8 What to do when your Jaguar is caught in the Web

This chapter covers

- Internet and web primer
- How to build web applications using Jaguar CTS
- An overview of PowerDynamo

8.1 What is the World Wide Web?

The Internet and the World Wide Web (or Web) have engulfed the information technology industry. In fact, it has started to dominate every aspect of our lives. We can buy, sell, and research just about anything on the Internet. Dot-com companies dominate commercials and most companies today at least advertise having a web site. Internet companies drive the stock market and boast gaudy P/E ratios and market caps while telecom companies and cable companies are trying to make Internet access faster and easier for everyone. In the software industry, the Internet has made getting trial versions of products easier to obtain, and most products and drivers have regular maintenance patches and upgrades available on the vendor's web site.

It is important to point out that the Internet and the Web are not the same, although the terms are becoming nearly synonymous in today's vocabulary. The popularity of the Internet has been spurred on by the ease of use provided by web technologies, particularly the web browser, which adds a graphical interface to the Internet environment.

The Internet is a global network of interconnected networks. This network provides access to servers, information, and services around the world. The Internet is based on the TCP/IP network communication protocol. All machines and servers need to have TCP/IP installed and configured in order to communicate on the Internet. Servers on the Internet can provide several different services to clients, some common ones include FTP, SMTP, Telnet, Gopher, and HTTP.

The World Wide Web, also known as the web, is not the Internet. The Web is a distributed computing architecture and is just one of many services (HTTP) that servers on the Internet provide. While the Web is often deployed on the Internet, it can also be implemented on a completely independent corporate network, known as an intranet.

DEFINITION INTRANET A network that is based on web technologies and is confined to an organization is called an intranet. An intranet is typically deployed over a corporate network and often provides access to the Internet. Internet access, however, is not a requirement. Because an intranet architecture is in a more controlled environment and is used to perform daily business tasks, the browser typically relies on plug-ins, Java applets, and database connectivity to provide a richer client-side application.

The web architecture relies on the client/server model of communication where a client application requests information and services from a server application. However,

the Web is based on a thin client that can access information anywhere using a web browser application, unlike the larger, more specialized GUI client applications that interact directly with a database and are typically what is meant by the term "client/server" in today's IT terminology.

8.2 How does the Web work?

The web architecture is defined by the technologies it needs, including the TCP/IP and HTTP communication protocols, web browsers, and a web server. In the web architecture the web browser requests an HTML page from a web server. The web server (or HTTP server) is designed to listen for a client's HTTP requests and to provide the documents they ask for. The web architecture is illustrated in figure 8.1.

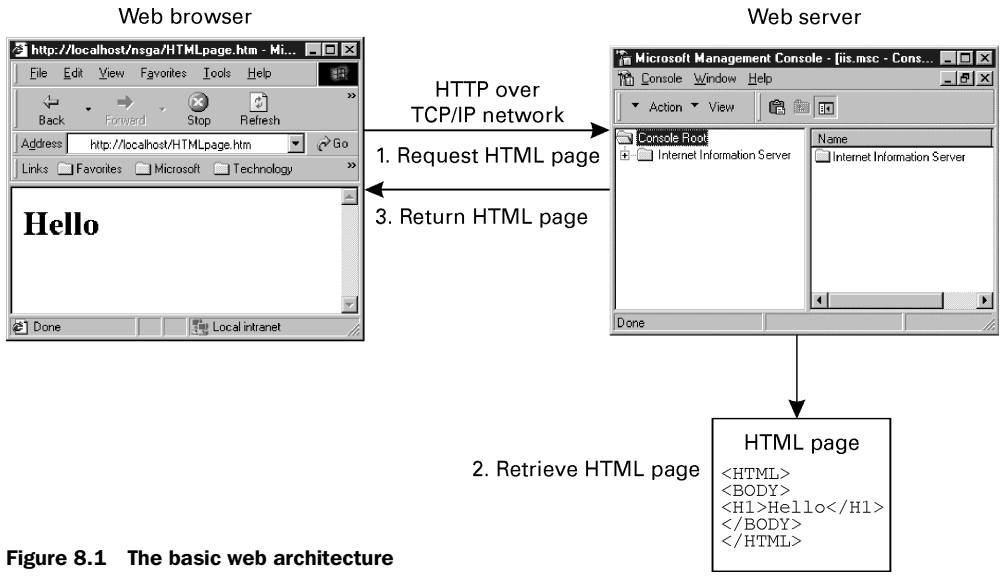

Figure 8.1 The basic web architecture

The web browser is a client application that can communicate with web servers over HTTP and request, receive, interpret, and display information from a web server. The capabilities of the Web have expanded greatly beyond the core web service of requesting and receiving HTML documents. Web browsers have been enhanced to handle more files types through either built-in capabilities or by installing plug-ins. Some examples of file types that most browsers can handle on their own are the JPG and GIF image formats while Adobe's Portable Document Format (PDF) files are widely supported through a plug-in, the Adobe Acrobat Reader. In addition to requesting and displaying files, the browser is now capable of executing

code on the client-side by interpreting scripts embedded in the HTML documents that are usually written in JavaScript. The leading browsers also have a built-in Java virtual machine making them capable of running Java programs.

Before discussing the details of building a web application with Jaguar CTS, it is important to have an understanding of some of the underlying technologies.

8.3 What is HTML?

Most information on the Web is stored in documents using HTML (HyperText Markup Language). These HTML documents are platform independent hypertext files that contain textual information marked with tags. These tags are used to tell the browser to do something special besides display text. They can be used to define the way the data is presented. They can also embed information and additional resources within them. These additional resources include images, files, scripts, controls, and programs. They can also provide links to other documents that can be on the same web server or on any other accessible web server.Here is a sample of HTML:

```
<html>
<body>
<h1>Hello</H1>
</body>
</html>
```

The browser translates this so that the user sees the display as shown in the browser in figure 8.1.

The HTML is based on Standard Generalized Markup Language (SGML), which conforms to International Standard ISO 8879. The World Wide Web Consortium manages the HTML language. For more information about HTML, visit the World Wide Web Consortium site at the following address:

http://www.w3.org/MarkUp/

8.4 What is a URL?

When a browser makes a request on the Web, it does so using the Uniform Resource Locator (URL) standard. The URL identifies the communication protocol, the file or resource that is being requested, and where the file or resource is located, including the server and the path. The syntax for the URL is listed below:

protocol://host {:port}/path/file_name

The protocol describes the service that is to be used to access the resource (i.e. HTTP, File, FTP). The next part of the request tells us the IP address of the server

and on what port the application we are interested in is listening on. A web server usually listens on port 80, which is the default for the HTTP service, so it is usually not required. The server location is usually expressed using a domain name rather than the IP address, but either can be used. The final part of the request tells us the path and name of the resource that is being requested. An example is:

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

In the example above we are requesting an HTML page named index.htm from the root directory on the server that hosts the www.aegisconsulting.com domain. For more information about URLs, see RFC 2396 and the World Wide Web Consortium site, at the following address:

http://www.w3.org/Addressing/

8.5 What is HTTP?

The HyperText Transfer Protocol (HTTP) is an application-level communication protocol that runs over TCP/IP. HTTP allows multimedia content to be exchanged across a network to diverse platforms. HTTP is a stateless, object-oriented protocol that uses the client/server model of issuing requests and returning a response. For more information about HTTP, see RFC 2616 and the World Wide Web Consortium site, at the following address:

http://www.w3.org/Protocols/

An HTTP transaction or session is made up of four basic steps.

1. Connect to the web server
2. Request a file/resource
3. Receive a file/resource or an error
4. Disconnect from the web server

The stateless character of HTTP arises from the fact that the client is disconnected from the server after a request has been resolved. Let's review this in the context of retrieving an HTML page with an image from a web server. When the client browser requests an HTML page, it connects to the web server and issues the request. The web server handles the request, locates the document, and sends the requested HTML page to the client application and then disconnects. The browser translates the HTML and presents the information to the user. During the translation, the browser notes that an image is required based on the `<IMG>` tags. When an HTML page requires additional resources, in this case an image, it must issue another request to the web server. The additional resource could also be any other

content type, for example a Java applet. To finish loading the HTML page, the browser must reconnect to the web server and request the image. After the image is received the browser is disconnected. This process is repeated for all the images and resources contained by the HTML document as illustrated in figure 8.2. This makes the web architecture inherently slow. A newer version of the HTTP protocol, HTTP 1.1, allows several requests to be made in the same connection before disconnecting, helping to speed up the retrieval of HTML documents. However, it is still not connection-based because the browser is disconnected from the web server after the page and all the files it requires are downloaded.

The steps in an HTTP 1.1 transaction that downloaded an HTML page and an image are shown in figure 8.2 and listed below:

1. Connect to the web server
2. Request an HTML page
3. Receive the HTML page
4. Request the image file
5. Receive the image file
6. Disconnect from the web server

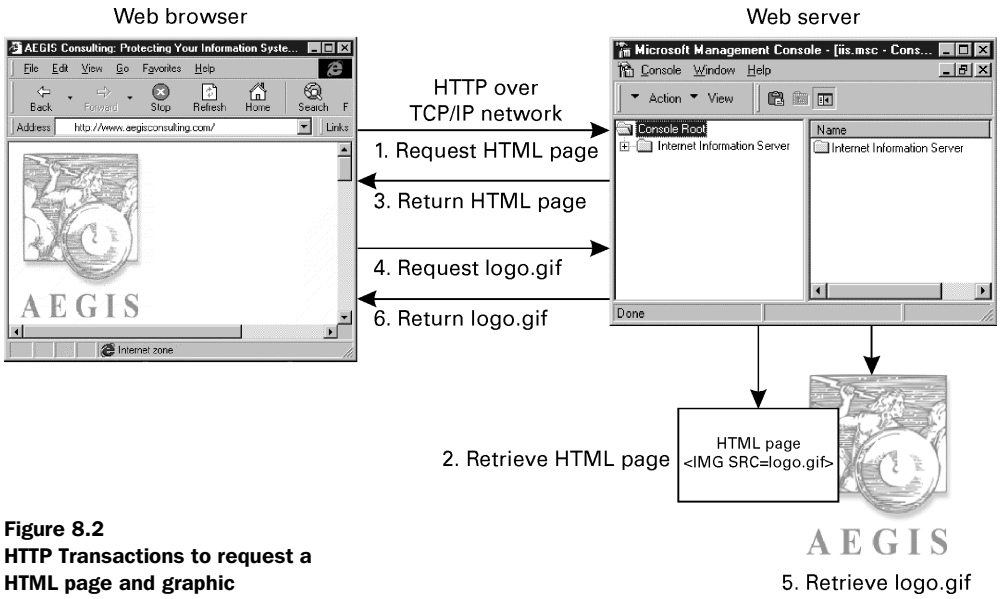

Figure 8.2
HTTP Transactions to request a HTML page and graphic

8.6 What is TCP/IP?

TCP/IP (Transmission Control Protocol/Internet Protocol) provides a set of network communication protocols that performs the tasks of both the transport and network layer of the Open Systems Interconnection (OSI) Reference Model. The OSI Reference Model, developed by the ISO (International Organization for Standardization), is used to divide the services provided by network protocols into seven layers. TCP/IP is one of the most popular networking stacks and is the standard protocol on the Internet. This is due to the wide support of hardware platforms including Windows, UNIX, DOS, and NetWare, and also the broad support of network protocols and physical architectures including Ethernet, Token Ring, Frame Relay, and ATM.

8.7 What is IP?

The Internet Protocol (IP) provides network-to-network communications, which is known as routing. To facilitate routing, IP is responsible for providing an internetwork addressing scheme and routing data to the correct network and machine. IP does not rely on the physical addresses of each node. Instead, it uses a logical addressing scheme that allows the physical network to change without having to propagate these changes to the upper layers of the network stack. The logical address is translated into a physical address using the Address Resolution Protocol (ARP).

