8. Packaging Servlets for Deployment

Contents

[1. Introduction 1](#_Toc17550)

[2. Creating Servlets in Packages 1](#_Toc32406)

[3. Package Servlets Deployment 1](#_Toc28640)

[4. Summary 1](#_Toc25229)

# 1. Introduction

=>slides: Pg. 1

Until now in the previous modules, we have understood how to create a Servlet page, access the query string and form data from the Servlets, how to add filters to the Servlet, and also we have discussed about tracking the session data. With all this information, we are in the position to create a basic Servlet application. But once the application is created, then we need to deploy that application.

=>slides: Pg. 2

So in this module, we shall understand how to create Servlets in packages and deploy the package Servlets.

# Creating Servlets in Packages

=>slides: Pg. 3

Now, let us understand how to create the Servlets package. To start, let me open the command prompt, and let me create a folder with the name psdemo.



So let me type in md psdemo. Let me change over to that folder,



so let me type in cd psdemo. Now let us create a simple Servlet page.



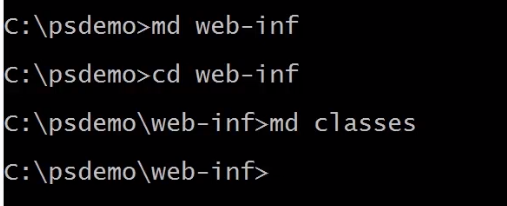
So let me type in start notepad HelloServlet. java.



Let me update this page with simple Servlet code to display a welcome message. We can observe a class with the name HelloServlet, which extends HttpServlet, and we have a doGet method where the content type has been set to text/html. And then we have defined an object for the PrintWriter class. And then we have displayed a simple welcome message, Hello From Servlet. Once the above code has been added, now let us Save this file and close the notepad. Before I compare this code, we need to remember that the job of a Servlet is not only is it responsible for executing, it should also provide the required resources.

=>slides: Pg. 4

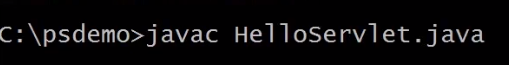
Now let us understand the folder structure required for the Servlet application to be deployed. First, we will have the name of the Servlet application as a folder name. And then to maintain the web content, we require a folder with the name Web Contents. And then we know the web deployment descriptor file will be maintained within the WEB-INF folder. So we require another folder with the name WEB-INF. Along with that, we require a folder to maintain all the compiled Servlet class files where the name of the folder should be Classes.



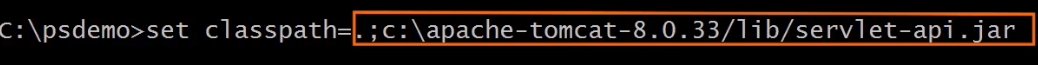
So within the command prompt, let me quickly create that folder, md web-inf, and then md classes.



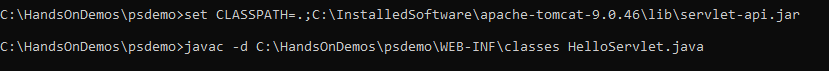
Now let me come back to the main folder, psdemo.



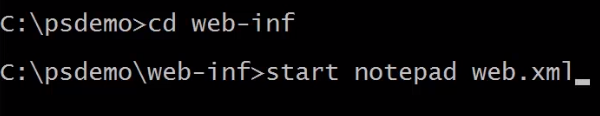
As the required folder structure is created now, let me compile that application. So let me type in javac HelloServlet. java. We can observe a whole lot of errors are being displayed. The reason for that is the Java compiler will not recognize the Servlet class definitions. So in order to make the things work, we need to set that classpath to refer the Servlet API. JAR, which will be located in that lib folder present at the Apache server folder. Let me type in set classpath=. ;.

Since the Apache server is copied to the C: drive directly, let me type c:\apache-tomcat-8. 0. 33/lib/servlet-api. jar. Once the classpath is set, now we can compile the Java Servlet code. And as it already informed the Servlet class definition should be maintained within that Classes folder, so let me compile the code and copy the Servlet class file to the Classes folder using a single comment.

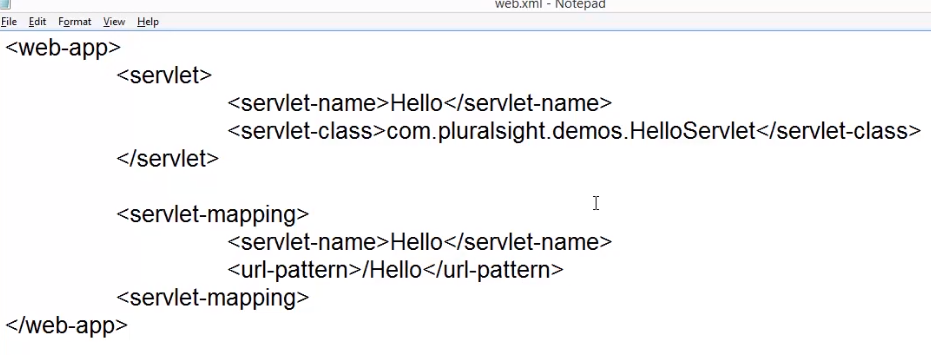




So let me type in javac -d c:\psdemo\web-inf\classes HelloServlet. java. Now let us observe the folder structure. Let me navigate to the Classes folder present within the WEB-INF folder. We can observe HelloServlet. class is present. Now, let me come back to the command prompt and change the directory to the WEB-INF folder.

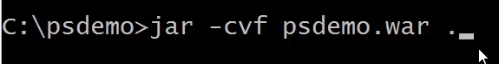


So let me type in cd web-inf. We need to add the web deployment descriptor file to provide the details about the Servlet. First, let us create the file. So let me type in start notepad web. xml. Now we need to provide the details about the Servlet.



