

Running your own Servlet Engine: Beginning JSPs and Servlets

by Brad Rippe

This is a brief introduction to running your own servlet engine. For more information on jsp's and servlets, please see the web site <http://staffwww.fullcoll.edu/brippe/cis226>.

Getting Started

1. To get started with running server side java you will need to download and install a Java Development Kit. You can download a Java Development Kit (JDK) release (version 1.2 or later) from: <http://java.sun.com/j2se/>
2. Set an environment variable JAVA_HOME to the pathname of the directory into which you installed the JDK release. This needs to be done through the control panel or your autoexec.bat.
3. Download and Install Tomcat 4.0. Tomcat. Tomcat can be downloaded at <http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/v4.0.6/bin/>
You need want to download the windows installer [jakarta-tomcat-4.0.6.exe](#)
Tomcat is in a version 4.1, but I'll use the 4.0 version for this demonstration.

The following is from the Tomcat README

Start Up Tomcat 4.0

There are two techniques by which Tomcat 4.0 can be started:

* Via an environment variable:

- Set an environment variable CATALINA_HOME to the path of the directory into which you have installed Tomcat 4.0.
- Execute the shell command:

```
%CATALINA_HOME%\bin\startup      (Windows)
```

```
$CATALINA_HOME/bin/startup.sh    (Unix)
```

* By modifying your current working directory:

- Execute the following shell commands:

```
cd %CATALINA_HOME%\bin           (Windows)  
startup                          (Windows)
```

```
cd $CATALINA_HOME/bin           (Unix)  
./startup.sh                     (Unix)
```

After startup, the default web applications included with Tomcat 4.0 will be available by browsing:

```
http://localhost:8080/
```

Further information about configuring and running Tomcat 4.0 can be found in the documentation included here, as well as on the Tomcat web site:

```
http://jakarta.apache.org/tomcat/
```

Shut Down Tomcat 4.0

There are two techniques by which Tomcat 4.0 can be stopped:

* Via an environment variable:

- Set an environment variable CATALINA_HOME to the path of the directory into which you have installed Tomcat 4.0.
- Execute the shell command:

```
%CATALINA_HOME%\bin\shutdown      (Windows)
```

```
$CATALINA_HOME/bin/shutdown.sh     (Unix)
```

* By modifying your current working directory:

- Execute the following shell commands:

```
cd %CATALINA_HOME%\bin              (Windows)
```

```
shutdown                            (Windows)
```

```
cd $CATALINA_HOME/bin              (Unix)
```

```
./shutdown.sh                       (Unix)
```

Creating a Context

1. Before we can begin with creating jsps, we should create a context for these jsps to run in. The context is a self contained location for all related jsps and servlets to run from. Example, <http://zeus.fullcoll.edu:8080/classSchedule/>, the “classSchedule” is the context where all jsps and servlets will reside.
2. To create a context we must modify the server.xml located in CATALINA_HOME/config directory. To add a new context we need to specify the following xml in the server.xml:

```
<!-- CIS 226 EXAMPLE 12/3 -->  
<Context path="/cis226" docBase="cis226" debug="0" privileged="false"/>  
<!--END NEW CONTEXT -->  
  
<!-- Tomcat Manager Context -->  
<Context path="/manager" docBase="manager" debug="0" privileged="true"/>
```

Add the new context before the Tomcat Manager Context as above. My context name is “cis226”, you can name appropriately. Remember, this is going to be a url, NO spaces.

3. Create a directory in the webapps with the same name as you given in the xml above.
4. Copy the WEB-INF directory from the ROOT directory into your new context directory. The WEB-INF directory contains a web.xml file that describes services and authentication mechanism for your web application. This web.xml from the ROOT context is empty.

