JavaServer Pages™ White Paper

Dynamic Generation for the Web

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
1 (800) 786.7638
1.512.434.1511

Introduction

JavaServer Pages™ (JSP) technology provides a simplified and quick method for creating Web pages that display dynamically-generated content. JSP technology was designed to make it easier and faster to build Web-based applications that work with a wide variety of Web servers, application servers, browsers, and development tools.

This paper provides an overview of JSP technology, describing how it was developed and the overall goals for the technology. It also describes the key components of a Java™ technology-based page, through a simple example.

Developing Web-based Applications: A Background

In its short history, the World Wide Web has evolved from a network of basically static information displays to a mechanism for trading stocks, buying books, etc. — in other words, e-commerce. There is virtually no limit to the possible uses for Web-based clients in diverse applications.

Applications that can make use of browser-based clients have several advantages over traditional, client/server-based applications, such as nearly unlimited client access and greatly simplified application deployment and management. To update an application, a developer only needs to change one server-based program, not thousands of client-installed applications. As a result, the software industry is quickly moving toward building multi-tiered applications using browser-based clients.

These increasingly sophisticated Web-based applications require changes in development technology. Standard HTML is fine for displaying relatively static content; the challenge has been creating interactive, Web-based applications in which the content of the page is based on a user request or system status and not pre-defined text.

An early solution to this problem was the CGI-bin interface. Developers wrote individual programs to this interface, and Web-based applications called the programs through a Web server. This solution has significant scalability problems — each new CGI request launches a new process on the server. If multiple users access the program concurrently, these processes consume all of the Web server's available resources, and performance slows to a grind.

Individual Web server vendors have tried to simplify Web application development by providing "plug-ins" and APIs for their servers. These solutions are Web server-specific and don't address the problem across multiple vendor solutions. For example, Microsoft's Active Server Pages (ASP) technology makes it easier to create dynamic content on a Web page, but only works with Microsoft IIS or Personal Web Server.

Other solutions exist, but they are not necessarily easy for the average page designer to deploy. Technologies such as Java Servlets, for example, make it easier to write server-based code using the Java programming language for interactive applications. A Java Servlet is a Java technology-based program that runs on the server (as opposed to an applet, which runs on the browser). Developers can write Servlets that take an HTTP request from the Web browser, generate the response dynamically (possibly querying databases to fulfill the request), and then send a response containing an HTML or XML document to the browser.

Using this approach, the entire page must be composed in the Java Servlet. If a developer or Web master wants to tune the appearance of the page, they must edit and recompile the Java Servlet, even if the logic is already working. With this approach, generating pages with dynamic content still requires application development expertise. Clearly, what is needed is an industry-wide solution for creating pages with dynamically-generated content. This solution should address the limitations of current alternatives by:

- Working on any Web or application server
- Separating the application logic from the appearance of the page
- Allowing fast development and testing
- Simplifying the process of developing interactive, Web-based applications

JavaServer Pages (JSP) technology was designed to fit this need. The JSP specification is the result of extensive industry cooperation between vendors of Web servers, application servers, transactional systems, and development tools. Sun Microsystems developed the specification to integrate with and leverage existing expertise and tools support for the Java programming environment, such as Java Servlets and JavaBeans™ components. The result is a new approach to developing Web-based applications that extends powerful capabilities to page designers using component-based application logic.

The JavaServer Pages Technology Approach to Web Application Development

In developing the JavaServer Pages specification, Sun Microsystems worked with a number of leading Web server, application server, and development tool vendors, as well as a diverse and experienced development community. The result is an approach that balances portability with ease of use for application and page developers.

JSP technology speeds the development of dynamic Web pages in a number of ways:

- *Separating Content Generation from Presentation*

  Using JSP technology, Web page developers use HTML or XML tags to design and format the results page. JSP tags or scriptlets are used to generate the dynamic content on the page (the content that changes according to the request, such as requested account information or the price of a specific bottle of wine). The logic that generates the content is encapsulated in tags and JavaBeans components and is tied together in scriptlets, all of which are executed on the server side. If the core logic is encapsulated in tags and beans, then other individuals, such as Web masters and page designers, can edit and work with the JSP page without affecting the generation of the content.

