INTEGRATING WEB SITES AND DATABASES

By Mike Morrison, Joline Morrison, and Anthony Keys

Web site developers creating 'data-based Web pages' that interact with organizational databases need to know server- and client-side processing.

Customers ordering from an e-commerce Web site need to be able to get information about a vendor's products and services, ask questions, select items they wish to purchase, and submit payment information. Vendors need to be able to track customer inquiries and preferences and process their orders. The requirements of data-intensive Web sites increasingly drive the merging of Web and database technologies. Web developers increasingly experiment with technologies promising to speed development and maintenance of Web sites. E-commerce organizations have also discovered that intranets provide a readily accessible interface to internal databases. Here, we examine popular commercially available technologies for creating Web pages that interact with organizational databases, offering guidelines for matching system requirements with these technologies [4].

In a static Web page, content is determined once, when the page is created. As users access a static page, the page always displays the same information. In a dynamic Web page, content varies based on user input and data retrieved from external sources. We use the term "data-based Web pages" to refer to dynamic Web pages deriving some or all of their content from data files or databases.

It is useful to distinguish between data-based Web pages from pages created through client-side scripting technologies, such as JavaScript and VBScript, that support such tasks as verifying data, displaying new browser windows, and providing animated graphics and sound, rather than interacting with files or databases.

Data-based Web pages support a variety of user requests, such as, "I want to buy 100 shares of Technology, Inc. at $99 per share" and "What is the current balance of my checking account?" These requests are usually serviced through programs running on Web servers that retrieve and display data from a file or a database. Though file-based data retrieval is within the scope of data-based Web pages, we focus on databases rather than file-based systems for data storage and retrieval because they are more robust and easier to maintain.

A data-based Web page is requested when a user clicks a hyperlink or submit button on a Web page form. If the request comes from clicking a hyperlink, the link specifies either a Web server program or a Web page that calls a Web server program. In some cases, the program performs a static query, such as "What was the Dow Jones average for the last five time periods?" Although this query requires no user input, the results vary depending on when the query is made. If the request is generated when the user clicks a Web page form's submit button, instead of a hyperlink, the Web server program usually uses the form inputs to create a query. For example, the user might enter an order to buy five shares of Cisco Systems stock, then submit the input to the Web server program. The Web server program then services the order, generating a dynamic Web page response to confirm the transaction. In either case, the Web server is responsible for formatting the query results by adding HTML tags. The Web server program then sends the program's output back to the client's browser as a Web page.

Web Page Programming Options

An e-commerce organization can create data-based Web pages by using server-side and client-side processing technologies or a hybrid of the two. With server-side processing, the Web server receives the dynamic Web page request, performs all processing necessary to create the page, then sends it to the

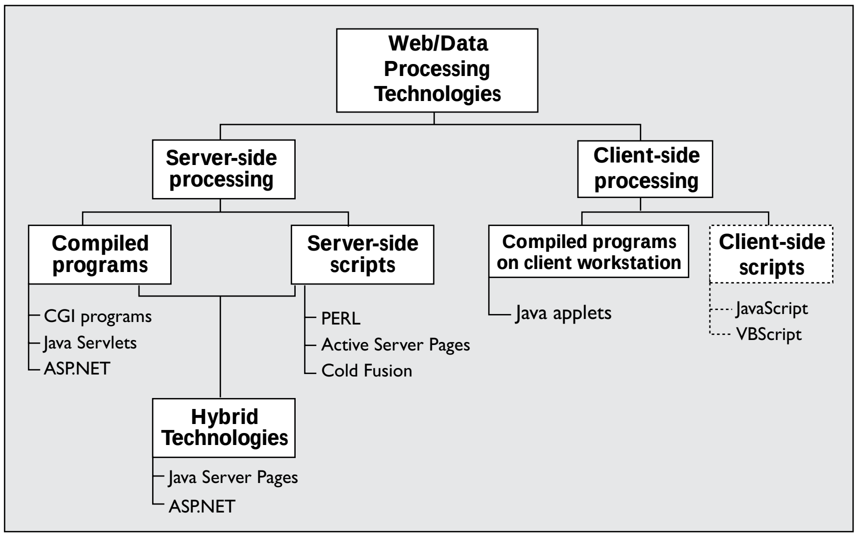


Figure 1. Popular Web/data processing technologies.