The IP address is a 32-bit number that is often represented in dotted decimal notation. A typical IP address in the dotted notation would be represented as follows:

161.125.23.116

Each portion of the IP address in the dotted notation represents the decimal value of a byte (8 bits) of information. The decimal number 161 is represented as 10100001 in binary. The entire IP address 161.125.23.116 can be represented by the following binary number, 10100001 01111101 00010111 01110100. These octets, as each portion of the IP address are sometimes known, can range in value from 0 to 255.

Identifying a machine (or node) on an internetwork requires two pieces of information–the network the machine is attached to (the network ID) and the address of the machine (the host ID) on that network. The IP address actually represents both of these. The first part of the address is called the network ID or net ID, and it defines a unique network address. The second part is the host ID, which defines a unique machine on the network identified by the net ID.

So which part of the IP address is the net ID and which part is the host ID? That depends on the IP address in question. IP addresses are divided into classes and

depending on the class an address belongs to, a different number of bits is allocated for each ID. The classification of IP addresses allows network administrators to design a network infrastructure that can have lots of networks (each with few hosts), or have a few networks (each with several hosts). There are five classifications of IP addresses, of which only the first three (A, B, and C) are put to use on the Internet. The remaining two are reserved for special functions. The IP address classifications are listed in table 8.1. Only machines on the same network can communicate without a router. Two machines are on the same network when they have the same net ID. If the net IDs are different, the two machines will not be able to communicate even if they are connected through the same hub.

Table 8.1 IP address classifications

Classification	First octet range	No. octets for Net ID	No. octets for host ID	Subnet mask
A	1–126	1	3	255.0.0.0
Reserved	127	-	-	-
B	128–191	2	2	255.255.0.0
C	192–223	3	1	255.255.255.0
D (Special)	224–239	-	-	-
E (Special)	240–255	-	-	-

8.8 What is TCP?

The Transmission Control Protocol (TCP) is a connection-based communication protocol that provides reliable two-way communication between hosts. It establishes a connection and makes sure that outgoing messages are divided into packets capable of being sent on the physical network. It also reassembles incoming packets in the correct sequence—error free to the layers above.

When IP transmits data, it is concerned with getting it from network A to network B and from machine A to machine B. The TCP protocol takes this process to the next step, making sure that the data gets to the correct application once it arrives at the correct machine. It does this using a port to identify the application the message is intended for. Each application on a server must have a unique 16-bit port number to identify itself. There are several well-known ports, such as those for FTP (21), HTTP (80), and Telnet (23). Your applications should stay away from these port numbers. A single machine can have several server applications listening on different ports. For example, a Windows NT server may be hosting IIOP for your Jaguar server on port 9000 and also be listening for HTTP (80), SMTP (25), FTP (21), and Telnet (23) clients. IP makes sure that the correct machine gets the

data you are sending, and TCP makes sure that the message intended for the Jaguar server does not get sent to the SMTP service as shown in figure 8.3.

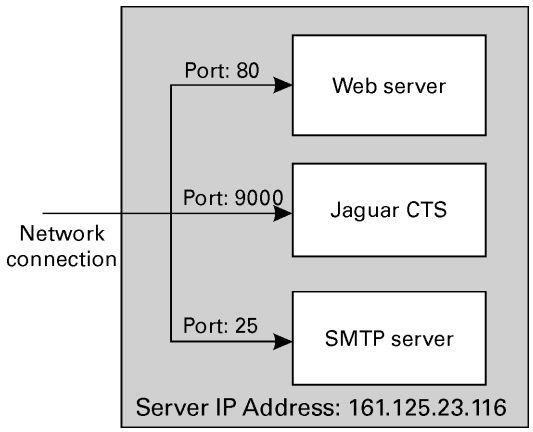

Figure 8.3 TCP/IP on a machine with several server applications

When a connection is established between two applications on a TCP/IP network it is called a socket. A socket is a unique connection formed by combining an IP address and a port number. There are numerous Request for Comments (RFC) documents that are available for more information on TCP (RFC 793), IP (RFC 791), and other Internet technologies. Check out the RFC search site at:

http://www.cis.ohio-state.edu/hypertext/information/rfc.html

DEFINITION UDP TCP/IP actually has two different transport protocols, TCP and UDP. The User Datagram Protocol (UDP) is a connectionless-based communication protocol that runs over IP. It is faster than TCP because it does not provide reliable communication between hosts. UDP does not guarantee that the data will get to its destination or will be error-free.

8.9 What is a dynamic web site?

Up to now we have looked at a basic web site, where a browser requested an HTML file from a web server. The web server receives the HTTP request for a document, retrieves the document from the local file system, and returns the document to the browser. The documents that the web server can retrieve and return at this point are static HTML pages. These HTML pages are fixed documents in which the content does not change without a developer editing the file and redeploying the document. Static web sites do not display real-time data or allow user interaction with the site.

As compelling as static web sites are, the real power is in providing a dynamic site that allows a user to interact with the site and access data in a database. A dynamic web site allows a web server to take a URL request from the browser (or another web server) and return a dynamic HTML page that is generated at runtime instead of returning a static document from the file system. A dynamic web page is

generated each time it is accessed. Because dynamic pages are generated at run time, they can respond to input from the web browser, return data requested by the user, and access information real-time. A dynamic HTML page typically will run scripts that execute on the server. These scripts can access a component on an application server or data in a database.

In general web servers do not process logic or access databases. That is not part of the functionality that is required to receive and process HTTP requests for documents. However, as the Internet and Web expand, web servers are providing more features including CGI, API extensions (ISAPI, NSAPI), data encryption (SSL), and server authentication (digital certificates) that enable developers to build dynamic web sites.

In order for a web server to return a dynamic HTML page, an interface is required between the web server and the program or application server that executes the script. The most common web server interface is the Common Gateway Interface (CGI), which is also the industry standard. In addition to CGI, some web server vendors provide proprietary interfaces to their servers, the two most common are NSAPI (Netscape Application Programming Interface) and ISAPI (Internet Server Application Programming Interface). ISAPI is the API supported by Microsoft's web server—Internet Information Server (IIS). In addition to ISAPI, Microsoft IIS also has its own built-in script execution engine called Active Server Pages (ASP).

8.10 What are CGI, NSAPI, and ISAPI?

The Common Gateway Interface (CGI) was designed to allow web servers to access external programs to produce dynamic content. These external programs can be any type of executable file including DOS batch files, UNIX shell scripts, and more commonly, Perl programs. The external program can also be a dynamic page server like PowerDynamo. CGI handles all the nuances of communicating between the web server and other applications, defining how data is passed to and from the web server and the external application, as well as describing how to invoke functions. The output that is returned by the CGI program to the web server is typically an HTML document, but it may also be any content or information that a web server can handle, including an image or a reference to another document. A CGI program can do just about anything, but usually creates a dynamic HTML document based on some input from the client and accesses a data source.

CGI creates a new process for each request, making it slow and inefficient. It also uses environment variables and standard input/output to gather and return information. Most web server vendors provide a proprietary interface, which allows the web server to execute programs in the same process space as itself, which makes it

much faster and more efficient. The two most common proprietary interfaces are NSAPI and ISAPI.

A CGI, NSAPI, or ISAPI program is invoked using a URL, similar to requesting a static HTML document. Instead of specifying an HTML document, a CGI program is specified. Most CGI programs are stored in a special directory called `cgi-bin` as shown below:

http://www.MySite.com/cgi-bin/cgi_program.exe

When the web server receives a URL that requests a CGI resource, the server calls upon the external application and passes off the request through the CGI interface. The results of the program are returned as an HTML document that is sent to the web server. The web server passes this document back to the browser as pictured in figure 8.4.

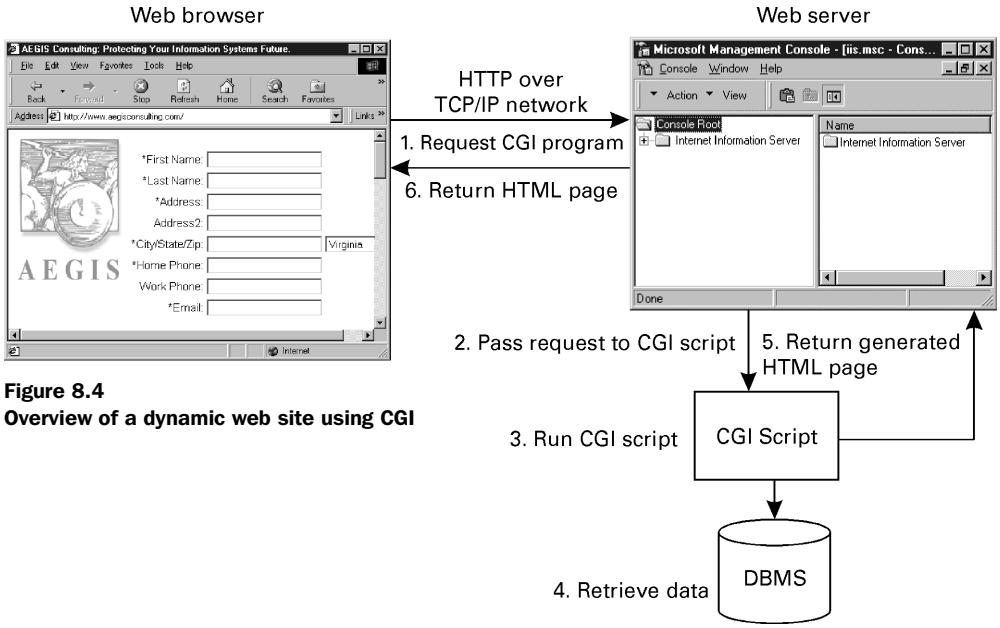

Figure 8.4
Overview of a dynamic web site using CGI

NOTE Java servlets are Sun's answer to CGI, ISAPI, and NSAPI. Servlets allow a web server to call special server-side Java programs to perform similar tasks. The PowerDynamo server does not have a servlet container and cannot execute servlets; however, it could redirect requests to a Java servlet.

8.11 How do I access Jaguar components from the Web?

The Jaguar CTS server is a CORBA-compliant application server designed to execute components in a high speed, multithreaded kernel. Each Jaguar component is accessible to clients using the CORBA Internet Inter-ORB Protocol (IIOP), which is designed to allow any application servers that adhere to the CORBA 2.x specifications to communicate with each other and access components.

Now that you have built reusable business and data access components for the Jaguar CTS server, how can they be accessed over the Web? There are three different approaches to designing a web architecture that can be implemented with Jaguar CTS.

- Jaguar-Java applet (medium client)
- Jaguar-PowerDynamo (thin client)
- Jaguar-Java servlet (thin client)

The first is the Jaguar-Java applet approach, which allows a web browser to make an HTTP request to a web server for a web page or file. The web pages that are returned by the web server have Java applets embedded in the page, which are also downloaded to the browser over the HTTP protocol. After being downloaded, the Java applet is the primary client used to access the Jaguar server and invoke component methods over the IIOP protocol. This approach is illustrated in figure 8.5.

TIP The Jaguar-Java applet approach can be implemented without a separate web server because Jaguar CTS supports HTTP requests and can serve static web pages and files (applets) to a web browser

REPORT CARD
Jaguar—Java applet

Pros	■ More robust GUI is possible with Java applets. ■ This method does not require an additional dynamic page server. ■ There is a direct persistent connection to Jaguar using IIOP.
Cons	■ Java applets require a Java-enabled browser and correct version of the JVM. ■ Java applets take longer to download then HTML pages. ■ Sandbox and firewall issues may make this unacceptable for Internet solutions.