  On the server, a JSP engine interprets JSP tags and scriptlets, generates the content required (for example, by accessing JavaBeans components, a database with JDBC™ technology, or including files), and sends the results back in the form of an HTML or XML page to the browser. This helps authors protect their proprietary code while ensuring complete portability for any HTML-based Web browser.

- *Emphasizing Reusable Components*

  Most JSP pages rely on reusable, cross-platform components — JavaBeans or Enterprise JavaBeans™ components — to perform the more complex processing required of the application. Developers can share and exchange components that perform common operations, or make them available to larger user or customer communities. The component-based approach speeds overall development and lets organizations leverage their existing expertise and development efforts for optimal results.

- *Simplifying Page Development with Tags*

  Web page developers are not always programmers familiar with scripting languages. JSP technology encapsulates much of the functionality required for dynamic content generation in easy-to-use, JSP technology-specific XML tags. Standard JSP tags can access and instantiate JavaBeans components, set or retrieve bean attributes, download applets, and perform other functions that are otherwise more difficult and time-consuming to code.

  JSP technology is extensible through the development of customized tag libraries. Over time, third-party developers and others will create their own tag libraries for common functions. This lets Web page developers work with familiar tools and constructs, such as tags, to perform sophisticated functions.

JSP technology integrates easily into a variety of application architectures, leveraging existing tools and skills, and scaling to support enterprise-wide distributed applications. As part of the Java technology-enabled family, and an integral part of the Java 2 Platform, Enterprise Edition architecture, JSP technology can support highly complex Web-based applications.

JSP pages have all of the benefits of Java technology, including robust memory management and security, because the native scripting language for JSP pages is based on the Java programming language, and because all JSP pages are compiled into Java Servlets.

As part of the Java platform, JSP technology shares the Write Once, Run Anywhere™ characteristics of the Java programming language. As more vendors add JSP support to their products, more servers and tools will become available, and these servers and tools will be able to be changed without affecting current applications.

When integrated with the Java 2 Platform, Enterprise Edition and Enterprise JavaBeans technology, JSP pages will provide the enterprise-class scalability and performance necessary for deploying Web-based applications across the virtual enterprise.

What Does a JSP Page Look Like?

A JSP page looks like a standard HTML or XML page, with additional elements that the JSP engine processes and strips out. Typically, the JSP elements create text that is inserted into the results page.

JavaServer Pages technology is best described using an example. The following JSP page is very simple; it prints the day of the month and the year, and welcomes the user with either “Good Morning” or “Good Afternoon”, depending on the time of day.

The page combines ordinary HTML with a number of JSP elements:

- Calls to a clock JavaBeans component
- Inclusion of an external file (for copyright information)
- JSP expressions and scriptlets

```
<HTML>
<%@ page language=="java" imports=="com.wombat.JSP.*" %>
<H1>Welcome</H1>
<P>Today is </P>
<jsp:useBean id=="clock" class=="calendar.jspCalendar" />
<UL>
<LI>Day: <%==clock.getDayOfMonth() %>
<LI>Year: <%==clock.getYear() %>
</UL>

<% if (Calendar.getInstance().get(Calendar.AM_PM) ====
Calendar.AM) { %>
Good Morning
<% } else { %>
Good Afternoon
<% } %>
<%@ include file=="copyright.html" %>

</HTML>
```

The page includes the following components:

- *JSP directive*: Passes information to the JSP engine. In this case, the first line indicates the location of some Java programming language extensions to be accessible from this page. Directives are enclosed in <%@ and %> markers.
- *Fixed template data*: Any tags that the JSP engine does not recognize are passed on with the results page. Typically, these will be HTML or XML tags. This includes the Unordered List and H1 tags in the example above.