client for display in the client's browser. Client-side processing is done on the client workstation by having the client browser execute a program that interacts directly with the database. Figure 1 outlines commonly used server-side, client-side, and hybrid Web and data processing technologies; client-side scripts are in dashed lines to indicate they are unable to interact directly with a database or file but are used to validate user input on the client, then send the validated inputs to the server for further processing.

Server-side processing. Most data-based Web pages use HTML forms to collect user inputs, submitting them to a Web server. When a HTML form is submitted to a Web server, a program running there processes the form inputs, dynamically composing a Web page reply. This program, which we call the servicing program, can be either a compiled executable program or a text script interpreted into machine language each time it is run.

Compiled server programs. When a user submits HTML-form data for processing by a compiled server program, the Web server invokes the servicing program. The servicing program is not part of the Web server listener or administration utilities but an independent executable program running on the Web server; it retrieves the input variables, processes them, stores the output values formatted with HTML tags in a server memory location, then terminates. The Web server then reads the output values, sending them back to the user's browser where they are displayed as a Web page.

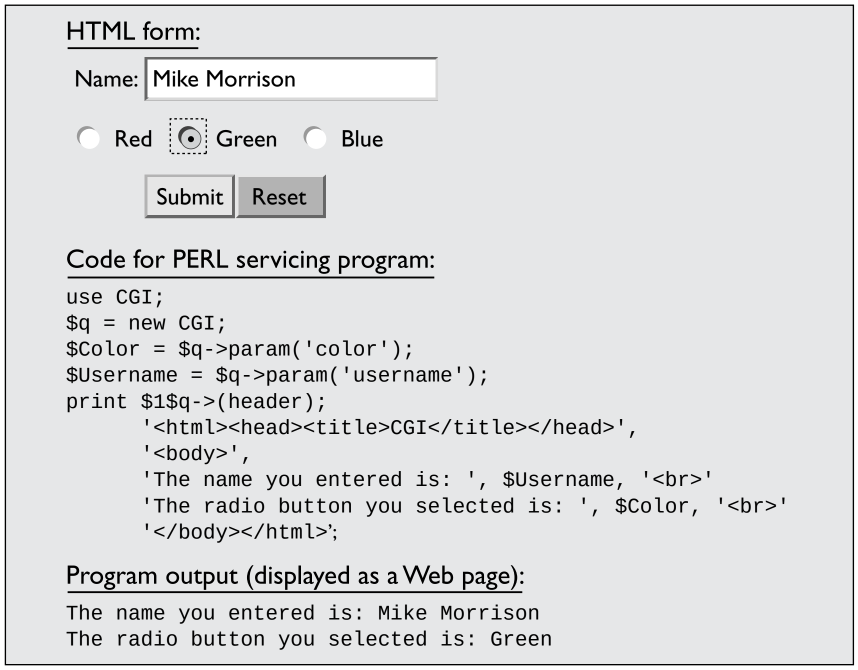
Popular languages for creating compiled server programs are Java, Visual Basic, and C++, but almost any language that can create executable programs can be used, providing it supports commands used by one of the protocols that establish guidelines for communication between Web servers and servicing programs. The first such protocol (introduced in 1993) for use with HTML forms was the Common Gateway Interface (CGI); many servicing programs on Web sites today still use CGI programs. However, a disadvantage of using CGI-based servicing programs is that each form submitted to a Web server starts its own copy of the servicing program on the Web server.