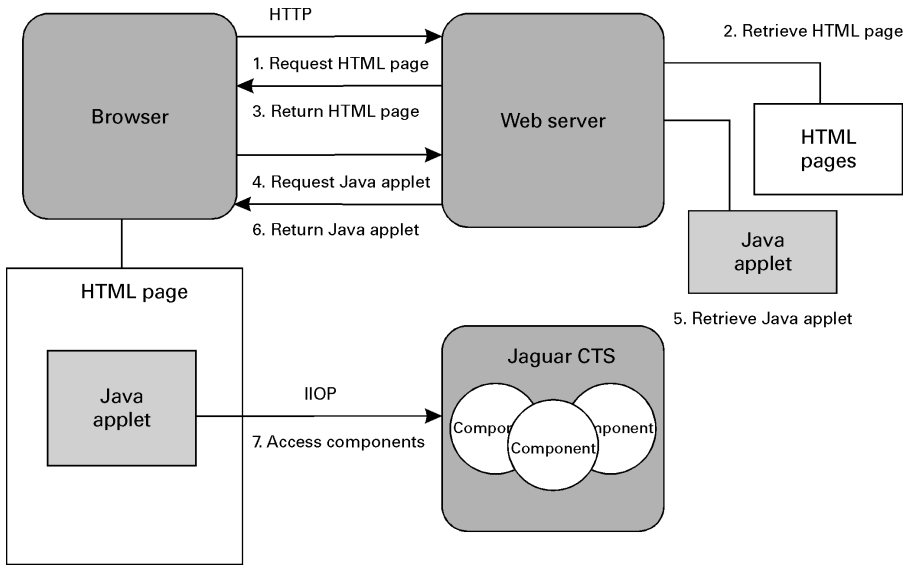

Figure 8.5 The Jaguar-Java applet architecture

An ultra-thin client application can be built by combining Jaguar CTS with PowerDynamo. PowerDynamo is a dynamic page server. PowerDynamo is based on templates that are files made up of a combination of HTML, PowerDynamo tags, and DynaScript. The DynaScript is executed by the PowerDynamo server and the results are used to build a dynamic web page at runtime. The script in the template can call Jaguar components, accessing them as a Java-CORBA client. The results from running the components can be used in the dynamic HTML page that is generated and returned to the client. This approach is illustrated in figure 8.6.

TIP In addition to PowerDynamo, Jaguar CTS can be accessed by any web server or dynamic page server that can access COM or CORBA objects, including Microsoft ASP and Allaire's ColdFusion.

REPORT CARD
Jaguar—PowerDynamo

Pros	■ There are minimal requirements on client machines (thin client). ■ HTML pages are typically small and download quickly.
Cons	■ HTTP requests are stateless and require state management by the dynamic page server or Jaguar CTS. ■ PowerDynamo DynaScript is an interpreted language.

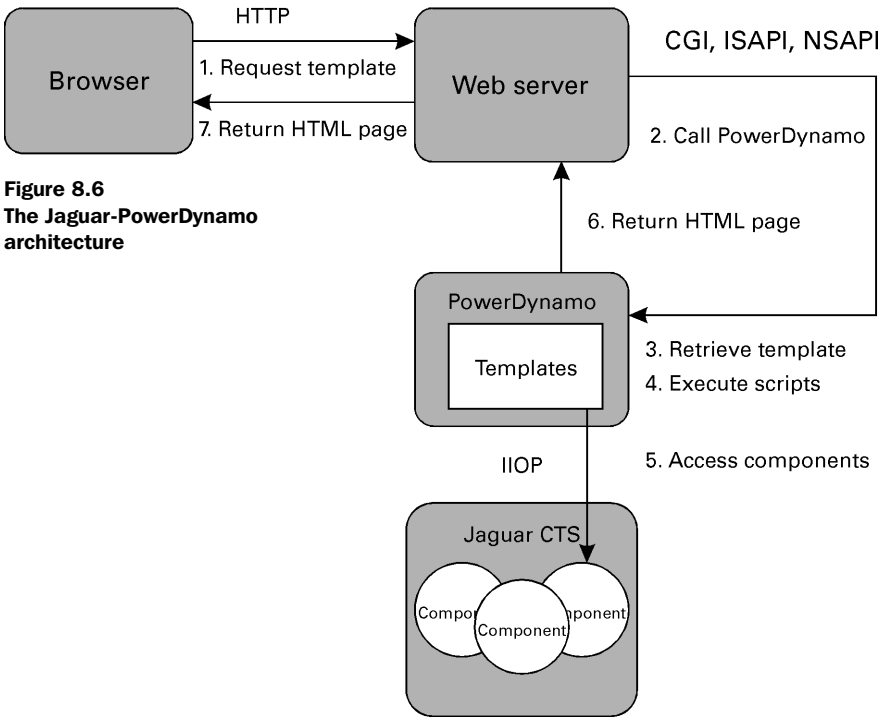

Figure 8.6
The Jaguar-PowerDynamo architecture

A third approach is to have the browser access Java servlets that are deployed on the Jaguar CTS server. Java servlets are not Jaguar components; they are a special Java program that is designed to be called by an HTTP request. The Java servlet is similar to the PowerDynamo template in that it can access Jaguar components and use the results to build a web page or file. However, the Java servlet is 100 percent Java, so it may be easier to work with than DynaScript if the developers already know Java. In addition, the Java servlet is compiled into bytecodes, which are faster than the interpreted Dynamo templates. One of the major downsides to this approach is that the Jaguar CTS application server is now acting as a web server. The Jaguar server does not have all the capabilities that a web server like Microsoft IIS or Netscape Enterprise Server have. This Jaguar-servlet architecture is illustrated in figure 8.7.

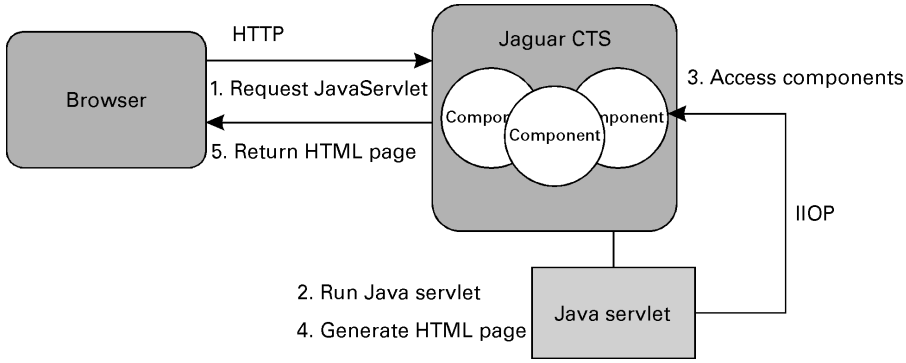

Figure 8.7 The Jaguar-Java servlet architecture

REPORT CARD
Jaguar—Servlet

Pros	■ There are few if any requirements on client machines (thin client). ■ This configuration allows Java classes to be called over HTTP. ■ Compiled Java is faster than interpreted pages. ■ HTML pages are typically small and download quickly. ■ Server-side coding is all done in Java.
Cons	■ HTTP requests are stateless and require state management by the servlet. ■ This method requires using Jaguar CTS as a web server to handle HTTP requests. ■ The Jaguar-servlet method requires HTML pages to be built in Java classes by concatenating strings of HTML and data. Note this is addressed with JSP.

It is important to point out that these three approaches are not mutually exclusive and can be combined to build a dynamic web application. In addition, there are other techniques that can be used to access a Jaguar CTS server over the Web; the ones presented here are the most straightforward and rely on industry standard practices and de facto standard technologies. The PowerDynamo approach is covered in this chapter and in chapter 9. The Java applet, servlet, and Sun's Java Server Pages (JSP) approaches are covered in chapter 10.

8.12 What is PowerDynamo?

Jaguar CTS is a component-centric application server, designed to host and run components. PowerDynamo is a page-centric application server, often referred to as a dynamic page server. It is designed to run HTML templates and scripts to generate dynamic HTML pages. It is similar to Microsoft's ASP, Sun's JSP, and Allaire's ColdFusion.

DEFINITION EAServer Enterprise Application Server (EAServer) is the name Sybase uses to refer to both Jaguar CTS and PowerDynamo

PowerDynamo generates dynamic HTML pages using a template. The template is a combination of HTML tags, Dynamo tags, and DynaScript. The Dynamo tags and DynaScript in the template are interpreted and run by the Dynamo server. The scripts can accept user input to determine which business logic and queries should be executed. The output of these scripts determines what the generated HTML page looks like to the person using the web browser. The process is illustrated in figure 8.8.

The steps below explain how PowerDynamo works:

1. The web server receives a URL request sent by a web browser.
2. The web server passes the request to PowerDynamo.
3. PowerDynamo retrieves the template from the database or file system.
4. PowerDynamo executes the scripts in the template to generate the HTML page.
5. PowerDynamo scripts may access a DBMS or application server for additional processing and data.
6. The generated HTML page is sent to the web server.
7. The web server returns the HTML page to the web browser.

PowerDynamo is a dynamic page server. It is not a web server and cannot receive HTTP requests directly. A web application built with PowerDynamo *requires* a separate web server. Web servers use extensions or gateways to interface with different programs and application servers. PowerDynamo supports CGI, ISAPI, and NSAPI and it is through these different interfaces that a web server can be configured to use PowerDynamo.

With PowerDynamo three different types of web applications can be built:

- Static content
- Dynamic content
- Dynamic content with components

PowerDynamo applications are built from various web resources including templates, scripts, HTML and XML pages, and images. These web resources can be deployed to a database or in a file system. Storing the web resources in a database can be a big advantage allowing the web site to be shared, backed up, replicated,

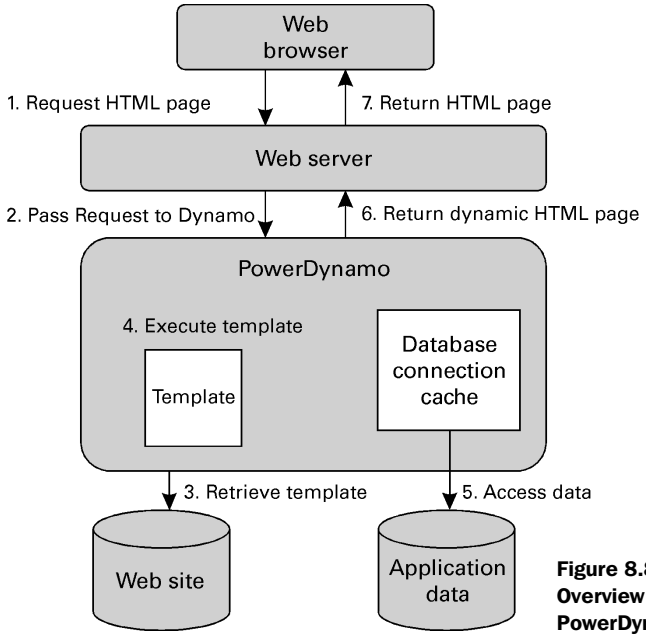

Figure 8.8 Overview of the PowerDynamo architecture

and secured easily with database technology. The database used to store the PowerDynamo web site resources does not have to be located on the same database or even use the same DBMS as the application data.

PowerDynamo is a high performance, extendable, page-based application server. It provides session management, access to application data on any DBMS that supports ODBC or Open Client (Sybase), and can cache dynamic pages and database connections. PowerDynamo is capable of supporting robust dynamic content web sites without another application server; however, it is not designed to run components, Java servlets, or provide access to non-Web clients.

PowerDynamo provides functionality that allows scripts to access both ActiveX (COM) and Java (Java, CORBA) objects. This lets the PowerDynamo scripts access Jaguar components. When PowerDynamo is teamed with Jaguar CTS, it enables all the power and advantages of the Jaguar server and the reusability of components to be moved to the Web. By placing logic and data access into components on Jaguar, a reusable set of business rules can be extended to any client or architecture. In addition, Jaguar has several features that PowerDynamo does not, including built-in load balancing and failover, transaction management, and support for several component languages and models.