- *JSP actions or tags:* These are typically implemented as standard or customized tags, and have an XML tag syntax. In the example, the jsp:useBean tag instantiates the clock JavaBeans component on the server.
- *Expression:* The JSP engine evaluates anything between <%== and %> markers. In the List Items above, the values of the day and year attributes of the clock JavaBeans component are returned as a string and inserted as output in the JSP file. In the previous example, the first list item will be the day of the year and the second item will be the year.
- S*criptlet*: A small script that performs functions not supported by tags or ties everything together. The native scripting language for JSP 1.0 software is based on the Java programming language. The scriptlet in the sample determines whether it is a.m. or p.m. and greets the user accordingly (for daytime users, at any rate).

The example may be trivial, but the technology is not. Businesses can encapsulate critical processing in server-side beans, and Web developers can easily access that information using familiar syntax and tools. Java technology-based scriptlets provide a flexible way to perform other functions without requiring extensive scripting. The page as a whole is legible and comprehensible, making it easier to spot or prevent problems and share work.

Following are a few of these components described in more detail.

JSP Directives

JSP pages use JSP directives to pass instructions to the JSP engine. These may include the following:

- JSP *page directives* communicate page-specific information, such as buffer and thread information or error handling.
- *Language directives* specify the scripting language, along with any extensions.
- The *include directive* (shown in the example) can be used to include an external document in the page. Files such as copyright files or company information files are easier to maintain in one central location and include in several pages rather than to update in each JSP page. However, the included file can also be another JSP file.
- A *taglib directive* indicates a library of custom tags that the page can invoke.

JSP Tags

Most JSP processing will be implemented through JSP technology-specific XML-based tags. JSP 1.0 technology includes a number of standard tags, referred to as the core tags. These include:

jsp:useBean

This tag declares the usage of an instance of a JavaBeans component. If the Bean does not already exist, then the JavaBeans component instantiates and registers the tag.

jsp:setProperty

This sets the value of a property in a Bean.

jsp:getProperty

This tag gets the value of a Bean instance property, converts it to a string, and puts it in the implicit object “out”.

jsp:include

jsp:forward

The 1.1 release will include additional standard tags.

The advantage of tags is that they are easy to use and share between applications. The real power of a tag-based syntax comes with the development of custom tag libraries in which tool vendors or others can create and distribute tags for specific purposes.

Scripting Elements

JSP pages can include small scripts, called scriptlets, in a page. A scriptlet is a code fragment, executed at request time processing that may be combined with static elements on the page (as in the example presented) to create a dynamically-generated page.

Scripts are delineated within <% and %> markers. Anything contained within those markers will be evaluated by the scripting language engine,. In the example, this is the Java virtual machine on the host.

The JavaServer Pages specification supports all of the usual script elements, including expressions and declarations.

Application Models for JSP Pages

A JSP page is executed by a JSP engine, which is installed on a Web server or a JSP technology-enabled application server. The JSP engine receives requests from a client to a JSP page, and generates responses from the JSP page to the client.

JSP pages are typically compiled into Java Servlets. Java Servlets are a standard Java extension, described in more detail at www.java.sun.com. The page developer has access to the complete Java application environment, with all of the scalability and portability of the Java technology-enabled family.

When a JSP page is first called, if it does not yet exist, it is compiled into a Java Servlet class and stored in the server memory. This enables very fast responses for subsequent calls to that page, and avoids the CGI-bin problem of spawning a new process for each HTTP request, or the runtime parsing required by server-side includes.

JSP pages may be included in a number of different application architectures or models and may be used in combination with different protocols, components, and formats. The following sections describe a few of the possibilities.

A Simple Application

In a simple implementation, the browser directly invokes a JSP page, which in turn generates the requested content (perhaps invoking JDBC to get information directly from a database). The JSP page can call JDBC or Java Blend™ components to generate results, and creates standard HTML that it sends back to the browser as a result.

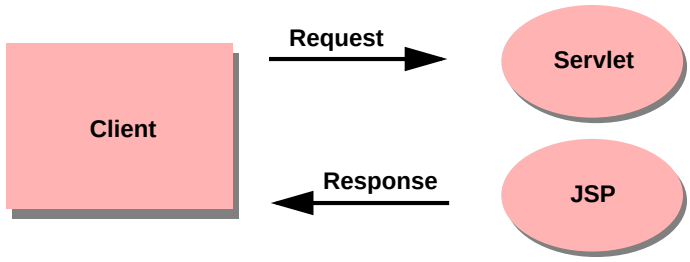
This model basically replaces the CGI-bin concept with a JSP page (compiled as a Java Servlet), and delivers the following advantages:

- It is simple and fast to program.
- The page author can easily generate dynamic content based on the request and state of the resources.