A busy Web server is likely to run out of memory when it services many forms simultaneously; thus, as interactive Web sites have gained popularity, Web server vendors have developed new technologies to process form inputs without starting a new copy of the servicing program for each browser input. Examples of these technologies for communicating with Web servers include Java Servlets [2] and Microsoft's ASP.NET [1]; they allow a single copy of the servicing program to service multiple users without starting multiple instances of the program.

ASP.NET has introduced many new capabilities to server-side Web programming, including a new category of elements called server controls that generate as many as 200 HTML tags and one or more JavaScript [5] functions from a single server control tag. Server controls support the processing of user events, such as clicking a mouse or entering text at either the client browser or the Web server. Server controls also encourage the separation of programming code into different files and/or areas from the HTML tags and text of a Web page, thus allowing HTML designers and programmers to work together more effectively.

ASP.NET allows the developer to choose between

FROM A PERFORMANCE STANDPOINT, BECAUSE COMPILED PROGRAMS EXECUTE FASTER THAN SCRIPTS, BUSY WEB SERVERS SHOULD USE COMPILED SERVER-SIDE PROGRAMS.



HTML form:

Name: Mike Morrison

Red Green Blue

Submit Reset

Code for PERL servicing program:

```
use CGI;
$q = new CGI;
$Color = $q->param('color');
$Username = $q->param('username');
print $1$q->(header);
      '<html><head><title>CGI</title></head>',
      '<body>',
      'The name you entered is: ', $Username, '<br>'
      'The radio button you selected is: ', $Color, '<br>'
      '</body></html>’;
```

Program output (displayed as a Web page):

```
The name you entered is: Mike Morrison
The radio button you selected is: Green
```

Figure 2. PERL script example.

Cold Fusion Example

```
<HTML><HEAD><TITLE>Cold Fusion</TITLE></HEAD><BODY>
The name you entered is: #form.username#
The radio button you selected is: #form.color#
</BODY></HTML>
```

ASP Example

```
<HTML><HEAD><TITLE>ASP</TITLE></HEAD><BODY>
The name you entered is: <%=request.querystring('username')%>
The radio button you selected is: <%=request.querystring('color')%>
</BODY></HTML>
```

PHP Example

```
<HTML><HEAD><TITLE>PHP</TITLE></HEAD><BODY>
The name you entered is: <?php $username; ?>
The radio button you selected is: <?php $color; ?>
</BODY></HTML>
```

Figure 3. Scripting compared.

a server-side hybrid processing model and a compiled model; this dual approach is why ASP.NET is included in both categories in Figure 1. The compiled model creates Web application projects, or collections of files providing the functionality for each application. Visual Studio.NET's development environment, called VB.NET, is almost identical to the development environment it provides for Web application projects. Included are reliable integrated debugging, graphic design tools, and project management. Since none of these functions is available when the hybrid model is used, ASP.NET is likely to be used as compiled rather than as hybrid technology.

Programs created through ASP.NET are not backward compatible with ASP scripts created through the original ASP server-side scripting technology [6]; upgrading older ASP scripts to ASP.NET requires substantial revision. ASP and ASP.NET programs can, however, run on the same Web server, as ASP.NET programs are distinguished with .aspx file extensions.

Server-side scripts. Web-based applications can also use server-side scripts to create dynamic Web pages that are able to retrieve and display database data and modify data records. The processing architecture is the same as the processing architecture used for compiled server programs, except the Web server processing is performed through an interpreted text script rather than a compiled program.

Scripts using the CGI communication protocol are usually written using the Practical Extraction and Report Language (PERL) scripting language, which contains features derived from Unix, C, and BASIC. The current version of PERL, Version 5, handles binary and text file processing on a variety of operating system and Web server platforms. The language itself is terse, with many operations and commands specified with one- or two-character commands; special characters (such as $, #, %, and /) are used frequently to indicate data types and operations.

Figure 2 shows an example of a PERL script designed to interact with a HTML form. The HTML form allows a user to enter a name in a text box referenced as username and select an option button in a radio button group referenced as color. When the user clicks Submit, the PERL script executes and displays the name the user entered, along with text displaying the label of the selected option button.

