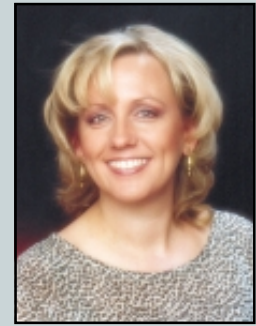


Before you build your Web site, you need a solid design.

# Blueprint for E-commerce Success



*Eden Watt*

## **At a Glance**

When you develop your first e-commerce site, you will discover that there are a few new things to learn about application development—Web style! Designing a functionally rich site requires an understanding of technical and architectural aspects of Web systems.

*By Eden Watt*

If your organization is not already in the midst of an e-business project, then you are probably evaluating the viability of such an endeavor. Given the phenomenal growth of the Web for commerce and business-to-business transactions, many are finding themselves drawn into this technology.

After you determine the best application for your organization to enter the world of e-business, the next step is to establish the architectural design, development approach, and tools you need to build your Web system. Many of the skills you have used to deliver robust and functional business applications are important as you enter the world of Web application development. Learning about the latest technologies can assist you in moving forward into your new system's medium. Some of the key technological concepts this article covers are: e-business architectural components; browser functionality; server-side programs; the statelessness of



HTTP applications; state management techniques, such as cookies; and Web system design considerations.

## **Application Architecture**

To understand an application developer's architectural view of Web serving's technological components, check out **Figure 1**. The three main components illustrated are a client workstation, a TCP/IP network, and a Web server. The language of the Web is HTTP, which must run over TCP/IP. The Internet is essentially a huge TCP/IP network with millions of interconnected systems. HTTP defines how a Web conversation can transpire, including syntax, functionality, standards, and security. In order to serve and request Web pages, a client must have browser software (like **Netscape Navigator** or **Internet Explorer**), and a Web server must be active. After a client workstation is connected to the Internet or intranet, the browser performs all the needed functionality for the user to carry on HTTP conversations with various servers. A user can request pages by clicking on

links or typing the URL into the browser's location or address field. The browser has many responsibilities and tasks, such as the following:

- Sending out requests for pages to servers on the network
- Passing authentication header (with information such as user name, browser type, IP address, etc.) to the server with each request
- Receiving, interpreting, and displaying those pages in the browser
- Interpreting and running JavaScript
- Requiring user sign-on if a server sends back a 401 error and forwarding to the server for authentication
- Creating and setting cookies (more on this later) at the request of server programs or JavaScript
- Managing file download requests
- Launching Java applets or other client applications associated with Multipurpose Internet Mail Extension (MIME) types

The beautiful part with regards to deployment and maintenance of these applications is that nothing need reside on the client workstation except a standard browser. The entire application, including Web pages, graphics, data, and application logic, typically resides on a server. The server-side application, in

Reprinted with permission from Midrange Computing, published by Midrange Computing, IIR Publications, Inc., in Carlsbad, CA; 760-931-8615; [www.midrangecomputing.com](http://www.midrangecomputing.com)

conjunction with the Web server, assimilates the required page and sends it down to the client. A Web server receives a variety of requests from a wide variety of clients in diverse locations, but all of these requests come in a uniform syntax. A Web server performs a number of basic functions, including the following:

- Retrieving a requested HTML page from a folder or library on the server system
- Launching a program that may be requested via Common Gateway Interface (CGI)
- Sending information to CGI programs via API calls
- Performing security functions, such as user authentication and restricting access to disallowed files and programs
- Launching Java servlets and responding to API calls from Java (if the server is Java-enabled)
- Encrypting pages if a Secure Sockets Layer (SSL) port is accessed
- Logging all activity and requests
- Sending a page back to the browser after it has been dynamically generated or retrieved
- Sending special instructions back to the browser through content headers to set cookies, request user sign-on, display errors, etc.
- Managing the jobs on the server and diverting requests to appropriate request processor jobs

### Statelessness

When you start developing Web applications, be prepared for some technological learning curves. In addition to figuring out how HTML works and how to code your server-side programs, you must understand some new paradigms with Web applications. Traditionally, many AS/400 shops have been coding native applications, which run within a job space. As code is run, oftentimes a user's input is required. For example, a screen appears while the program remains active; after the user enters the screen, the next line of code in the program runs. All the program variables, open files, and record pointers retain their values during this period. This is a typical procedural style application.

