WHAT IS JAVASERVER PAGES?

Java Server Pages (JSP) is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.

A Java Server Pages component is a type of Java Servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.

Using JSP, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages and sharing information between requests, pages etc.

WHY USE JSP?

Java Server Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offer several advantages in comparison with the CGI.

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having a separate CGI files.
- JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.
- Java Server Pages are built on top of the Java Servlets API, so like Servlets; JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.
- JSP pages can be used in combination with Servlets that handle the business logic, the model supported by Java Servlet template engines.

Finally, JSP is an integral part of Java EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

ADVANTAGES OF JSP:

Following is the list of other advantages of using JSP over other technologies:

- **vs. Active Server Pages (ASP):** 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:** It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.
- **vs. Server-Side Includes (SSI):** 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 but can hardly interact with the web server to perform complex tasks like database access and image processing etc.
- **vs. Static HTML:** Regular HTML, of course, cannot contain dynamic information.

A development environment is where you would develop your JSP programs, test them and finally run them.

This tutorial will guide you to setup your JSP development environment which involves following steps:

SETTING UP JAVA DEVELOPMENT KIT

This step involves downloading an implementation of the Java Software Development Kit (SDK) and setting up PATH environment variable appropriately.

You can download SDK from Oracle's Java site: Java SE Downloads.

Once you download your Java implementation, follow the given instructions to install and configure the setup. Finally set PATH and JAVA_HOME environment variables to refer to the directory that contains java and javac, typically java_install_dir/bin and java_install_dir respectively.

If you are running Windows and installed the SDK in C:\jdk1.5.0_20, you would put the following line in your C:\autoexec.bat file.

```
• set PATH=C:\jdk1.5.0_20\bin;%PATH%
• set JAVA_HOME=C:\jdk1.5.0_20
```

Alternatively, on Windows NT/2000/XP, you could also right-click on My Computer, select Properties, then Advanced, then Environment Variables. Then, you would update the PATH value and press the OK button.

SETTING UP WEB SERVER: TOMCAT

A number of Web Servers that support Servlets are available in the market. Some web servers are freely downloadable and Tomcat is one of them.

Apache Tomcat is an open source software implementation of the Java Servlet and Java Server Pages technologies and can act as a standalone server for testing Servlets and can be integrated with the Apache Web Server. Here are the steps to setup Tomcat on your machine:

Download latest version of Tomcat from http://tomcat.apache.org/.

Once you downloaded the installation, unpack the binary distribution into a convenient location. For example in C:\apache-tomcat-5.5.29 on windows and create CATALINA_HOME environment variable pointing to these locations.

