

Patterns for Producing Web-Enabled Applications

Excerpted from SAS Institute White Paper

Introduction

In the chess game of applications deployment, the Web has put your King in check. As you scramble to make sense of all your options, we'd like to take a few minutes to show you how using SAS software allows you to bring your game to a winning conclusion.

It's the lure of easy accessibility that makes the Web so compelling. Anyone with a browser can get to your application if you Web-enable it. The distribution of the software for your application is then taken care of by virtue of the connection between your user's browser and the Web server. The data for the application, as well as the actual executables, is under centralized (read "your") control on a server, minimizing the impact of programmatic and content changes on your user community. When you come up with a new and better version of an application, or when the data changes, your users experience no change in the way they access and use your application. That sounds great, but how do you put the pieces together for your application? For instance:

- When is CGI better than Java?
- When should you consider Dynamic HTML instead of Java Applets or Active X controls for Web clients?
- At what point does report generation with dynamic SQL (HTMSQL) queries run out of steam?

These are real questions, faced by anyone interested in using the Web to deliver decision support capabilities to their users. The scope of SAS software's Web infrastructure gives you a big advantage when answering these questions. You have more choices when developing Web-enabled applications when SAS software is part of your solution. This paper provides a framework for answering implementation questions. It also points you to the components of the SAS System that are most appropriate for addressing the needs of specific application scenarios.

The Playing Board

Let's start by briefly examining what you may be trying to accomplish by Web-enabling your application. You probably want your users to be able to access information in their browser, manipulate information in their browser, and execute applications from their browser.

You may have some, or all, of those results in mind as you read this paper. Before we discuss matching technologies with results, let's briefly review some of the tools that you will need in order to make the Web an integral part of your solution.

Web Environment

Do you have an intranet? It's basically a Web server connected via TCP/IP to one or more Web browsers. This phrase can also be used to describe the Internet, but the connected browsers

can be anywhere in the world. As you can see there are two basic elements in the Web environment: the Web server and the Web browser. We'll mention one more element, the firewall, which is optional from a simple communications perspective, but essential from the security perspective.

Web Servers

There are lots of **Web servers** in the market to choose from, and most offer similar functionality in their core deliverables:

- **HTTP** as the transmission protocol. HTTP stands for HyperText Transport Protocol, and is a standard for Web-based transmissions.
- **CGI** as a framework for invoking executables in the server. The Common Gateway Interface (CGI) is the ad hoc standard method for a Web server to call another process into execution. A CGI program can be written in any of several programming languages, and is supported by all Web servers.
- **Java servlets** and **Java Server Pages** provide an alternative method of invoking server-based Java executables. Think of Java servlets as CGI programs written in Java, with better features for performance and state management. Servlets are now supported more readily by most Web servers. Plug-ins can be downloaded and installed on those which provide no automatic support in order to enable the execution of servlets.
- **Active Server Pages (ASP)** is another server-based method of invoking executables. This is a feature of Microsoft's Web servers, although there are some ongoing efforts to port it to other platforms.

Web Browsers

There are also many **Web browsers** to choose from, but most organizations find themselves choosing between Microsoft's Internet Explorer and Netscape's Navigator. Although there are many differentiating features, we will focus only on their similarities in this paper:

- **HTML** is the common language spoken by Web browsers. HTML is the acronym for the HyperText Markup Language. Content-written HTML is interpreted by the Web browser in order to format a **Web page**. HTML provides simple formatting and form processing capabilities, font control, table generation and population, hypertext linking, and syntax to support the delivery and execution of Java applets.
- **Dynamic HTML (DHTML)** is supported. However, there are differences in implementation. DHTML is an advanced version of HTML, supporting the Document Object Model that is proposed by the Internet Engineering Task Force

(IETF) in HTML 4.0. Along with superior formatting capabilities, DHTML provides conditional logic, and dynamic execution capabilities that are absent from HTML 3.2.

- **Cascading Style Sheets (CSS)** are built upon the use of one or many style sheets. This construct is a set of guidelines that tell a browser how an HTML documents viewing can be influenced. A style sheet can influence the color, font, size, spacing, and placement of objects. Several different style sheets can be attached to a document. They can cascade, hence the term Cascading Style Sheets.
- **JavaScript** is supported, but with differences in implementation. JavaScript has no relationship with the Java language other than the first four letters of its name. It was first presented for use in the Netscape Navigator Web browser as a way of providing conditional logic, and more control over the interactivity of the Web page than could be realized with straight HTML. It has since been standardized as ECMAScript, and has been adopted for use in Microsoft's Internet Explorer as Jscript.
- **Java applets** are supported by an embedded Java Virtual Machine (JVM), with differences in implementation. A Java applet is an executable program written in the Java language and designed to execute inside the browser using the browser's JVM as a runtime interpreter. We will not go into more detail on the construction of the applet, or the JVM runtime environment. Suffice to say that using Java applets is a way of delivering highly interactive user interfaces to the Web browser.
- **Active X Controls** are the marketing name for the 3rd Generation Object Linked Embedding (OLE) framework. This allows software components to co-operate. Active X provide the plumbing between distinct software objects.
- **Multimedia** can be displayed in the browser, either natively, e.g., GIF or JPEG files, or via "helper" applications that are available on the browser machine and dynamically called into execution by the browser. Other multimedia events that can be displayed include sound files, streaming audio, video files, streaming video, and Virtual Reality Markup Language (VRML) files.

