3080
How To Leverage Your Existing Assets on the Web

James H. Zisch
Acucobol, Inc.
Corporate Headquarters
7950 Silverton Avenue, Suite 201
San Diego, California 92126-6344
(619) 689-7220

Abstract

Intended for beginners, this presentation includes step-by-step instructions for deploying existing assets across the Internet. The final application uses an HTML front end to interact with an application running on a UNIX-based application server. Attendees will learn the basics of HTML, CGI, and Web server software. The presentation highlights practical issues surrounding the migration of commercial applications to the Internet. Special focus is placed on issues of security, application design, and performance.

The paper begins with a brief discussion of the primary components comprising "Deployment of An Application On The Web." The objective is to describe a Web-based HTML front end interfacing with an application.

Introduction

The success of every company in the New Information Age is dependent on its ability to serve customers across the Internet, today. The Internet, specifically the World Wide Web, has so fundamentally altered the way business is done that a company who cannot sell its product, as well as its concept, on-line has absolutely no future in this digital world. Ignoring Internet commerce is now similar to ignoring the telephone; the impact that the Web has had on business is just as profound. Consumers are becoming increasingly comfortable with on-line transactions and also with product research on the Internet. Widespread implementation of global Intranets has made the deployment of internal database applications vital to successful interaction within today's enterprise.

While the proliferation of Web-enabled technologies and requirements has spawned a wealth of software that encourages the development of new applications, it is important not to ignore the value of existing applications and their suitability for deployment across the "Information Superhighway." Existing commercial applications can be migrated to the Internet with existing staff and technologies, dramatically reducing the development time and costs associated with outsourcing such a project.

The Primary Components

- HTML (input and output)
- CGI (process)
- Server (files: programs and data)

HTML

HTTP (HyperText Transfer Protocol) describes the standard for the workings of the World Wide Web and its parts. An important part of HTTP is the file naming conventions.

HTML (HyperText Markup Language) falls within the scope of the HTTP standard. The prefix "Hyper" began occurring in the computer industry in the early eighties and is indicative of increased function. HTML is a text "markup" language. Markup languages have their origins within the publication industry, beginning centuries ago. Within the context of the computer industry, markup languages established a presence providing stylized text formatting in early word processor applications in the form of embedded command printer driver directives.

Standardization of HTML is maintained by the W3 Consortium. The standard has undergone several formalized reviews and revisions and is currently defined by the HTML 3.2 Specification. The specification consists of the HTML markup "tags" as well as style guides.

The basic format of the HTML tag set consist of a series of formatting directives that are transmitted to a client-side browser application that interprets the markup and uses the directives to format the display of the content of a Web page.

As an example:

```
<HTML>
<HEAD>
<TITLE>Sample Page</TITLE>
</HEAD>
<BODY>
<P>This is a sample page</P>
</BODY>
</HTML>
```

Note in the previous example the <TITLE> and </TITLE> markup tags. The <TITLE> begins the markup directive; the tag prefixed with a forward slash ("/") indicates termination of the TITLE markup. Also note the terminators for the HEAD, BODY, P (paragraph) and HTML tags. This is the basic format of the HTML syntax. Most, but not all, HTML tags have a terminating tag. All HTML markup tags are wrapped in less than (<) and greater than (>) characters, with one exception.

The exception is a special notation that supports an expanded alphanumeric character set. For characters in this expanded set, the tags begin with an ampersand and terminate with a semicolon. For example, the copyright character is represented by the markup "©", and quotes are represented by """. The exception character set is robust.

Many HTML tags also support a set of parameters that specify various attributes. HTML parameter format is the parameter name followed by an equals sign and a value.

Accepting Input from a Web Page

The FORM tag defines input areas within a Web page. A simple FORM containing one input field would appear as follows:

```
<FORM>
<INPUT TYPE="TEXT">
</FORM>
```

The result of the above example would be the display of an input field. The form does not provide any function beyond merely the display of a field.

To add functionality to the field, expand the FORM tag to include some parameters.

```
<FORM ACTION="program-name">
```

The ACTION parameter defines the program to receive control when the form is submitted. The program is resident on the Web server's system.

The value specified by the user in the INPUT field is passed to the program in one of two formats. These formats are determined by the method used to pass the parameters. This method is specified in the METHOD parameter of the FORM definition.

