



# Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

# Java Server Pages (JSP) Support Guide



# Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

## General Information

### What is JSP?

From: JSP 101: Introduction to JavaServer Pages by JJ Kuslich - [http://developer.netscape.com/viewsource/kuslich\\_jsp/kuslich\\_jsp.html](http://developer.netscape.com/viewsource/kuslich_jsp/kuslich_jsp.html)

### JSP Basics

Java Server Pages (JSP) technology allows the combination of regular, static HTML and XML and Java code to produce dynamic, portable, scalable and easily maintained pages.

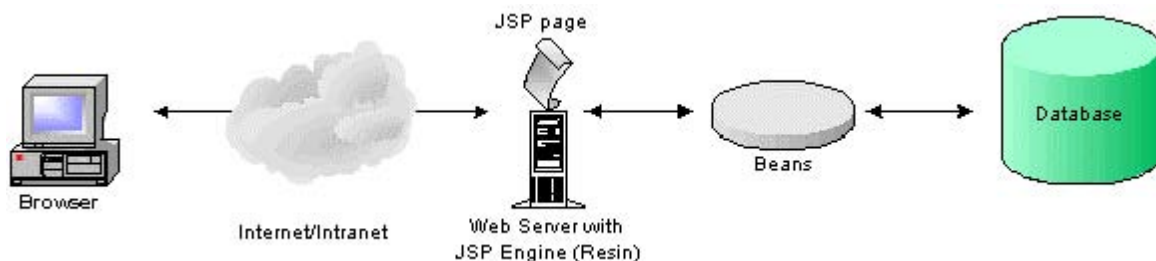
JSP technology acts as a top layer above Java servlets on the servlet engine. In this way, developers can combine HTML code with server-side scripting to create dynamic pages.

### Incorporating Components

JSP is based on components. It uses JavaBeans and Enterprise JavaBeans (EJB) components which hold the business logic for an application, and it composes tags and a scripting platform for presenting the content generated and returned by the beans into HTML pages. The components structure means that non-Java developers can use JSP to manipulate beans that developers have built. On the other hand, Java developers can use these beans and Java in JSP pages for more advanced productions based on the beans.

### Request Model

The following model is based upon a diagram from the following page [http://developer.netscape.com/viewsource/kuslich\\_jsp/kuslich\\_jsp.html](http://developer.netscape.com/viewsource/kuslich_jsp/kuslich_jsp.html). It follows the path of a JSP request. A user requests a page through a browser. The browser sends out the request for a JSP page. The request goes via a web server which contains a JSP engine (for example, Resin). Resin compiles the servlet. The servlet goes through the Beans, to a database to retrieve the relevant information. Once it collects the information from the database, it travels back to the web server, where this information is collated with the servlet information into the correct format and is sent back to the browser as a JSP page.



What is really happening in this process? A user calls a JSP page through their browser, so Resin (for instance) located on the web server, converts JavaScript, JSP tags, and HTML into segments of Java code, which the engine then consequentially arranges into a core Java servlet. This core servlet is therefore pre-assembled and works 'behind the scenes'. It is called every time that particular page is requested by a user, saving valuable recompiling time. The next time a user calls that page from their browser, because the JSP code has already been compiled, it does not have to convert the JavaScript,



# Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

HTML, etc each time. That is, the servlet engine needs only to produce that servlet once, or after the last code change was updated. In this example, the JSP Engine, Resin acts to eliminate duplicate effort

[http://www.caucho.com/articles/resin\\_js.xtp](http://www.caucho.com/articles/resin_js.xtp)

## What engine is Web Secure using?

Web Secure is using Resin.

## What special features does Resin have?

◆ Resin provides fast response times, as the faster that pages load the more likely customers are to look at your entire site and to make a return visit. ◆ With fast loading pages, the traffic to your site will in turn grow, and then Resin has the capacity to allow your site to expand accordingly.

◆ Resin promotes the use of XSL (XML Stylesheet Language) which supports the split between servlets and JSP, that is content and style. It utilises a servlet filtering process to convert XML through the use of XSL stylesheets.

