4. Using the Request and Server Context through Built-in Objects

Contents

[1. Introduction 1](#_Toc22889)

[2. Request 1](#_Toc10735)

[3. Response 1](#_Toc17192)

[4. Out 2](#_Toc28555)

[5. PageContext 2](#_Toc30952)

[6. Session 2](#_Toc3375)

[7. Application 2](#_Toc27508)

[8. Config 2](#_Toc15052)

[9. Demo: Extending Virtual Training Company Application Using Built-in Objects 2](#_Toc7165)

[10. Summary 3](#_Toc8624)

# 1. Introduction

=>slides: Pg. 1

JSP built-in objects are the Java objects that the JSP container makes available to developers in each page. These are given to the developers for free, so we don't have to create them explicitly and the developer can use them directly without declaring them. These objects can be used within the code JSP service methods, that is, these built-in objects can be used within the scriptlet tags and also within the expression tags, but they cannot be used within the declaration tag. As we know that the code written within the declaration tag will be written in the global declaration section of the generated servlet after the translation phase.

=>slides: Pg. 2

In this module we shall understand the built-in objects provided by the JSP API. We shall first understand the individual built-in objects provided by the JSP and once we have understood the usage of built-in objects, then we shall extend our virtual training company web application by adding the registration and login functionalities using the built-in objects. But now let us start understanding the JSP built-in objects.

# Request

=>slides: Pg. 3

Whenever a client sends a request to the JSP page, then the request will be submitted along with some header information and the body information. And once the JSP page has been translated into the servlet, then we know that JSP service is the method which will handle the request and response of the client. If we observe the syntax of JSP service method, we can notice that the method is accepting two parameters. The first parameter is the request, which is defined for the HTTP servlet requestor class and to refer the HTTP servlet request associated with the client request directly, we can use the request built-in object provided by the JSP API within the JSP page. The request object provides methods to get HTTP header information including form data, cookies, and HTTP methods.

=>slides: Pg. 4

Now let us understand some of the important methods that we frequently use from request object. SetAttribute, it is used to set a value within the request page scope of the JSP page. GetAttribute returns the value of the named attribute as an object or null if not attribute for the given name exists. GetCookies returns an array of all cookie objects containing within the client's request. GetHeaderNames returns an enumeration of all the header names this request contains. GetHeader of String name returns the value of the specified request header as a string. GetMethod returns the name of the HTTP method with which this request was made, for example, getPush, put or delete. GetParameter of String name returns the value of a request parameter as a string or null if the parameter doesn't exist. GetQueryString of String name returns the query string present in the request URL after the fact. GetLocale returns the preferred locale for the client. Now let us understand how to use the request built-in object.

=>slides: Pg. 5

Say that we have a guestbook. jsp page and let us assume that we wanted to collect the name of the guests so we might define a form tag and a textbox to collect the value.

=>slides: Pg. 6

For example, input type=text, name=guestName. To submit the data we require a submit button, input type=submit, name=btnsubmit. And we need to inform that where the data should be submitted. For that we need to set the action attribute at the form tag. For example, action equal to welcome. jsp. Now once we have created the guestbook. jsp,

=>slides: Pg. 7

you would like to create the welcome. jsp page where we need to collect the value of the guest. So within the welcome. jsp page, let me type in h3 welcome and let me define an expression tag and in order to collect the value of the guest, we can use the request built-in object by calling request. getParameter of guestname, and let us display some more additional information about the guest and I wanted to know what type of browser and -- of the guest being used. So let me type in paragraph Request User Agent and an expression, request. getHeader of User-Agent. And let us say I wanted to know the client's browser language, so I can type in paragraph Guest Language an expression and then request. getLocale.

=>slides: Pg. 8

If we execute the guestbook. jsp page and if I provide the guestname as Sekhar and click on Submit, =>slides: Pg. 9

we can observe the guest name, the user agent details, and also the locale used by the client. In the next section let us understand response built-in object.