There are two specific methods: GET and POST. The GET method passes the input data as an argument. The POST method puts the input data in the server system's memory stack of the indicated program. The addition of this parameter to the FORM definition with the POST method would appear as follows:

```
<FORM ACTION="program-name" METHOD="POST">
<INPUT TYPE="TEXT">
</FORM>
```

To complete the form, a control must be added to allow the user to specify the submission of the form data. This causes the program defined in the ACTION parameter to be invoked. There are several techniques supported for providing this functional control element. For this example, the INPUT TYPE of SUBMIT is used.

```
<INPUT TYPE="SUBMIT">
```

The completed form would appear as follows:

```
<FORM ACTION="program-name" METHOD="POST">
<INPUT TYPE="TEXT">
<INPUT TYPE="SUBMIT">
</FORM>
```

The form above supports display of an input field with a button labeled SUBMIT. The form would be included within the BODY of an HTML page and would appear as follows:

```
<HTML>
<HEAD>
<TITLE>Sample Page</TITLE>
</HEAD>
<BODY>
<FORM ACTION="program-name" METHOD="POST">
<INPUT TYPE="TEXT"><BR>
<INPUT TYPE="SUBMIT">
</FORM>
</BODY>
</HTML>
```

Note: The
 results in a line break, causing the SUBMIT button to display on the line following the INPUT field. This example does not contain any static text, nor does it associate the INPUT field with a variable name.

To finalize the example, instructional text has been added, and the INPUT field definition has been associated with a variable name, "Account Number," via the NAME parameter. And, the "program-name" has been replaced with the name of a real program.

```
<HTML>
<HEAD>
<TITLE>Sample Page</TITLE>
</HEAD>
<BODY>
<P>Please enter an account number, then click submit.</P>
<FORM ACTION="sample.cgi" METHOD="POST">
<INPUT NAME="Account Number" TYPE="TEXT"><BR>
<INPUT TYPE="SUBMIT" VALUE="ACCESS ACCOUNT">
</FORM>
</BODY>
</HTML>
```

This completes the building of a very basic Web page input form that supports input from a user and invokes a program named "sample.cgi" when the user clicks the SUBMIT button. Note that in the definition of the SUBMIT button in the example above, the VALUE parameter has been defined; because of this, the button itself will display the text “Access Account,” instead of the word “SUBMIT.”

Summary of HTML Discussion

The HTML language provides the Web browser application for displaying, accepting, and passing data from a Web page to a remote server. The above discussion describes the elements supporting the construction of a Web-based user input screen, referred to as a “Web page." A Web page in its HTML format may exist as an ASCII file or as generated output from a Web server-based application program. Supporting application programs of this type are referred to as CGI programs.

CGI

Common Gateway Interface (CGI) refers to the standard that defines the platform-independent communication and control between client-based and server-based components. The components involve a client-based Web browser application and a server computer system. The elements communicated between these components consist of data. The elements of control are the process controls performed by both the Web browser application (on the client) and the Web server system.

The CGI Standard is a relatively simple set of protocols defining the behavior of the interactions between a Web browser and the Web server that is addressed when a user accesses a Web page resident on a particular Web server, thus invoking a server-based application.

Part of the standard includes the definition for the input and output handling in this interface.

CGI Program Input

A CGI program receives control upon invocation and can receive input from the invoking client browser application. Input data may be passed within an HTML page containing a form. Also, input may be passed in the format of command line type parameters within the context of a hyperlink markup that invokes a CGI program, such as:

```
<A HREF="http://www.Website-domain.net/sample.cgi?Account%20Name>Get Account Name</A>
```

The HTML section earlier in this paper did not discuss the hyperlink markup structure, so an explanation of the example shown above may be helpful. The result of the example is the display of the text "Get Account Name". The text is displayed as underscored, indicating that when it is clicked, an action will occur. In this example, clicking the text invokes the program "sample.cgi," and the value "Account Name" is passed as an input parameter. Note the "%20" between the words "Account" and

"Name". The percent symbol (%) is an escape character, and "20" is the hexadecimal ASCII equivalent for a space. Without this protection of the space delimiter, the input parameter is truncated by the space, resulting in the word "Name" being dropped.

In addition to the client browser application originating the input data, the Web server also supports access to several items of information pertaining to the client/server interface, contained within an "environment" array. The CGI standard defines the Environment variable set. These Environment variables are Web server dependent, meaning not all Environment variables defined in the standard are necessarily set by the Web server.