◆ The following servlet specification is supported by Resin, servlet 2.3.

◆ DB Pooling

## What features does Resin support?

From: Caucho Technology Products - <http://www.caucho.com/products/index.xtp>

Resin supports the following features:

### Resin Core

- ◆ JSP 1.2 support
- ◆ Fast Servlet 2.3 engine
- ◆ Automatic Servlet and Bean compilation
- ◆ Distributed Sessions
- ◆ Caching
- ◆ XML Application Server
- ◆ EJB client for third-party EJB servers
- ◆ SSL Security

### Serif (XTP)

- ◆ Compiles XML to JSP
- ◆ Combines JSP performance with stylesheet flexibility



# Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

- ◆ Separates content from formatting
- ◆ Configures sites easily for different clients (e.g., Palm Pilots)
- ◆ XSL filtering

## Resin-XSL

- ◆ Xalan 2.2 D14
- ◆ Saxon 6.5
- ◆ XT
- ◆ Supports standalone XPath API
- ◆ Simplifies XSL syntax with StyleScript

## Resin-XML

- ◆ Parses XML or HTML
- ◆ Parses XML or HTML Supports SAX 2.0, DOM 2.0, and JAXP 1.0
- ◆ Flexible Virtual File System
- ◆ Perl 5.x Regular Expressions

## Resin-JavaScript

- ◆ Supports JavaScript 1.3
- ◆ Supports ECMA-262
- ◆ Compiles to Java
- ◆ Full access to Java libraries
- ◆ Object oriented classes
- ◆ Reliability with exceptions, synchronization, and import

## What are the advantages of JSP?

From: Servlets and JavaServer Pages (JSP) 1.0: A Tutorial - <http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/>

◆ **vs. Active Server Pages (ASP).** ASP is a similar technology from Microsoft. The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS-specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.

◆ **vs. Pure Servlets.** JSP doesn't give you anything that you couldn't in principle do with a servlet. But it is more convenient to write (and to modify!) regular HTML than to have a zillion println statements that generate the HTML. Plus, by separating the look from the content you can put different people on



# Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

different tasks: your Web page design experts can build the HTML, leaving places for your servlet programmers to insert the dynamic content.

◆• **vs. Server-Side Includes (SSI).** SSI is a widely-supported technology for including externally-defined pieces into a static Web page. JSP is better because it lets you use servlets instead of a separate program to generate that dynamic part. Besides, SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.

◆• **vs. JavaScript.** JavaScript can generate HTML dynamically on the client. This is a useful capability, but only handles situations where the dynamic information is based on the client's environment. With the exception of cookies, HTTP and form submission data is not available to JavaScript. And, since it runs on the client, JavaScript can't access server-side resources like databases, catalogs, pricing information, and the like.

◆• **vs. Static HTML.** Regular HTML, of course, cannot contain dynamic information. JSP is so easy and convenient that it is quite feasible to augment HTML pages that only benefit marginally by the insertion of small amounts of dynamic data. Previously, the cost of using dynamic data would preclude its use in all but the most valuable instances

## Java and Servlet Versions Supported

### *JSP Version Supported*

Web Secure supports JSP version 1.2.

### *Servlet Version*

Web Secure supports servlets written to the 2.3 version of the servlet specification.

### *JDK Version*

Web Secure uses a JDK version 1.4.2 compliant VM, servlets and JSP are compiled using jikes version 1.15 with optimisation enabled. Jikes can compile JSP and servlets up to three times faster than legacy Java compilers.

## Database Specifications

### **Which Databases Does Web Secure Support?**

Currently we are supporting both MySQL and PostgreSQL databases. See below for connectivity information.

### **Connecting to a Database**

Tutorials are currently being written on PostgreSQL, and can be found at: <http://jdbc.postgresql.org/>

From: Documentation for MM.MySQL - Programming Information - <http://mmysql.sourceforge.net/doc/mm.doc/c106.htm#AEN118>



# Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

To begin working with JDBC, a Connection Object needs to be acquired. The Connection Object symbolises the network connection to the database that holds the data to be manipulated.