# Response

=>slides: Pg. 10

Whenever a client sends the request for the JSP page to the server, the JSP container will create two objects, one to handle the client request and the other to provide the response to the client, and we know that these objects will be provided to the JSP service method for processing. Like how we have request object as an instance for HTTP servlet request, similarly JSP API has provided response object as an instance to the HTTP servlet response. Response object is used mainly to set the content type of the response or to set the HTTP headers that can be used to set the cookies and also response object can be used to redirect the user from one page to the other page.

=>slides: Pg. 11

Now let us understand some of the important methods that we use with the response object. SetContentType of String type, it is used to set the content type of the response being sent to the client. For example, response. setContentType of text/html. SetIntHeader of String name, Int value. It is used to set a response header with the given name and value. For example, response. setHeader of Refresh, 5. SendRedirect of String location, it is used to send a temporary redirect response to the client using the specified redirect location URL. For example, response. sendRedirect of success. jsp. AddCookie of Cookie. It is used to add the cookie object to the response, for example, response. addCookie of userCookie.

=>slides: Pg. 12

Now let us understand how to use the response object.

=>slides: Pg. 13

Assume that we wanted to refresh the JSP page for every 3 seconds,

=>slides: Pg. 14

then we can define an expression and we can set the response header using response. setIntHeader of Refresh, 3. And I just wanted to display the date and time to show the last refreshed date time. To do, we can define an object for the date by typing Date currentDate=new Date, and to get the current date and time in a proper formatted way we can use the simple data format. So we can type in SimpleDateFormat ft=new SimpleDateFormat of the format to be used for displaying the date and time. And now let me define a variable to hold the current date and time. So let me type in String currentDateTime=ft. format of currentDate. And now let us display the value. So we can type in paragraph Page Last Refreshed: an expression CurrentDateTime and when we save this JSP page and execute,

=>slides: Pg. 15

then we can observe this page will be refreshed for every 3 seconds. Later in our course we will be using the response object for various requirements while extending the virtual training company web application. In the next section we shall understand the Out built-in object.

# Out

=>slides: Pg. 16

When we used to work with servlets, in order to display the content in a response we used to instantiate an object for the print writer class with the support of response. getWriter method.

=>slides: Pg. 17

For example, Printwriter out=response. getWriter. But within JSP you don't have to perform all this. We have been provided with the built-in object Out to achieve the same functionality where Out is an instance of Javax. servlet. jsp. jspWriter object, which is created with the support of pagecontext. getOut method. And it is used to send content in a response. The initially jspWriter object is instantiated differently depending on whether the page is buffered or not. We know that buffering can be easily turned off by using the buffered equal to false attribute of the page directive. The jspWriter object contains most of the same methods as the Java. io. Printwriter class, however jspWriter has some additional methods designed to deal with buffering. Unlike the Printwriter object, jspWriter throws IOExceptions.

=>slides: Pg. 18

Some of the important methods we use without built-in objects are print. It is used to display the value. Println, it is used to display the value and once the value is displayed, then a new line brake will be inserted at the end of the content. Flush, it is used to flush the content present within the buffer.

=>slides: Pg. 19

For example, to display the current date and time we can use a scriptlet out. print of Today is Java. util. Calendar. getInstance. getTime. In the next section we shall understand about the page context.

# PageContext

=>slides: Pg. 20

The PageContext object is an instance of a javax. servlet. jsp. pageContext object and it is used to represent the entire JSP page. This object is intended as a means to access information about the page while avoiding most of the implementation details. PageContext reference to getAttributes from any scope including the page scope for attributes bound to the pageContext. The methods that work with other scopes take an int argument to indicate the scope. Although the attribute access methods come from JSP context, we can find the constants for the scope inside the PageContext class.

