8. Encapsulating Core Functionality Using JSP Standard Tag Library(JSTL)

Contents

[1. Introduction 1](#_Toc28096)

[2. Core Tags 1](#_Toc20830)

[3. Formatting Tags 1](#_Toc19142)

[4. SQL Tags 2](#_Toc23858)

[5. JSTL Functions 2](#_Toc10165)

[6. Demo: Extending Virtual Training Company Application Using JSTL 2](#_Toc2509)

[7. Summary 2](#_Toc2513)

# 1. Introduction

=>slides: Pg. 1

In the last module we have understood what is Expression Language, but in order to use Expression Language efficiently, we need to understand one more powerful feature, JSP standard tag library, popularly known as JSTL of JSP. Let us first understand what is JSTL and why we need to use JSTL. Java server pages tag library is the part of the Java Enterprise Edition web application development platform.

=>slides: Pg. 2

The main features of JSTL technology is to provide a set of standard tags for performing the most common task such as loops, conditional execution, database access, internationalization, and XML processing while developing JSP pages. It can use an Expression Language for accessing the server side objects and also it provides the mechanisms for defining extensions to the JSP language. Like many libraries, JSTL tag library is further divided into various sections, which are properly called as function areas.

=>slides: Pg. 3

JSTL supports five functional areas core, formatting, database, functions, and XML. Where each functional area will target a specific functionality and will have a specific prefix and a URI to inform the page which functionalities can be used within the page. In this module we shall understand how JSTL will help JSP developers for encapsulating the core functionality with the support of JSTL code tags in detail and then we shall understand the various formatting tags, SQL tags, and JSTL functions. Once we get a fair idea on JSTL tags

=>slides: Pg. 4

we shall extent the virtual training web application

=>slides: Pg. 5

to display all the courses present within the library using JSTL and Expression Language. Now let us first understand the JSTL core tags.

# Core Tags

=>slides: Pg. 6

The core group of tags are the most frequently used JSTL tags and in order to use the core tags in our application, we need to include a tag library.

=>slides: Pg. 7

The syntax to include the tag library will be returned within the directive taglib prefix=c, which is the common name provided for the core tag library. And then URI=http://Java. sun. com/jsp/jstl/core, which provides the data type definitions for the core tags.

=>slides: Pg. 8

There are 14 core tags supported by JSTL which helps in encapsulating the JSP page efficiently. Now let us understand these core tags one by one. Out tag is used to display the result of an expression, it is similar to the expression tag. Set tag is used to assign a value for the object or invariable. Remove tag is used to remove a scoped variable from a particular scope if specified. Catch tag is used to catch any throwable that occurs within its body and optionally exposes it. If tag is a conditional tag which is used to execute its body if the provided condition is true. When there is a requirement for multiple if statements to be executed, usually in Java we can take the support of switch case and in order to achieve the same using JSTL we can use choose, when, and otherwise tags. Where choose tag is a simple conditional tag which establishes a context for mutually exclusive conditional operators marked by when and otherwise. This tag doesn't have any attributes. When is a subtag of choose that executes the body if the condition evaluates to true, and this tag supports only one attribute test where we need to provide the expression to be evaluated, and finally otherwise tag is also a subtag of choose tag that follows when tags and runs only if all the prior conditions evaluated to false and this tag also doesn't have any attributes. Import tag provides all of the functionality of the include action, but also allows for inclusion of absolute URLs. JSTL code also supports tags to control the flow and also allows to repeat the execution of the statements. To achieve that, JSTL code provides foreach tag where foreach tag is a basic iteration tag which works similar to forloop in Java. Similar to foreach tag we have another code tag which is used for iteration called as forTokens tag. It is used to iterate over tokens separated by the supplied delimiters. Param tag allows proper URL request parameter to be specified with URL. Also it performs the necessarily URL encoding if required. Redirect is used to redirect to a new URL. URL is used to create a URL with the optional parameters. Now let us understand how to work with JSTL code tags.