In the '90s, a few new computing concepts have altered our previous modularized, procedural view of the programming world. One such invention of the last decade is the event model, which is popular with desktop and client/server programming languages (such as Visual Basic). In this situation, the application sits in an idle state until the user performs some action, which could be a number of different events. When an event occurs, such as moving the mouse or clicking on an object, the appropriate code snippet comes to life. Object-oriented programming and object-oriented design techniques also have enhanced a developer's bag of tricks.

Most applications operating within HTTP behave in a manner similar to the event model. A user sitting at a client workstation on the Internet can make a request to any server on the Internet by simply sending out the appropriate URL. This URL may ask for a static page, or it may launch a server-side program that builds a dynamic page in response to the

user's request. Essentially, the server responds to requests, retrieves requested objects, and sends them back to the browser – then forgets the user exists! In many cases, this is a wise thing for the server to do because it may get requests from thousands of visitors a day. And visitors could, at any time, jump to another server and forget about the conversation they had initiated with the previous server. Generally speaking, Web users do not need to *sign off* from a server they go to.

Often, this Web characteristic is referred to as *statelessness*. By default, each request operates independently without any knowledge of previous requests or responses.


### State Management Techniques

From an application development perspective, the statelessness of the Web can be somewhat limiting. A typical business system has a series of input and

*Things to do When Mid-Range is Your Business Partner: #11*

# PLAY WASTEPAPER BASKETBALL

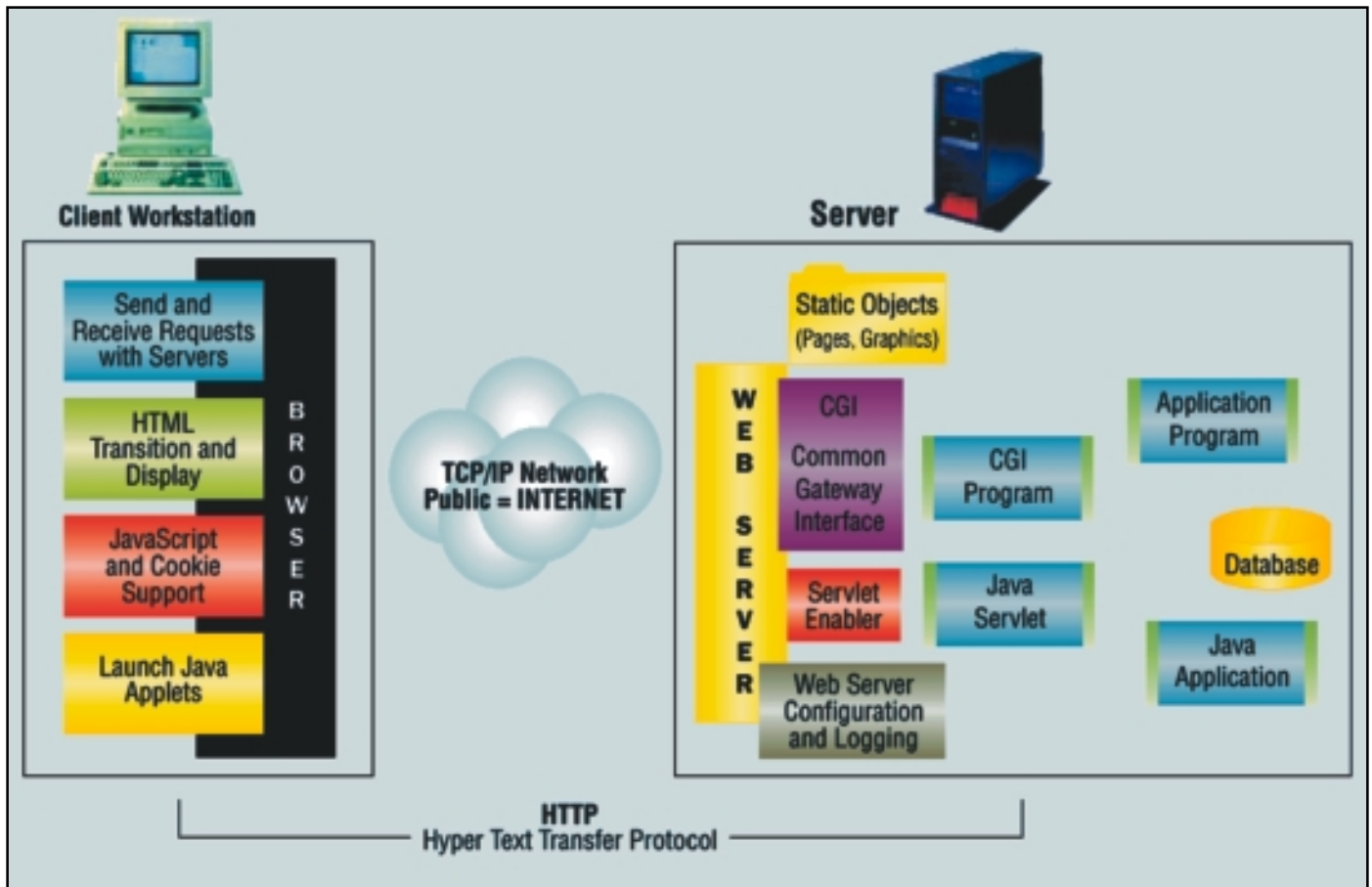
Because you're going to have a lot of time on your hands. At Mid-Range we're experts at keeping your iSeries 400 – AS/400 operating at peak performance. From application software recommendations / hardware upgrades and performance tuning through logical partitioning, operational support /education and disaster recovery we have what you need. *Call us so that you can spend more time practicing your jump shot.*