The power of a PowerDynamo application is in the templates and scripts that are used to build a web site. The templates and scripts are written using HTML tags,

PowerDynamo tags, and PowerDynamo's own scripting language, DynaScript. The template scripts contain the business logic, database access logic, and HTML to generate dynamic HTML pages. The templates are run in the PowerDynamo interpreter on the server before it is sent to the client. This eliminates the need to worry about client-script compatibility issues. In addition, the client receives only the generated HTML, not the server-side scripts. Thus, application logic is protected.

REPORT CARD
PowerDynamo

Pros	■ PowerDynamo supports session management. ■ There is a database deployment option to manage web resources. ■ PowerDynamo is easily managed through Sybase Central. ■ PowerDynamo supports CGI, Win-CGI, ISAPI, NSAPI. ■ Database connection caching is supported. ■ An interface to ActiveX, Java, and Jaguar objects is provided.
Cons	■ Business logic is limited to web applications. ■ Database connectivity is limited to ODBC and Open Client.

8.13 How do I set up a web site in PowerDynamo?

When setting up a web site in PowerDynamo, there are two deployment options, deploying the web site resources into a database or deploying them as files. This is a deployment issue and it does not affect the development of the web site itself. One of the benefits of using a database is that a web site can easily be shared, backed up, or replicated to other servers.

Although PowerDynamo can access any database that supports ODBC for application data, PowerDynamo supports only Adaptive Server Anywhere (ASA) and Adaptive Server Enterprise (ASE) as the deployment database. In order to create a web site in PowerDynamo that uses an ASA database to store its web site resources, we need to follow these simple steps:

1. Create an ASA database to store the web resources.
2. Create an ODBC profile for the ASA database.
3. Create a connection profile for the web site in Sybase Central.
4. Connect to the web site using the connection profile to create system files/tables.
5. Create a mapping for the web site.

When using ASE, an ODBC profile is not required. Instead configure an Open Client profile, but otherwise the steps are the same. In order to create a web site in PowerDynamo that uses a file system to store resources, we need to follow these simple steps:

1. Create a directory to store the web site resources.
2. Create a connection profile for the web site in Sybase Central.
3. Connect to the web site using the connection profile to create system files.
4. Create a mapping for the web site.

8.13.1 Creating a connection profile

In order to create a connection profile for the PowerDynamo web site, open Sybase Central, which is listed as "Manage PowerDynamo" under the Windows Start Menu. Select the Tools | Connection Profiles… menu option as shown in figure 8.9. From the Connection Profile dialog, click on the New button to create the new profile. The next dialog will open and prompt you for the profile name and type. The profile name is used only in the Sybase Central tool and does not impact the PowerDynamo web site. Make sure that you select PowerDynamo from the list of options listed for the Type field.

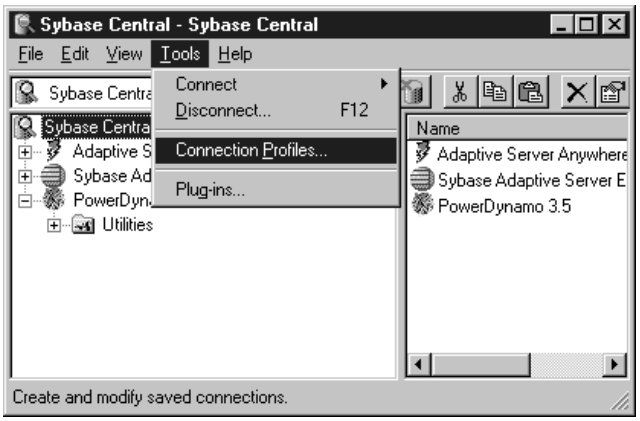

Figure 8.9 Creating a new connection profile

Once the new connection profile is created, the properties need to be configured. The Properties dialog for a connection profile is pictured in figure 8.10. The connection type determines the deployment option for the PowerDynamo site. The choices are ODBC used for the ASA database option, Open Client used for the ASE option, and Dynamic file, which is used for the file option. The ODBC option will require the name of the ODBC profile and the database user ID and password. The

ODBC data source is the name of the ODBC profile created to connect to the ASA database that will store the PowerDynamo web site resources. The default user ID and password for an ASA database are *dba* and *sql* respectively. Once the connection profile is set up, it will appear in the list of available connection profiles as illustrated in figure 8.11.

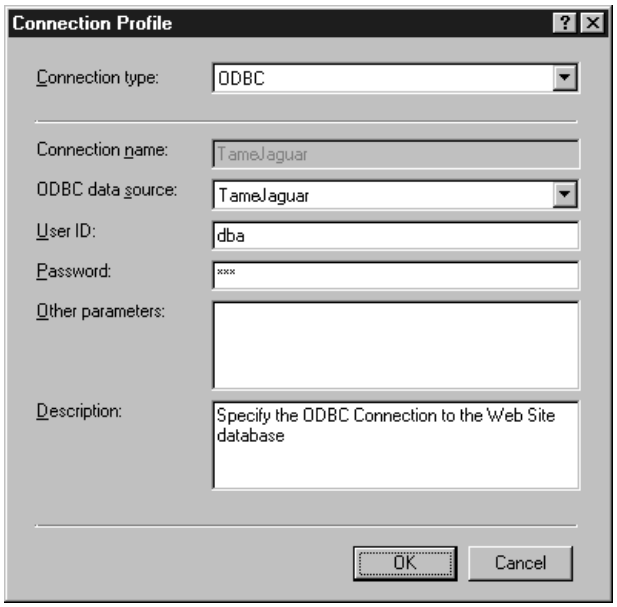

Figure 8.10
PowerDynamo connection profile properties

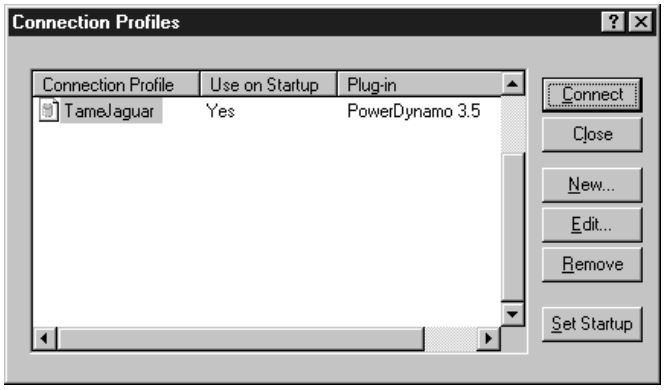

Figure 8.11
PowerDynamo connection profiles

Now click on the Connect button to connect Sybase Central to the PowerDynamo web site. Because this is the first time that the web site is being accessed, there will not be a web site stored in the database. Sybase Central will prompt you

to create the web site. Click the Yes button to create the web site tables and import the PowerDynamo system files. As the web site is created, Sybase Central will prompt you for the name of the root folder. This is the main folder for the web site from which all resources and subdirectories that will be available as part of the web site will be placed. The default name for the folder is Site. Once the web site is created and all the system files are imported, the PowerDynamo site setup is nearly complete and it can be managed through the Sybase Central application. Note that both the database and file deployment options manage the web site using a root folder and subdirectories that are under the root folder.

Figure 8.12 illustrates the PowerDynamo web site in the Sybase Central application. Notice the Site folder and the system and utils subdirectories. The system subdirectory contains all the system files and scripts that PowerDynamo needs to run the site. In general the system files and scripts should not be changed; however, there will be some exceptions. All application-specific files for the web site should be placed in the Site folder or in a separate subdirectory under the Site folder. Application-specific files should not be placed in the systems or the utils folder.

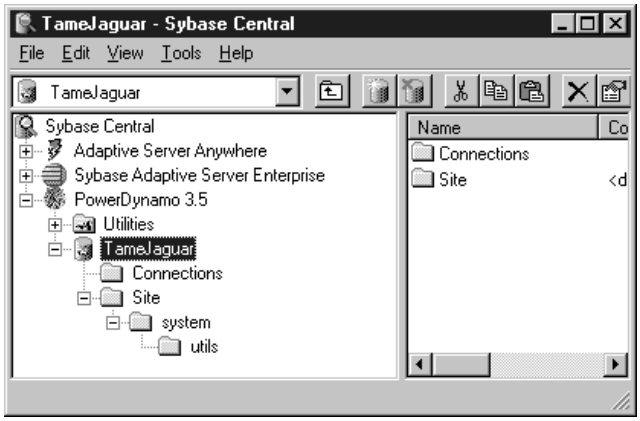

Figure 8.12
The PowerDynamo web site

8.13.2 Adding a mapping

Before the PowerDynamo web site can be accessed from a web server, it needs to have a mapping. The URL prefix specified in the mapping is used in the URL sent from the web browser or another web server to tell PowerDynamo which web site the request is meant for. The URL was explained in the sections 8.4 and 8.10.

When the web server receives an HTTP request, the server examines the URL to determine how to handle the request. The URL lets the web server know that the request should be passed on to PowerDynamo through the URL prefix, which is specified in the mapping. The PowerDynamo mapping name is placed after the host

and port portion of the URL and before the path and file name portion of the URL as shown below:

protocol://host {:port}/PowerDynamo_Mapping/path/file_name

To add a mapping to a web site, click on the Configuration folder located under the Utilities folder for PowerDynamo. The available configuration utilities for PowerDynamo are listed as treeview items as illustrated in figure 8.13. Choose the Mappings folder to display all the mappings that are already set up for other PowerDynamo web sites. Now, click on the Add Mapping utility. The dialog shown in figure 8.14 will open.

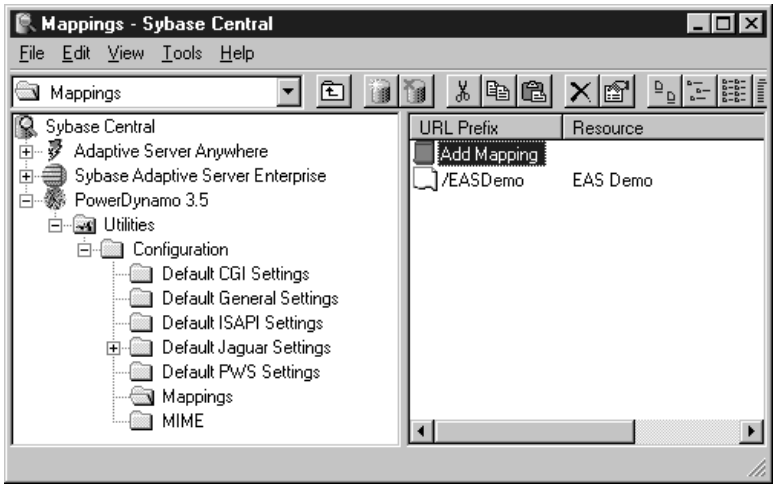

Figure 8.13 Creating a PowerDynamo mapping

In the URL prefix field, enter the name that will be used in the URL to access the PowerDynamo web site. The name must be prefixed with a backslash (/) as shown in figure 8.14. Once the mapping is created, a PowerDynamo web site that is running locally through the Personal Web Server with an HTML page named index.htm can be accessed from a web browser with the following URL:

http://localhost/safari/index.htm

The site type determines how the documents in the web site that the URL prefix points to are stored. The options are:

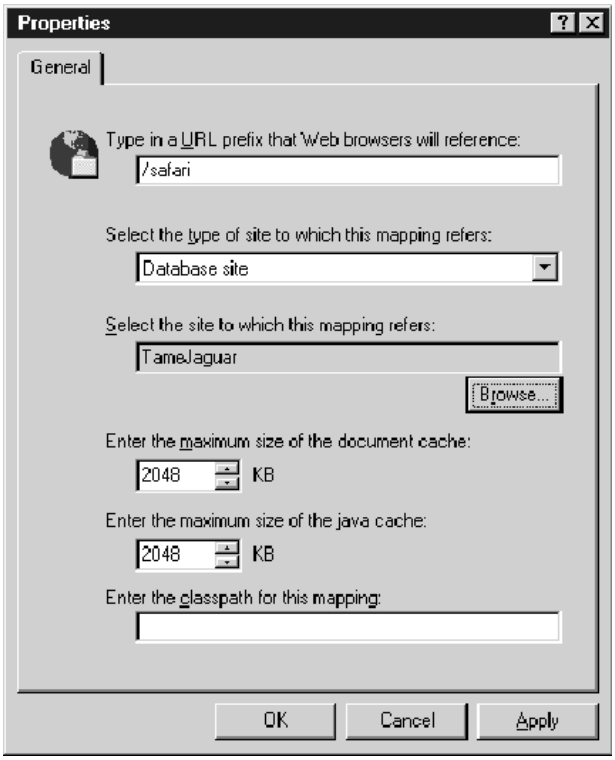

Figure 8.14
The PowerDynamo mapping properties

- Database site: The documents are stored in a database and templates/scripts are executed.
- Dynamic file site: The documents are stored on disk and templates/scripts are executed.
- Static file site: The documents are stored on disk and templates/scripts are not executed.

