2. Building and Deploying Your First Servlet

Contents

[1. Introduction 1](#_Toc728)

[2. Creating a Basic Servlet - Hello World! 1](#_Toc25237)

[3. Deploying Servlet in Web Server 1](#_Toc10962)

[4. Summary 1](#_Toc12906)

# 1. Introduction

=>slides: Pg. 1

In this module, we will understand the importance of HttpServlet in detail by working with form and query string data. By the end of this module, the user will understand how to work with HTTP-GET and HTTP-POST requests and handle the query string data, and also we shall learn how to create an HTML page and pass the data from an HTML page to the Servlet and access the form data from the Servlet. Now let us get started. The first thing that we need to understand is when we have a GenericServlet, then why we have to go for extending an HttpServlet. So let us understand that first. GenericServlet is protocol independent. That is, it supports all types of protocols such as HTTP, SMTP, FTP, etc. Whenever we use the GenericServlet class for the web application's development, then we know that we have to handle the service method, which can handle only simple requests. That is, GenericServlet can't track the session data, which is very essential for any web application development. I will explain the importance of session data and why we need to track the data later in this course by explaining the tracking session data module. Since HttpServlet is specially designed for handling the HTTP protocol and supports all the HTTP verbs such as HTTP-GET, HTTP-POST, HTTP-PUT, HTTP-DELETE, etc., it is advisable for a Servlet class to extent HttpServlet class for web application development than extending GenericServlet class. Now let us understand how to work with HttpServlet in detail. In order to work with HttpServlet, a class has to extend HttpServlet class. For example, public class HttpServletName extends HttpServlet. HttpServlet class provides various doXXX methods to handle the client's request such as doGet, doPost, doDelete, doPut, etc. We don't have to override service method by working with HttpServlet class. It has been implicitly provided with a definition to invoke the relevant doXXX method based on the HTTP \_\_\_\_\_ used for submitting the request. That is, if HTTP-GET method is used to submit the request, then doGet method will be invoked. And if HTTP-POST method is used to submit the request, then doPOST method will be invoked by the service method. All these methods accept two parameters, HttpServletRequest and HttpServletResponse, and throws ServletException and IOException. In order to get the values from the requests, HttpServletRequest variable should be used. And in order to provide the response from the Servlet, HttpServletResponse variable should be used. The request variable of the doXXX method provides the following methods to read the form data using Servlets. getParameter method--it accepts a string that is the parameter name and returns the value of a request parameter as a string. And this method will return null if the parameter does not exist. getParameterValues method--it accepts a string that is the parameter name and return an array of string objects containing all of the values for the given parameter name. And this method also will return null if the parameter does not exist. getParameterMap method--it returns a java. util. Map of the parameter of this request. getParameterNames method--this method returns an enumerator of string objects containing the names of the parameters contained within the request. Now let us understand how to work with form data and query string data practically.

# Creating a Basic Servlet - Hello World!

=>slides: Pg. 2

In order to create a Servlet page,

=>slides: Pg. 3

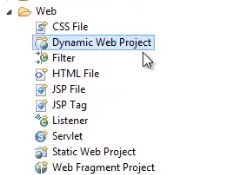
we know that it is mandatory for a class to implement Servlet interface and override its methods or the class has to extend GenericServlet or HttpServlet class. To make the concept simple, let us first understand creating a basic Servlet by extending the GenericServlet class.

=>slides: Pg. 4

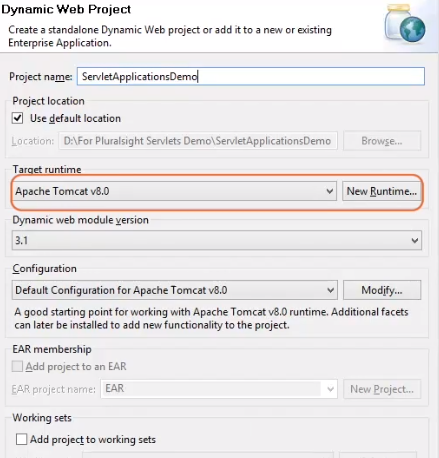
We can do that by defining that class, for example, HelloWorld, and extends GenericServlet. Whenever a class inherits GenericServlet class,

=>slides: Pg. 5

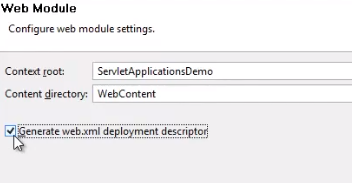
we know that service method is the one which will be invoked for every client request. The service method accepts two parameters, one to represent ServletRequest and the other to represent ServletResponse. And this method throws ServletException and IOException. If we wanted to accept the input provided from the user, then we need to use the request variable. And in order to provide the response to the client, we need to use the response variable. And the code that we write for the service method will be considered as the definition to be processed. Now let us understand practically how to create a basic Servlet HelloWorld.