**MID-RANGE** 

*Working For Your Peace of Mind*

Call: 1-800-668-6470 [www.midrange.ca](http://www.midrange.ca)

See our booths (#25 & 26) at the TEC Showcase, April 24



**Figure 1**

output screens as well as various paths an application can take. Each screen presented is based on previous user input or results. To overcome the limitations associated with stateless applications, various techniques have been developed. The most prevalent of these techniques are parameter passing, cookies, environment variables, user authentication, and transaction processors.

**Parameter Passing.** A URL that calls a program can pass parameters by appending the URL with parameters that are concatenated with plus operators and preceded with a question mark, like so:

```
www.domain-name.com/cgi-bin/program-name?parm1+parm2+...
```

When a page is served dynamically, it means the page was not sitting in an HTML file on a server directory but rather a program on the server built the page on the fly (launched via CGI or servlets).

When the server-side program is constructing the next page, it can affect any of the code on the page to be served. So, if there is a link to another program from the page to be served, then the program about to serve the page can set up the parameters in the next call. In this way information can be sent back to the next program about the previous activity. Another way to modify the value of the parameters to be passed is to use JavaScript. JavaScript is embedded in the HTML, which the client's browser processes. Alternatively hidden fields could be set up on the HTML forms that are passed back to the host program.

**Cookies.** A popular, and on some sites, overused method of overcoming the stateless nature of Web applications is to use cookies. Cookies are small pieces of text that the browser can place on a client PC at the request of the application. All current browsers support this functionality. Although a user has the ability to disable cookies or force a

warning and decide at each instance whether or not to accept the cookies, this process can detract from the elegance of your application. Netscape stores all cookies in a file called *cookies.txt*, located in the Netscape directory. Internet Explorer, however, stores them as separate text files in the cookies directory under the Windows directory. The implementation of this is transparent to both the user and the developer.