=>slides: Pg. 21

Now let us understand some of the important methods belongs to PageContext. SetAttribute, it is used to set the attribute value at the different scopes. GetAttribute, it is used to read the value from the attributes from different scopes. FindAttribute, it is used to search for an attribute at the PageContext and if the attribute is not present, then the method starts looking in other scope from most restricted to least restricted scope, that is first request scope, then session, and finally application scope. The first one it finds with the name wins. RemoveAttribute, it is used to remove an attribute from all the scopes. Other than these methods we have merely 40+ methods such as getRequest, getServletConfig, getServletContext, getSession, et cetera. Some of these methods while understanding other built-in objects I will be explaining. Along with the methods, we have been provided with some static final fields such as application scope, page scope, request scope, session scope to represent the various scopes. =>slides: Pg. 22

Now let us understand some examples using PageContext to get and set attributes. To set a page-scoped attribute within the scriptlet, let us assume we have a float variable number defined with the value 62. 5. To set the page-scoped attribute we can use pageContext. setAttribute of number, number and to read a page-scoped attribute within the expression, we can use pageContext. getAttribute of number. If we wanted to set as a session-scoped attribute, then we need to pass the session scope as the third parameter, for example pageContext. setAttribute of number, number, PageContext. SESSION\_SCOPE, and to read the session-scoped attribute value within the expression, pageContext. getAttribute of number, PageContext. SESSION\_SCOPE. In the next section let us understand the session built-in object.

# Session

=>slides: Pg. 23

Whenever we have a requirement to maintain the information provided in one JSP page across the other JSP pages present within the web application, we need to perform session tracking. By default in JSP pages, session tracking is enabled and a new http session object is instantiated for every new client request.

=>slides: Pg. 24

The JSP container exposes the HTTP session object to the JSP developer through the session built-in object where session object is an instance of javax. servlet. http. HttpSession created with the support of pageContext. getSession. Using the session object, we can set our get or remove attribute from the session information.

=>slides: Pg. 25

Let us understand some of the important methods that we need to remember while using session built-in object. SetAttribute of String name, Object value. It is used to bind a value to the session using the specified name. GetAttribute of Stringname, this method returns the object bound with the specified name in the current session or returns null if no object is found with that name. RemoveAttribute of String name, this method removes the object bound with the specified name from this session.

=>slides: Pg. 26

For example, in order to set the session using the pageContext, within the scriptlet pageContext. setAttribute of number, value, PageContext. SESSION\_SCOPE we have used. The same if you wanted to achieve using session built-in object, we can write as session. setAttribute of number, value. In the next section we shall understand what is application built-in object?

# Application

=>slides: Pg. 27-28

The Application object is a direct wrapper around the servlet context object for the generated servlet and in reality it is an instance of Javax. servlet. ServletContext object defined based on the pageContext. getServletContext. For the people were not aware of what is Servlet Context, just to understand that the servlet context is an object that contains meta information about our web application. Just like session object, we can store attributes in the servlet context.

=>slides: Pg. 29

The attributes stored in the servlet context are available to all servlets in our application and also between requests and sessions, that means that the attributes are available to all visitors of the web application, whereas session attributes are just available to a single user. The servlet context attributes are stored in the memory of the servlet container.

=>slides: Pg. 30

To set the value in application object, application. setAttribute of String name, Object value should be used.

=>slides: Pg. 31

And to read the value from the application object, we can use application. getAttribute of String name. By adding an attribute to application built-in object, we can ensure that all JSP files that make up our web application will have access to the information. In the next section we shall understand the next built-in object that is config.

# Config

=>slides: Pg. 32

Config built-in object is used for getting configuration information for a particular JSP page. Using application implicit objects, we can get application wide initialization parameters, however, using Config we can get initialization parameters of an individual servlet mapping.

=>slides: Pg. 33

The config object is an instantiation of javax. servlet. ServeletConfig created with the support of pageContext. getConfig, and it is a directive wrapper around servlet Config object for the generated servlet.