Firewalls

If your Web environment extends outside your company to the Internet, you may have a firewall that protects the portion of your Web that is company confidential from the eyes and keyboards of potential intruders. A firewall is a special type of Web server that intercepts requests from browsers, selectively transmits the requests to internal Web servers for resolution, and returns the resulting datastream to the requesting browser. The selective nature of the firewall is what enables it to prevent unknown users, or unauthorized communications methods from entering the private area of your Web.

The Pieces

Think of the Web environment described in the last section as the board in a chess game. If you don't have enough squares on the board, it's really hard to play the game. In this section we will describe the "pieces" that we use to create your applications. Each piece has intrinsic capabilities that make it suitable for one or more aspects of your application.

Language

The language(s) that you use in Web-enabling your application has a direct impact on the features you have at your disposal. You will probably end up using more than one of the languages listed below, since they address different aspects of the Web enabling process.

- **Standard Generalized Markup Language (SGML)** — is a document language. It is considered the parent definition of all markup languages.
- **Hypertext Markup Language (HTML)** — is an interpreted language responsible for managing the visual aspects of Web documents.
- **JavaScript** — also an interpreted language that can be used to provide more client-side interactivity to your user interface than HTML can provide.
- **Java** — a language that is interpreted by a JVM and capable of delivering high levels of interactivity to a Web browser in the form of applets.
- **SAS** — the SAS programming language provides support for both procedural language artifacts and syntax for calling industry standard analytic sub-routines. SAS software also provides a macro language for pre-compilation code generation.
- **SCL** — the SAS Component Language, a language developed by SAS Institute Inc. to provide a component architecture for delivering SAS server capabilities to distributed application environments, including the Web.
- **SQL** — the Structured Query Language, is an ad hoc standard method for extracting information from relational databases (RDBMS). Proficiency in SQL makes it easy for you to develop Web-based applications that generate dynamic reports based on data stored in SAS as well as in relational databases like Microsoft Access, IBM's DB2, Oracle, Sybase, and Informix.
- **htmlSQL** — a markup language delivered as part of SAS/IntrNet software that allows the embedding of SQL queries in an HTML page. The SQL is parsed out by a CGI program that communicates with the SAS server. The CGI program then makes the request for data, formats the result set, and then passes an HTML formatted page containing the resulting report back to the browser.

- **Visual Basic (VB)** — a Microsoft language that provides an ease-of-use paradigm that extends from the development environment through to the resulting applications. It is only available on Microsoft Windows platforms, and is only a factor in Web-based applications when the programs conform to the ActiveX specification.
- **XML** — the eXtensible Markup Language, is a recent addition to the list of languages available to Web applications. It complements most other languages because its primary focus is the structure and transmission of data across the Web. XML provides support for managing the data-dependent aspects of content once it is delivered to the browser.
- **PERL** — an excellent scripting language for CGI programs. It is in the public domain and can be used and distributed with minimal hassles.
- **C/C++** — these languages can be used to tackle the most daunting and complex programming. If efficiency in execution is paramount in your development efforts, C/C++ are your best choices.

Component Models

Your choice of language for your application determines the approach to component architecture that is available to you. Writing an application using components is a great way to modularize the different functions that are required to produce the appropriate results for your users. In addition, components that are written to adhere to a standard set of Application Programming Interfaces (APIs), can be re-used in other applications requiring similar functionality. There are three component architectures in vogue today for Web-based applications:

- **JavaBeans** — when you write programs in the Java language, adhering to the JavaBeans specification allows your program to be treated as a component, giving it the ability to interact with other components to form a loosely coupled (from a development perspective) highly integrated (from the user's perspective) application. The effort to scale the JavaBeans component model to the enterprise was the impetus for the Enterprise JavaBeans (EJB) specification. EJB promises to be a significant factor in the unification of object oriented protocols, especially since the Object Management Group (OMG) has elected to use it as a template for the CORBA 3.0 specification. When that occurs, CORBA compliance means EJB compliance and vice versa, which is very attractive to anyone writing multi-platform applications.
- **COM/DCOM/ActiveX** — when you write programs in either Microsoft C/C++, VB, or Delphi can save components that can be activated inside the Microsoft Internet Explorer Web browser. These components are called ActiveX controls, and can be used to deliver platform-specific (Windows only) functionality and interactivity

inside the browser, as well as be used for active server pages (ASP) and database access.