PERL includes functions for manipulating text strings and processing files with descriptive names, making it easier to understand and work with the code. Although it is possible to write well-documented, maintainable PERL scripts, the language encourages cryptic shortcuts. PERL scripts can therefore be difficult to understand and maintain. Moreover, they use the CGI protocol, so each time a script is requested a new copy of it is automatically invoked.

Macromedia's Cold Fusion solves the CGI problem of starting a new copy of the program each time it is requested by allowing a single running script to serve multiple requests. It also provides a different approach from PERL for creating server-side scripts. Instead of embedding HTML outputs in coded print statements, as PERL does, a Cold Fusion script looks like an ordinary HTML page with code embedded in the HTML; this approach allows HTML page designers to create Web pages and add code as needed.

```
JSP Example
<%@ page import="login.inputHandler" %>
<jsp:useBean id="abean" scope="page" class="login.inputHandler" />
<HTML><HEAD><TITLE>CG13</TITLE></HEAD><BODY>
The name you entered is:
<jsp:getProperty name="abean" property="username"/>
The radio button you selected is:
<jsp:getProperty name="abean" property="color"/>
</BODY></HTML>
```

Figure 4. JSP script example.

A number of newer scripting technologies developed since 1995 have taken the Cold Fusion approach to embedding code within HTML by allowing a single running script to service multiple requests. Figure 3 includes examples of such scripts using Cold Fusion, Microsoft's ASP, and the Apache Software Foundation's PHP; all do the same thing as the PERL script in Figure 2—access the username entered and the color selected on the HTML form and write them to a new Web page. Although the syntax varies, the approach is the same—embed code statements within ordinary HTML.

If needed, a developer can have a single Web server process a variety of scripts written with any or all of these technologies. The Web server knows which script interpreter to invoke by taking note of the requested script's file extension. Cold Fusion files have .cfm file extensions in their name, PERL scripts have .pl file extensions, and ASP scripts have .asp extensions.

Server-side hybrid processing. Compiled server-side programs offer two main advantages: They are first compiled and stored in a machine-readable format; they do not need to be translated into a machine-readable format each time they execute, so they usually run faster than scripts. Second, compiled programs are usually created in integrated development environments that provide debugging utilities, thus making it easy to locate and correct errors. The advantage of using scripts is that their modification requires only a text editor rather than installation of an associated development environment. Hybrid server-side programming strives to combine the advantages of compiled server-side programs and server-side scripts; a server-side script is created but not compiled. The first time a user accesses a Web page calling the script, the script is compiled into machine-readable format and stored as an executable file. With this approach, the developer works with ordinary text files and does not need to install an integrated programming development environment to modify the script. Performance is improved because the program does not need to be translated into machine language each time it runs.

Soon after Microsoft introduced ASP technology in 1997, Sun Microsystems introduced Java Server Page (JSP) technology. Unlike ASP scripts, JSP source code is automatically compiled into machine-readable format the first time a user accesses it. The Web server saves the compiled JSP program, using it (rather than the source code) the next time anyone else tries to access this particular JSP. This scheme reduces both the processing performed and the time the user has to wait to view a response from the Web server. If a programmer modifies the JSP source code, the Web server notes that the source code file has been modified since the compiled version was created, compiling and saving the compiled program the next time a user accesses the page; Figure 4 shows a JSP page equivalent of a script that is functionally equivalent to the scripts in Figure 3. The examples in Figure 3 are complete, but the JSP example in Figure 4 would require writing an additional program (a JavaBean) defining getProperty and other required methods.

Choosing server-side processing. From a performance standpoint, because compiled programs execute faster than scripts, busy Web servers should use compiled server-side programs. Why use scripts at all? Scripts are preferable when a large program is not required for processing HTML form inputs; creating a short script is faster than creating a short compiled program. For example, a programmer can readily modify a script using a text editor. To modify a compiled program, programmers must have the associated programming environment installed on their workstations, as well as the program's original source code. After the program source code is modified, it must be recompiled. This makes modifying a short compiled program more complicated and time-consuming than modifying a short script.

