3. Working with Form and Query String Data

Contents

[1. Introduction 1](#_Toc2550)

[2. Handling GET Method in Servlets 1](#_Toc27438)

[3. Handling Form Data in Servlets 1](#_Toc9867)

[4. Handling POST Method in Servlets 1](#_Toc12439)

[5. Summary 2](#_Toc4730)

# 1. Introduction

=>slides: Pg. 1

In this module, we shall understand the importance of HttpServlet in detail by working with form and query string data.

=>slides: Pg. 2

By the end of this module, the user will understand how to work with HTTP-GET

=>slides: Pg. 3

and HTTP-POST request

=>slides: Pg. 4

and handle the query string data,

=>slides: Pg. 5

and also we shall learn how to create an HTML page and pass the data from an HTML page to the Servlet

=>slides: Pg. 6

and access the form data from the Servlet. Now let us get started.

=>slides: Pg. 7

The first thing that we need to understand is when we have a GenericServlet, then why we have to go for extending 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 development, then we know that we have to handle the service method, which can handle only simple requests. That is, GenericServlets 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 while explaining that 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 extend 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.

=>slides: Pg. 8

For example, public class HttpServletName extends HttpServlet. HttpServlet class provides various do methods to handle the client's request such as doGet, doPost, doDelete, doPut, etc. All these methods access two parameters, HttpServletRequest and HttpServletResponse, and throws ServletException and IOException.

=>slides: Pg. 9

In order to get the values from the requests, HttpServletRequest variable should be used.

=>slides: Pg. 10

And in order to provide a response from the Servlet, HttpServletResponse variable should be used. We don't have to override service method by working with HttpServlet class.

=>slides: Pg. 11

That is, if HTTP-GET method is used to submit that request, then doGet method will be invoked.

=>slides: Pg. 12

And if HTTP-POST method is used to submit that request, then doPost method will be invoked by the service method.

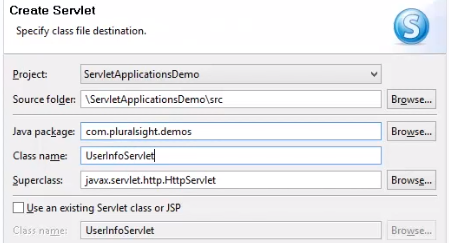
=>slides: Pg. 13

HttpServletRequest provides the following methods to read the form data using Servlets. getParameter method--it acts as a string that is the parameter name, which can be either a query string \_\_\_\_\_ or the form element name, which has been posted from any HTML or Servlet page, and it returns the value of the 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 returns 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--this method returns a java. util. Map of the parameters of this request. getParameterNames method--this method returns an enumeration 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.

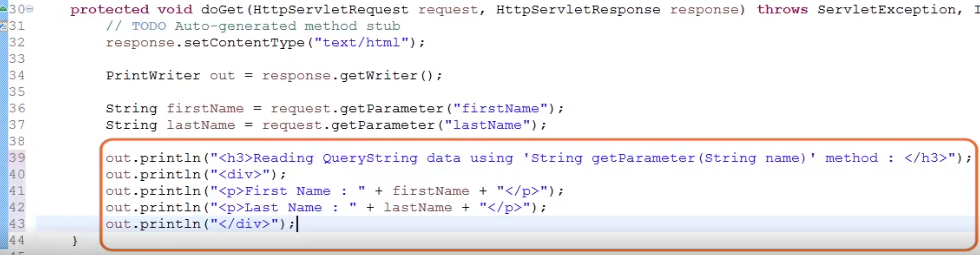
# Handling GET Method in Servlets

=>slides: Pg. 14

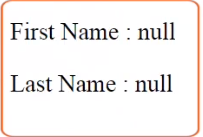
In order to create a Servlet, let me right-click on our application and



click on New, Servlet, provide a meaningful package name. For example, I provide com. Pluralsight. demos. And then we need to provide a meaningful class name. For example, let me provide UserInfoServlet. And by default, as a superclass's HttpServlet, I'm not changing the superclass option. So let me click on Next button. If any mappings have to be done, then we can update here. As I don't want to make any changes, let me click on Finish to create the Servlet class. By default, whenever the user types in a URL at the browser or clicks on a link present within the web page, then it will submit a GET request. And in order to handle that GET request by the HttpServlet, we need to handle the doGet method assuming that the user will pass firstName and lastName as query string data at the URL. Let us write the code. We already discussed that in order to read the data, we have been provided with various GET methods. Let us understand how to use these methods in practice.