- **Remote Object Class Factory (ROCF)** — a methodology distributed with SAS Institute's AppDev Studio, a complete Java development environment that permits the creation of dynamic bindings between information delivery Java applets or applications, and SCL objects that run on the SAS server. Using ROCF as a component architecture eases the migration of some SCL-based applications to the Web.

Data Communications Protocols

Web-based applications make frequent use of data stored in databases, especially when the requirements include dynamic report generation and distribution. There are three principle protocols in use today that satisfy the needs of most Web-based reporting applications:

- **Open Data Base Connectivity (ODBC) protocol** — Microsoft's method of allowing applications to communicate with an RDBMS. Think of ODBC as the wire connecting your application with the data; queries can pass over the wire to actually make the connection and get the data for your report.
- **Java Data Base Connectivity (JDBC) protocol** — the Java method of allowing or applications written in Java (the applets, servlets, or executables) to communicate with an RDBMS. The JDBC-ODBC bridge, part of the Java specification, allows a Java servlet to use ODBC on the server, or the database vendor may provide direct JDBC bindings. JDBC is the functional equivalent to ODBC, but since it is Java-based, it is platform independent.
- **SAS/SHARE** — The multi-user data server for the SAS System. Delivers synchronous, queued responses to client requests. When combined with SAS/IntrNet it is responsive to both ODBC, JDBC and hmsQL clients. For SAS software-based applications, SAS/SHARE is the most efficient means of creating SQL-driven content.
- **SAS/CONNECT** — The access point for requesting synchronous or asynchronous compute services from the SAS server. When used in conjunction with SAS/IntrNet, Web applications using HTML or Java clients can perform two-way communications with the SAS server.

Development Environments

One last player that you may need in your quest to build a Web-based application is a development environment. Although many hard-core programmers insist on using bare-bones text editors to create and debug their programs, there are many sophisticated Integrated Development Environments (IDEs) for the languages and protocols listed above. This paper is not intended to be an exhaustive review of them.

The following list is a sample of the more popular ones, and their relevance to developing Web-based applications:

- **Java IDEs** — this group of development environments focuses on allowing you to produce Java applications, servlets, or applets. Most of the vendors in this category deliver visual environments that allow both experienced and novice programmers to become adept with Java. Leaders in this market include:

Visual Café — Symantec
Jbuilder — Inprise (formerly Borland)
VisualAge™ — IBM
Supercede — Supercede
webAF — SAS Institute
Visual J++ — Microsoft

- **Visual Basic (VB)** — Microsoft is the only vendor to support VB, but it is one of the most widely used desktop languages. Like the Java IDEs mentioned in the previous bullet, VB is visually oriented and to a large extent shields developers from syntax.
- **SAS/AF** — a development environment from SAS used for SAS software-centric development. It provides a framework for the creation of SCL objects and applications that can deliver SAS server capabilities to distributed environments.
- **SAS/EIS** — a visual development environment from SAS that allows predefined classes written in SCL to be easily assembled into applications.
- **C/C++** — there are several vendors in this market, many of them are listed in the Java IDE section above. If your staff is experienced in these languages, C/C++ may be viable alternatives to Java, VB, and SCL in developing Web-based applications. ActiveX controls can be written in C/C++. Program modules that interface with the SAS System and provide customized capabilities can be written in C using SAS/TOOLKIT. C/C++ programs can be written to interoperate with Java components.

Matching Your Application Needs With Available Technology

Once you have the pieces needed to assemble a Web-based application, the next step is to develop patterns of use that maximize your skill set and your target environment. To do this, we will adapt the pattern boilerplate developed by Gamma, et al., in their groundbreaking book, *Design Patterns*¹. To get started with patterns, let's briefly describe the boilerplate that we'll be using.

A design pattern is a method of describing a recurring problem, and outlining a generalized solution that allows the problem to be resolved in different contexts. The design patterns we are

describing in this paper all center around the work that needs to be done in order to Web-enable some or all of a SAS software-based application. To describe the patterns we will examine the following aspects of each:

- **Intent** — describes the problem addressed by the pattern
- **Motivation** — a short description of how the pattern solves the problem
- **Applicability** — tells in what situations the pattern can be used
- **Participants** — names the components that comprise the pattern
- **Collaborations** — tells how the components interact
- **Implementation** — names potential problems that may arise when using the pattern
- **Samples** — actual application fragments that demonstrate the pattern in action

Pattern 1 — Web Publishing

Intent

You want to deliver static information to a Web browser. Note that this static information can be updated/replaced on any appropriate scheduled basis, e.g., by a nightly or weekly production process. For example, you may have some SAS jobs that create tables or reports. Using the Web publishing pattern, you can make them accessible to anyone with a Web browser.

Motivation

If you have relatively stable information that needs wide dissemination, this pattern will show you how to assemble the pieces required to create a static Web page that can be viewed by anyone with a Web browser installed on their desktop and access to your Web server.

