

# Java<sup>™</sup> Servlet and JavaServer Pages<sup>™</sup> Technology

---

*Comparing Methods for Server-Side Dynamic  
Content*



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

Copyright 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Java, JavaBeans, Enterprise JavaBeans, EJB, JavaServer Pages, Java Community Process, JDBC, JDK, JavaMail, Java Naming and Directory Interface, JVM, J2EE, and Write Once, Run Anywhere are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

**RESTRICTED RIGHTS:** Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

**DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.**

Copyright 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, Californie 94303 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, Java, JavaBeans, Enterprise JavaBeans, EJB, JavaServer Pages, Java Community Process, JDBC, JDK, JavaMail, Java Naming and Directory Interface, JVM, J2EE, et Write Once, Run Anywhere sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



Please  
Recycle



Adobe PostScript

## Introduction

---

As the Web has evolved, it has changed from a collection of static pages to an environment filled with constantly changing personalized content. The challenge for Web developers has been to dynamically generate content for Web pages. Many approaches have been tried to solve this problem, some easier to use than others. For example, Web server-based APIs are powerful solutions tied to individual Web servers, however, their complexity restricts the usefulness of these APIs.

To avoid becoming overly dependent on a single vendor, organizations often prefer solutions that have broad industry acceptance or are open source solutions. Popular choices today include common gateway interface (CGI) programs, `mod_perl` programs, and PHP for page authoring. These methods correct many of the limitations of earlier approaches.

This white paper compares CGI, `mod_perl`, and PHP solutions with the Java™ Servlet and JavaServer Pages™ (JSP) technologies for creating dynamic content with Java™ technology. Java Servlet and JSP technology are products of the Java Community Process™, an open process used by Sun Microsystems since 1995 to develop and revise Java technology and specifications in cooperation with the international Java technology community.

From both the server-side processing and page authoring perspectives, this paper considers and compares the alternatives. However, it does not provide an exhaustive comparison nor a comprehensive survey of page generation alternatives.

Individual preferences also affect which solutions are selected. Perl developers often have a strong preference for Perl-based solutions. Rather than address individual preferences, this paper assesses the broader requirements of an organization's platform strategy, such as ongoing maintenance, portability, and security.

# Comparing Server Mechanisms for Dynamic Content

---

## Java Servlet, CGI, and mod\_perl Programs

This section compares the server-side processing mechanisms of CGI scripts, the mod\_perl Apache module, and Java Servlet APIs.

CGI programs allow the Web server to get information from other applications before responding to the browser. CGI programs are usually written in the C or Perl languages. The mod\_perl plug-in for the Apache Web Server integrates Perl code with the Web server, so programmers can write extensions in Perl. Many developers use mod\_perl as a replacement for the CGI interface because it addresses some of CGI's limitations. Java Servlet extensions are server-side components. And JSP components are compiled automatically into Servlets, making them the server-side mechanism for JSP technology.

Each solution enjoys a loyal following. This paper compares these alternatives across several broad dimensions: portability; performance; ease of development, deployment, and maintenance; and security.

## Portability

A portable application environment gives organizations the flexibility to migrate servers and swap tools as business needs change. Portability also enables developers to share their work with a wider audience.

The CGI interface is supported on a wide variety of Web servers. However, CGI programs themselves are not inherently portable across platforms, so careful design is required to construct portable CGI applications.

By definition, the `mod_perl` solution is an Apache Web Server module, restricted to the Apache Web Server. This is an open source application, providing a higher comfort level than committing to a vendor-specific solution, but still restricting the choice of Web servers. And these programs are not easily portable to different platforms.

Java Servlet programs adhere to the Write Once, Run Anywhere™ philosophy of the Java family of technologies. Servlets run on any Web server, on any platform, and may access portable Java components or classes. Sun released the Java Servlet and JSP source code to the Apache and Java technology developer community, ensuring the availability of a free, open-source engine and supporting tight integration with the Apache Web Server. Most major Web and application servers support Java Servlet APIs as well.

## Performance

Everyone hopes their Web site will grow in popularity, so it's best to start with scalability in mind. Performance becomes an issue when the number of concurrent users accessing the Web page increases.

For high-volume sites, performance is the CGI application's fatal flaw. In a traditional CGI environment, the Web server spawns a new process every time a client requests a CGI application, loading and executing a Perl interpreter for each request. As the number of concurrent users grows, performance slows.

Both `mod_perl` and Java Servlet pages avoid this limitation. Each program is compiled once, remaining in memory for subsequent requests. Servlets also make it easier to write multithreaded applications than traditional Perl and `mod_perl` programs. Most Java™ virtual machine (JVM™) implementations automatically take advantage of additional processors if they are available.

## Ease of Development and Deployment

By supporting all of the Java language's benefits, Java Servlet programs make application development easier. For example, programs written in the Java language do not experience memory access violations that can lead to server crashes. Java technology also simplifies exception handling. Other benefits can be traced to the fact that Java is an inherently object-oriented programming language, offering better support for team development than Perl's traditional scripting approach.