When using JDBC, the generation of connections is created by the DriverManager, which needs to be instructed about which JDBC drivers it should connect to. This can be done simply by using Class.forName() on the Class that instigates the interface for the java.sql.Driver. For example MM.MySQL uses the Class called org.gjt.mm.mysql.Driver.

Code Example - Registering the Driver with the DriverManager and Obtaining a Connection from the DriverManager\*

```
import java.sql.*;

// Notice, do not import org.gjt.mm.mysql.* // or you will have problems!

public class LoadDriver

{ public static void main(String[] Args) {

try { // The newInstance() call is a work around for some // broken Java implementations

Class.forName("org.gjt.mm.mysql.Driver").newInstance(); [use this code for MySQL]
Class.forName("org.postgresql.Driver").newInstance(); [use this code for PostgreSQL]

}
catch (Exception E) {

System.err.println("Unable to load driver."); E.printStackTrace();

try {

Connection C = DriverManager.getConnection( "jdbc:mysql://caelum.Web
Secure.com.au/database_com_au? user=user_name&password=password");** [use this code for
MySQL]

Connection C = CreateConnection( "jdbc:postgresql://caelum.Web
Secure.com.au/database_com_au? user=user_name&password=password");** [use this code for
PostgreSQL]

// Do something with the Connection

....

}

catch (SQLException E) { System.out.println("SQLException: " + E.getMessage());
System.out.println("SQLState: " + E.getSQLState()); System.out.println("VendorError: " +
E.getErrorCode());

}

}
```



# Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

\* Note that in this example the **red** text denotes MySQL and the **green** text depicts PostgreSQL. \*\*  
Your username and password are available from Web Secure Technical Support at [support@WebSecure.au.com](mailto:support@WebSecure.au.com).

It is important to register the driver with the DriverManager, and to acquire a URL for the database using the following syntax:

**jdbc:mysql://caelum.Web Secure.com.au/database\_com\_au?**  
**user=user\_name&password=password** [use this code for MySQL]

**jdbc:postgresql://caelum.Web Secure.com.au/database\_com\_au?**  
**user=user\_name&password=password**

[use this code for PostgreSQL]

When the URL is set up and identified, it is forwarded to the DriverManager.getConnection() method so that a Connection Object can be attained.

For more information on JDBC, please visit the following sites: Sun's JDBC Technology -  
<http://www.javasoft.com/products/jdbc/>

Documentation for MM.MySQL - Programming Information

<http://mmyysql.sourceforge.net/doc/mm.doc/c106.htm#AEN118> PostgreSQL -  
<http://jdbc.postgresql.org/>

## Database Pooling

From: Resin Tutorial - [http://www.caucho.com/products/resin/java\\_tut/index.xtp](http://www.caucho.com/products/resin/java_tut/index.xtp)

If you would like this to be set up, please contact Web Secure's Technical Support Team at [support@Web Secure.au.com](mailto:support@WebSecure.au.com) for assistance.

Database Pooling allows a multitude of users to access a database at a single time. This provides great connectivity than a shared connection which allows only one user to connect to a database at one time, and therefore restricts other users.

Web Secure is using Resin as our web server, which provides an interface for a database connection pool if the Java DataBase Connection (JDBC) protocols are followed. In fact, Resin's configuration for databases connects together configuration from a user's application and places it into the configuration files where it should be. This configuration allows the user to make an unlimited number of named database pools [http://www.caucho.com/products/resin/java\\_tut/db-config.xtp](http://www.caucho.com/products/resin/java_tut/db-config.xtp).