I already opened Eclipse and also configured the Eclipse with the Tomcat web server. Now the first step we need to perform is add the dynamic web project. To do, right-click on the Project Explorer and then click on New. We can observe the recent project types which are open. Assuming that this is the first time, let us click on Other, return the available wizard, scroll down, and then expand the Web folder. 

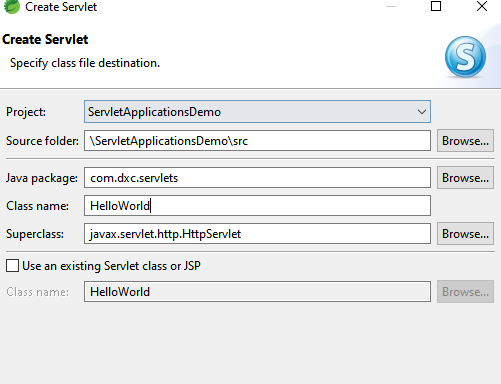
Select the Dynamic Web Project, and click on Next.



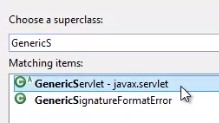
Now we need to provide a project name. Provide some meaningful name, for example, ServletApplicationDemo. We can observe the target runtime has been configured to Apache Tomcat version 8. Click on Next button. It provides the details about the source folders. We just need to click on the Next button.



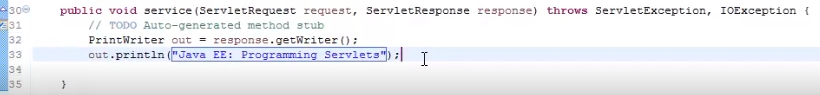
Check the Generate web. xml deployment descriptor checkbox. I will explain about the importance of web. xml later in this course. For now, let me click on the Finish button.



In order to create a Servlet page, we need to right-click on the application name, and then click on New and select Servlet. Provide a meaningful name for the package, for example, demo. Pluralsight. com. And then provide the class name, for example, HelloWorld. And we can observe by default the Servlet superclass is taken as HttpServlet.



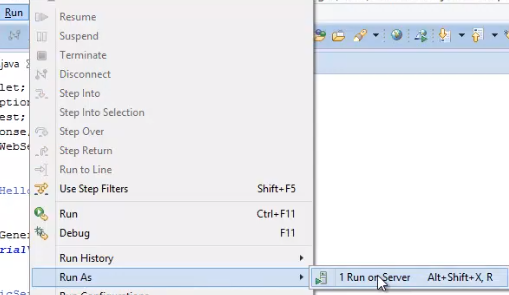
As I wanted to create a GenericServlet, let me click on the Browse button, and at the Choose a superclass textbox, let us type in Generic. We'll find some filtered matching items. We need to select GenericServlet - javax. Servlet. And click on OK button. And then click on Finish button. We can observe some boilerplate code has been generated. Now as our requirement is to display a basic output HelloWorld, we need the response. To get, we have response. getWriter method, which returns now PrintWriter class object.



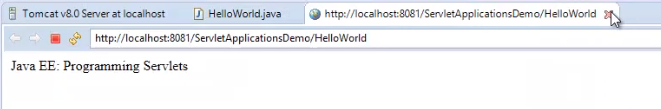
So let me type in PrintWriter out = response. getWriter. Now I would like to display a simple welcome message. So let me type in out. println ("Java EE: Programming Servlets"), quite a simple code to start with. In the next clip, we shall understand how to deploy this Servlet and execute the Servlet page.

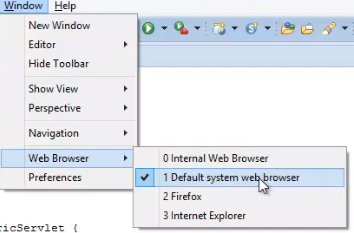
# Deploying Servlet in Web Server

Once the Servlet page has been created, now in order to deploy the Servlet, we have two methods. One is manual deployment, and the other one is automated deployment using the Eclipse. The manual deployment we shall understand later in this course. And to get started with, let us use the automated deployment using the Eclipse. As we have already configured Eclipse with the Tomcat web server, so to execute the server page, it is quite simple task to do.



Click on the Run menu, and then click on Run As, Run on Server. The above step will prompt the user to select a server name to be used for handling the request and response. I select the default option, Choose an existing server, and select Tomcat version 8, Server at localhost. This window will keep on bugging you every time you execute the Servlet page. To avoid, we can check the checkbox, Always use this server when running this application, and then click on Next. We can observe our application has been configure on the server. Click on Finish button.

We can observe an inline web browser has been opened, and we got the welcome message. In case if we require our output to be displayed on external web browsers, then we need to update the default web browser.



To do, let me click on the Window menu, Web Browser, then we need to select the browser to be used for displaying that web page. So I select Default system web browser. Now let me execute the page. Once again, we can observe the default browser of my system has been opened, and we got that welcome message from the Servlet page.

# Summary

=>slides: Pg. 6

In this module, we have understood how to create a basic Servlet page and deploy that Servlet page. In the next module, we shall understand more in detail of developing some advanced Servlet pages by working with form data and handling the GET and POST requests.

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