Applicability

You want your users to be able to access information in their browser, without requiring the installation of special software on their desktops.

Participants

- **Web browser** — The content viewer, by default only talks to the Web server.
- **Web server** — The content broker, accepts requests for information from the Web browser, and delivers files to the browser for viewing.
- **content generator** — This participant is the heart of the pattern, since it creates the value of the application. The simplest content generator could be someone that writes some HTML, and saves the file in a Web server directory. At the other end of the spectrum, SAS software's powerful decision support and data warehousing muscle can be harnessed as a content generator, simply by creating its output as HTML and transporting the resulting files to a

¹*Design Patterns*, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Addison-Wesley, 1995

Web server directory. In this pattern, the content generator is a batch or background process that occurs prior to the user's request that the information be displayed in their browser.

- **file transport mechanism** — The communications between the Web browser and the Web server facilitates file transport via HTTP, so that is not a consideration in this pattern. A method of delivering files from the content generator to the Web server is needed. File transfer protocol (FTP) is the most widely used, but there are others that may be available to you.

Collaborations

The content generator creates files that are suitable for the Web, and uses the file transport mechanism to place them in directories that are accessible by the Web server. Through a Web browser, the user requests the files from the Web server and displays their contents. HTML files are the most common means of delivering static information to a Web browser, in the form of a Web page. Web pages can be embellished by using graphics files as backgrounds, or as decorative elements used to highlight information, or as navigational elements. Graphics can be used as content, too, delivering the proverbial "picture worth a thousand words" in the form of GIF- or JPEG-formatted files. With the use of hypertext links that allow the Web page reader to click to other parts of the same page, or other Web pages, you can give your users many options to get to as much or as little of the information as they find necessary or appropriate.

Implementation

Known issues for Web publishing include:

1. Update frequency should match the expectations of users. Web publishing may yield to the next pattern, Dynamic Content Creation, if users expect to have frequent updates to the pages.
2. Use the least capable browser as your lowest common denominator when designing content. If you don't know who will be using your Web page, include a feedback point, usually a "MAILTO:" link, that lets your page's visitors tell you what they like or don't like about your content.

Samples

The SAS Output Delivery System (ODS) provides an excellent framework for publishing the output of SAS solutions to the Web. This sample shows how to publish a simple report:

Pattern 2 — Dynamic Content Creation

Intent

Provide a framework for delivering dynamically generated reports to a Web browser in response to user requests for information. This is different from the first pattern, Web Publishing, only in the timing of the response, not the nature of the content. The Web

Publishing pattern creates content in a process that is disconnected, or asynchronous, from the viewing of the content. This pattern creates content for immediate viewing, so content creation and content viewing are synchronously linked processes.

Motivation

The ability to generate reports on demand enlarges user expectations of a Web-based application to include ad hoc and heuristic information seeking. This expands the user's concept of what the application can be used for, and adds more flexibility to querying and reporting applications without direct interactions with data managers and application developers.

Applicability

You want your users to be able to manipulate information in their browser. One of the primary reasons for wanting this capability in your application is to avoid the costly and time-consuming regeneration of all the possible static Web pages from data repositories that users may need or want. Use the report distribution pattern to ensure that your users have precisely the information they need, derived from the most recent data available.

Participants

- **Web publishing pattern** — all the participants of the first pattern, including the Web browser, Web server, file transport mechanism, and the content generator. The Web server has an additional role in the pattern, in addition to delivering files upon request, it can initiate processes via CGI or native interfaces in order to satisfy more sophisticated requests. In this pattern, the content generator dynamically interacts with the browser by acting as either a query processor or a compute processor.
- **query processor** — one instance of a content generator that accepts SQL queries as input, and delivers formatted result sets as output. SAS/SHARE is the query processor for the SAS System. The htmSQL component and the SAS/SHARE Driver for Java, both part of SAS/IntrNet, provide entry points for sending SQL queries from the Web to the SAS/SHARE server.
- **compute processor** — another instance of a content generator that accepts programmatic directives as input and delivers formatted results as output. The primary difference between the query processor and the compute processor is the former relies on SQL to structure input to the process, where the latter can use whatever programming directives the compute processor is designed to accept. The SAS/IntrNet Application Dispatcher is the compute server for CGI-based Web applications, providing an entry point for requesting services from the SAS server that runs the gamut of SAS System capabilities. It uses SAS source code as programming directives for the SAS server.
- **data repository** — a place to store data that is accessible to the network, and accessible to the query and/or compute processor. It may be as a SAS data set, a SAS Multi-dimensional Data Base (MDDb), a DB2 or Oracle table, or any other format.

Collaborations

The user requests information from the Web server in the form of a URL that houses either an embedded query or a pointer to a program. The query or program is used to extract data from the data repository by either the query processor or the compute processor. The results are formatted and delivered to the browser via the file transport mechanism.