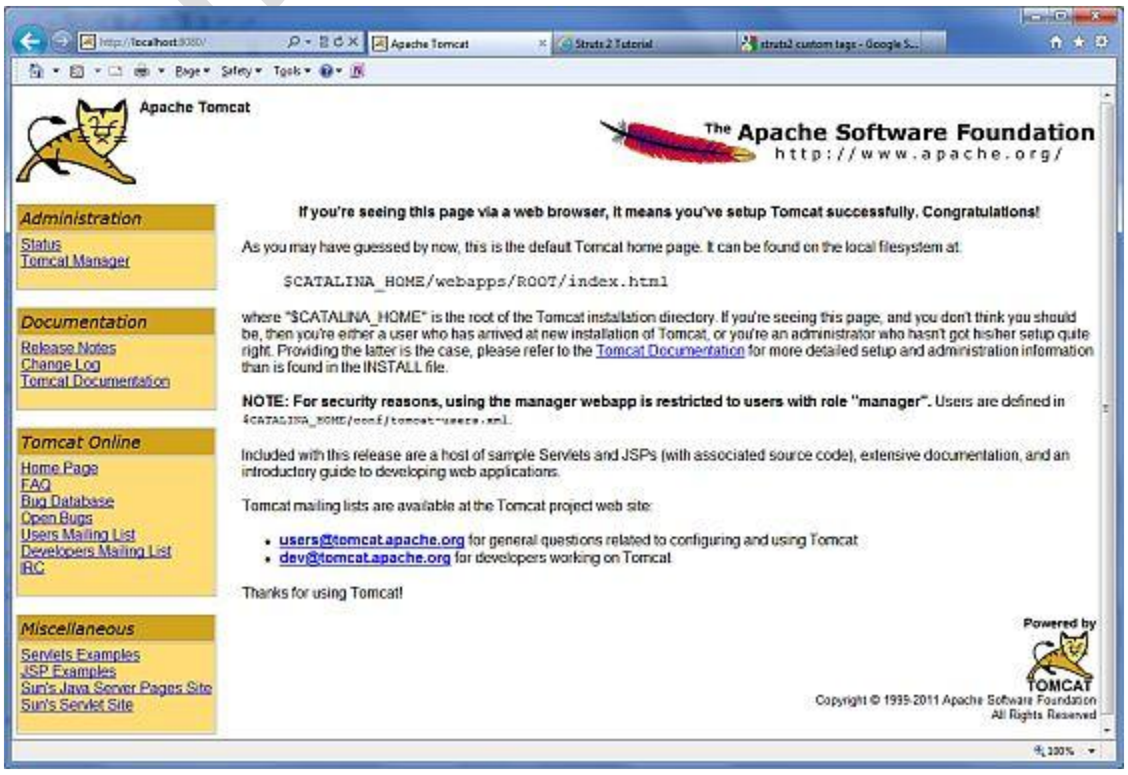
Tomcat can be started by executing the following commands on windows machine:

```
%CATALINA_HOME%\bin\startup.bat

or

C:\apache-tomcat-5.5.29\bin\startup.bat
```

After startup, the default web applications included with Tomcat will be available by visitinghttp://localhost:8080/. If everything is fine then it should display following result:



If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.html
```

where "$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Documentation for more detailed setup and administration information than is found in the INSTALL file.

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- users@tomcat.apache.org for general questions related to configuring and using Tomcat
- dev@tomcat.apache.org for developers working on Tomcat

Thanks for using Tomcat!

Copyright © 1999-2011 Apache Software Foundation
All Rights Reserved

Further information about configuring and running Tomcat can be found in the documentation included here, as well as on the Tomcat web site: *http://tomcat.apache.org*

Tomcat can be stopped by executing the following commands on windows machine:

```
C:\apache-tomcat-5.5.29\bin\shutdown
```

SETTING UP CLASSPATH

Since Servlets are not part of the Java Platform, Standard Edition, you must identify the Servlet classes to the compiler.

If you are running Windows, you need to put the following lines in your C:\autoexec.bat file.

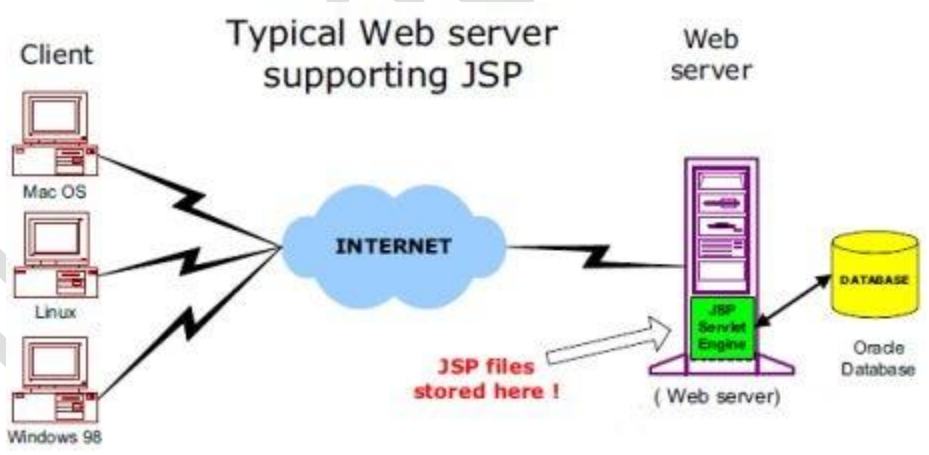
```
set CATALINA=C:\apache-tomcat-5.5.29
set CLASSPATH=%CATALINA%\common\lib\servlet-api.jar;%CLASSPATH%
```

Alternatively, on Windows NT/2000/XP, you could also right-click on My Computer, select Properties, then Advanced, then Environment Variables. Then, you would update the CLASSPATH value and press the OK button.

The web server needs a JSP engine i.e. container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. This tutorial makes use of Apache which has built-in JSP container to support JSP pages development.

A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that part of JSPs.

Following diagram shows the position of JSP container and JSP files in a Web Application.



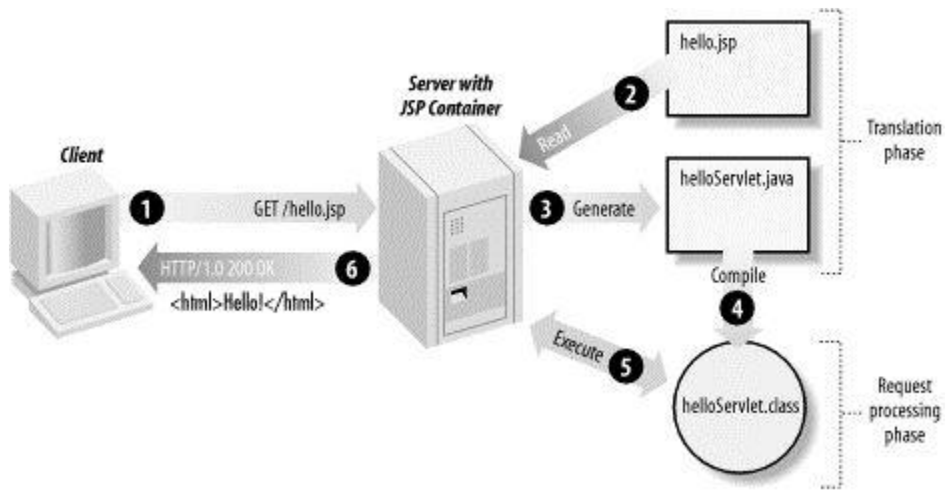
JSP Processing:

The following steps explain how the web server creates the web page using JSP:

- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of .html.
- The JSP engine loads the JSP page from disk and converts it into Servlet content. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.
- The JSP engine compiles the Servlet into an executable class and forwards the original request to a Servlet engine.
- A part of the web server called the Servlet engine loads the Servlet class and executes it. During execution, the Servlet produces an output in HTML format, which the Servlet engine passes to the web server inside an HTTP response.

- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.

  All the above mentioned steps can be shown below in the following diagram:



Typically, the JSP engine checks to see whether a Servlet for a JSP file already exists and whether the modification date on the JSP is older than the Servlet. If the JSP is older than its generated Servlet, the JSP container assumes that the JSP hasn't changed and that the generated Servlet still matches the JSP's contents. This makes the process more efficient than with other scripting languages (such as PHP) and therefore faster.

DRAWBACKS OF JSP

1. *JSP pages require about double the disk space to hold the page.*

Reason: *Because JSP pages are translated into class files, the server has to store the resultant class files with the JSP pages.*

2. *JSP pages must be compiled on the server when first accessed. This initial compilation produces a noticeable delay when accessing the JSP page for the first time.*

Reason: *The developer may compile JSP pages and place them on the server in compiled form (as one or more class files) to speed up the initial page access. The JSP developer may need to bring down the server to make the changed classes corresponding to the changed JSP page.*

3. *Debugging JSP programs is still a major challenge. You have to remember that a jsp page will be first converted to a .java file and then compiled. Thus, any errors will be pointing to line numbers in the converted .java file and not the .jsp page.*
4. *Database connectivity is not as easy as it should be.*
5. *JSP makes it tempting to put Java code in the web page, even though that's considered bad design. Or JSP can actually demand putting Java code in the page.*
6. *Doing even a simple task such as header and footer includes is overly difficult with JSP.*
7. *Looping is overly difficult in JSP.*
8. *JSP requires a compiler be shipped with the web server.*
9. *JSP consumes extra hard drive space and extra memory space.*