The site location is where the name of the connection profile for the web site that is deployed to a database is specified. For the web site that is deployed using files, the site location is the path to the root directory of the site. The site location field is grayed out indicating that it is not editable; however, the field is editable and is required for the mapping to work.

The document cache size and the Java cache size settings indicate the maximum amount of memory that PowerDynamo should reserve for the document and Java caches respectively. The Java classpath is used to specify the class path for class files that are stored in the database.

TIP In order for a new mapping to be recognized the web server/PowerDynamo must be stopped and restarted.

8.14 How do I configure a web server to use PowerDynamo?

PowerDynamo is a dynamic page server, it is not a web server. Once the PowerDynamo web site is set up, a web server must be configured to use PowerDynamo so that the web site is available over the Web. The Personal Web Server (PWS), which ships with EAServer, is automatically configured to use PowerDynamo. The PWS can be started from Sybase Central by clicking on the PWS icon under the Utilities folder for PowerDynamo.

Any web server that supports CGI, Win-CGI, ISAPI, or NSAPI can be configured to use PowerDynamo. The easiest way to configure the web server to use PowerDynamo is to install the web server before installing PowerDynamo. During the installation of PowerDynamo, the setup routine will detect the presence of the web server, edit the system path to include the PowerDynamo DLLs, and configure the web server automatically. Check out the PowerDynamo User's Guide for more information on how to configure CGI, Win-CGI, ISAPI, and NSAPI web servers.

TIP When deploying a PowerDynamo web application, the web server and the PowerDynamo server must *always* be installed on the same machine.

8.15 How does a web server know to use PowerDynamo?

From a technical point of view, the web server receives a URL from a web browser or web server like the one listed below. The request is forwarded to PowerDynamo through either CGI, ISAPI, or NSAPI:

http://www.MySite.com/safari/StartPage.stm

The URL is initially sent as an HTTP request to the web server listening on port 80 that is on the machine specified by the address www.MySite.com. The web server, which is configured to use PowerDynamo, invokes the appropriate PowerDynamo ISAPI, NSAPI, or CGI program and sends the rest of the URL request to PowerDynamo. The URL prefix that corresponds to the mapping, in this case, */safari*, determines which PowerDynamo Web site gets the request (there could be more than one). The request that is passed to PowerDynamo is the path and name of the resource, which in this example is StartPage.stm. PowerDynamo looks for this document in the

root folder and executes the document. The generated HTML page is returned to the web server.

The URL that the web browser or web server sends must specify the appropriate ISAPI, NSAPI, or CGI program as part of the URL. Which one and how this is done depends on the web server and the extensions that are being used. Web servers that use CGI have an executable directory where CGI programs are kept. This directory has an alias, typically, /cgi-bin, that can be used as part of the URL prefix. A URL call for PowerDynamo using CGI would look something like this:

http://www.MySite.com/cgi-bin/dycgi03.exe/safari/StartPage.stm

The dycgi03.exe file is the CGI program provided with PowerDynamo so that it can interface with a web server using CGI. In order for a web site to interface with PowerDynamo through CGI, the CGI Helper must also be running.

In order to make a call to PowerDynamo through ISAPI, the URL would look like this:

http://www.MySite.com/Scripts/dyisa03.dll/safari/StartPage.stm

The dyisa03.dll file is provided with PowerDynamo so that it can interface with a web server using ISAPI, and is usually placed in an executable directory with an alias, typically Scripts. An ISAPI filter can be used to hide the /Scripts/dyisa03.dll part of the URL, making it easier to call PowerDynamo. NSAPI is similar to ISAPI and also has a way to mask the internal process of routing the request to PowerDynamo from the web server.

8.16 What is a PowerDynamo template?

PowerDynamo templates are text files that are used to build the dynamic web site. PowerDynamo templates provide the simplicity of tag-based page design with the power and extensibility of the DynaScript language. Templates often have the file extension .stm.

A PowerDynamo template can contain one or more of the following:

- Static HTML
- Dynamo tags
- DynaScript

The static HTML is always presented to the client the same way each time the page is displayed, because it is sent without being changed by the PowerDynamo server. The Dynamo tags and the DynaScript are interpreted and run on the PowerDynamo server before anything is sent to the client. The results of the scripts are used to generate the dynamic portion of the HTML page.

Templates often contain static HTML tags that define the basic layout of the page. The script on the template typically uses user input to run queries against a database or call components. The outcome of the queries and components is merged with the static HTML to form a completed HTML page.

8.17 What is a Dynamo tag?

Dynamo tags, like HTML tags, are used to create web documents. The difference between Dynamo tags and HTML tags is that the Dynamo tags are processed on the server by PowerDynamo, while HTML tags are processed on the client by the browser. Dynamo tags allow some functionality to be built into the web site without using DynaScript. There are several PowerDynamo tags including the following:

- <!--SQL-->
- <!--SCRIPT-->
- <!--INCLUDE-->
- <!--COMPONENT-->

Each Dynamo tag is embedded within an HTML comment, so they start with <!-- and end with a -->. The tag that is most commonly used is the SCRIPT tag, which is used to embed DynaScript in a template or script file. Most Dynamo tags have scripting equivalents, so some tasks can be accomplished with either a tag or a script. For example, the CreateComponent method on the java object can be used in place of the COMPONENT tag to access Jaguar components. The scripts are much more flexible than the tags. For more information on Dynamo tags, see the PowerDynamo User's Guide.

8.18 What is DynaScript?

DynaScript is PowerDynamo's server-side scripting language. It is a superset of ECMAScript, which is the standard that JavaScript is based on. As a result, much of the syntax is similar to JavaScript. This allows developers familiar with HTML and JavaScript to become proficient in writing PowerDynamo web sites very quickly.

DynaScript is used to write business logic, access components, and access data in a database. DynaScript is either embedded in a template (.stm) or stored in a separate script file (.ssc). Scripts can be written so that they generate an entire document including the HTML tags, or they can be combined with static HTML in a template to form a dynamic HTML document. Placing common scripts in a separate file allows the file to be included in other scripts or templates, making the code reusable throughout the site.

DynaScript is a language with its own set of literals, variables, expressions, statements, operators, functions, and objects. Some nuances of the DynaScript language include:

- script must be written within the SCRIPT tags
- language is case-sensitive
- variables do not need to be declared
- object-oriented, so developers can create their own objects

There are several predefined objects supported by DynaScript that make it easier to use to develop a web site. These objects provide important functionality such as session management, document redirection, and access to Java objects and Jaguar components. Some of the more common predefined objects include the following:

- document object
- java object
- session object
- site object

It is beyond the scope of this book to delve into all the nuances of DynaScript syntax and programming. Instead, this book will focus on the basics of generating dynamic HTML and how to access Jaguar components from a template script. Check out the PowerDynamo Reference for additional information.

NOTE For those familiar with PowerSite, it has its own object model that is converted at deployment time into PowerDynamo syntax. This book will stick to using the PowerDynamo syntax as we are not directly covering the PowerSite IDE. PowerSite is a complete web application development tool and its scope lies outside of an advanced Jaguar book.

8.19 How do I create a PowerDynamo template?

PowerDynamo ships with Sybase Central, which is the management tool for PowerDynamo web sites. Sybase Central provides a basic interface for developing a template. In addition to the Sybase Central tool, PowerDynamo applications can be built with Sybase's PowerSite, a team-based web application development tool, which is fully integrated with the PowerDynamo product and can deploy web resources directly to the web site. At this point we will focus on Sybase Central and build a simple template.

In order to add a template to our web site, we need to open Sybase Central and connect to our PowerDynamo application through the connection profile. Click on

the Site folder, which may be named differently depending on the name chosen for the root folder when the web site was connected for the first time. The right panel will display the utilities available for the web site as shown in figure 8.15. Select Add Template to open the New Template wizard.

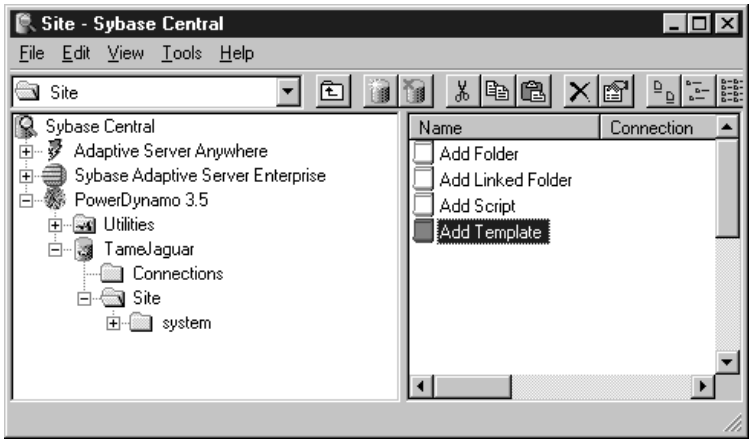

Figure 8.15 Add a template to PowerDynamo site

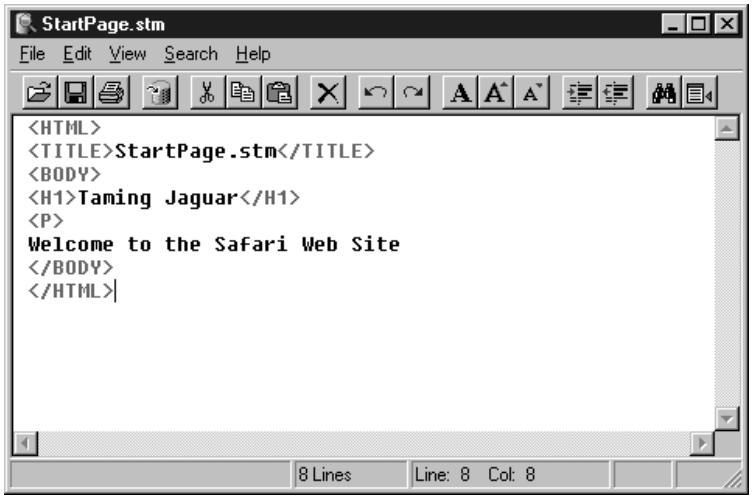

Figure 8.16 Editing the template in Sybase Central

The New Template wizard will prompt you for the name of the template. The wizard will also prompt you for information on which PowerDynamo database connection to use and what SQL query (if any) will be added to the template. (Because

we will not be accessing data from this page, just accept the defaults.) Eventually you will arrive at a dialog where you can create an XML or HTML document. Choose HTML and click the Next button, and on the next dialog accept the default HTML document and then click the Finish button. After the wizard is finished, the Script editor will open with some basic HTML tags for a simple page. After some slight modifications, we have the page shown in figure 8.16. At this point we are building a simple HTML page without any special Dynamo tags or scripts.