Like Web publishing, dynamic content creation relies on the creation of suitable files for viewing in a Web browser, so HTML and graphics formats are fundamental structures of this pattern. Additionally, the use of dynamic processes for content creation introduce two new structures: The process spawner (mechanism that allows a Web server to spawn a new process. This comes in several flavors — CGI, native Web server APIs like ISAPI and NSAPI, Java servlets, and Active Server Pages), and distributed content viewers (Java applets and ActiveX controls).

Process Spawners

Process spawners are currently the most commonly used method of distributing reports via the Web. They are simple in concept, reliable, and reasonably efficient in terms of processor overhead and bandwidth consumption.

The four aforementioned flavors all have the attribute of being able to initiate server-based processes, albeit using different methods. CGI is the preferred method for applications that are required to run on different vendors' Web servers, since the CGI specifications are vendor independent. As an example, the SAS/IntrNet component htmSQL is a very compact CGI program that can be executed on virtually any Web server on the market because of the universal acceptance of the CGI specs.

The two Web server APIs, ISAPI and NSAPI are typically used in applications that require transactional throughput rates, supporting small, repetitive content creation requests. Using the APIs may yield a slightly more efficient transactional application, but it is only portable among Web servers that support the same API, and in some cases only the same Web server vendors implementation of the API. SAS/IntrNet does not support these native APIs because they lack portability and provide only negligible performance gains over its highly optimized CGI implementations.

Java servlets are similar to CGI, except they are written in Java, and require a Java Virtual Machine in order to execute. If a Web server supports Java servlets, a JVM is usually included with the Web server, simplifying implementation logistics. As the CORBA and Enterprise JavaBeans specifications merge in the CORBA3.0 specification, Java servlets will become more popular as a means of enabling platform independent server-based functionality. Servlet support will arrive in the V8 timeframe for the SAS System.

Active Server Pages (ASP) are Microsoft's specification for enabling server-based functionality via the Web. An ASP is an executable

file that runs on Windows, and can communicate with other services that are available on the host machine, e.g., the Microsoft Transaction Server for transaction support, or MSMQ, Microsoft's message-queue product for asynchronous content creation. ASP support is included in SAS Integration Technologies.

Distributed Content Viewers

Java applets are packets of executable code that can be delivered to a Web browser embedded in an HTML file. Once the HTML file arrives at the browser, the browser initiates another file transfer from the Web server for the applet code. The applet code is then "executed" by the Web browser, allowing the applet to do what it does using the browser's JavaVirtual Machine (JVM) as its runtime environment.

ActiveX controls are similar in function to Java applets, but are limited to executing on Windows-based clients, and are capable of becoming resident on the client machine. The latter feature is desirable in corporate settings as a software distribution mechanism, but considered a severe security limitation when used as a distribution agent for executable code. In addition, ActiveX controls can interact with system services and peripherals once they begin executing on the client. This is a double-edged sword, since the same control that can print on a local printer can also delete files on the local hard drive.

Implementation

Known issues for dynamic content creation include:

- It's important to match the capacity of the content generator with the capacity of the viewer, e.g., a thousand-line report is probably best viewed in small pieces, or mulled over on paper while jetting to your next overseas meeting.
- Minimize the number of required steps to get the desired report. Too many options can reduce the effectiveness of distributed reports by delaying their delivery.
- Query processors that use the SQL language provide a good, generalized mechanism for translating many report requests into reports. SQL code can be generated by the application, shielding the user from the vagaries of the language. The htmSQL component and the SAS/SHARE Driver for Java in conjunction with the SAS/SHARE server, provide excellent query processing capabilities for Web-enabled SAS applications.
- Compute processors, such as the server capabilities of the SAS System, provide a wide array of tools to access virtually any data source, manipulate it, and then produce the desired report in a wide range of Internet content types, e.g., HTML, GIF, JPEG, CSV, etc.
- For maximum flexibility and utility, design or use content generators that are robust enough to format reports with embedded links that provide additional report generation capabilities from within the original generated report.

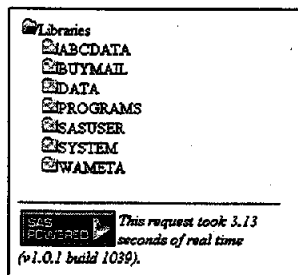


Figure 1. Directory Tree of Libraries

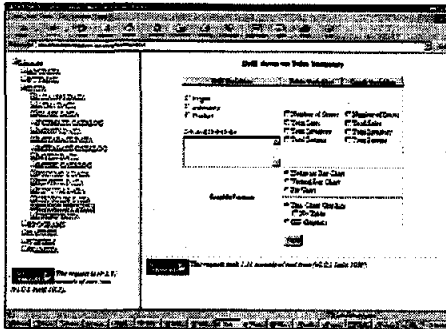


Figure 2. Generated Web page

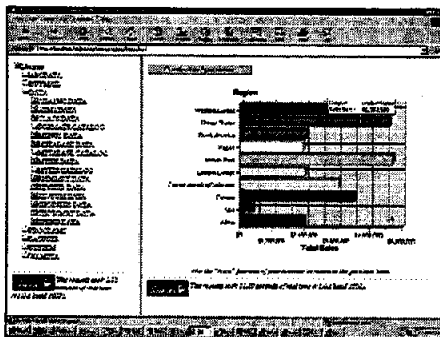


Figure 3. Generated Output

Samples

We will use Xplore, one of the samples from the SAS/IntrNet product package, to illustrate this pattern. Xplore is a Web-based explorer designed to detect SAS catalog entries and perform type-specific operations on them.