=>slides: Pg. 34

Now let us understand some of the important methods we use with the config implicit object. GetInitParameter of String name, it returns the value of initialization parameter for a given parameter name. GetInitParameterNames returns an enumeration of initialization parameters. GetServletContext, this method returns a reference to the servlet context. GetServletName, this method returns the name of the servlet which we defined in the web. xml file inside servletName tag. As we have understood the built-in objects provided by JSP,

=>slides: Pg. 35

in the next section we shall use these built-in objects practically by extending our virtual training company application to support registrations

=>slides: Pg. 36

and login.

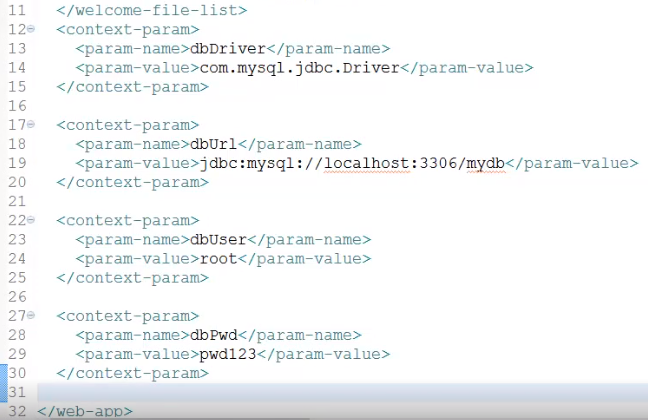
# Demo: Extending Virtual Training Company Application Using Built-in Objects

=>slides: Pg. 37

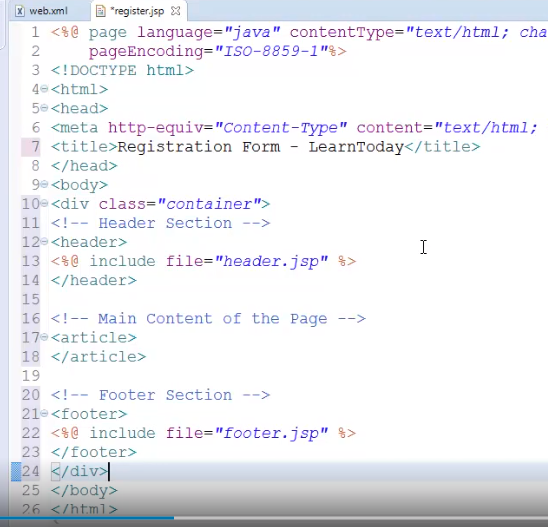
Let us first observe the finished application. In the home page we have got two links, one for the new users to get registered and the other for the existing users to login. Now let us observe the register and the login page. We have some user input elements to accept the username and the password.

=>slides: Pg. 38

And when we click on the Register button, we need to add the user to our application and when we click on Login, we need to authenticate if the user is valid or not. Both these pages uses the same table users, which has username and password as the columns. I have taken the liberty to create the table users within my SQL database. Since we would like to use the same database for all the pages, let us first configure the database settings within web. xml file such that these details can be accessed from all the JSP pages.



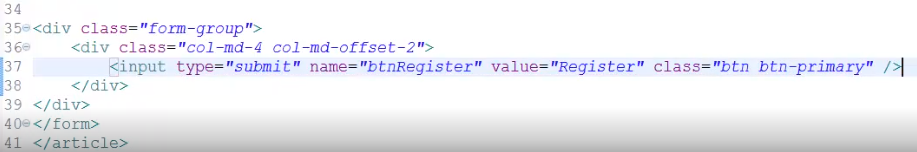
Now let me open the web. xml file and let me type in the details for the database driver. Context-param, param-name, dbDriver, param-value com. mysql. jdbc. Driver. Similarly, let me add the code for the database URL, database user, and the password.



Now let us first create the registration form. Let me right click on the WebContent folder and click on New, JSP File and let me provide the name for the JSP pages, register. jsp, and let us update the title for the page as Registration Form - LearnToday. We require the header and footer and the article section to provide the user input elements. So let me add the code we can observe and include for the header and a blank article section and an include for the footer.