The nice thing about cookies is that you, the developer, must specify how long they should remain on the user's workstation. So, you can use cookies to help track users and user movements or requests in your active application. Also, you can use cookies to help you remember certain information about a visitor every time that visitor comes to your site. You could get in trouble if you develop your application with the assumption that every cookie you send actually gets stored on the client. Users might refuse the cookie, or they might be on an older browser that does

not support cookies. Also, there are limitations on how much cookie information a browser can store before it starts replacing older cookies. For example, a limit on a client work-station is 4K or 100 cookies, and a server domain cannot send out more than 20 cookies to one user.

When using cookies, store only the identifying attributes in the cookie. When your server-side program retrieves the cookie, this usually serves as a key to retrieve other information about the user from your server's database files.

**Environment Variables.** A browser and server exchange a certain amount of information on every transmission. The browser sends the server some basic data about the client who is making the request in the request header. Web servers support access to this information via environment variables. Examples of information your server program can request include IP address, browser type, and user ID (if they are signed on – more

about this shortly). If you are within an intranet, you can use the IP address as an identifier for a user. But, for the Internet this method is not reliable. Often the user is behind a proxy server, which means all the users from that company have the same IP address. For example, if you rely on this method, everyone at IBM will seem to be the same person from outside IBM's firewall.

**User Authentication.** If you secure your directories and programs with user authentication, then you have a reliable and easy way to track visitors and their activities. You configure user authentication via the Web server. Typically, you do this when you want to restrict access in certain areas of your site to select people who have a user ID and password. You can set up user identification for tracking purposes. With user authentication, the server sends a *401 Unauthorized* error to your browser, causing a sign-on window to appear. After users have signed on, their user name is always accessible from the server-side via

the `REMOTE_USER` environment variable as long as they stay within the realm you have identified for the user set. If they jump to a non-authenticated area of your site, then the `REMOTE_USER` value is blank during that request. If users come back to the authenticated pages, they don't have to re-sign in because the user name has a value again.

**Transaction Processors.** Many application vendors – including **Microsoft, IBM, and LANSA** – have products with built-in capabilities that can overcome the stateless nature of the Web. In the LANSA for the Web product, if you choose to compile your applications to maintain a persistent connection, programs that serve Web pages stay active until the user comes back. The next request is then processed within the same job space, at the next line of code, with files and variables still active. Using this technology allows you to develop Web applications more rapidly because you are not doing special processing to keep track of visitors.



# ASTECH Solutions

**OUR SERVICES**

- Project Services**  
World-class application design, application development, project management, and implementation services.
- Technology Consulting**  
Our experts provide 'Clear Strategic Thinking' in our practice specialties - Application Development, Architecture, e-Business, and Project Management.
- Learning Services**  
Development and delivery of customer education by certified instructors.

**I.T. Consulting and Project Services**

For more information, call (905) 727-2384 or visit [www.atech.com](http://www.atech.com)

See our booths (#29 & 30) at the TEC Showcase, April 24

There are often limitations, however. For example, there can be some inconsistencies in the program logic when a person uses the browser Back button or jumps in and out of the application with a persistent state Web application. The server-side application has no control or knowledge over browser functions such as Backward and Forward. The general recommendations I give my clients is to use the persistent connection style application for intranets through which you have some control over your users and can give them instructions. For Internet applications intended for the general public, developing and deploying systems by using the independent event model is a better idea. The other techniques, described previously, for state management (such as cookies), can be used in these situations, without

compromising user functionality and navigational ability.

### Server-side Program Initiation

Although you can launch server-side programs by issuing commands from your browser, your browser does not interact directly with your server-side programs. Instructions are sent to the Web server and then server-side programs interact with your Web server to get any required information. An endless array of back-end tools, programs (compiled), scripts (interpretive), platforms, and Web servers exist, but there are really only a couple of different methods of invoking server-side programs from the browser interface:

1. Jumping to a program on a server through a direct URL link, such as:

- The <A HREF> tag in HTML
- The server-side include (SSI) EXEC in HTML
- The entry of a URL in browser Address Entry

2. Submission of an HTML form to a server program

You can construct the program URL in different ways, depending on whether parameters are being passed into the program (typical) and whether the program is initiated via CGI, Java servlets, or a vendor-specific built-in interface, such as Domino.