The panel in the first figure (Figure 1) shows a directory tree of all the SAS libraries that are currently allocated to the SAS session. Folder icons can be double clicked with a mouse to reveal the contents of the directory or catalog (Hint: the little red ball at the bottom right of an icon indicates it is a SAS artifact, not an operating system directory).

Different member types are distinguished with different icons, and each type has an action associated with it that determines the behavior of Xplore when that member type is double clicked.

A SAS data set, for example, can be displayed as a table in the browser by double clicking on its icon. A stored graph can

be displayed, a source code entry can be browsed, etc. Xplore has a special action for output data sets from the SUMMARY procedure that first determines the CLASS and analysis variables. Then it displays a Web page that allows the user to request tabular or graphical representation of the data with drilldown points determined by the CLASS variables that are selected. Figure 2 shows the generated Web page and the selections marked by the user. The SUBMIT button initiates the creation of content shown in Figure 3, using Java applets as content viewers that support the use of mouse clicks to drill down into the underlying data.

Pattern 3 — Application Distribution

Intent

Distribute executable application code on demand, while maintaining central control of source management.

Motivation

Application users are becoming more scattered and disconnected from centralized IT resources. To ease the maintenance burden of managing distributed applications, IT is shifting from deployment via software installation to deployment via either "push" or "pull" methods of file transfer. Although outside the scope of this pattern, many of the concerns elevated by the total cost of ownership (TCO) industry initiative have been addressed with configuration management methodologies that center around a combination of push and pull software installation and maintenance. This pattern is an instance of that effort, in that it satisfies the sometimes transient needs of the desktop users instead of the institution of desktop administrators. Push deployment is proactive in the placement of data and executables onto client machines, while pull deployment passively makes data and executables available for use on demand. This pattern embraces both methods, but for sake of brevity we will only discuss pull style deployment in this paper.

Applicability

You want your users to be able to execute applications from within their browser.

Participants

- **Web publishing pattern** — all the participants of the first pattern, including the Web browser, Web server, file transport mechanism, and the content generator. In this pattern, the content generator, or some portion of that function, may be delivered to the browser or desktop at which point content generation takes place.
- **Push/Pull mechanism** — the push mechanism is a transmission point in a network that targets desktops via IP address, catalogs information about the configuration of applications, and actively updates application software, all without requiring interaction from the desktop user. The pull mechanism is more passive, and only broadcasts the

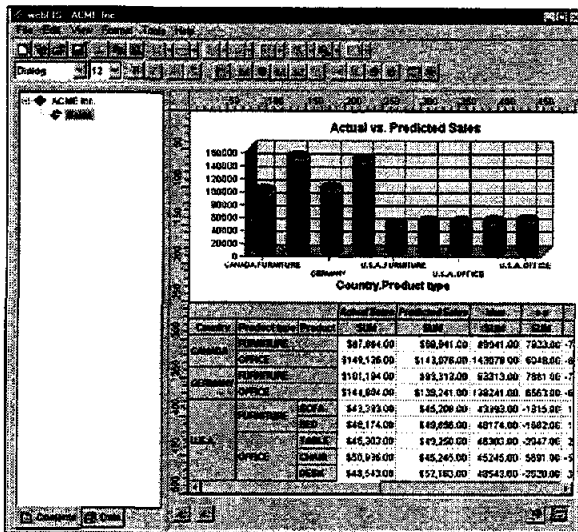


Figure 4. Distribution of application code on demand

availability of new data or application software. A combination of push and pull is referred to as a publish/subscribe loop where the content generator broadcasts the availability of new data or software, and any active subscribers to that thread of information will automatically receive the new information. In that case, the content generation process is still passive, but desktop users can activate it automatically by subscribing to the process.

- **Application repository** — this is the corollary structure to the data repository. It provides a place to store and retrieve application logic. The management of the repository can range from simplistic, as in the use of a SAS catalog or partitioned data set, to sophisticated. The more sophisticated management regimes typically employ either data base management systems for filing and retrieving program locations, or if the application uses object orientation, an object request broker, perhaps coupled with a directory server for finding objects that are scattered across the Web.