When a CGI program receives control, the parameters are available to it for access. As with any other application system environment, the input parameters must be read into the application program's data space. The technique for reading in the input parameters is dependent on the method by which the input parameters are passed. As discussed in the HTML section of this paper, there are two METHODs of data input passing: GET and POST.

The GET method passes the input data as an argument character string. The method for reading the input parameters is specific to the programming language used to create the program. The same technique used to receive input arguments passed as command line parameters would be used to assign the data to a variable. No descriptive information, about the data passed in, is included with the data, so the program must parse the data to determine whether multiple parameters have been passed. For this reason, with the GET method, the parameter format must be predefined and known to the program so that it can successfully parse the data. This might involve the use of a dedicated delimiter or a fixed length offset scheme with a defined data structure.

The POST method causes data item to be readable as standard input with a length specified in the Environment array (CONTENT-LENGTH). This method is designed specifically to support the variable name and value pairs to be passed. As discussed in the HTML FORM section above, the field name is associated with the value and passed as field-name=field-value. The example in that section uses "Account Name=x", where x equals the value entered by the user.

In addition to passed parameters, the CGI program also has all connected, assigned, and authorized devices available for access as input devices; including disk for data file or database input and output.

CGI Program Output

CGI Programs are capable of sending output to the client browser application in a variety of data formats. These formats include HTML marked up Web page content, image formatted data, and a variety of other data formats as well as supporting audio and video. For this paper, only the HTML formatted text data is discussed.

In most programming languages, program output by default is sent to standard out. As part of the CGI standard, a special output string causes the Web Server system to direct standard out to the remote client browser application. Identification of the remote user is accomplished through the maintenance of the originator’s identification parameters being stored in the Environment array by the Web Server. The special output string used for transmission of HTML text content appears as follows:

"Content-type: text/html"

The "Content-type:" portion of the above string is the directive, and "text/html" is a parameter. The parameter is referred to as a MIME type (Multi-purpose Internet Mail Extensions), describing a standardized internal format used in construction and interpretation of e-mail documents transceived (transmitted and received) on the Internet. Although the use of MIME in this application does not apply to electronic mail, as it occurs in the definition of the acronym, the standard that had been established prior to the HTTP World Wide Web standard is applied. There are numerous MIME types defined within the HTTP standard in support of special handling for the various internal data formats

used during preparation, transmission, reception, and post processing. The MIME type supported by a Web Server is installation-specific. User defined MIME types can also be implmented.

The transmission of the "Content-type: text/html" string via a programming language's print command causes all subsequent standard out print commands to be directed to the client browser application. The client browser application then processes the transmitted output data as if it had acquired it from a Web Server resident text file.

CGI Client Browser and Server Control Handling

The client/server interaction within a Web environment is similar to the interaction handled by any other interactive programming project. Process control is based on user interaction. Either an application system is polling for user interaction, or the user performs an interaction that invokes an application system process. The primary difference with a Web-based application is that the invoked application retains control until it is completed. Specifically, the process invoked by the user that causes a server-resident CGI application to receive control, results in control being maintained by the CGI program, without polling the client browser, until completion of the process. The process must achieve completion by transmitting content back to the client Web browser. This control mechanism is part of the CGI standard, which is part of the current HTTP standard.

Cutting Edge Technology and Legacy Systems' Common Ground

The advent and standardization of multi-channeled remote processing provides the opportunity to alleviate the restriction of the "wait state" via process controls in the not too distant future. However, in the interim this is acknowledged as the common link between what is referred to as the legacy systems environment and what is currently the most advanced of networked computer telecommunications processing, the information superhighway. Until the "wait state" restriction is alleviated, true multi-processing computing is limited to multitasked processes on distributed systems.

CGI Summary

The Common Gateway Interface technology provides for cross-platform independence of data input and output and defines the control mechanisms for process controls. As applied to the World Wide Web, it provides Web server-resident programs with the ability to be invoked by a remote client application, which passes control to the Web server-resident CGI application program. It also provides the controls to cause the client browser application to wait for completion of the CGI application program's process (indicated by the transmission of output to the client browser application).

Web Server

The most basic Web server includes a central processor and disk storage. The system is augmented beyond a stand-alone system with standard telecommunication components, including local control devices, switches, and routers, and often multiple processors. Telecommunications connect through either a hub connection or direct connection to the Internet trunk line. Completing the configuration is an operating system subsystem application referred to as the Web Server. The Web Server itself is software; however, without the supporting devices it is non-functional, hence what is often referred to as a Web Server is in actuality a Web Server System composed of both software and hardware.