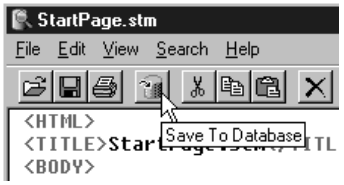

Figure 8.17
Saving the template in Sybase Central

Once the template changes are finished, we need to save the file. Using the File | Save menu option will prompt you to save the template as a file outside of the web site. When you want the changes to be effective in the web site, click the Save to Database toolbar button as pictured in figure 8.17 or the File | Save to Database menu option. Once the template is saved we will be able to see the file under the Site folder as part of the web site.

This file is now accessible by a web browser using the URL:

http://localhost/safari/StartPage.htm

Before running this test, be sure to start the Personal web Server.

8.20 How do I turn off the directory listing?

When a URL is sent to a web server and the name of the file or resource is not specified, the web server must determine which document to return to satisfy the request. The default document is the document that the web server displays when a document is not specified as part of the URL request.

When PowerDynamo uses the database to house the web site and no document is specified, the script Site/system/contview.ssc is run. This script is a system script, which is imported into the site when it is first created. The contview.ssc script will list all of the documents in the directory that is specified in the URL (or in the root directory if no directory is specified). For example, let's run the PowerDynamo site that we just created by typing the following URL in the browser:

http://localhost/safari

This URL will bring up the directory listing for the Site folder as shown in figure 8.18. In a web site, we typically do not want to show the user the listing of the directory. In order to turn this off, we need to make some changes to one of PowerDynamo's system files.

Figure 8.18
Directory listing of a PowerDynamo site

One of the system files imported into the PowerDynamo web site is the Site/system/autoexec.ssc script, which is automatically run when the web server is started. The autoexec.ssc file is used to initialize the PowerDynamo web site. One of the options that is set is the handling of requests for a directory. This is done with the site object. The site object in PowerDynamo is used to manage the web site. The OnEvent method of the site object is used to add an event handler to the web site. The Get event of the Directory item is triggered each time a directory is requested instead of a document, image, or some other resource. In the autoexec.ssc file, find the following line of script and remove it. You can do this by commenting out the line by prefixing it with a //:

```
site.OnEvent('Directory', 'Get', '', '', 'contview.ssc');
```

After making a change to the autoexec.ssc script, remember that the web server must be restarted before the changes take effect. At this point, trying to access the web site without specifying a document will result in a file-not-found error as shown in figure 8.19. In order to correct, this we will need to specify a default document.

Before moving on, it is important to point out that turning off the directory listing for PowerDynamo does not affect the web server itself. In order to handle a

Figure 8.19
File-not-found error after turning off directory listing

URL that does not specify a PowerDynamo mapping (http://www.MySite.com), the directory listing option will need to be turned off at the web server as well. How this is done depends on the web server.

8.21 How do I set the default document?

When a URL that does not specify a document or some other web resource is sent to PowerDynamo, the script contview.ssc is run. This script will display the file listing of the directory that is specified in the URL. In order to replace the directory listing with a default document, we need to change the autoexec.ssc file.

In the autoexec.ssc file, find the line shown below. If you are following along with this chapter, it is commented out:

```
site.OnEvent('Directory', 'Get', '', '', 'contview.ssc');
```

Replace the contview.ssc file name with the name of a default script or document, including the directory relative to the Site folder. An example is shown below:

```
site.OnEvent('Directory', 'Get', '', '', '/default.ssc');
```

Using a script file allows the default document page to be changed without changing the autoexec.ssc file. Therefore, stopping and restarting the web server will not be required. An example of the default document is listed below:

```
<!--SCRIPT
document.redirect="/safari/StartPage.stm";
-->
```

After making a change to the autoexec.ssc script, remember that the web server must be restarted before the changes take effect. At this point, trying to access the web site without specifying a document will result in the default document being returned to the web browser. This web document could be another script that generates an HTML page as well as a PowerDynamo template. In our example, let's run the PowerDynamo site that we just created by typing the following URL in the browser:

http://localhost/safari

This URL will cause PowerDynamo to execute our default script, which will redirect the browser to the StartPage.stm template and return the generated HTML page as shown in figure 8.20.

Figure 8.20
StartPage.stm in a browser

Before moving on, it is important to point out that setting the default document for PowerDynamo does not specify the default document that the web server will return when only the domain name or IP address is specified. In order to handle a URL that does not specify a PowerDynamo mapping (http://www.MySite.com) the default document will need to be handled at the web server as well. How this is done depends on the web server.

TIP With IIS, we can use ASP to set up the default document on the web server to redirect the web request to our PowerDynamo web site. This will allow a web browser to send requests to our PowerDynamo web site using just the web server domain name or address (http://www.MySite.com).

8.22 How do I add DynaScript to a template?

DynaScript is added to a template using the SCRIPT tag. All the script is then written between the <!--SCRIPT and --> tags. A simple example is shown below:

```
<HTML>
<TITLE>HelloWorld.stm</TITLE>
<BODY>
<!--SCRIPT
   document.Write( "<H1>Hello world</H1>" );
-->
</BODY>
</HTML>
```

The script in the example uses the document object to access the document being generated. The Write method of the document object is used to add the string "<H1>Hello World</H1>" to the document itself. The string that is added is a combination of HTML tags and text that will be sent to the web browser as follows:

```
<HTML>
<TITLE>HelloWorld.stm</TITLE>
<BODY>
<H1>Hello world</H1>
</BODY>
</HTML>
```

Notice that the actual script is not sent to the web browser. It is executed on the PowerDynamo server and the output is applied to the document.

8.23 How do I create a PowerDynamo script?

A PowerDynamo script is a file that contains only DynaScript. By placing common logic, functions, and class definitions into a separate script file, the code is reusable between templates because the script file can be imported. Scripts often have the file extension .ssc.

In order to add a script to our web site, we need to open Sybase Central and connect to our PowerDynamo application through the Connection Profile. Click on the Site folder, which may be named differently depending on the name chosen for the root folder when the web site was connected to for the first time. The right panel will display the utilities available for the web site. Select Add Script icon to open the New Script wizard. Type the name and the description of the script.

The script will be saved and placed in the web site under the Site folder. Open the script by double-clicking on it in Sybase Central. All the logic is written between the <--SCRIPT and --> tags. In this example, we will create a separate script file, GetDate.ssc, to get today's date. The code is shown below:

```
<!--SCRIPT GetDate.ssc

function getTodayDate()
{
    myDate = new Date();
    iMonth = myDate.getMonth();
    iDate = myDate.getDate();
    iYear = myDate.getFullYear();
    switch (iMonth) {
    case 0:
        sMonth = "January";
        break;
    case 1:
        sMonth = "February";
        break;
    case 2:
        sMonth = "March";
        break;
    case 3:
        sMonth = "April";
        break;
    case 4:
        sMonth = "May";
        break;
    case 5:
        sMonth = "June";
        break;
    case 6:
        sMonth = "July";
        break;
    case 7:
        sMonth = "August";
        break;
    case 8:
        sMonth = "September";
        break;
    case 9:
        sMonth = "October";
        break;
    case 10:
        sMonth = "November";
        break;
    case 11:
        sMonth = "December";
        break;
    }
    CurrentDate = sMonth + " " + iDate + ", " + iYear;

    return CurrentDate;
}

-->
```

This script can now be used in any template to display the current date.

8.24 How do I import a script into the template?

If we want to import the script GetDate.ssc we can add a SCRIPT tag to a template and use the import function. The import function takes a string argument that contains the name of the script file that is being imported. The script file needs to have the path specified in relation to the root of the web site. For the GetDate.ssc file which is in the root directory just prefix the name of the file with a / as shown in the code sample below:

```
<HTML>
<TITLE>StartPage.stm</TITLE>
<BODY>
<!--SCRIPT
// Import scripts
import('/getdate.ssc');
-->

<H1>Taming Jaguar</H1>
<P>
Welcome to the Safari Web Site - today is
<!--SCRIPT
// Write out the data
document.writeln(getTodayDate());
-->

</BODY>
</HTML>
```

The function getTodayDate can now be called from other scripts in the template.

8.25 How are page parameters passed?

A web application is built by passing information between the web client and the web server and also between HTML pages. This information can be user input from an HTML form or a hyperlink. This information can also be data previously collected and being forwarded to the new page due to the stateless nature of the Web.

Any information from the web browser is sent to the web server as part of the URL when a hyperlink (link) or a button on a form is clicked by the user. The address and the web resource portion of the URL described in section 8.4, "What is a URL," are separated from the arguments being passed to the page by a question mark to signify that a list of values follows. The arguments are listed in name/value pairs and are separated by an ampersand when passing more than one. The name/value pairs are in the format name=value where the argument name is followed by an "=" sign and the value being passed. The URL syntax is shown below:

```
http://www.MySite.com/Dynamo_Mapping/
template-name?arg1=value1&arg2=value2
```

This URL can be sent from an HTML *form* or *anchor* tag, so let's quickly examine these. If you are interested in learning more about these HTML tags and others, we recommend checking out additional HTML resources like *HTML the Definitive Guide,* published by O'Reilly.

The HTML form tag is the most common way to send requests to a web server. It allows the user to enter information in a variety of formats that are read by the browser and appended to the end of the URL automatically. The HTML form is delimited by the tags <FORM> and </FORM>. Everything in between the tags is considered part of the form. Anything outside the form tags will be ignored when the form is submitted.

The form tag has several attributes but the ACTION and METHOD are the most important. They determine the address and resource that will receive the request and how the request will be sent as described in table 8.2.

Table 8.2 FORM attributes

Attribute	Description
ACTION	This is the address and resource to send the URL to when the user submits the form.
METHOD	This specifies the way the request is sent. The valid values are GET and POST.

The GET and POST values of the METHOD attribute define how the data on the form is sent from the web browser to the web server and from the web server to the CGI, ISAPI, or NSAPI program. The GET method sends information in the URL, while the POST method passes data as a stream. The GET method has a limit as to the number of characters that can be sent, determined by the browser being used. This can be a problem with large forms and TEXTAREA tags (similar to multiline edit). The POST method does not have this limitation and is the preferred means of passing information. In addition to the limit on the size of the input that can be sent, GET and POST require different methods of reading in the data by the CGI program. However, PowerDynamo will automatically handle reading the input from the form based on the METHOD and pass the values to the template.

Another important aspect on passing information from an HTML form is that the GET value for the METHOD attribute passes information so that it can be seen in the address field of the browser, while the POST value does not as shown in figures 8.21 and 8.22. Although both the GET and POST methods pass the same information, they pass it differently. This is important when passing sensitive information like passwords or hidden information so that the values cannot be seen in the browser. The information, however, is not encrypted when passed over the network as this requires a secure socket using the Secure Socket Layer (SSL).