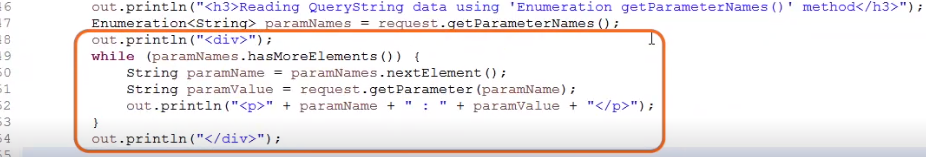
First, we need to set the response part of the Servlet page as HTML content. To do, we need to set the content type of the response using the text/html. So let me type in response. setContentType of text/html. We know that in order to bring the output of the page, we have response. getWriter method, which returns the PrintWriter class. So let me type in PrintWriter out = response. getWriter. We can observe an error in PrintWriter. Let us import a package. To do, let me press Ctrl+Spacebar. Now let us read the values from the request. The best approach to read the values from the request parameter is using the getParameter method. So let me define a string variable to hold the value of firstName. So let me type in String firstName = request. getParameter("firstName"). Similarly, to get the value of lastName, let me define another string variable, String lastName = request. getParameter("lastName"). Now, let me write the code to display the value where h3 tag is used for displaying the heading and div tag is used to create a logical block. And p tag is used for creating the paragraph. Now let me execute the application.



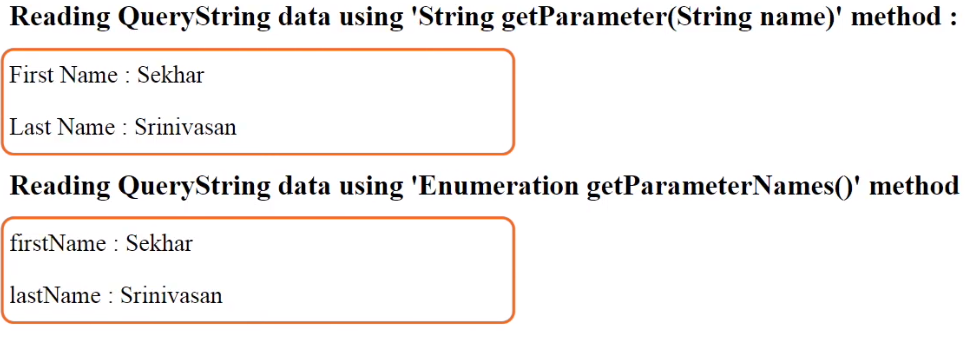
We can observe the null values because the default value is null if the parameter names don't exist. Let me type in the query string data at the URL. To provide the query string at the URL, we need to open the URL with the question mark followed by the parameter name equal to value. And if more than one parameter exists, then we need to separate the parameters using ampersand.



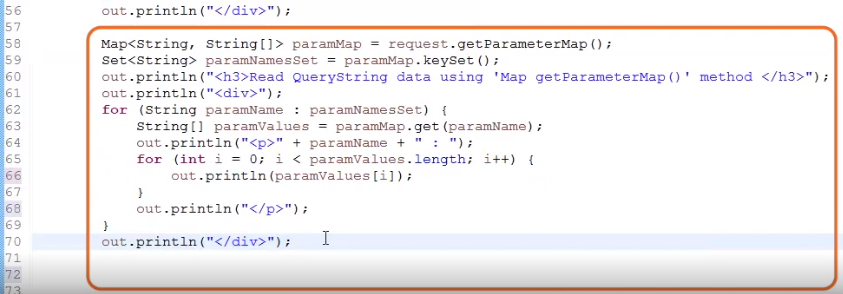
So let me type in? firstName=Sekhar&lastName=Srinivasan. And let me press Enter. We can observe that the Servlet has returned values. Servlets also support other alternate methods to read the values from the parameters submitted to the Servlet page. Assume that we are not aware of the parameter names, but we wanted to extract the values from those parameters. Then this alternate method will help us to read the values. Let me flip to Eclipse again, and this time let me use getParameterNames method. We know that this method will return an enumeration. Now let us write a simple out statement to display Reading QueryString data using enumeration getParameterNames method. So let me type in out. println, h3, Reading QueryString data using 'Enumeration getParameterNames' method. First, let us get the list of parameter names.



