

# QuickStart Tool

## JavaServer Pages (JSP)

Getting a good start in any new technology or programming language often depends on finding the best available information. The Builder.com QuickStart Tools give you the information you need to quickly grasp the fundamentals of developing in a new IDE, using a new programming language, or working with a new development tool.

JavaServer Pages (JSP) is a server-side scripting technology that uses the Java programming language to write dynamic content. JSP version 1.x gives developers the tools to build platform-independent Web-based applications. Besides explaining the basics, this Builder.com QuickStart Tool shows common tasks, exposes strengths and weaknesses, demonstrates some of the best uses of JSP, and lists a variety of other online and offline resources that can help you build a solid foundation of practical knowledge.

## Table of contents

<b>Fundamentals .....</b>	<b>3</b>
<b>Common tasks .....</b>	<b>5</b>
<b>Strengths .....</b>	<b>7</b>
<b>Weaknesses .....</b>	<b>7</b>
<b>Best uses .....</b>	<b>8</b>
<b>Online resources.....</b>	<b>8</b>
<b>Other resources .....</b>	<b>9</b>
<b>Additional information .....</b>	<b>9</b>
<b>About Builder.com .....</b>	<b>10</b>

## Fundamentals

JavaServer Pages (JSP) is a server-side scripting technology that uses the Java programming language to write dynamic content. This content is written within tags and interpreted by an application server. When the page is accessed for the first time, pages are compiled into servlets, which deliver server-side dynamic content. JSP version 1.x was developed by Sun Microsystems as an extension of the Java Servlet technology. Sun Microsystems has made JSP specifications available to all developers free of cost as a component of the Java 2 Platform, Enterprise Edition ([J2EE](#)). It is useful for all developers and programmers who want to create dynamic Web pages for Web sites or portals.

### Hello World program

The recommended file extension for a JSP source file is .jsp. A JSP page can be written in any text editor. For this example, we will create the standard “Hello World” program, using JSP version 1.x.

#### Begin

Launch your favorite text editor and type the code shown in the following sections using scriptlet, declaration, and expression tags.

#### Using the scriptlet tag

```
<html>
<head>
<title>Hello World Program</title>
</head>
<body>

<h1>
<% out.println("Hello World") %>
</h1>

</body>
</html>
```

In the above code, JSP’s scriptlet tag ( <% and %> ) is used for the “Hello World” program.

#### Using the declaration tag

```
<html>
<head>
<title>Hello World Program</title>
</head>
<body>

<%!
String msg = "Hello World";
%>

<h1>
<% out.println(msg) %>
</h1>

</body>
</html>
```

In the above code listing, the declaration tag ( <%! and %> ) is used to declare a string object (or variable) with the value “Hello World”. Next, the string value is displayed using the scriptlet tag.

### Using the expression tag

```
<html>
<head>
<title>Hello World Program</title>
</head>
<body>
<%!
  String msg = "Hello World";
  %>

<h1>
<%=msg %>
</h1>

</body>
</html>
```

In the above code listing, the declaration tag is used to declare a string object (or variable) with the value "Hello World". The string value is displayed using the expression tag (<%= and %>) instead of the scriptlet tag.

### Running your program

Save one of the above code examples as a file named hello.jsp. The Java Development Kit (JDK) and a Java Application Server (e.g., Tomcat) are required to run the program. You can download the JDK from Sun and Tomcat from [Jakarta](#). Copy the file hello.jsp to the webapps\ROOT subdirectory of the Tomcat installation directory. Open a Web browser and enter the URL <http://localhost:8080/hello.jsp>. The browser will display "Hello World".

## Common tasks

Common tasks refer to actions that are frequently performed by developers. Listed below are some of the common tasks associated with JSP version 1.x:

### Task

Connecting to a MySQL database

### Steps

JSP connects to a MySQL database using the MySQL **Connector/J** drivers.

