SERVLETS - PACKAGING

The web application structure involving the WEB-INF subdirectory is standard to all Java web applications and specified by the servlet API specification. Given a top-level directory name of myapp, Here is what this directory structure looks like:

```
/myapp
    /images
    /WEB-INF
        /classes
        /lib
```

The WEB-INF subdirectory contains the application's deployment descriptor, named web.xml. All the HTML files live in the top-level directory which is *myapp*. For admin user, you would find ROOT directory as parent directory as myapp.

Creating Servlets in Packages:

The WEB-INF/classes directory contains all the servlet classes and other class files, in a structure that matches their package name. For example, If you have a fully qualified class name of **com.myorg.MyServlet**, then this servlet class must be located in the following directory:

```
/myapp/WEB-INF/classes/com/myorg/MyServlet.class
```

Following is the example to create MyServlet class with a package name *com.myorg*

```
// Name your package
package com.myorg;

// Import required java libraries
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {

  private String message;

  public void init() throws ServletException
  {
      // Do required initialization
      message = "Hello World";
  }

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
            throws ServletException, IOException
  {
      // Set response content type
      response.setContentType("text/html");

      // Actual logic goes here.
      PrintWriter out = response.getWriter();
      out.println("<h1>" + message + "</h1>");
  }

  public void destroy()
  {
      // do nothing.
  }
}
```

Compiling Servlets in Packages:

There is nothing much different to compile a class available in package. The simplest way is to keep your java file in fully qualified path, as mentioned above class would be kept in com.myorg. You would also need to add this directory in CLASSPATH.

Assuming your environment is setup properly, go in **<Tomcat-installation-directory>/webapps/ROOT/WEB-INF/classes** directory and compile MyServlet.java as follows:

```
$ javac MyServlet.java
```

If the servlet depends on any other libraries, you have to include those JAR files on your CLASSPATH as well. I have included only servlet-api.jar JAR file because I'm not using any other library in Hello World program.

This command line uses the built-in javac compiler that comes with the Sun Microsystems Java Software Development Kit *JDK*. For this command to work properly, you have to include the location of the Java SDK that you are using in the PATH environment variable.

If everything goes fine, above compilation would produce **MyServlet.class** file in the same directory. Next section would explain how a compiled servlet would be deployed in production.

Packaged Servlet Deployment:

By default, a servlet application is located at the path <Tomcat-installation-directory>/webapps/ROOT and the class file would reside in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/classes.

If you have a fully qualified class name of **com.myorg.MyServlet**, then this servlet class must be located in WEB-INF/classes/com/myorg/MyServlet.class and you would need to create following entries in **web.xml** file located in <Tomcat-installation-directory>/webapps/ROOT/WEB-INF/

```
<servlet>
   <servlet-name>MyServlet</servlet-name>
   <servlet-class>com.myorg.MyServlet</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>MyServlet</servlet-name>
   <url-pattern>/MyServlet</url-pattern>
</servlet-mapping>
```

Above entries to be created inside <web-app>...</web-app> tags available in web.xml file. There could be various entries in this table already available, but never mind.

You are almost done, now let us start tomcat server using <Tomcat-installation-directory>\bin\startup.bat *onwindows* or <Tomcat-installation-directory>/bin/startup.sh *onLinux/Solarisetc*. and finally type **http://localhost:8080/MyServlet** in browser's address box. If everything goes fine, you would get following result:

```
HELLO WORLD
```

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js