CGI is the standard interface for launching programs in the HTTP 1.0 standard. Script or program libraries are configured in the Web server, and programs (written in any language) would need to reside in the allowable libraries. Some critics claim CGI launched programs are slower than newer techniques. CGI programs are slower for a couple of reasons. First, traditional CGI programs were written in interpretive languages like Perl. Second, many such programs have been designed to start up on each request and then terminate. Java servlets have been gaining a lot of attention lately as a more current way to launch the server-side programs in a Web application. If the Web server is Java-enabled, then servlet class files can interface with the Web server. Typically, a Java servlet stays in memory between requests and so the execution time should be faster. For more information on Java servlets, refer to at [www.javasoft.com](http://www.javasoft.com) or [www.as400.ibm.com/developer/java/servlet1.html](http://www.as400.ibm.com/developer/java/servlet1.html). The actual URL format for a server-side program may look something like this, if it is a cgi-bin call:

```
www.domain-name.com/cgi-bin/
lansaweb?procfun+frunews01+
new0101+web+englf
```

If it is called via a Java servlet, it may look like this:

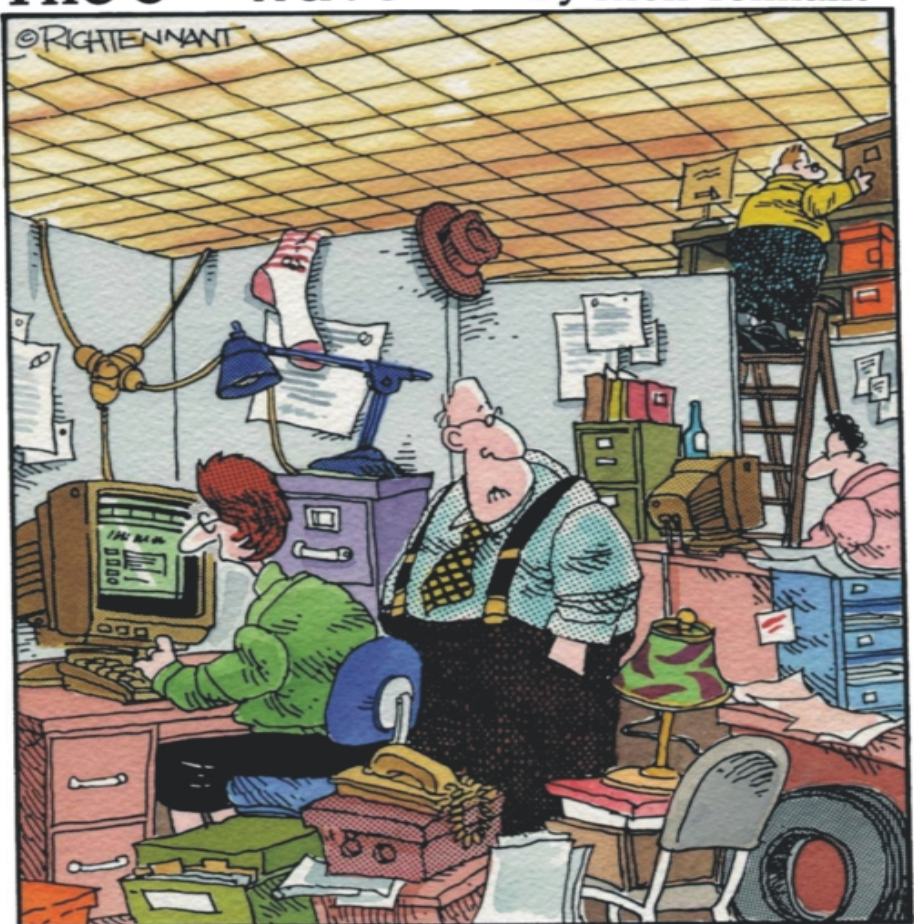
```
www.domain-name.com/lansaweb?
procfun+frunews01+new0101+web+eng
```

For a Domino database, the actual URL may look like this:

```
www.domain-name.com/
notesdb.nsf?OpenDatabase
```

©The 5th Wave, www.the5thwave.com

## The 5th Wave By Rich Tennant



"Just how accurately should my Web site reflect my place of business?"