So let me type in Enumeration of String paramNames = request. getParameterNames. Now the list of parameter names will be maintained within the paramNames enumeration. Now let us write the code to extract the value and display on the page. I have opened the div block, and then I have defined a while loop which will iterate until there is an element present within that paramNames enumeration. ParamNames. nextElement method will return the name of the parameter. As we got the parameter name, now in order to extract the value based on the parameter name, I have used request. getParameter method to get the parameter value and displayed the name of the parameter and the value within that simple paragraph. Finally, close the div block. Let me execute the page again.



We can observe the values are displayed. We have one more alternate method to extract the values from the parameters without knowing the parameter name, that is, with the support of getParameterMap. Now let us understand how to use this method for extracting the values from the parameters. So let me flip to Eclipse and let me update the code.



I have a simple out statement to display the heading Read QueryString data using 'Map getParameterMap' method. The getParameterMap method of request variable will return a map collection that maintains the parameter name as a key and parameter values as a string array values. Map of String, String array, paramMap = request. getParameterMap. The keySet method will return the set of keys present within the map. I have opened a div block, then I have defined a loop for iterating each parameter present within that parameter namespace. Now we need to extract the value from the map based on the parameter name key. Get is a method which accepts a key from the set and returns the value as a string array. Once we've got the value, I have just prepared a simple paragraph to display the parameter name and the parameter value. Now let me once again execute the page. We can observe the values are extracted and displayed. We can use any of the other methods for extracting the values, but it is always advisable to use getParameter method to extract the value. In the next clip, we shall understand how to create an HTML page for accepting the user input and submit a request to the Servlet page, and also we shall learn how to access the form data from the Servlet page.

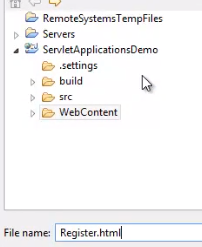
# Handling Form Data in Servlets

=>slides: Pg. 15

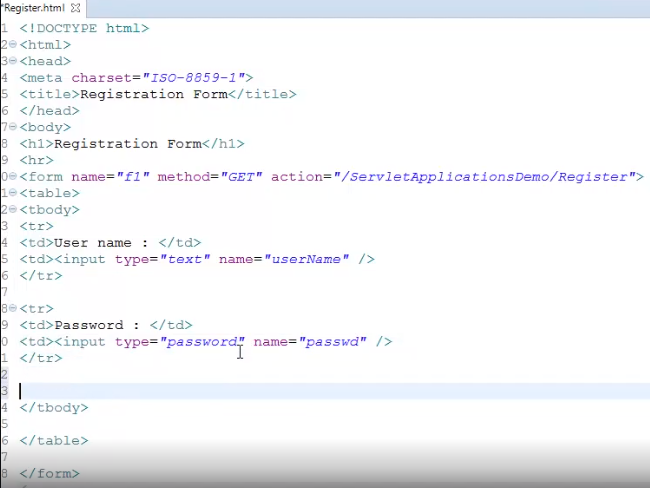
Whenever we try to pass the data the way we had performed in the previous clip, it will not be that easy for the users to submit. Moreover, many users may not have an idea of the parameter names to be used within the query string and how to manually prepare the query string. So it is always advisable to have an HTML form to accept the user input and submit the data to the Servlet page. In a Servlet application, whenever we have a requirement to maintain any static content such as HTML pages, stylesheet files, images, etc., that can be maintained within the WebContent folder, create an HTML page for accepting the user input.



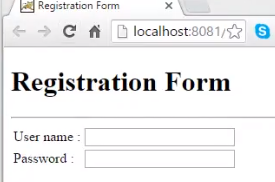
Right-click on the WebContent folder



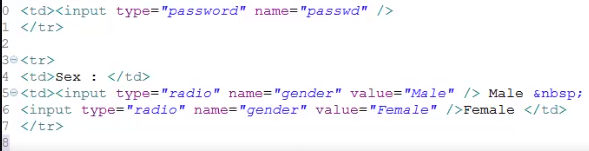
and click on New, HTML File. Provide a meaningful name for the file, for example, Register. html.