Collaborations

The user makes a selection in the browser, which causes the Web server to initiate a request for content. The fulfillment of the request can take place on the client itself, in which case the server delivers the executable and then does no further processing for that request. Or fulfillment may require a program execution either on the Web server processor or another server on the network. The application repository, responding to the Web server, locates and makes the program available for execution. The delivery of content to the browser is delivered through the Web server by either the application repository, or the executing application, depending on the design of the system. As in the previous two patterns, HTML remains the primary transport mechanism for application distribution. Highly interactive applications may require the use of more sophisticated tools that can be embedded into the HTML file that is delivered to the browser. Dynamic

HTML (DHTML), JavaScript, VBScript, Java and ActiveX can all be used to make a Web page more dynamic and interactive.

Applications need not be delivered in total to the client, but can be partitioned with some functions occurring on the client and some on the server. This is a useful technique to minimize the transfer time for startup on the client.

A new structure, the application repository, is introduced with this pattern. The application repository can be thought of as the library from which applications can be checked out and used by clients, and checked in and maintained by content generators. The complexity of the repository is dependent on the usage patterns exhibited by the user community. In its simplest form, the repository is a read-only directory located on a server. More complex forms might include a directory service to provide look-ups for both authorization and search fulfillment. Other instances may use messaging middleware to notify servers of pending requests for their services. Generally speaking, more complex designs are warranted by needs of scalability and availability. However, some form of an application repository is required for distributed deployment of applications.

Implementation

Known issues for Application Distribution:

- Bandwidth is a limiting factor in any scenario that relies on moving data from one place to another via a network. Application distribution assumes minimal latency for transferring the executable application file to the client, an assumption that should not be overlooked. If the size of the file is large, in excess of 1 MB, mobile users connected at 28.8 kbps will experience wait times measured in minutes. LAN or WAN users of the same-sized application may only have to wait seconds, depending on the state of the network at the time.
- It's important to factor in both the length of download with the frequency and length of use. For example, a two-minute download for an application you will be running for a long time (hours or even all day) is tolerable. But the same download for something that you will only be using briefly is not acceptable.
- Client capabilities/portability may preclude ActiveX, or limit JavaScript/Jscript usage. In that case Java becomes the most viable option for distributing program logic.

Samples

One of the components of AppDev Studio, webEIS, demonstrates this pattern by providing a framework for the creation of dynamic content. Applications can then be deployed using the tenets of this pattern. For brevity we will skip the steps used to create the application, and go directly to what happens when someone wants to use it.

As with all Web-enabled applications, we start with a URL in a browser window, which launches the file transfer component and starts moving program logic (in the case of webEIS applets, the program logic is in Java classes packaged and compressed into Java Archive (JAR) files). Once the applet is initialized, the user begins to see the shape of the information being presented. In our example (Figure 4), the applet is using a tree diagram to depict files being used, and a table to view the information. Once in place and executing in the browser, the applet can be directed to show other files in the table viewer, or to change the display to a graph.

The primary difference between this pattern and the previous one, Dynamic Content Creation, is that content displayed using the previous pattern was created with a server process and delivered to the browser for viewing. Using the Application Distribution pattern, the content generation process is delivered to the browser, and content creation takes place inside the browser (or on the desktop if the content generator is not an applet or an ActiveX control).

Products from SAS Institute

To summarize, and put all this in context, the following SAS products can play a role in Web-enabled applications. The table that follows shows in general which product or component can be used to address which pattern.

SAS Product or Component	Pattern		
	Web Publish	Dynamic Content	Application Distribution
Base SAS*	✓	✓	✓
– ODS	✓		
– Web Publishing Macros	✓		
SAS/MDDB Server*		✓	✓
SAS/ACCESS* products		✓	✓
SAS/CONNECT*			✓
SAS/SHARE*		✓	✓
SAS/IntrNet*		✓	✓
– Application Dispatcher		✓	✓
– htmSQL		✓	
– SAS/CONNECT Java driver		✓	✓
– SAS/SHARE Java Driver		✓	✓
webAF™			✓
webEIS™		✓	
SAS Integration Technologies		✓	✓
IOM		✓	✓
MQInterface		✓	✓
Directory Support		✓	✓
Pub/Sub		✓	✓
Enterprise Reporter™	✓	✓	

Figure 5. Title – SAS Web Enablement Products

- **Base SAS** — in the context of this paper includes:

Data repository using SAS Data Sets
Application repository using SAS Catalogs
Web Publishing Tools — for Web publishing in V6 and beyond
Output Delivery System (ODS) for Web publishing in V7 and beyond

- **SAS/MDDB Server** — specialized data repository for summarized multidimensional data
- **SAS/ACCESS products** — gateway to third-party data repositories
- **SAS/SHARE** — multi-user data repository, manages SAS Data Sets, MDDBs, or third-party repositories.
- **SAS/CONNECT** — file transport mechanism, SAS client and SAS server communications
- **SAS/IntrNet** — the Web-enabling product for the SAS System, includes:
 - Application Dispatcher* — CGI-based SAS application server
 - htmSQL* — CGI-based query processor which enables publishing/formatting of the result set as HTML