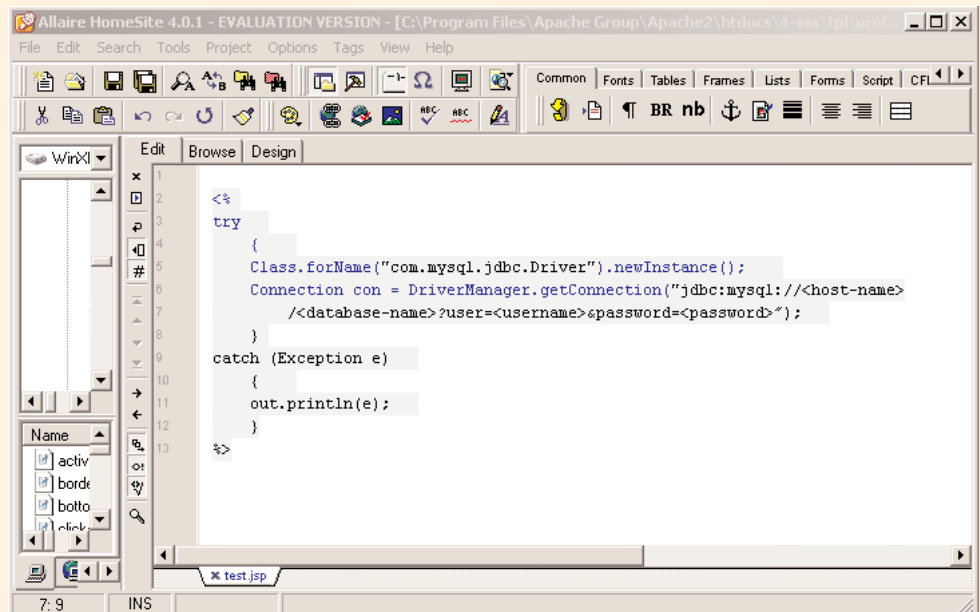
```
<%
try
{
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    Connection con = DriverManager.getConnection
        ("jdbc:mysql://<host-name>/<database-name>?user=<username>
        &password=<password>")
    }
catch (Exception e)
{
    out.println(e);
}
%>
```

The JDBC driver to load is `com.mysql.jdbc.Driver`. The connection string is in the format:

```
jdbc:mysql://<host-name>/<database-name>?user=<username>
&password=<password>
```

The `<host-name>` is the name or IP address of the machine running the MySQL database server. The `<database-name>` is the name of the database to use, and `<username>` and `<password>` are the credentials required to connect to the database. Note that database connections may throw exceptions, so a try/catch block is included.

**Figure A**



Connecting to a MySQL database

## Using a JavaBean

JSP can take advantage of a JavaBean by using the **useBean** tag as follows:

```
<jsp:useBean id="bean" class="JavaBean" />
```

Here "bean" is the designated identity and "JavaBean" is the name of the class file that contains the compiled code for the JavaBean.

## Setting JavaBean properties

The properties of a JavaBean can be set within JSP using the **setProperty** tag. Suppose the JavaBean has a property named "var" and it has to be set to a value of 1. This is accomplished as follows:

```
<jsp:setProperty name="bean" property="var" value="1" />
```

## Accessing session data

An implicit object called **session** can store, retrieve, or remove objects in an HTTP session.

To store an integer value of 25 in the session:

```
session.setAttribute("sessionVar", new Integer(25));
```

To retrieve a value stored in the session:

```
Integer sessionVar = (Integer)session.getAttribute("sessionVar");
```

To delete the value stored in the session:

```
session.removeAttribute("sessionVar");
```

## Include external files

The JSP **include** tag can be used to include static HTML files:

```
<jsp:include page="header.html" />
```

It can also be used to include dynamic content, such as another JSP page:

```
<jsp:include page="footer.jsp" />
```

This capability is commonly used to insert consistent headers and footers across a Web site's pages.

## Accessing HTML form data

Data posted to a JSP version 1.x file from an HTML form can be accessed using the **request** object. For example, the data typed in an HTML form's textfield named "text" can be accessed as:

```
request.getParameter("text");
```

## Strengths

Strength	Description
Powerful technology for display of static and dynamic content	JSP is used for developing static and dynamic Web pages. JSP uses text-based documents containing tags and constructs to describe how to process a request and construct a response. A JSP Web page also combines static tags using HTML and dynamic coding using complex methods for accessing Java Database Connectivity (JDBC) objects or Remote Method Invocation (RMI) objects.
Enhanced performance	JSP version 1.x pages are compiled in the Java servlet class. These classes remain in the server memory for a longer duration; therefore, if the page is accessed again, recompilation is not required, which saves time and enhances application performance.
Extensible	JSP uses customized tag libraries, which provide an advantage to developers because they can work with familiar tools and constructs.
Offers “write once, run anywhere” capability	JSP takes advantages of code reuse, because the same code is usable on different platforms.
Easy integration with HTML	JSP scriptlets integrate with HTML code to create dynamic Web pages.
Supported by all Web servers	JSP can be executed on any Web server. All major Web servers, including Apache and IIS, support JSP version 1.x.