## Browser Interface Design

Decisions you make about HTML page design and standards for your Web site can have an affect on your e-business application design. After working closely with IBM on their AS/400 Web site, I'd like to share a couple of standards or initiatives that had a impact on development.

**Frames vs. No Frames.** Many sites on the Web use frames, which means the page is divided into sections, and different pages and programs (i.e., URLs) can fill in each section of the page. Every call to a server can direct the results to be placed in a particular frame. A common example would be a three-framed page. The frame menu is on the left, a masthead or header bar is on the top frame, and the content frame is on the right, which takes up most of the real estate of the entire page. As items in the menu are selected, the masthead and menu remain the same, but the content area is refreshed with the latest request. Using frames in your HTML pages has a number of distinct advantages but one unfortunate disadvantage. The advantages of frames are simple – you can target to refresh different areas of the page so you do not have to waste transmission time for static page information. Also, it can be easier from an HTML programmer perspective to break the page up in this manner rather than managing tables. The major disadvantage is that visitors to that page can only bookmark the frameset, not each individual content page that is delivered.

**HTML Template Using Server-side Includes.** Because of the bookmark limitation with its framed site, IBM set new standards for HTML and devised a template in which all pages on the site were static HTML templates. Specific content, menus, and program logic all had to be requested via HTML #include statements. IBM underwent a major redevelopment effort to accommodate its new standards and templates, which solved the bookmark issue. All pages on the site had a unique bookmark and a nice clean URL (e.g., domain/directory/filename.html). Unfortunately, the new standards posed some serious restrictions on the site's intelligent programs. Because program calls were only initiated via include statements from static HTML, any parameters that were passed had to be static as well. The programs had to rely on cookies to pass information between calls.

## Design Considerations

In this article, I've reviewed some new techno-logical concepts that may affect design decisions during an e-business project. Clearly, there may also be some new products, languages, or tools to learn. However, an e-business system development project has a lot in common with any other business application development project you have managed. Some of the key elements that you should cover in your project planning include the following:

- Site objectives
- Audience
- Application requirements
- Architectural requirements

- Integration with existing systems
- Site standards
- Tools and technologies to be used
- Timelines and important milestone dates
- Effective staffing of project team

Typical applications that organizations are deploying to the Web include order-entry and order-status checking; specialized search engines over objects, data, and pages; discussion forums and chat rooms; ad banner engines; and specialized partnerships or broker-style sites. Given the wide variety of functionality, there are various tools and approaches for AS/400 Web development, and not all are suitable in all cases. An understanding of the technologies involved should assist you in making good decisions about building your own e-business site. [TUG](#)

*Eden Remme Watt is the services director for LANSA, Inc. She is responsible for a large staff of consultants who develop AS/400-based native, client/server, and e-business systems for clients. Eden has been involved in the design and man-agement of some significant e-business systems, including projects for Shell, IBM, and COM-MON. You can reach her by email at eden.watt@lansa.com.*

The graphic features the Radius Information Systems Ltd. logo at the top left, with contact information at the top right: 295 The West Mall, Toronto, On M9C 4Z4, [T]416.626.9556, [F]416.626.9565. A central blue area lists services: E-Business Solutions, ERP Integration (BPCS Specialists), Financial Brokerage Solutions, EDI Solutions, Business Intelligence Integration, and AS/400 Technology Specialists. A green curved banner on the left reads 'Our Services'. The entire graphic is framed in green.

See our booth (#20) at the TEC Showcase, April 24