Special Focus

- **Security**

Web Server data security issues are fundamentally the same as issues surrounding any computer system facility with external access. The opportunities for a breach of security are significantly increased by the number of potential culprits.

Web Server systems often implement minimal security and data protection measures in the interest of reducing system overhead and thus improving throughput rates.

Due to the increased vulnerability of Web-based systems, it is imperative that adequate security measures be implemented as warranted by the system, tasks, and data resident on the Web server. Implementation of firewalls is strongly recommended for protection of internal networks connected to systems with external access.

Data en route during transmission between client browser applications and Web server CGI applications is a key area of concern. Any sensitive data should be encrypted, as supported by most client browser applications and Web Server systems.

A number of security protocols, such as Server Secured Linkage (SSL), are available and should be evaluated for any Web-based system. These help ensure that adequate measures are taken to protect information and data in accordance with its sensitivity.

- **Application Design**

When process controls and user transaction paths are designed, the interaction controls that are currently in place should be given due consideration.

Existing transaction-based application systems can often be readily migrated to Web based CGI applications. Standard data dictionaries can be easily transformed into Web page content forms. Often, minor modifications to existing data and program modules can transform data files and database management systems into Web-based Intranet and Internet solutions.

Three primary aspects involving Web based application design:

1. User interface ease of use. Many of the aspects of field traversal, field validation, and invalid value handling are supported via built-in client browser applications. However, most require HTML embedded and/or included programmatic language functionality. These aspects may be externalized from the application program and incorporated into the Web page content. It is preferable to perform field validation at the client browser application side, with an emphasis on capturing the error at the time it occurs and reducing or eliminating data input error checking at the application level.

2. Presentation display aspects. With the robust capabilities for text and graphical content, migration of existing programs must consider the presentation and display aspects of program design, above and beyond mere GUI interface design aspects. The incorporation of high resolution graphics, video, and audio into the presentation of data requires a multimedia approach. The established software design and development team must augment their resources, incorporating the necessary multimedia expertise such as graphic artists, marketing, and multimedia production talents. The objective of the overall existing application system must be revisited and evaluated during a move to Web-based solutions. What once was an application that crunched numbers and displayed statistics can easily come to life as a product brochure, marketing and electronic commerce system, with the expansion of the graphical aspects and the combined output of multiple applications into a robust Web-based solution. The value of the existing applications is the expansive data and the power of the access and manipulation of that data.

3. Performance. Web page performance is crucial for a successful implementation. The amount of resources being used within a Web page determines the amount of time it takes for its content to transfer from the Web server to the client browser application. With the continuing increases in desktop processor speeds and modem throughput rates, the issue of performance takes on new aspects. When evaluating overall performance, determine your user base and their standard levels of technology. If your user base has predominately ISDN level connections, such as an Intranet solution, then page content loading time is not a crucial design consideration. If your user base is the entire

World Wide Web, you might want to obtain the latest survey information to determine your user base's equipment with respect to monitor size and resolution, processor speed, and modem connection rates.

And finally, with respect to I/O devices, CGI applications should use a rapid access method to obtain data using buffered I/O, opening files reading all required data, then closing files. Files should not be left open during data manipulation and processing functions within the application. The same holds for outputting data. The entire content of the output should be formatted and prepared prior to the performance of output functions. The issue here holds true with any multi-user environment; however, a successful implementation may have a user access rate that exceeds expectation, and applications should be designed with optimum performance in mind. This is critically important in the area of I/O.

A COBOL CGI Application Example

The following describes the six basic steps involved in accessing data from a database through a Web browser.

1. Browser sends request through the Internet to the Web Server.
2. Web Server sends information to the COBOL CGI program.
3. COBOL CGI program accesses database.
4. Database returns data to COBOL CGI program.
5. COBOL CGI program composes HTML document and sends it to the Web Server.
6. Web Server returns the HTML document to the requesting browser.

The Process in Detail:

Mike's TV and Appliance Example

The following example demonstrates a small application where a user would select the type and size of television from "homepage.html" and expect the price of that television to be returned. This example creates the file "cobol.htm" and returns the requested information back to the user.

Step 1 Browser sends request through the Internet to the Web Server.