This architecture works well for many applications, but does not scale for a large number of Web-based clients simultaneously accessing scarce enterprise resources. Each client must establish or share a connection to the content resource in question. For example, if the JSP page accesses a database, it may generate many connections to the database, which can affect the database performance.

A Flexible Application with Java Servlets

In another possible configuration, the Web-based client may make a request directly to a Java Servlet, which actually generates the dynamic content, wraps the results into a result bean, and invokes the JSP page. The JSP page accesses the dynamic content from the bean and sends the results (as HTML data) to the browser.



This approach creates more reusable components that can be shared between applications, and may be implemented as part of a larger application. There are still scalability issues in terms of handling connections to enterprise resources, such as databases.

Scalable Processing with Enterprise JavaBeans Technology

The JSP page can also act as a middle tier within an Enterprise JavaBeans architecture. In this case, the JSP page interacts with back-end resources via an Enterprise JavaBeans component.

The Enterprise JavaBeans component manages access to the back-end resources, providing scalable performance for high numbers of concurrent users. For e-commerce or other applications, the Enterprise JavaBeans manages transactions and underlying security. This simplifies the JSP page itself. This model will be supported by the Java 2 Platform, Enterprise Edition.

Integrating XML Technology in JSP Pages

JSP pages can be used to generate both XML and HTML pages.

For simple XML generation, developers can include XML tags and static template portions of the JSP page. For dynamic XML generation, server-based beans and customized tags are used to generate XML output.

JSP pages are not incompatible with XML tools. Although Sun designed the JavaServer Pages specification so that JSP pages would be easy to author, even by hand, the JSP specification also provides a mechanism for creating an XML version of any JSP page. In this way, XML tools can author and manipulate JSP pages.

JSP pages may be used with XML-based tools by converting JSP tags and elements to their XML-compatible equivalents. For example, a scriptlet can be included within <% and %>, or within the XML-based tags <jsp:scriptlet> and </jsp:scriptlet>. In fact, it is possible to convert a JSP page into an XML page by following a few simple steps, including:

- Add a JSP root element
- Convert elements and directives into XML-compatible equivalents
- Create CDATA elements for all other (typically non-JSP) elements on the page

With this XML-compatible alternative approach, page designers creating HTML pages still have an easy-to-use environment for quickly creating dynamic Web pages, while XML-based tools and services can integrate JSP pages and work with JSP technology-compliant servers.

The Future for JSP Technology

JavaServer Pages technology is designed to be an open, extensible standard for building dynamic Web pages. Developers will use JSP pages to create portable Web applications that can run with different Web and application servers for a variety of markets, using whatever authoring tools fit their market and their needs.

By working with a consortium of industry leaders, Sun has ensured that the JSP specification is open and portable. JSP pages may be authored anywhere and deployed them anywhere, using any client and server platforms. Over time, tool vendors and others will extend the functionality of the platform by providing customized tag libraries for specialized functions.

The 1.0 release of the JSP specification is the first step toward this vision of an open, industry-standard method for dynamic Web page generation. The 1.0 release lays the groundwork, with a core set of tags and implicit objects and the basic functionality required to start creating dynamic Web pages. Several Web server, application server, and development tool vendors are adding JSP 1.0 support to their products, so that JSP technology has initial and immediate support within the industry at large.

The 1.1 release, to be completed later in 1999, extends this vision with more extensive XML support, customizable tags, and integration with Java 2 Platform, Enterprise Edition. Vendors may choose to extend and expand the basic, required functionality in the JSP specification. The JSP engine can potentially support multiple scripting languages and object models. As the industry expands and leverages the power of JavaServer Pages technology, Sun is committed to ensuring that JSP technology remains inherently portable across platforms and servers.

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303

1 (800) 786.7638
1.512.434.1511

http://java.sun.com

September 1999