Creating your first jsp

1. Type the following html code:

```
<% @ page contentType="text/html; charset=iso-8859-1" language="java" %>
<% @ page import="java.util.*" %>
<html>
<head>
<title>HelloJSP - CIS 226</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<h1>Hello from my First JSP</h1>
<% Date today = new Date(); %>
Today's date is : <%= today.toString() %>
</body>
</html>
```

Save and name your file “HelloJSP.jsp” in the context directory that you just created.

2. Start Tomcat by running startup.bat from the bin directory.
3. Open your browser and type the url:
<http://localhost:8080/cis226/HelloJSP.jsp>
4. You should see the title and today’s date.

Now let me discuss a little of what just happened.

Creating a servlet

1. Create a directory called “classes” in your WEB-INF directory.
2. Type the following java code and save the file as HelloServlet.java

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {

    public String getServletInfo(){
        return "Servlet description";
    }

    public void service(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException{

        response.setContentType("text/html");
        PrintWriter out = new PrintWriter(response.getOutputStream());
        out.println("<html>");
        out.println("<head>");
        out.println("<title>HelloServlet - CIS 226</title>");
        out.println("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859-1\">");
        out.println("</head>");
```

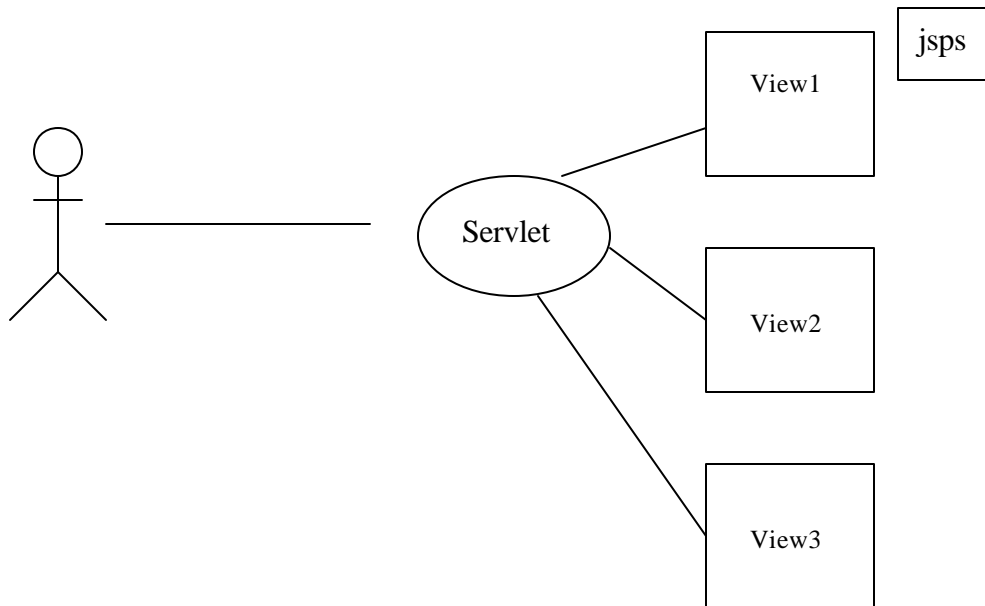
```
out.println("<body>");
out.println("<h1>Hello from my First Servlet</h1>");

Date today = new Date();
out.println("Today's date is : " + today.toString());
out.println("</body>");
out.println("</html>");
out.flush();
}
}
```

This servlet needs to be compiled and the class file must be in the classes directory. Once compiled the servlet can be accessed via a web browser at

<http://localhost:8080/cis226/servlet/HelloServlet>

Architectural idea behind servlets and jsps are basically servlets will handle traffic control and/or the logic to the web application. Jsps will provide the view for the user responses (Servlets often call additional object to handle business logic. These are known as Java Beans). The following is an overview of how such an application might look. This model can be replicated for common features in a web application. This model is often referred to as Model 2, due to the servlet handling the request and forwarding it. Model 1, the JSP handles the request.



For more information on servlets see:

http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets.html

For more information on jsps see:

http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/JSPIntro.html