Let me also open the browser and execute the page to have a look at the form while designing. Within the HTML page, we can observe some boilerplate code is already written. Now, let us start with our coding. Let me update the title as Registration Form. Let me add a heading for the page within the body tag. So let me type in Registration Form and then just to beautify, I would like to place a horizontal row. Whenever we need to submit the data to any page, we'll require the form type. So let me type in the form type, form name="f1" method. We need to specify the HTTP verb to be used for submitting the request, that is, GET or POST or PUT or DELETE. We need to use GET method whenever we need to read the data, POST method to submit the data for adding a new resource, PUT method to update or change the existing resource at the server, and the DELETE method to remove the resource. For this example, let us use GET. Later we shall update to POST method. And I will also explain in detail when to use GET and POST methods. And then we need to set the action. Well, we have to provide the name of the Servlet page where the data has to be submitted. So let me type in action="/ServletApplicationsDemo/Register". Let me use a simple table for designing the form. But it is advisable to use Bootstrap and CSS for efficient form designing. Now let us start with that table. There are no headers for this form, so let me start with the body of the table, tbody. Let me start the row, tr, td Username, td input type="text" name="userName". Now let me copy the last two and paste and update the values, the text as Password and input type as password, and the name as passwd. Let me refresh the browser.

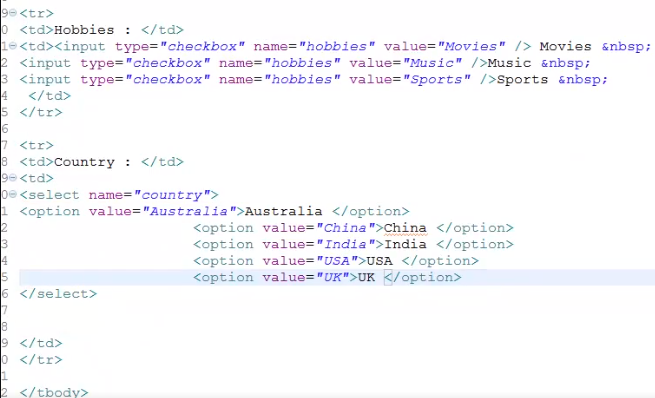


We can observe the form elements are rendered.



Once again, let me paste the code and update the text as Sex and let me remove the textbox. And, instead, let me provide two radio buttons. So I type in input type="radio" name="gender" value="Male" and the text Male. And I would like to provide a single whitespace, so let me type in &nbsp;. And then, again, input type="radio" name="gender" value="female", and the text as Female. I refresh the browser. 

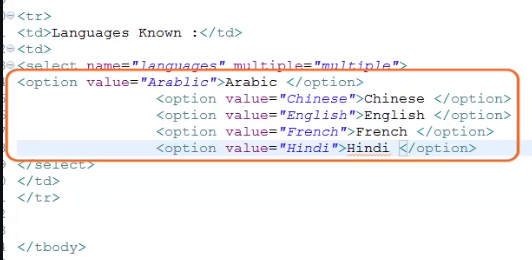
Now let me copy the above code and paste.



Let me update the text as Hobbies and then replace that radio button as checkboxes to provide the hobbies as Movies, Music, and Sports. Let me refresh the browser again to view. Now let me have another row for accepting the country. So let me type in tr, td Country, td, select name="country". I would like to add various countries as options. So let me update the code. I have added some countries as options.

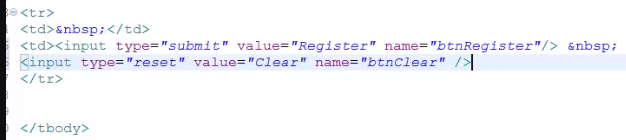


Let me refresh the browser. We can observe the various countries.

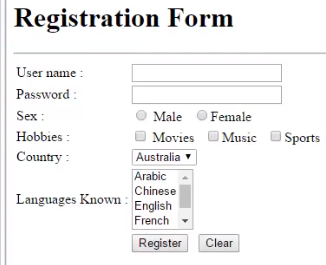


Now let me add one more row for accepting the languages known. So let me type in again tr, td Languages Known, td, select name="languages", and I would like to provide the user for selecting multiple languages. So let me type in multiple="multiple". Let me paste some languages as the options. 