## Weaknesses

Weakness	Description
Preliminary knowledge of Java required	A developer working efficiently in JSP must have knowledge of Java technology.
Tags not rendered	Various JSP authoring tools are supplied by vendors such as Macromedia. However, in this process, sometimes JSP version 1.x tags, or customized tag libraries created in an authoring tool, are not recognizable by different authoring tool vendors. This inconsistency sometimes results in non-rendered JSP tags.
Underdeveloped debugging tool	The debugging tools for JSP version 1.x are not mature enough to help the developers debug the code.
Shortage of full-service authoring tools	Many authoring tools for JSP do not offer complete tag library support.
No common language for creative and technical teams	In an environment with separate creative and technical teams, the creative team may not necessarily know JSP or Java syntax, and the technical team may have problems implementing the HTML files delivered by the creative team. Both teams lack a common language to describe their requirements for the rendering of each page in a JSP environment.
Less efficient when compared to desktop applications	JSP-based Internet applications are not as efficient as traditional desktop applications because the current software architecture supporting these applications can only receive real-time data by repeatedly refreshing the page. This repetition reduces the overall performance and scalability of the application.

## Best uses

### Programmers with existing Java skills

JSP version 1.x uses the Java programming language to write dynamic content. Therefore, development environments with programmers who possess a thorough understanding and knowledge of the Java programming language and Java servlets will be best served by JSP.

### Separate programming logic from design issues

Because it uses both static HTML tags and dynamic components created with JavaBeans and RMI, JSP is most useful in projects that require developers to separate programming logic issues using EJB and RMI from the layout issues of HTML tags.

### Programmers looking to deploy software applications on the Internet

Because JSP is platform independent and renders dynamic components at the Web server, one of its best uses is for efficiently deploying Internet applications.

## Online resources

### [The JavaServer Pages™\(JSP™\) 1.2 Specification Tag Extensions: Design Patterns and Web Architectures](#)

This PDF details the JSP tag specifications.

### [Overview of Servlets and JavaServer Pages](#)

What are JavaServer Pages, when and why JSP should be used, and software installation and setup are just some of the topics covered in this PDF.

### [Web Application Development with JSP and XML Part I: Fast Track JSP](#)

JSP allows you to separate front-end presentation from business logic (middle and back-end tiers). It is a great Rapid Application Development (RAD) approach to Web applications. This white paper provides a hands-on tutorial explaining how to develop modern Web applications.

### [JSP Design Notes](#)

This white paper reviews client-server design considerations with respect to the use of JSP.

### [JSP Translation and Deployment](#)

This site discusses the operation of the Oracle JSP translator.

### [An Introduction to JSP Custom Tags](#)

This white paper describes a JSP model in which requests are submitted to a servlet controller that performs business logic and generates an appropriate data model to be displayed.

## Other resources

### [Core JSP](#)

By Damon Hougland (Author), Aaron Tavistock (Author), 2000, ISBN: 0-13-088248-8

### [JavaServer Pages](#)

By Larne Pekowsky (Author), 2000, ISBN: 0-201-70421-8

### [Java Servlet & JSP Cookbook](#)

By Bruce W. Perry (Author), 2004, ISBN: 0-596-00572-5

### [JavaServer Pages, 3rd Edition](#)

By Hans Bergsten (Author), 2000, ISBN: 0-596-00563-6

## Additional information

### [JavaServer Pages \(JSP\) 1.0](#)

### [Facilitate database access in your JSP application design with JDBC](#)

### [Implementing a form in a JSP Page](#)

### [Intro to JavaServer Pages: Learn the syntax basics](#)

### [Intro to JavaServer Pages: Create a JSP site](#)

### [Intro to JavaServer Pages: Utilize the implicit object set](#)

### [Template taglib ver. 1.3](#)

## About Builder.com

Thank you for downloading this Builder.com tool; we hope it helps you make better technology decisions. If you need help with the site or just want to add your comments, [let us know](#).

Builder.com provides actionable information, tools, trialware, and services to help Web and software developers get their jobs done. Builder.com serves the needs of developers on all major development platforms, including .NET and J2EE, and supplies the knowledge and tools every developer needs to build better and more robust applications.

**[Free how-to articles:](#)** Covering everything from open source software to the latest .NET technologies, Builder articles give developers the breadth and depth of coverage they need to build better applications.

**[Free e-newsletters:](#)** Keep up to date on any aspect of the developer industry—from Web services to Visual Basic—with Builder's e-newsletters, delivered right to your e-mail inbox.

**[Free trialware:](#)** We've collected all the latest trialware and free downloads—covering everything from application servers to HTML editors—to make your job easier.

**[Discussion Center:](#)** Open a discussion thread on any article or column, or jump into preselected topics, such as Java, HTML, and career management. The fully searchable Discussion Center brings you the hottest discussions and threads.

**[Your free Builder membership](#)** opens the door to a wealth of information. Our online developer community provides real-world solutions, the latest how-to articles, and discussions affecting developers everywhere. Get access to full-text books, exclusive downloads, and posting privileges to our discussion boards, all free!