A few specific examples of the differences between the Java technology-based and Perl language approaches in development and deployment follow.

## *Sharing State and Maintaining State Between Requests*

By definition, the HTTP protocol is stateless, which causes problems for Web-based applications. Programmers must use techniques such as cookies or URL rewriting to determine session state. With both CGI and mod\_perl, sessions must be implemented manually using these techniques.

The Java Servlet solution defines a session interface by which the servlet container (the servlet-enabled Web server) tracks session state. This makes it much easier to build interactive, Web-based applications.

## *Database Connections*

Both CGI and mod\_perl applications rely on relatively ad-hoc database interfaces. For example, using an Oracle-specific configuration and setup to access data in an Oracle database makes it difficult to add database sources using different database engines or switch sources without reworking the application platform.

Servlets access databases through the standard JDBC™ interface, which is supported by all major database vendors. There is no database-specific configuration and the developer can easily swap or add database sources without rebuilding applications. In addition, servlets support persistent database connections, providing faster database access and better use of enterprise resources.

## *Type Safety*

Perl is a flexible, powerful programming language. Experienced programmers can use its open structure to come up with creative solutions. This is referred to as “doing magic” with Perl. The flip side of this flexibility is that in less-experienced hands, unintended side effects may occur.

In Perl, passing the wrong kind of argument creates problems that may not show up until runtime. However, the Java language has inherent type safety. All values are treated individually and strongly typed, so type errors are discovered during compilation, not runtime. The result is a more predictable development environment.

## *Development Tools*

Many integrated development environments (IDEs) support the Java language and Java Servlet technology, including a number of visual development tools. These can help speed technology adoption and application development time.

## *Deployment*

For CGI and mod\_perl applications, the process of actually installing and deploying the application is manual and time-consuming, involving several administrative tasks on the Web server.

The Java Servlet 2.2 interface speeds deployment by providing a truly portable deployment mechanism. A Web application archive (.war) file packages the application for installation on Java Servlet 2.2 technology-compliant Web servers. This eliminates the server-specific installation processes, so developers can write and test applications on a laptop, then deploy in production quickly and accurately. It also makes it easier for developers to package and distribute applications.

## *APIs and Networking Interfaces*

Mature solutions need a wide range of supported interfaces to access enterprise sources. Both Perl and Java languages have extensive, growing libraries of classes and APIs. Servlets have access to the complete set of Java APIs and components, including:

- Java™ Transaction Service
- Java Naming and Directory Interface™
- Java™ Message Service
- JavaMail™ API
- Enterprise JavaBeans™ (EJB™) and EJB Containers
- JDBC

## *Ease of Maintenance*

Time spent maintaining an application contributes to its ongoing cost. Significant differences exist between Perl applications and the Java Servlet technology-based solution. Many differences are due to the nature of the Java language. As an inherently object-oriented language, Java is simpler to maintain than Perl.

For most developers, it is easier to read someone else's Java code and figure out what they have done than to read a Perl script. Java portable application components (utility classes or JavaBeans™) that perform specific tasks can be reused. And developers can create customized JSP tag libraries to distribute reusable functions to page authors. Java technology has fewer unexpected or hard-to-discover side effects that may be obvious to the script's author but unclear to the person who maintains the script.

## Security

For ISPs and others hosting Web-based applications, key security issues include:

- Can the server run untrusted code and prevent it from accessing other data on the server?
- Is it possible to limit damage to the server caused by the application (wasting memory or CPU cycles, system crash, etc.)?

All of the solutions address the first measure of security. For servlets, the Java virtual machine (JVM) can restrict the environment available to the servlet so it cannot access other critical data from the Web server. Perl's taint checking mechanism is used to detect potential security problems in Web-based input. And administrators may use operating system-level controls (such as the suEXEC in UNIX<sup>®</sup>) to restrict a program's access.

However, the JVM provides a greater degree of control over application behavior than the Perl or CGI approaches. For example, the JVM can control whether an application is able to open or listen to a socket, a level of control not available from the operating system. With the JDK<sup>™</sup> 1.1 software implementation, these controls are applied at a global level. With Java<sup>™</sup> 2 SDK, Standard Edition v1.2, these controls apply per servlet.

Feature	CGI/Perl	mod_perl	JSP/Servlet
Portable across Web servers	Yes	No	Yes
Portable across hardware/OS platforms	No	No	Yes
Runs multiple concurrent sessions with-out spawning separate processes for each	No	Yes	Yes
Protects against memory leaks	Yes	No	Yes
Scripting language	C, Perl	Perl	Java
Type safety	Loosely typed	Loosely typed	Strongly typed; all values treated individually
Maintains state between requests	No	No	Yes
Maintains persistent database connections	No	Yes	Yes
Portable database interfaces	Ad hoc	Ad hoc	Yes, JDBC
Runs entrusted code safely	No	No	Yes

