, you benefit from this simplicity without losing power or speed.

M Technology, with its recognized capabilities to develop highly complex, high-performance and resilient systems at relatively low cost, makes it an ideal technology for developing Web applications. The rapid rise of the Internet, I believe, has created a unique situation, in which the market has moved to M Technology, opening up enormous possibilities for expansion of the M community.

M

Jamie Newton is Micronetics Design Corp.'s Project Manager for MSM-PDQweb, the product that makes MSM databases accessible to the World Wide Web.