Java Servlets and a Stand Alone Server

Stand Alone Server

A free stand alone server can be downloaded from the Apache open software project. It is Tomcat 4.0.4, and it is available from http://jakarta.apache.org/tomcat/tomcat-4.0-doc/ . Directions for installation can be found at http://archive.coreservlets.com/Using-Tomcat.html.[1] Make sure that you download the regular version, not the LE version. The latter requires an XML parser and doesn't work without it.

The directions at coreservlets also tell you which directories to use for your html files and Java servlets. The html files should be placed in the webapps/ROOT subfolder of the Apache folder. The servlets class files belong in the webapps/ROOT/WEB-INF/classes folder. If you are reading from a file, the file goes into the *bin* folder. You might want to create a subdirectory of bin where you store your text files.

The Action Attribute

There are a few changes required for your html file. The main one is that the action attribute must include the term, servlet.

action="http://localhost/servlet/client-server.EmailServlet/"

Note: This assumes that Tomcat has been modified to receive input on port 80. If you leave it on port 8080 (as initially configured), the action attribute must include the 8080.

action="http://localhost:8080/servlet/client-server.EmailServlet/"

The rest of the form is the same as before.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
    <head><title>E-Mail Form</title></head>

<body>
    <h3>Enter your name and e-mail address.
    <br />Then click the Send button to send the data to the server.</h3>
    <form method = "get" action="http://localhost/servlet/client_server.EmailServlet">
        <p><input type = "text" name = "name" value = "" size = 30> Name </p>
        <p><input type = "text" name = "email" value = "" size = 30> E-Mail Address </p>
        <p><input type="submit" value="Send" /></p>
    </form>
</body> </html>
```

The servlet.jar package

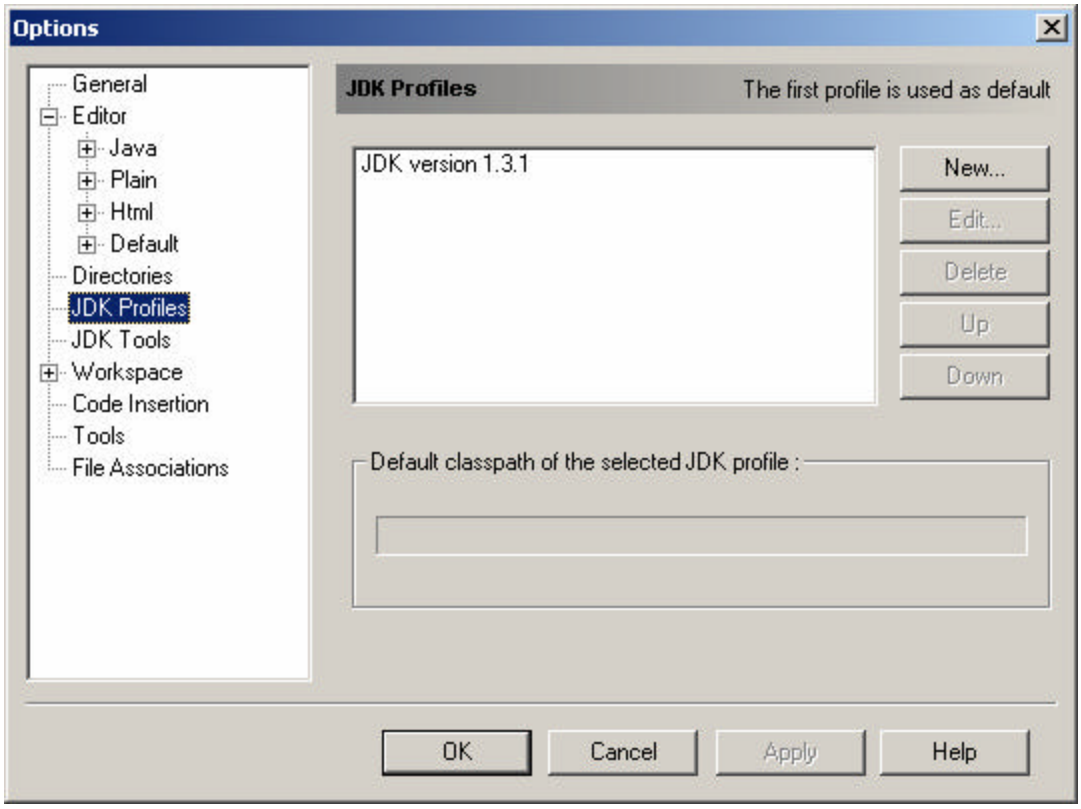
Unlike java.sql, javax.servlet does not come with the usual Java setup. It does come with Tomcat, but if you have problems downloading it, you will find a copy on my website. The package, servlet.jar, contains the java files needed for compiling Java servlets. The classes that are needed are javax.servlet and javax.servlet..http. They must both be imported into your file.

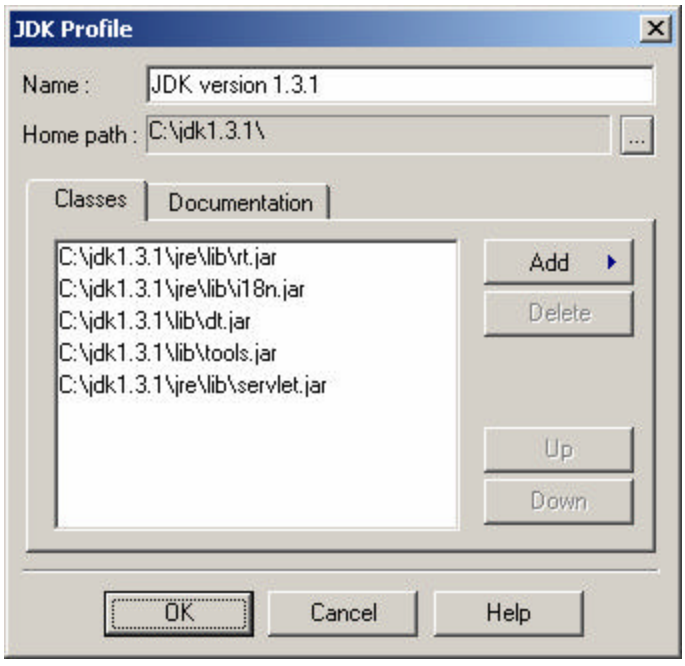
---

[1] A copy of the coreservlets page can be found on my web site in the Documents folder. It is titled *Configuring Tomcat*. If you have trouble downloading Tomcat, a copy of it is in the same folder as well.

In addition, the Java compiler has to know where to locate these files. You can put the file anywhere, but probably the best thing is to keep it with the other Java packages. So place it in a folder under your Java installation.

After that you will have to configure the classpath of your IDE (Integrated Development Environment) so that servlet.jar can be found when servlets are compiled. In JCreator, this is under Java Profiles. First click on **Configure/Options** and then on **JDK Profiles**. You should see the following screen.





Click on the compiler, here JDK version 1.3.1 and then **Edit**. Click on **Add/Add Package** and then navigate to the folder where you stored servlet.jar. When you click on servlet.jar, it will be added to the JCreator classpath. When done, it should look similar to the screen on the left.

The Java Se rvlet

Only a few changes are required in order to modify the programs written for the WebServer. You have to import both javax.servlet and javax.servlet.http.