Programmers using a CGI program should be aware that CGI server-side programs can be written in any language supporting the CGI protocol, and a CGI program can be used with any Web server and operating system supporting the language in which the CGI program is written; C, C++, and PERL are popular choices. Visual Basic is a popular choice for Web servers running in the Microsoft Windows environment. However, there are no VB compilers for other operating systems.

CGI's main drawback is its inefficient use of Web server resources. A number of public and proprietary Web servers and Web server add-on products have been developed to enable a single CGI program loaded into memory to service multiple submissions of the same form. One such product, called Persistent CGI is included in Zope, an open source Web server available from Zope Corp. in Windows and Unix/Linux versions. Another solution to the CGI performance problem is FastCGI, an open source extension for existing Web servers; links to FastCGI add-ons for Microsoft, Netscape, and Apache Web servers are available at www.fastcgi.com.

ASP.NET has impressive new capabilities, though it involves a much steeper learning curve than many other competing products, including Microsoft's own ASP; ASP.NET will, however, be a major player in the coming years as developers learn more about it.

Client-Side Processing

Client-side Web page processing is achievable through compiled programs downloaded, installed, and executed on the client workstation or by creating scripts within the HTML Web page commands interpreted by the client browser.

Downloading and running compiled programs on client workstations. When a user clicks a hyperlink on a Web page associated with a compiled client-side program, the user's browser must have the ability to run the executable program file; this program interacts with the user, sending and retrieving data from a database server as needed. The Java applet is the most popular technology for creating programs users can download from Web sites and install on their own computers.

Java applets are self-contained Java programs running in the generic Java runtime environments supplied by most Web browsers. For security reasons, Java applets receive and send data from a Web server but cannot read data from or write data to any files on a user's workstation. Java applets are intended to run uniformly on any operating system and with any Web browser. However, the Java runtime environment is implemented inconsistently among browsers, even within different versions of the same browser. This lack of consistency makes it difficult to create a Java applet that runs in the same way within all browsers.

Client-side scripts. In a client-side script, source code written in such languages as JavaScript and VBScript is embedded in a HTML document, along with the static HTML text; it is placed within delimiter tags to indicate to the user's browser that the text is code rather than Web page text. If the user's browser is able to recognize and interpret the code, it is processed. If the browser is unable to recognize and interpret the code, it is displayed as text on the Web page, unless the script author encloses it in a HTML comment.

IF BEING ON THE LEADING EDGE IS OF NO CONCERN TO THE E-COMMERCE ORGANIZATION DEVELOPING A SITE, ITS DEVELOPERS SHOULD CONSIDER USING *ASP.NET.*

Although client-side scripts cannot be used by a Web page to interact with a remote database, they are often used to validate user inputs entered on HTML forms submitted for processing by a server-side program; for example, a script running on a client workstation might check the inputs users submit to a Web page to make sure they entered all required data and appropriate data values. This approach avoids transmitting inputs to the Web server that are incomplete or include errors, while offloading error checking and handling from the Web server program to the client workstation.

Client-side scripts can also be used to create advanced Web page features, including: image maps allowing users to move their cursors over an image and click to access different Web page links; a new browser window displaying an alternate Web page; a timer displaying an image or playing a sound; or cookies storing data on users' computers about their actions while browsing a particular Web page.

JavaScript is the most commonly used client-side scripting language and is supported by most browsers. It can also support server-side processing through Web-server-specific languages derivatives; a version of JavaScript called LiveWire can be used on Netscape Web servers for server-side processing of form inputs; and Jscript, a Microsoft-specific version of JavaScript, can be used in ASP pages on Microsoft Web servers. Although JavaScript's core language is similar to Java's core language, JavaScript uses loosely typed variables and is designed to work mainly within Web browsers. Java is strongly typed and includes an extensive collection of object and language features for creating standalone programs. JavaScript client-side scripts can be added to standard HTML Web pages through special HTML tags.