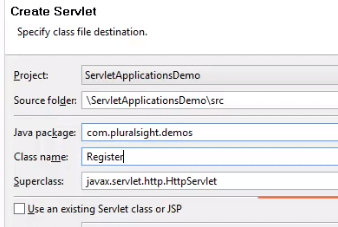
Once again, let me refresh the browser. We can observe the Languages Known is also displayed properly. Now let me add the buttons for submitting that data and to reset the data.



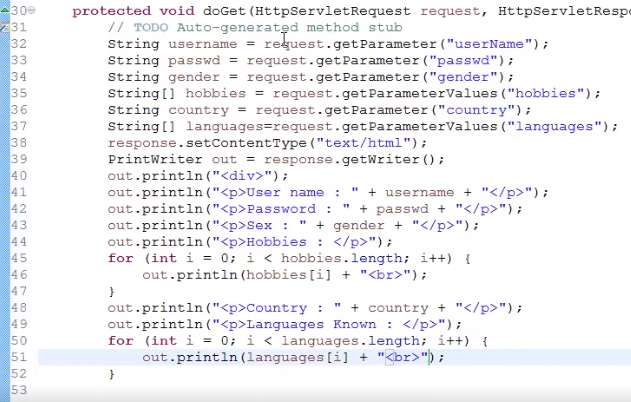
So let me add one more row, tr, td, and I don't require anything in the first column. So let me type in a whitespace, &nbsp;, td input type="submit" value="Register" name="btnRegister", a whitespace, and then the Reset button, input type="reset" value="Clear" name="btnClear".



Once again, let me refresh the browser to view the complete form. Once the HTML page has been designed, the next step is to add a Servlet page for accepting that request and to process the request and to provide that response. So let us add a Servlet page.

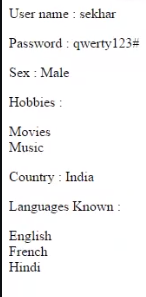


I right-click on the application again and click on New, Servlet. We need to provide the package name and the class name. So let me type in the package name as com. Pluralsight. demos and then the class name as Register. Let the superclass be HttpServlet itself, and if I click on Finish button at this stage, then it will generate the methods for doGet and also for doPost. Since I will use both methods for this demo, let me click on Finish button. Since I was submitting the data using GET method, we need to provide the definitions to handle adding the doGet method. First, let us collect the values from the controls present within the HTML page.



To collect the username, String username = request. getParameter of userName. And to collect the password, String passwd = request. getParameter of passwd. And to collect the value from the radio button Sex, String gender = request. getParameter of gender. The next form element we have is checkboxes hobbies. Since the checkbox allows the user to select multiple values, we need to use getParameter method, which returns a string array. So let me type in String array hobbies = request. getParameterValue of hobbies. And the next form element is select with a single item selection. So let me type in String country = request. getParameter of country. And then the next element is a select with multiple attributes, which again allows the user to select multiple values. So let me type in String array languages = request. getParameterValues of languages. Once we have collected the values, now let us format the output and display on the page. First, let me set the content type of the page's HTML. So I type in response. setContentType of text/html. Since I want to display the output on the page, let me type in PrintWriter out = response. getWriter, and then let me type in a set of out. println statements to display the username, password, and sex. Since the values of hobbies is maintained within a string array, it may use a for loop to iterate and display the hobbies. So let me type in for (int i = 0, i is less than hobbies. length; i++), out. println of hobbies of i. To display the country, let me add another out. println statement. And, again, to display the languages known, let me use the for loop to iterate and display the selected languages.

Now let us execute the Register HTML page. Let me provide the values for the form elements. Username--sekhar, Password--let me type in something, Sex--I select Male, Hobbies--I select Movies and Music, Country--I select India. And the Languages known--I select English, French, and Hindi. Let me click on the Register button.



We can observe the form data has been submitted to the Servlet page using the GET method, and the page has been generated, and the values are displayed. In the next clip, we shall understand how to handle the POST method in Servlets.