<b>Feature</b>	<b>CGI/Perl</b>	<b>mod_perl</b>	<b>JSP/Servlet</b>
<b>Ability to limit program capabilities</b>	Limited (OS level)	Limited (OS level)	Extensive with Java 2 SDK, Standard Edition v1.2 on a per-Servlet basis
<b>Team development</b>	Difficult	Difficult	Easy
<b>Ongoing maintenance</b>	Difficult	Difficult	Easy
<b>Deployment</b>	Cumbersome	Cumbersome	Easy with .war files (Java Servlet 2.2)
<b>IDE support</b>	No	No	Yes

**TABLE 1** Server-side Generation Comparison

## Comparing Page Templating Systems

---

### JSP Technology, EmbPerl, and PHP

Writing `mod_perl` scripts or Java Servlet programs are tasks best-suited to programmers with Perl or Java language experience. These tasks are not appropriate for the average, HTML-literate page author. Fortunately, a number of approaches are available to simplify the page templating process. This section compares three: JSP, EmbPerl, and PHP.

JSP technology extends the Java Servlet approach by encapsulating application logic within the XML or HTML page itself using JSP tags that call reusable Java components such as servlets and JavaBeans components. Pages created with JSP technology are automatically compiled into servlets when they are invoked.

EmbPerl processes Perl code embedded in an HTML document. To simplify the process of creating `mod_perl` pages, including static HTML templates, EmbPerl can run under `mod_perl` and Apache. It may also run as a standard CGI interface with other Web servers.

PHP is an open-source, server-side scripting language for creating dynamic Web pages. It uses a syntax similar to Perl or C, but is a distinct scripting language that runs embedded in Apache and works with other Web servers as well. PHP code appears in the HTML pages embedded within simple delimiters.

## Easier Page Authoring

Ideally, a page templating system should separate the page format and design (the template) from the application logic. The greater the separation, the easier it is to support a tiered development approach: application developers work on the application logic and page authors build the pages that call this logic. All three solutions considered here provide some degree of content/logic separation

With both EmbPerl and PHP, developers write scripts within the HTML page that control the application logic. Scripting in Perl or PHP requires some programming expertise, so both of these approaches still require a developer to create the dynamic pages.

Pages created with JSP technology can call external logic components (JavaBeans or custom JSP tags) that encapsulate the application logic. The page author need only know how to call the component and what arguments to pass — no programming skills are required. JSP also supports Java scriptlets for implementing very simple logic or tying together different parts of the page with conditional processing. The result is that an average Web developer can read and understand a JSP page without difficulty and may easily write pages that call the necessary application logic.

## Extensibility

Custom JSP tags provide built-in extensibility to Web pages. Developers and tool vendors may write and distribute custom tags to perform specific functions, while page authors invoke the tag with simple, HTML-like calls. Behind the scenes, the tags access enterprise resources, make transactions, or perform complex processing. With this mechanism, JSP technology supports the emergence of more specialized tools and solutions that will further speed application deployment. The other solutions do not provide this type of methodology for packaging and distributing extensions to the template system.

## Application Maintenance

The separation of content and logic in JSP technology enables page authors to make design or content changes that do not affect application logic and application developers to enhance functionality without changing every page. Changes to a page design do not require a developer's time to implement or recompile. With EmbPerl and PHP, a programmer must make the application changes.

## Tool Support

The JSP technology is part of the Java™ 2 Platform, Enterprise Edition (J2EE™), and as such enjoys widespread support from third-party tool vendors. This means there will be a growing number of development tools supporting JSP technology, making JSP adoption and application development easier and faster. EmbPerl and PHP lack a broad base of tool support.

Feature	EmbPerl	PHP	JSP
<b>Separation between content and logic (enabling tiered development)</b>	Partial	Partial	Full
<b>Target user</b>	Programmer	Programmer	Web developer, page author
<b>Industry-wide support from third-party tool vendors</b>	No	No	Yes
<b>Language in page</b>	Perl	Perl	Java
<b>Extensible through third-party components</b>	No	No	Yes

TABLE 1 Page Template System Comparison

## Conclusion

---

Java Servlet and JSP technologies represent the latest entries in a long line of solutions for creating dynamic content on the Web. Pages created with JSP technology extend the power of Java Servlet extensions, opening dynamic page generation capabilities to a wider variety of page authors. Together, the JSP and Java Servlet solution solves many of the problems of earlier techniques by providing:

- Platform- and server-independent methods
- Portable, reusable logic components
- High performance for multiple concurrent requests
- Easy deployment
- Easy maintenance

The best solution for an organization depends on short-term and long-term needs. If a number of Perl experts are available to work on Web-based applications, using a CGI or mod\_perl approach may help meet short-term goals.

If ongoing maintenance of applications is a concern, and the organization wants to expand Web-based development throughout the enterprise, the JSP and Java Servlet solution offers considerable long-term benefits. And because these technologies are products of the Java Community Process — a cooperative development process involving the international Java technology community — a broad base of industry support in the future is assured.



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

1 (800) 786.7638  
1.512.434.1511

<http://java.sun.com>

January 2000