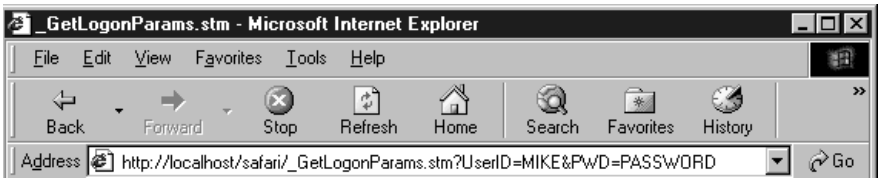

Figure 8.21 Address field of browser using the GET method

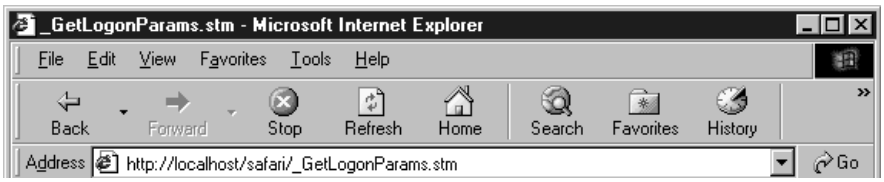

Figure 8.22 Address field of browser using the POST method

An example of a simple HTML form is an HTML page that requests a user ID and a password from the user. When the user clicks on the Submit button, the URL specified in the ACTION attribute is sent to the web server along with the data in the INPUT tags. Here is the template StartPage.stm, which has an HTML form tag added to it that collects the UserID and password of the user.

```
<HTML>
<TITLE>StartPage.stm</TITLE>
<BODY>
<!--SCRIPT
// Import scripts
import('/getdate.ssc');
-->

<H1>Taming Jaguar</H1>
<P>
Welcome to the Safari Web Site - today is
<!--SCRIPT
// Write out the date
document.writeln(getTodayDate());
-->

<P>
<FORM ACTION="_GetLogonParams.stm" METHOD=GET>
<TABLE>
<TR>
    <TD align=right>UserID:</TD>
    <TD align=left><INPUT TYPE=text NAME=UserID></TD>
</TR>
```

```
<TR>
    <TD align=right>Password:</TD>
    <TD align=left><INPUT TYPE=password NAME=PWD></TD>
</TR>
<TR>
    <TD align=center colspan=2><INPUT TYPE=submit VALUE=Logon></TD>
</TR>
</TABLE>
</FORM>

</BODY>
</HT\ML>
```

The HTML page is displayed in the browser as shown in figure 8.23. Once the user types in a user ID and password and clicks the Submit button, the following URL is generated and sent to the web server:

http://localhost/safari/_GetLogonParams.stm?UserID=MIKE&PWD=PASSWORD

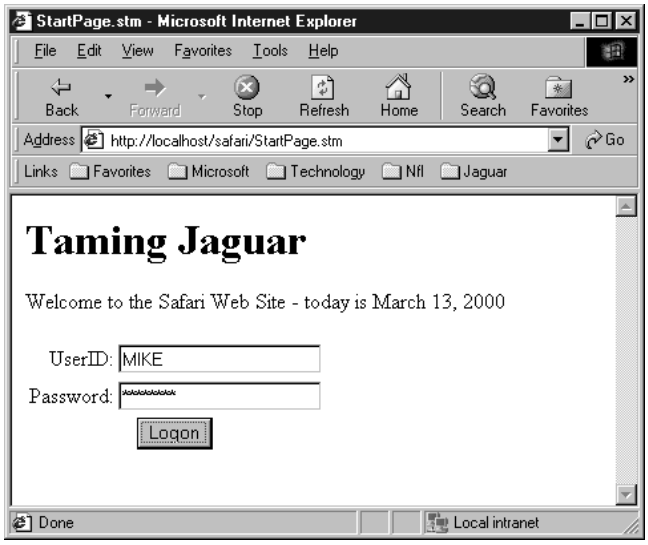

Figure 8.23
An HTML form in a browser

Notice that the argument names match the INPUT tag NAME attributes used in the HTML form. The INPUT tag is the basic building block of an HTML form. The INPUT tag specifies what elements make up the form. Several attributes of the INPUT tag are listed in table 8.3.

Table 8.3 Input tag attributes

Attribute	Description
CHECKED	Used by radio button and check box inputs to determine if the object is selected.
MAXLENGTH	Used by the text input to limit the number of characters that are accepted in the field.
NAME	Identifies the argument name that the name/value pair should be passed as.
SIZE	Used by the text input to determine the width of the field shown.
TYPE	Specifies the type of input object to be displayed.
VALUE	The initial value of the input (or for buttons the name shown on the button face).

The INPUT tag is quite versatile, allowing for several different input types. The type of input displayed is specified using the TYPE attribute. Valid values for the TYPE attribute of the INPUT tag are shown in table 8.4.

Table 8.4 Values for the TYPE attribute

Value	Description
TEXT	A single line edit field.
CHECKBOX	A checkbox.
RADIO	A radio button.
PASSWORD	A single line edit field, which hides each character the user types with a *.
HIDDEN	A hidden text field.
SUBMIT	The Submit button, which sends the URL specified by the action to the web server.
RESET	The Reset button, which clears all the elements in a form.
BUTTON	Creates a custom button.
IMAGE	Same as the Submit button, but an image can be used in place of the button.
FILE	A single-line edit that accepts a file name and uploads the actual file. Requires the FORM attribute ENCTYPE= "multipart/form-data".

The Submit button is required to trigger the action specified by the form. When the Submit button is clicked, the browser collects all the values from the form elements and appends them to the URL before sending the request to the web server. In addition to the INPUT tag, there is the text area (<TEXTAREA>) tag and the select (<SELECT>) tag. The text area tag is used to display a multiline edit field. The select tag is used with the option (<OPTION>) tag to create a drop down list.

The anchor tag <A> is used to create hyperlinks in an HTML document. These links can direct the browser to other portions of the same document or to other documents, images, or application programs that can reside on both the local machine or on a remote machine. Hyperlinks, defined by the anchor tag, are the basic way a user navigates around the web. HTML documents contain links to other documents, usually related to the information on the current page. The HTML page that contains the hyperlink is considered the source document. The document

that is referenced by the URL in the anchor tag is the target document. The anchor tag has several attributes that can be used to control how the link will work as shown in table 8.5.

Table 8.5 Anchor tag attributes

Attribute	Description
HREF	Used to specify the URL of the target document.
NAME	Used to name a section of the current document as a hyperlink target so that other links can point to it.
TARGET	Used to direct the browser to display the target document in a specific frame.

The text between the <A> and </A> tags defines the text area that a user may click on to jump to the new URL or subsection. Most browsers show this text in a different color and underline it. A code sample of a hyperlink is shown below:

```
<HTML>
<TITLE>Sample.stm</TITLE>
<BODY>
<H1>Taming Jaguar</H1>
<H2>Sample<H2>
<P>
<A HREF="HelloWorld.stm">Go to HelloWorld</A>
</BODY>
</HTML>
```

This is displayed in a browser as shown in Figure 8.24.

Figure 8.24
The Anchor tag in a browser

An anchor tag can also specify an image in between the <A> and </A> tags. This image is now a link that the user can click on to jump to another URL. The code sample for an anchor tag as an image is shown below:

```
<A HREF="HelloWorld.stm"><IMG SRC="JagHead.jpg" BORDER=0></A>
```

The NAME attribute is used to define a target for a hyperlink. The HREF attribute is used to define the URL of the new resource. The NAME and HREF attributes should not be used in the same tag. The URL can specify an absolute address or a relative address. An absolute address contains the protocol, domain name of the server and resource. The relative address is based off of the location of the current document, usually the HTML directory on the web server.

Here is an example of an anchor tag that uses an absolute address to point to a document on another web site:

```
<A HREF="http://www.sybase.com">Sybase Inc.</A>
```

An anchor tag that references a document on the same server could use a relative address as shown below:

```
<A HREF="about.stm">About Taming Jaguar</A>
```

The HREF in the example above will look for the about.stm page in the same directory that the current document was found. This makes HTML page management easier.

To include arguments in an anchor tag, they must be hard-coded onto the end of the URL using the same syntax as the <FORM> URL. The URL can be dynamically created in a Dynamo script. To add arguments to the URL, add a "?" after the document name followed by the arguments in name/value pair format. The argument values are listed one after the other and are separated by an ampersand (&) as shown below:

```
<A HREF="_GetLogonParams.stm?UserID=MIKE&PWD=PASSWORD">Logon</A>
```

In order to build this dynamically, the HTML must be constructed using DynaScript and written out to the document as shown in the code sample below:

```
<!--SCRIPT

// Get the values to be passed in Anchor Tag
UserName = "MIKE";
PassWord="PASSWORD";

// Build Anchor Link
Link='_GetLogonParams.stm?UserID=' + UserName + '&PWD=' + PassWord

// Write Anchor tag to document
document.writeln('<A HREF="' + Link + '">Logon</A>');

-->
```

This anchor tag will call the _GetLogonParams.stm page passing in the arguments UserID and PWD. When passing an anchor tag with information, it is important to note that the URL can be seen in the address field of the browser. This is similar to the GET method.

8.26 How are page parameters received?

The page parameters that are passed into a template can be accessed directly in HTML or in a script through the document object. The name of each page parameter is determined by the name of the argument passed in through the URL. The StartPage.stm HTML FORM will pass a URL to the web server as shown below after the user types in the information and clicks on the Submit button:

http://localhost/safari/_GetLogonParams.stm?UserID=MIKE&PWD=PASSWORD

The page parameters are named UserID and PWD with the values MIKE and PASSWORD respectively.

The value of a page parameter can be directly inserted into the HTML portion of a document by preceding the name of the page parameter with a dollar sign. For example, to show the UserID and Password passed to the _GetLogonParams page from the StartPage.stm form, we can code the following:

```
<HTML>
<TITLE>_GetLogonParams.stm</TITLE>
<BODY>
<P>
The UserID passed in is $UserID;
<BR>
The Password passed in is $PWD;
</BODY>
</HTML>
```

This template,—when returned to the browser after typing in a UserID of MIKE and a password of PASSWORD in the form supplied by the StartPage.stm—generates the output illustrated in figure 8.25.

Page parameters are accessed inside scripts through a property of the document object. The syntax to access a page parameter from the document property is as follows:

```
document.value.variable
```

The variable portion of the syntax is where the name of the page parameter is specified. This property of the document object is read-only. The following code sample

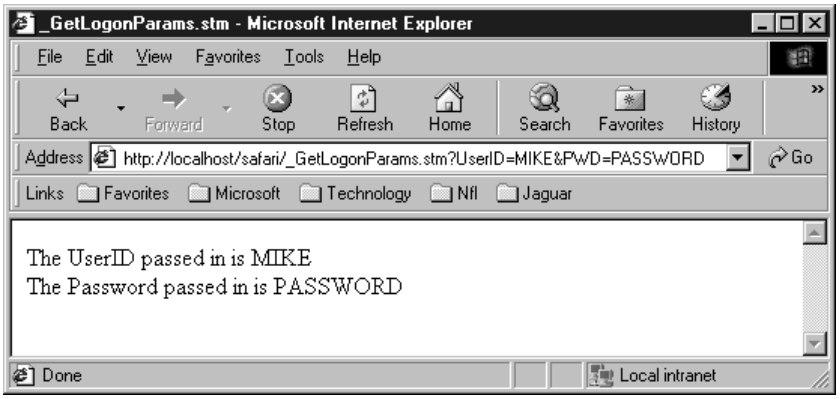

Figure 8.25 Displaying page parameters

illustrates how to use the document object to get the user input of UserID and PWD that were passed into the template. The code sample also illustrates how to write them out to the document:

```
<HTML>
<TITLE>_GetLogonParams.stm</TITLE>
<BODY>
<P>
<!--SCRIPT
var UserName= document.value.UserID;
var Password = document.value.PWD;

document.writeln('The UserID passed in is ' + UserName);
document.writeln('<BR>');
document.writeln('The Passoword passed in is ' + Password);

-->
</BODY>
</HTML>
```

This template will result in the same browser output as shown in figure 8.25. Sometimes a template that expects page parameters that are not passed in can cause errors when it attempts to access them. For example, type the following URL in a web browser:

http://localhost/safari/_GetLogonParams.stm

This will ask PowerDynamo to run the _GetLogonParams.stm template without passing in any page parameters. When this happens, the document.value.variable property is undefined and generates a web page with errors as shown in figure 8.26.