# Handling POST Method in Servlets

=>slides: Pg. 16

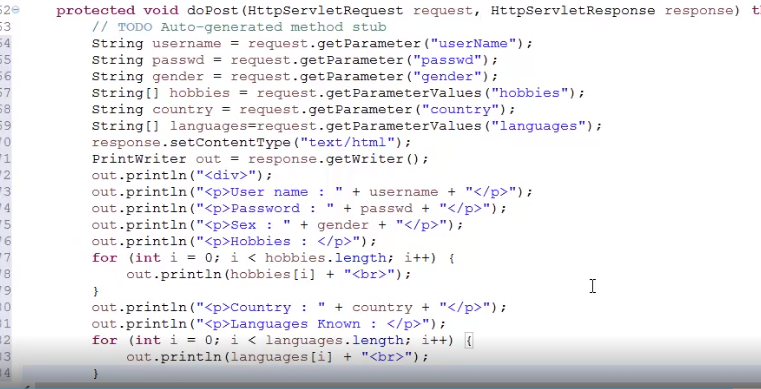
We have observed how to handle the HTTP-GET method request. And before we understand how to handle the HTTP-POST method request,

=>slides: Pg. 17

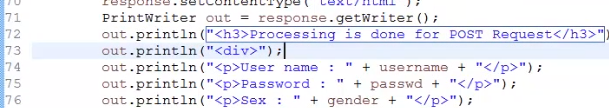
let us first understand the differences between HTTP-GET and HTTP-POST so that we will have a clear idea when to use the GET method and when to use the POST method. Whenever we click on the back button or reload the page, the GET method is harmless. But in POST method, the data will be resubmitted, and the browser will alert the user that the data are about to be resubmitted. GET requests can be bookmarked, whereas POST requests cannot be bookmarked. GET requests can be cached, and POST requests won't support cache. In GET requests, the parameter values will be maintained at the browser history, whereas in the POST request, the parameters are not saved in the browser history. Whenever we submit the data using the GET method, there will be a restriction on the size of data that has been submitted to the server, maximum URL length will be 2048 characters, whereas POST method doesn't have any size restrictions. In GET method, only ASCII characters are allowed, whereas in POST requests, no restrictions on that data type even binary data is allowed. GET method is less secure because the data is sent over the URL, and the data is visible to everyone. And, hence, it is not advisable to use GET method while using password or any other secure information, whereas POST method is safer than the GET method. As the data will not be shown at the URL, and it won't be stored in the browser history or other web server locks. Now let us understand what are the changes we need to perform to work with POST method. Let me first flip to the Register. html file at the Eclipse.



And the only change we need to perform in the HTML page is within the form element tab, we need to update the method from GET to POST. So let me update that. Since the HTML page now will use HTTP-POST method for submitting the data at the server page, we need to use doPost method for handling the request. So let me open the Register. java Servlet page.



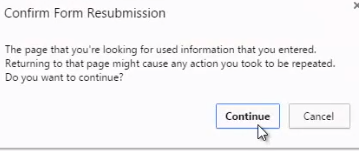
Whatever the code we have written in doGet method will work for the doPost method also. Let me copy the code from the doGet method and paste the code at the doPost method.



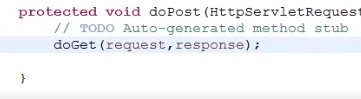
Just to have the difference, let me use an out statement to display processing this from a POST request. So let me type in out. println, Processing is done for POST Request. Now let me execute that registration form. And let me provide the values for the form and click on Register. We can observe the data has been displayed.



And at the URL, we can observe the data is not appended.



When we try to reload the page, we can observe and elect for the data reload, which will help us in preventing from accidental post back to the server. Instead of copying the code to the doPost method, we can also alternatively call the doGet method from the doPost method. Let me open the Eclipse again, open the Register. java page.



Let me remove the code, which I have copied, and, instead, let me invoke the doGet method. To do, let me type in doGet of request, response. Let me execute the registration form again and provide the values. Let me click on the Register button. We can observe the same output.

# Summary

=>slides: Pg. 18

In this module, we have learned how to work with HTTP-GET requests and how to work with HTTP-POST requests and also understood how to handle the query string and form data. In the next module, we shall understand how to handle HTTP requests and responses.

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