Example from Resin Tutorials - Storing a Survey in a Database.

```
<%@ page language=java %>
<%@ page import='java.sql.*' %>
<%@ page import='javax.sql.*' %>
<%@ page import='javax.naming.*' %>
```



# Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

```
<%  
Context env = (Context) new InitialContext().lookup("java:comp/env");  
  
DataSource source = (DataSource) env.lookup("jdbc/your_domain_name");  
  
Connection conn = source.getConnection();  
  
try {  
  
Statement stmt = conn.createStatement();  
  
String name = request.getParameter("name");  
String color = request.getParameter("color");  
  
stmt.executeUpdate(  
"insert into COLORS values (" +  
"" + name + ", " + color + ")"  
);  
  
} finally { conn.close(); }  
  
%>  
  
<h1>Thank you, <%= name %></h1>
```

The standard Java API to acquire new database connections is DataSource.

Ensure that the database connection is closed, so that it automatically returns to the pool. Using the **try** through to **finally** syntax will guarantee that the connections are returned to the database connection pool.

## Secure Pages - SSL Information

Redirect to [https://secure-jsp.Web Secure.net/your\\_domain\\_name](https://secure-jsp.Web Secure.net/your_domain_name). This is available only with certain products. Please contact the Technical Support Team at [support@Web Secure.au.com](mailto:support@Web Secure.au.com) for assistance with this.

## Technical Assistance

From: Resin's Classloading and Compilation FAQs - <http://www.caucho.com/products/resin/faq/classloader.xtp>.

## Where do I put my Classes and Jars?

Application classes are stored in *WEB-INF/classes*. Application jars are stored in *WEB-INF/lib*. Servlets are not enabled by default. Contact Web Secure Technical Support Team at [support@Web Secure.au.com](mailto:support@Web Secure.au.com) if you wish them to be enabled.



# Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

## What Classes does Web Secure have?

- ◆• dnsjava
- ◆• JavaMail
- ◆• PostgreSQL
- ◆• MySQL
- ◆• jdom
- ◆• jsse
- ◆• sax
- ◆• jakarta-regexp
- ◆• jce
- ◆• jndi
- ◆• jnet

## Web.xml

Web.xml provides servlet support and belongs in the WEB-INF. If you would like this to be enabled, please contact the Web Secure Technical Support Staff at [support@Web Secure.au.com](mailto:support@Web Secure.au.com).

## SMTP Information

When using JavaMail (<http://java.sun.com/products/javamail/>), set the SMTP host to:

```
package com.Web Secure.util;
```

```
Message mailMsg;
```

```
Properties props = System.getProperties();  
props.put("mail.smtp.host", smtpHost);  
Session session = Session.getDefaultInstance(props, null);  
session.setDebug(false);
```

```
this.mailMsg = new MimeMessage(session);
```



# Web Secure

PO Box 2726 Cairns, Qld 4870 Australia

P: 1300 73 83 01 F: 07-3036-5741

```
InternetAddress fromAddress = new InternetAddress(mfrom);

InternetAddress[] toAddress = new InternetAddress[mto.length];

for (int i = 0; i < mto.length; i++)
{
toAddress[i] = new InternetAddress(mto[i]);
}

this.mailMsg.setFrom(fromAddress); this.mailMsg.setRecipients(Message.RecipientType.TO,
toAddress); this.mailMsg.setSubject(subject);

this.mailMsg.setSentDate(new java.util.Date());

StringBuffer sb = new StringBuffer();

// set the body
this.mailMsg.setText(sb.toString());

// send the email
Transport.send(this.mailMsg);
```

## Troubleshooting

### I can't write to my web page

Contact Technical Support ([support@websecure.net.au](mailto:support@websecure.net.au)) for solutions to this problem, as by default Web Secure does not allow the JDK to write to the document root of web sites. This is a security measure.

### Error redirections

There are a number of different pages that your site will be redirected to in the following situations. Below are some

sample sites which you can customise to suit your pages: **The servlet engine is down:** [errors/server\\_down.html](#) **An error in your code (java.lang.NullPointerException):** [errors/internal\\_error.html](#) **File not found:** [errors/file\\_not\\_found.html](#)

For Resin troubleshooting queries, please visit Resin FAQ - <http://www.caucho.com/products/resin/faq/index.xtp>.