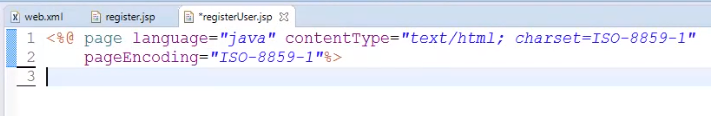
Now within the article, let us add the code to provide the user elements. Let me first add a header for the page, so let me type in h1 class=page-header and the Registration Form. Then we require a form element for submission. So let me type in form name=registrationForm, class=form-horizontal. When we click on the Register button, I would like to submit the data to the register user page. So let me type in action=registerUser. jsp and the method for submitting the data I would like to use is post. Let us add a form group for accepting the username. Let me type in div class form-group and then a label, class= and this label I wanted to occupy 2 cells in the grid, so let me type in col-md-2 control-label, then the User name. Now let me add another div and I want this to occupy 4 cells, so let me type in the class as col-md-4, then I require the textbox for accepting the username. So let me type in input type=text class=form-control name=username, and I want the user to provide the username before submission. So let me type in required=required. I want to accept the password also. So let me copy the above code and paste the code. Now let me change the username text as Password and the input type should be password, so let me update the text to password, and the name I want to provide for the element is password. So let me change the username to password.



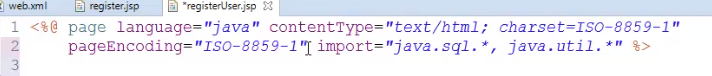
Finally, I want to provide a Submit button to submit the data to the registered user. jsp page. To do, let me type in div class=form-group and then a div class=col-md-4 with a col-md-offset-2 so that he content will skip 2 columns to place the button. Now let me type in input type=submit name=btnRegister value=Register class=btn btn-primary.



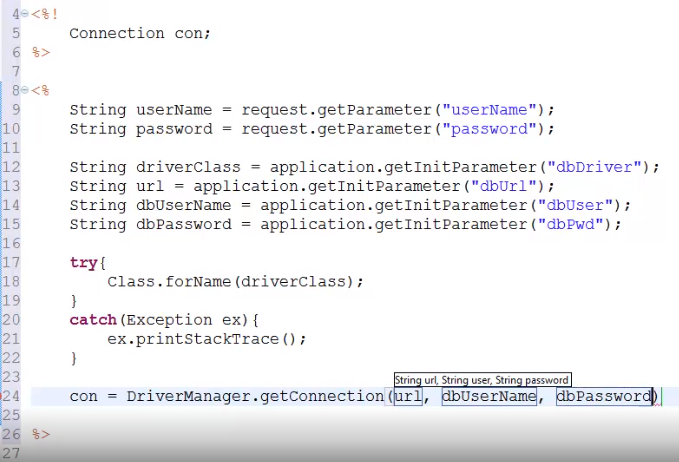
Now let me create the registerUser. jsp page.



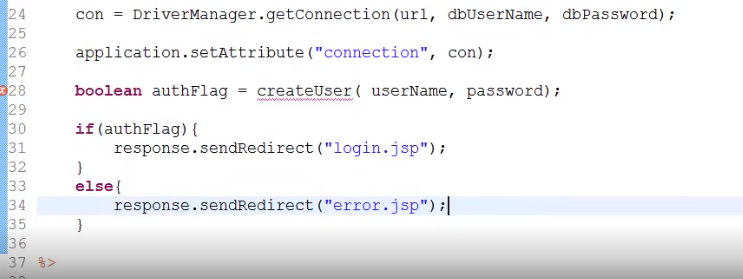
Since this page requires the interaction with the database using JDBC,



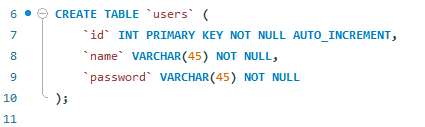
let me add the input attributes for the page directive. Import=Java. sql. \*, Java. util. \*. Let me first define a declaration expression and define a variable for the connection.