So let me type in web-app, servlet, servlet-name, and let me provide a reference name for the Servlet. For example, let me type in Hello. And then we need to specify the Servlet class, so let me type in servlet-class. And we need to specify the class name to be mapped to it, that is, com. Pluralsight. demos. HelloServlet. Once we have defined the Servlet, then we need to map the Servlet to a specific URL-pattern. So I need to type in servlet-mapping. Then we need to specify the Servlet name, so let me type in servlet-name, Hello, and then we need to specify the URL-pattern to be used for accessing the Servlets. So let me type in url-pattern/Hello. Let me save the web. xml file and close the Notepad. The final step for creating the Servlet packaging will be creating the WAR file.



Let me come back to the root folder that is psdemo, and let me create the Servlet package file. To do, 

let me type in jar -cvf psdemo. war and a. where. means all the files present within the folder. Now let me check out the directory. We can find the WAR, that is, the Servlet package file. As we have understood how to create a Servlet package, in the next clip, we shall understand how to deploy the Servlet package.

# Package Servlets Deployment

=>slides: Pg. 5

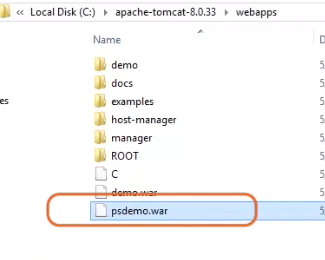
In order to execute our application, it must be deployed on a Servlet container, and this is true even during the development.

=>slides: Pg. 6

Servlet's can be deployed in two ways-- hot development method, and the other one is cold deployment method. Whenever we deploy and un-deploy web applications without restarting the server, then it is considered as hot deployment. The advantage of hot deployment is it prevents us from manual restart of production server if we make any changes to the contents of the WAR file being deployed, and the business is never stopped. But the limitations of hot deployment are that hot deployment deploys simultaneously to all the nodes, resulting in down time, and hot redeployment of an application sometimes results in out-of-memory errors. Whenever we require one or more instances of the server to restart to reflect the changes, then it is said to be cold deployment. The advantages of the cold deployment are there's no caching and also no out-of-memory issues will be there, even after subsequent deploys during testing. And the limitation is outage cost is more. We will use Tomcat to provide the execution environment.

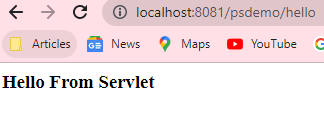
=>slides: Pg. 7

A web application can be deployed in Tomcat by one of the following approaches. Copying the web application archive file into directory webapps/ of Tomcat. When Tomcat starts, it will automatically expand the web application archive file into its unpacked form and execute the application. This approach would typically be used to install an additional application provided by a third party vendor or by our internal development team into an existing Tomcat installation. Now let us understand it practically. I already opened the psdemo folder where we have the psdemo. war file. And I also opened the webapps folder of Apache web server.



Now let me copy the psdemo. war file to the webapps folder. Then let us start the Tomcat web server. 

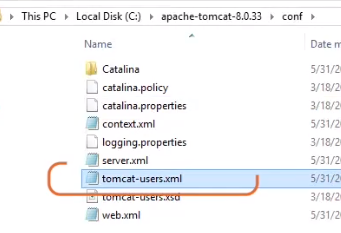
To do, let me navigate to the bin directory of the Apache web server folder, and we need to execute the startup. bat file. Once the Apache web server has been started, then we can observe the WAR deployment of the web application. And when we open the webapps folder, we can also observe the copy of our application is hosted.



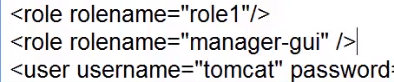
Now let me open the web browser and type in the URL--http://localhost/psdemo/hello. We can observe the output is displayed. We need to remember one important point. That is, if you use this approach and wish to update our application later, we must replace both the web application archive file and delete the directory that Tomcat created, and then restart Tomcat in order to reflect the changes.

=>slides: Pg. 8

Now let us understand another efficient method to deploy the Servlet application using the Tomcat Manager web application to deploy and un-deploy web applications. Tomcat includes a web application deployed by default on context path/manager that allows us to deploy and un-deploy applications on a running Tomcat server without restarting it. Now let us understand practically. Now let me remove the deployed Servlet application from the Apache web server. Now let us understand how to use Tomcat Manager for web application deployment. To do, first we need to configure the users and the roles for accessing the Tomcat web server.



So let me open Apache Tomcat web folder/conf folder. Within that, we can find a file with the name tomcat-users. xml.



Let me open that file for adding the entry role rolename="manager-gui". And then let me update the user "tomcat".



Let me type in the password as "tomcat" and the role as "tomcat, manager-gui". Let me update the "tomcat" password for other users also. Let us save the file and execute the startup. bat file. Now let me open the browser, open localhost:8081. We can observe the Apache web server home page.



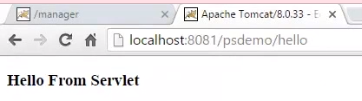
Then click on the Manager App button. The above step will prompt the user to provide the username and password. So let me type in username as tomcat, and the password is also tomcat. And then let me click on Log In. Once we have logged into that Manager app,



then scroll down the page until we see the title as WAR file to upload. Click on the Choose File, and select the WAR file from the local system, and click on the Deploy button.



The above step will deploy the Servlet application to the Apache web server. Let us open the webapps folder of the Apache web server. We can find that application has been deployed. Let me open the web browser once again and type in the URL, http://localhost:8081/psdemo/hello.



We can observe the output has been displayed.

# Summary

=>slides: Pg.

In this module, we have understood how to create a Servlet package and deploy the Servlet package. In the next module, we shall understand how to debug the Servlets.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*