Use of a client-side scripting language depends on user operating system, browser platforms, and developer expertise. If the Web pages in question are to be accessed by a variety of users over the Internet, JavaScript is probably better than VBScript, as JavaScript is the only scripting language able to run on nearly all browsers. If the Web pages are to be accessed on an intranet and if the organization has standardized on Microsoft's browser and Web server, VBScript is a satisfactory scripting language for creating client-side scripts.

Database connectivity. A general approach for linking Web pages with a database involves creating a database connection, then issuing SQL commands. A code module, which can be in a server-side compiled program or in a client-side Java applet, must first execute commands for establishing a connection with the database. Subsequent commands cause SQL queries to be executed. The records retrieved can then be displayed within a Web page. The code for establishing a database connection and issuing SQL commands is language- and database-specific.

Conclusion

If being on the leading edge is of no concern to the e-commerce organization developing a site, its developers should consider using ASP.NET, a compiled technology with many innovative and timesaving features suitable for building large Web applications. Developers should also consider using compiled server-side programs (including ASP.NET and the hybrid JSP) rather than server-side scripts when performing extensive or sophisticated database and Web processing operations. Compiled server-side programs are often written in Java, C++, or Visual Basic but can be written in any language supporting the appropriate Web server communication protocols.

Compiled server-side programs often use the server-independent CGI communications protocol. Although CGI does not use Web server resources as efficiently as newer scripting and compiled technologies, a few Web servers include a modified form of CGI that solves the older CGI efficiency problems. Additional public and proprietary products are available for upgrading the efficiency of CGI programs running on Microsoft and Unix/Linux Web servers.

If a particular Web server does not have to process thousands or even tens of HTML forms per second and its Web programs tend to be small, the developer should use server-side scripts to take advantage of their relatively easy development and maintenance. PERL scripts tend to be cryptic and difficult to maintain but are supported on almost all operating systems and Web servers. Cold Fusion and ASP scripts are easier to understand and maintain.

Client-side processing can be used to perform database processing if Java applets are used. However, although this approach offloads much of the processing from the Web server to the client workstation, it also involves limitations; for example, Java applets behave differently depending on browser type and version and cannot interact with data on the client workstation. JavaScript and VBScript can't interact with a remote database but can be used for data verification and error checking.

While other Web and database technologies are available and continue to emerge, the underlying concepts and implications of server-side and client-side processing discussed here remain constants in the rapidly evolving environment of Web development. C

References

1. Anderson, R., Francis, B., Homer, A., Howard, R., Sussman, D., and Watson, K. *Professional ASP.NET.* Wrox Press, Ltd., Birmingham, U.K., 2001.
2. Brown, S., Burdick, R., Falkner, J., Galbraith, B., Johnson, R., Kim, L., Kochmer, C., Kristmundsson, T., and Li, S. *Professional JSP, 2nd Ed.* Wrox Press, Ltd., Birmingham, U.K., 2001.
3. Kaparthi, S. and Kaparthi, R. *Cold Fusion.* Course Technology, Cambridge, MA, 2002.
4. Morrison, M. and Morrison, J. *Database-driven Web Sites.* Course Technology, Cambridge, MA, 2000.
5. Wagner, R., Daniels, K., Griffin, G., Haddad, C., and Nasr, J. *JavaScript Unleashed, 2nd Ed.* SAMS Net, Indianapolis, IN, 1997.
6. Walther, S. *Active Server Pages.* SAMS Net, Indianapolis, IN, 1998.

Mike Morrison (morriscm@uwec.edu) is an associate professor in the Department of Management Information Systems at the University of Wisconsin, Eau Claire.

Joline Morrison (morrisjp@uwec.edu) is an associate professor in the Department of Management Information Systems at the University of Wisconsin, Eau Claire.

Anthony Keys (keysac@uwec.edu) is an assistant professor in the Department of Management Information Systems at the University of Wisconsin, Eau Claire.