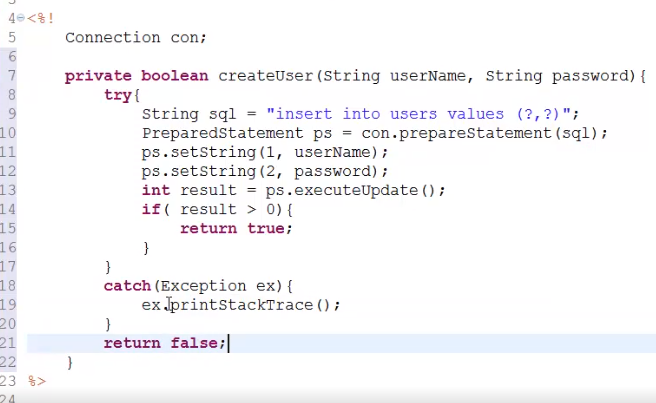
So let me type in declaration Connection con. Now let me add the scriptlet to read the values of username and password. So let me type in String username = request. getParameter of userName, String password = request. getParameter of password. We have provided the database connection information within the context of web. xml file. So let me type in String driverClass = application. getInitParameter of dbDriver. String url = application. getInitParameter of dbUrl, String dbUserName = application. getInitParameter of dbUser, String dbPassword = application. getInitParameter of dbPwd. Let me register the driver class. Since the statement throws an exception, let me use a try block and within that block let me type in Class. forName of driverClass, catch of Exception ex, ex. printStackTrace. From JDBC4 onwards, registering the driver is automated, so you won't require the above code. Now let us establish the connection with the database. So let me type in con = DriveManager. getConnection of url, dbUserName, dbPassword.



Now let me add the connection object to the applications code so that that object will be available for other JSP pages. So let me type in application. setAttribute of connection, con. For maintainability, let me invoke a method to create the user and collect the result in a flag. So let me type in Boolean flag equal to createUser of userName, password. If the user is registered, the flag value will be true, else the flag value will be false. So let me type an if of flag response. sendRedirect of login. jsp, else response. sendRedirect of error. jsp.



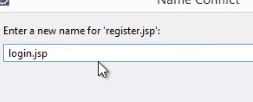




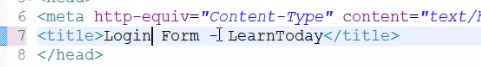
As we have used createUser method, let us define the method, so within the declarations let me type in private Boolean createUser of String userName, String password. Let me add a try block. So let me type in try String sql = insert into users values, and I need to provide the values for username and password, so let me type in?,?. Since the SQL statement is not complete, let us use a prepared statement, so let me type in PreparedStatement ps = con. prepareStatement of sql. And since we have two parameters, let us set the parameters, ps. setString of 1, userName, ps. setString of 2, password, int result = since the statement is an insert statement, let us use execute update. So let me type in ps. executeUpdate. To provide a confirmation, let me type in system. out. println of statement executed. If the result is greater than 0, then I would like to return true. So let me type in if of result greater than 0, return true. Let me add a catch block, catch of Exception ex ex. printStackTrace, and finally let me return false.

=>slides: Pg. 39

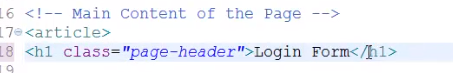
Now we’ll try to design the login. jsp page for providing the user interface and authenticate. jsp page for performing the validations. Hope you have completed the login and the authenticate. jsp pages. If you have completed, very good, else just follow along with me to complete the task.



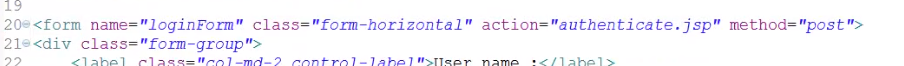
Let me first copy the register. jsp and paste. Let me provide the name for the file as login. jsp.