To avoid any errors or references to undefined variables, we can check for the existence of the page parameter first. Note the following code sample:

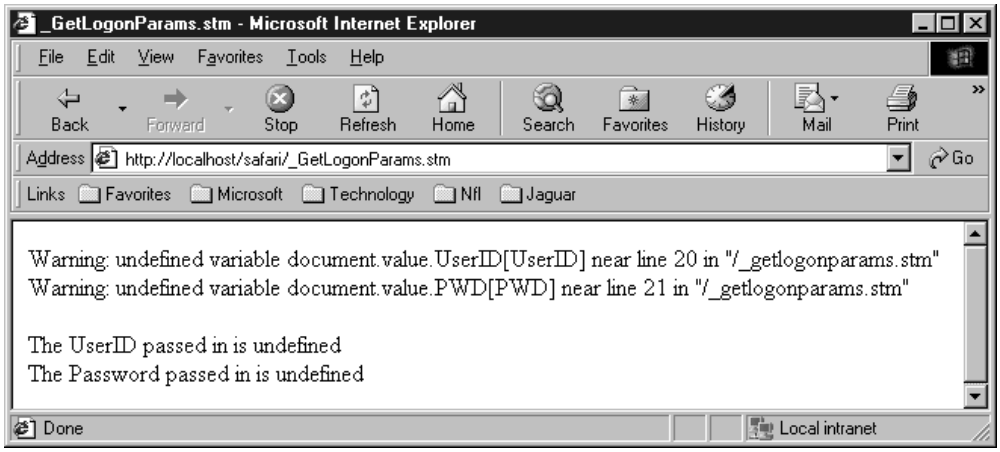

Figure 8.26 Error accessing undefined page parameters

```
// Check for UserID page parameter
if (exists(document.value.UserID)) {
   UserName= document.value.UserID;
}
else {
   UserName= "";
}

// Check for PWD page parameter
if (exists(document.value.PWD)) {
   Password= document.value.PWD;
}
else {
   Password= "";

}
```

8.27 How can I set URL page parameter case-sensitivity?

While DynaScript is case-sensitive, the page parameters passed in through the URL that PowerDynamo accepts can have case-sensitivity turned on and off. Under the PowerDynamo Utilities folder, click on the Configuration folder and highlight the Default General Settings folder. One of the available settings is "URL arguments case sensitive." The value can be set to *yes* to have case-sensitivity turned on and set to *no* to have case-sensitivity turned off.

For example, given the following code and a page parameter passed in as PWD=PASSWORD, we will get two different results based on the value of this PowerDynamo setting.

```
document.writeln('<P>');
document.writeln('The Password passed in is ' + document.value.PWD);
document.writeln('<BR>');
document.writeln('The Password passed in is ' + document.value.pwd);
```

When case-sensitivity is turned on, the page parameter document.value.pwd is considered undefined and we get the following result:

```
The Password passed in is PASSWORD
The Password passed in is undefined
```

When case-sensitivity is turned off, the page parameter document.value.pwd is considered the same as the page parameter document.value.PWD and we get the following result:

```
The Password passed in is PASSWORD
The Password passed in is PASSWORD
```

8.28 How do I deal with special characters in a URL?

When passing data back and forth in a URL, you will probably run across some characters that need to be handled specially in order to successfully pass them from a web browser to the web server. Some of these characters include the space character, colon, semicolon, backslash, equal sign, and double quote. These characters need to be encoded in a URL using their ASCII hexadecimal equivalents, prefixed by a % sign. For example, a space character is ASCII 20, so all space characters in a URL should be replaced with a %20.

PowerDynamo provides two system functions to handle encoding and decoding a URL. These are the escape and unescape functions. The escape function is used to encode special characters so they can be used in URLs. The unescape function is used to decode special characters from URLs.

So for example if I wanted to pass the value `"TEST = 1:TEST%` as part of my URL, I would code something like the following:

```
<!--SCRIPT

// Get the values to be passed in Anchor Tag
Value = escape('"TEST = 1:TEST%');

// Build Anchor Link
Link='URLTest.stm?URLArg=' + Value;

// Write Anchor tag to document
document.writeln('<A HREF="' + Link + '">Click to Send URL</A>');

-->
```

The value returned by the escape function and passed in the URL would be encoded with the following `%22TEST%20%3d%201%3aTEST%25`.

TIP Only use *escape* on the value that you are trying to pass and not on the entire URL. Otherwise, the escape character will encode parts of the URL that should not be encoded, like the // in http:// or the "?" that separates the address and resource name from the page parameters.

8.29 How are multiple values passed using the multiple SELECT tag or checkbox?

When page parameters are passed from an HTML page to the web server, and ultimately, to PowerDynamo, the name of the page parameter often appears only once as a name/value pair in the URL. We have seen this in previous examples where we have looked at an HTML form used to log in to a web site. The form contains an input tag with the name UserID and another with the name PWD. Sometimes, however, the name of a page parameter can appear multiple times in an HTML form, which will show up as several name/value pairs with the same name in the URL as well. An example of this is seen when trying to collect information with a checkbox input tag or a multiple select statement.

An HTML form can be used to collect information with a checkbox or multiple select that indicates that more than one option applies for a given parameter. For example, if we were collecting information on the pets that visitors to our web site had, we might present a list of animals probably generated from a result set that was retrieved from a database. Each specific animal would have a unique ID and a description. Because some users might have more than one type of animal as a pet, we would need to allow them to enter more than one value. The HTML page below shows how this information could be collected with an HTML form and the checkbox input tag. Notice that each input tag has the same value for the NAME attribute, pets_id, but has a different value for the VALUE attribute associated with the unique pet ID stored in the database. This allows us to collect one or more pet ID values and store them in a table that associates pets with our web visitor. This technique is easier than generating a unique NAME attribute for each checkbox that is shown on the page, because managing the number of unique names for a large set of data could be cumbersome, especially when trying to process the page parameters in the script or template the HTML form calls.

```
<HTML>
<TITLE>CheckBox</TITLE>
<BODY>
<H1>Check all that apply</H1>
```

```
Which pets do you own?
<FORM ACTION=getmultivalue.stm METHOD=GET>
    <TABLE border=1>
        <TR>
        <TD>Select</TD>
        <TD>Pet</TD>
        </TR>

        <TR>
        <TD><INPUT TYPE=checkbox NAME=pets_id VALUE=1 ></TD>
        <TD>Dog</TD>
        </TR>

        <TR>
        <TD><INPUT TYPE=checkbox NAME=pets_id VALUE=2 ></TD>
        <TD>Cat</TD>
        </TR>

        <TR>
        <TD><INPUT TYPE=checkbox NAME=pets_id VALUE=3 ></TD>
        <TD>Bunny</TD>
        </TR>

        <TR>
        <TD><INPUT TYPE=checkbox NAME=pets_id VALUE=4 ></TD>
        <TD>Iguana</TD>
        </TR>

        <TR>
        <TD colspan=2 align=center><INPUT TYPE=submit VALUE=OK></TD>
        </TR>
    </TABLE>
</FORM>
</BODY>
</HTML>
```

The code sample below shows an HTML page that uses a multiple select tag to do the same thing as the HTML page that uses the checkbox. With the select tag, there is only one NAME attribute associated with the tag so a multiple select tag will always pass back the values chosen with the same name in the URL.

```
<HTML>
<BODY>
<H1>Check all that apply</H1>
Which pets do you own?
<FORM ACTION=getmultivalue.stm METHOD=GET>
    <SELECT NAME=pets_id multiple>
        <OPTION VALUE=1>Dog
        <OPTION VALUE=2>Cat
        <OPTION VALUE=3>Bunny
        <OPTION VALUE=4>Iguana
    </SELECT>
    <BR>
```

```
    <INPUT TYPE=submit VALUE=OK>
</FORM>
</BODY>
</HTML>
```

The HTML page above produces the output in a browser as pictured in figure 8.27.

Figure 8.27
A multiple select tag

Once the user selects the appropriate choices and clicks the Submit button, the HTML form generates a URL as shown below:

http://localhost/safari/getmultivalue.stm?pets_id=1&pets_id=3

Because two pets were selected, the URL contains the name/value pair for pets_id twice.

8.30 How do I get the values from a multiple SELECT tag or checkbox?

When page parameters are passed into PowerDynamo, they are accessed through the document object. The document object has a value property that has a variable property defined for each page parameter with corresponding names. For details on this, see section 8.26, "How are page parameters received." However, when a name appears more than once in the URL, PowerDynamo handles this in the document.value.variable, which is similar to an array.

```
document.value.variable[i];
```

So if only one item is selected or there is only one value passed in with the same name, the value is held in document.value.variable. When two or more items are selected, the values are stored in the document.value.variable as document.value.variable[0], document.value.variable[1], and so on.

This is easier to see in an example. The following URL was generated by either the checkbox or multiple select HTML pages described in the previous question and shown below:

http://localhost/safari/getmultivalue.stm?pets_id=1&pets_id=3

We would have the following in our document object:

```
document.value.pets_id[0] = 1
document.value.pets_id[1] = 3
```

Although the way PowerDynamo uses the document.value.variable to hold multiple values looks like an array, there is no length property. Thus, it is easier to work with the page parameters if the contents are copied into a PowerDynamo Array object first. A script that contains a function that converts a page parameter into a PowerDynamo array is shown below:

```
<!--SCRIPT

function PageParam_Array(PageParam)
{
    // convert a page parameter into an array
    // expects a page parameter with multiple values
    PDArray = new Array();
    i = 0;

    if( typeof( PageParam ) == "object" ) {
        while( exists( PageParam[i] ) ) {
          PDArray[i] = PageParam[i];
          i++;
        }
    }
    else {
        PDArray[i] = PageParam;
    }

    return PDArray;
}

-->
```

The script is designed to handle page parameters that have only one value as well as those that contain several. The PowerDynamo template that expects multiple values for a given page parameter name can import the script and use the PowerDynamo array to manage the values. A sample template that uses the script is shown below:

```
<HTML>
<TITLE>Get Multi Value</TITLE>
<BODY>
<H1>Your selections:</H1>
<!--SCRIPT
  import ArrayHelper.ssc;

  if (exists(document.value.pets_id)) {
    myArray = PageParam_Array(document.value.pets_id)

    for(var i=0; i<myArray.length; i++) {
        document.writeln( "value[" + i + "] = " + myArray[i] );
        document.WriteLn( "<BR>");
    }
  }
  else {
   document.writeln( "Page parameter document.value.pets_id does not exist.");
  }
-->

</BODY>
</HTML>
```

8.31 How do I redirect a document in PowerDynamo?

When a browser asks for a particular web page, it may be necessary to redirect the request to another web page that it did not ask for to satisfy the request. In addition, an application may have a standard web page to handle error reporting to the user. As a script processes, it may need to redirect the client to the error page when it encounters a problem. The PowerDynamo document object has a redirect property that allows the script to specify the URL to which the current request should be directed.

```
document.redirect=URL_String
```

Continuing with the login example, we can implement a simple validation routine that checks to see if the UserID is equal to the password passed in from the HTML form. Based on the success or failure of the login attempt, the web browser is directed to the next appropriate page. The client, however, did not directly request the page that was received.

```
if (UserID==Password) {
    // valid logon
    document.redirect="HelloWorld.stm";
}
else {
    // invalid logon
    document.redirect="LogonError.stm";
}
```

The redirect will load and run the page specified after the current script and all the scripts on the current page have been processed. In order to get the current page to immediately direct the request to the new URL, use the exit statement after the redirect. The exit statement stops processing the current document from that point forward.

```
document.redirect="StartPage.stm";
exit;
```

Using PowerDynamo, the redirect property can be used to direct the user of the web site to any valid URL. This includes other web resources stored in a PowerDynamo web site, or any other web site. The redirect property can even be used to call Java servlets as shown below:

```
document.redirect="/servlet/HelloWorld";
exit;
```