This is done through the use of embedded forms within the HTML document. In this case the HTML document uses the "GET" method to call a CGI program. The following is the HTML source code to the "homepage.html" document that will request the information.

homepage.html

```
<HTML>
<HEAD><TITLE>COBOL-cgi Example</TITLE></HEAD>
<BODY>
<H1>Welcome to the MIKES TV and Appliances</H1>
 <FORM ACTION="/cgi-bin/cobolcgi.cbx" METHOD="post">
<P>
       TV Type
      <SELECT  NAME="tv_type">
        <OPTION>color
        <OPTION>blwht
       </SELECT>
       TV Size
      <SELECT NAME="tv_size">
        <OPTION>15
        <OPTION>20
       </SELECT>
```

```
 <INPUT TYPE="SUBMIT" VALUE="Run Query">
</FORM>
</BODY>
</HTML>
```

Step 2 Web Server sends information to the COBOL CGI Program.

Based on information from the browser, the Web Server sets the QUERY_STRING environment variable and runs the COBOL program. The COBOL program reads the environment variable QUERY_STRING, unstrings it, and determines what information is requested.

Different Web Servers use different methods of associating extensions. If your Web Server does not have this ability, you may have to run your program from within a script. For example, if you are running NCSA's Server, you could use the following script called "cobolscript" to launch your program.

```
#!/bin/sh
A_TERM=vt100
export A_TERM
runcbl cobolcgi.cbx
```

You would then have to replace "cobolcgi.cbx" with "cobolscript" in the ACTION statement of "homepage.html"

Step 3 COBOL CGI program requests data.

COBOL Source Code

```
 identification division.
 program-id. cobolcgi.
 remarks.
 environment division.
 input-output section.
 file-control.
select data-file
       assign to "cobolcgi.dat"
       organization is indexed
       access mode is dynamic
       record key is primary-key.
 data division.
 file section.
 fd  data-file.
 01  data-record.
       03  primary-key              pic 9.
       03  tv-type-key              pic x(5).
       03  tv-size-key              pic x(2).
       03  tv-price                 pic x(5).
 working-storage section.
 01  html-record                    pic x(120).
 01  cgi-environ-variables.
       03  query-string-name        pic x(40).
       03  query-string-value       pic x(250).
 01  data-values.
       03 tv-type                   pic x(20).
       03 final-type                pic x(5) value spaces.
       03 tv-size                   pic x(20).
```

```
        03 final-size                   pic x(2) value spaces.
        03 ws-tv-price                  pic $zzzzz.
        03 stop-reading         pic x(3).
 01  garbage                           pic x(10).
*
 procedure division.
*
 main-logic.
        perform accept-environment.
        perform get-request.
        perform get-data.
        perform make-html.
stop run.
*
 make-html.
        move "Content-type: text/html" to html-record.
        display html-record.
        move spaces to html-record.
        display html-record.
        move "" to html-record.
        display html-record.
        move spaces to html-record.
        move “MIKES TV and Appliances" to html-record.
        display html-record.
        move spaces to html-record.
        move "The price of the television is: " to html-record.
        display html-record.
        move spaces to html-record.
        move ws-tv-price to html-record.
        display html-record.
        move spaces to html-record.
        move "" to html-record.
        display html-record.
end-make-html.
*
 get-data.
        open input data-file.
        perform until stop-reading = "yes"
              read data-file next record
              at end
                    move "yes" to stop-reading
                    move "9999" to ws-tv-price
              not at end
                    if tv-size-key = final-size
                          and tv-type-key  = final-type
                    then
                          move tv-price to ws-tv-price
                          move "yes" to stop-reading
                    end-if
              end-read
        end-perform.
        close data-file.
 end-get-data.
*
accept-environment.
        call “cgiread” using query-string-name query-string-value.
```

```
end-accept-environment.
*
 get-request.
       unstring query-string-value
             delimited by "&"
             into    tv-type
                     tv-size
       end-unstring.
       unstring tv-type
             delimited by "tv_type="
             into    garbage
                     final-type
       end-unstring.
       unstring tv-size
             delimited by "tv_size="
             into    garbage
                     final-size
       end-unstring.
 end-get-request.
*
```

Step 4 Database returns data to COBOL CGI program.

The get-data procedure of the "cobolcgi.cbx" program performs this step.

Step 5 COBOL CGI program composes HTML document and sends it to the Web Server.

Step 6 Web Server returns the HTML document to the requesting browser.