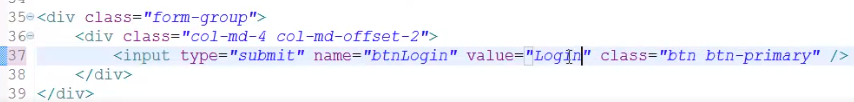
Let me update the title for the page, Login Form LearnToday,



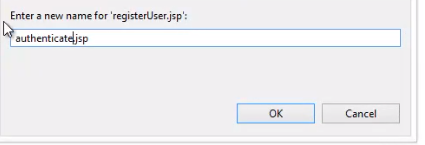
and then let me update the form name as Login Form



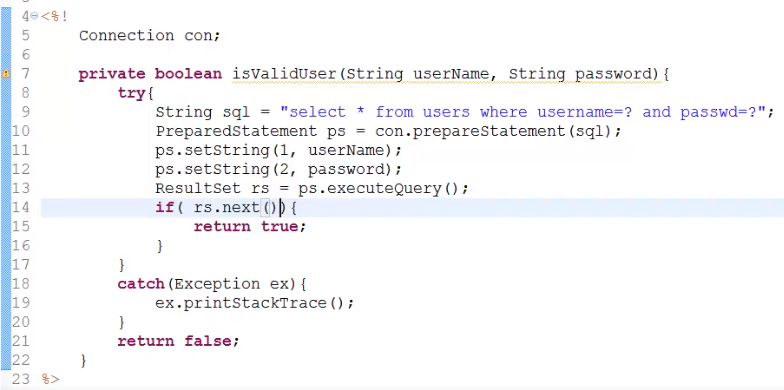
and the action value as authenticate. jsp.



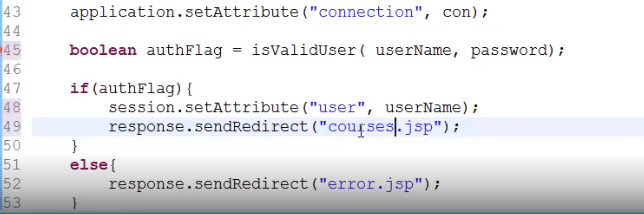
And finally the Submit button value as Login. Let me save the login page.



Now let me copy the registerUser. jsp file and paste and let me provide the name for the page as authenticate. jsp.



Now within the declarations, let me update the name of the method as isValidUser and since our requirement is to verify if the provided user credentials is valid or not, let me update the SQL statement as select \* from users where username=? and passwd=?. Since the SQL statement is a query, in order to execute we need to use the executeQuery method. So let me type in ResultSet rs = ps. executeQuery. And let me update the condition also. If the user is valid, then we would have retrieved a single record, else the ResultSet will be null. So let me update the condition as if of rs. next, we'll return true, else the method returns false.



Finally, let us update the method name present at the scriptlet to use the isValidUser. So let me change the createUser method to isValidUser, and if the flag is true, let me set the userName into the session. So let me type in session. setAttribute of user, userName and then once the user is authenticated, I would like to redirect the user to courses page. So let me type in response. sendRedirect of courses. jsp. Now let us save the file. If we have done all these changes, then it is really nice, else you don't have to worry, we are still in the beginning. Now let us execute our application. Let me first click on the Register link and let me provide the username as Sekhar and the password as Srinivasan and click on Register. Once the user is registered, we have been navigated to the login page. Let me provide my credentials and click on Login. We can observe an error because we don't have the courses page yet.

# Summary

=>slides: Pg. 40

In this module we have understood the importance of JSP built-in objects and its usage by exploring the individual objects in detail and using them effectively within our virtual training company web application. In the next module we shall understand how to handle the exceptions in JSP and also let us update our virtual training company application to support the exception handling and prepare a custom error page for our application.

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