```
import javax.servlet.*;
import javax.servlet.http.*;
```

The processor class now must extend HttpServlet instead of WebRequestProcessor.

```
public class EmailServlet extends HttpServlet
```

The main method in the servlet is now either *doGet* or *doPost* and the parameters are HttpServletRequest and HttpServletResponse.

```
public void doGet (HttpServletRequest request, HttpServletResponse response)
```

Finally this method always throws an IOException that must be caught or re-thrown.

he EmailServlet Example

```
package client_server;

/* EmailServlet processes a request from a web page.  It responds to the request by echoing back the
name and email address that was sent in. */
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class EmailServlet extends HttpServlet
{
    String name, email;
    PrintWriter out;

    public void doGet (HttpServletRequest request, HttpServletResponse response)
    {
        try
        {
            out = response.getWriter ();
            name = request.getParameter ("name");
            email = request.getParameter ("email");

            // Get a PrintWriter object and respond to the request.
            //PrintWriter out = response.getWriter ();

            // The Page class is contained in the client_server folder.
            Page.createHeader (out, "Test Data");
            out.println ("<h3>Hello.</h3>");
            out.println ("<h3>" + name+ "</h3>");
            out.println ("<h3>Your email address is " + email + "</h3>");
            Page.createFooter (out);
        }catch (IOException e) {System.out.println ("Servlet Exception");}
    } // doGet
} // EmailServlet
```