Java classes, for inclusion in user-written applications, including:

- *Java Driver* for SAS/CONNECT — Java class for accessing a SAS application server from Java
- *Java Driver* for SAS/SHARE — Java class for report distribution

Java Tunnel Feature — CGI-based HTTP Tunnel, for drilling through firewalls

- **AppDev Studio** — Windows-based developers toolkit, includes developer versions of the following SAS products:

BASE SAS	SAS/CONNECT
SAS/GRAPH	SAS/SHARE
SAS/AF	SAS/IntrNet
SAS/FSP	SAS Integration Technologies
SAS/EIS	

AppDev Studio also contains the following components:

- **webAF** — Java Integrated Development Environment, includes the Remote Object Class Factory (ROCF — Using ROCF, users can access SAS/AF models seamlessly in a networked computing environment. ROCF provides the ability to automatically encapsulate AF models in Java so that a client application can utilize them in an open solution without having to write any specialized code. For application distribution.

- *webEIS* — Java Applet/Application builder, for report distribution.
- **SAS Integration Technologies** — SAS Integration Technologies is a collection of enabling technologies, all designed to leverage the portability and back-end strengths of the SAS System's core technologies, extending access and presentation models beyond those available within the current V6/V7 batch and interactive frameworks. These technologies include:
 - *Interface Object Model (IOM)* — gateway support for CORBA, COM, and DCOM communications with the SAS server.
 - *MQInterface* — interface support for IBM's MQSeries and Microsoft's MSMQ, message-oriented middleware software products. Allows SAS applications written in SCL or DATA step language to place messages in the queue, or retrieve messages from the queue. Messages can be used to deliver and/or retrieve data, instructions for processing data, or executable binary files.
 - *LDAP Interface support* — directory service support for Lightweight Directory Access Protocol (LDAP) compliant directories. Allows the SAS server to find information stored in an LDAP directory, or to store information in an LDAP directory. This is a key scalability factor in the use of application repositories.
 - *Publish/Subscribe Package Framework (V8)* — Enables the programmatic collection, serialization and compression of result sets, and provides language support for publication and subscription services within the SAS System.
- **Enterprise Reporter (ER)** — In V6/V7, provides Windows-client-based report distribution, and standalone client reporting functionality including Web publishing. In V8, ER provides a similar feature set, but uses SAS Integration Technologies to access the SAS server.

Conclusions

First, let's answer the questions posed at the beginning of this paper:

- *When is CGI better than Java?* When you've got it working, and your application performs to your specifications.

CGI should be considered the first choice for getting started with Patterns two and three, since it is simple in both concept and implementation. In that context it is better than Java simply because you will finish your project in less time. There will be times, probably after your first try at Web enablement, when Java will be the only way to endow your client with the interactive features that your specifications require, or the only way to address client and server functionality using the same programming resources. You'll see.

- *When should you consider Dynamic HTML instead of Java for Web clients?* Unlike JavaScript, Java is very different from Dynamic HTML in a number of ways. Java is a full implementation of an object oriented programming language, and gives an application developer immense control over program flow and application development. From a programming capability perspective Dynamic HTML cannot compete with the Java language.
- *At what point does report generation with dynamic SQL queries run out of steam?* The answer really lies in how much analytic power do you need in order to assemble the reports that your users require on the Web. If you can give them what they want with SQL queries, you haven't run out of steam. But if their needs extend to ad hoc or preformatted analysis, sophisticated graphics, or anything that the SQL language can't do for them, your SQL-based content generator just ran out steam. When that happens, use the SAS/IntrNet Application Dispatcher to get your users the content they require. It harnesses the entire scope of data management and analytic capabilities of the SAS server to generate content from your data warehouse.

To the users, the ubiquity of the Web has simplified the information consumption process. But as you have probably guessed by now, those same compelling opportunities have not simplified the process of delivering information to users. You will probably need a mixture of products to cover all your bases. Some of those products will be from SAS Institute Inc. Our path to open technologies, demonstrated by the availability of enterprise connectivity products like SAS/IntrNet and SAS Integration Technologies, assures your continued success in creating Web-enabled, integrated solutions that address your business initiatives for data warehousing, business intelligence, and enterprise decision support.



SAS World Headquarters
SAS Campus Drive
Cary, NC 27513 USA
Tel: (919) 677 8000
Fax: (919) 677 4444
U.S. & Canada sales:
(800) 727 0025

SAS Europe, Middle East & Africa
PO Box 10 53 40
Neuenheimer Landstr. 28-30
D-69043 Heidelberg, Germany
Tel: (49) 6221 4160
Fax: (49) 6221 474850

SAS Asia/Pacific & Latin America
SAS Campus Drive
Cary, NC 27513 USA
Tel: (919) 677 8000
Fax: (919) 677 8144

www.sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © 2000, SAS Institute Inc. All rights reserved. 35758US.0800