=>slides: Pg. 9

In order to work with JSTL code tags, first we need to include the tag library. So let me add the code to include the tag library. Now let me define a variable to hold the value 10. So let me type in c:set var=no1 value=10. Let me display that value. So let me type in paragraph value of no c:out var=no1. Now let me check out what will happen if I divide this number by 0. So let me type in c:catch var=errorMessage, within the catch body let me add a scriptlet int x = $ curly braces of no divided by 0, and if we get an error, I would like to show that, so let me type in paragraph $ curly braces errorMessage. I wanted to check whether the given number is even or not. To do, let me add a condition, c:if test=$ curly braces no1 modulus 2==0. Within the if body, let me type in paragraph $ curly braces no1 is an even number. Now let us understand how to use the choose, when, and otherwise tags also. So let me type in c:set var=no2 value=5,

=>slides: Pg. 10

and then let me add the code for the choose, when, and otherwise. I have provided two when conditions, the first one verifying if no1 is greater than no2. Then I'm displaying the no1 is greater than no2, and the second when condition I'm verifying is no2 is greater than no1. And if this condition is satisfied, then I'm displaying a message, no2 is greater than no1. And if both conditions are failure, then I have added an otherwise where I'm displaying no1 and no2 both are equal.

=>slides: Pg. 11

Let me add a foreach tag to display the values from 1 through 5. To do, let me type in c:forEach var=i, begin=1, end=5. Within the foreach body, let me type in paragraph Item c:out value=$ curly braces of i. And finally, let us also observe how to work with forTokens. So let me type in c:forTokens items=JDBC, Servlets, JSP delims=, var=courseName. Now I wanted to display the courseName, so within the forTokens body, let me type in paragraph c:out value=$ curly braces of courseName.

=>slides: Pg. 12

Now let us execute the page. We can observe the following result. We have observed some of the most frequently used JSTL code tags. In the next section we shall understand JSTL formatting tags, which helps in formatting the JSP page contents effectively.

# Formatting Tags

=>slides: Pg. 13

Once we have got a fair idea on the core JSTL tags, now let us understand the formatting tags supported by the JSTL. JSTL formatting tags are used to format and display the text, date, time, and numbers for internationalized websites.

=>slides: Pg. 14

In order to use the JSTL formatting tags, we need to include taglib prefix=fmt the uri= http://java. sun. com/jsp/jstl/fmt.

=>slides: Pg. 15

Now let us understand various JSTL formatting tags. FormatNumber used to render numerical value with specific precision of format. ParseNumber is used to parse the string representation of number, currency or percentage. FormatDate is used to format a data or time using the supplied styles and pattern. ParseDate is used to parse the string representation of a date or time. Bundle is used to load a resource bundle to be used by its tag body. SetLocale is used to store the given locale in the locale configuration variable. SetBundle is used to load a resource bundle and store it in the named scope variable or the bundle configuration variable. TimeZone is used to specify the time zone for any time formatting or parsing actions nested in its body. SetTimeZone is used to store the given time zone in the time zone configuration variable. Message, it is used to display an internationalized message. RequestEncoding, it is used to set the request character encoding.

=>slides: Pg. 16

Now let us practically observe how to use the JSTL formatting tags to format the numbers and date. We have a simple JSP page. Now in order to use the JSTL formatting tags, we need to add the tag library directive. So let me type in taglib directive prefix=fmt uri=http://java. sun. com/jsp/jstl/fmt. Now within the body, let me set two variables, set variable=now value= expression new java. util. Date and some values for the date and time. Let me define one more variable to define a number. So let me type in set var=no value=4. 5. Now let us understand how to format the above date and number using the JSTL formatting tags. So let me type in Currency formatNumber type=currency value=$ curly braces of no. Now I would like to display the number in a specific number pattern, so let me type in paragraph Number Pattern formatNumber type=number pattern= pattern to be used for formatting the number value=$ curly braces of no. Now if I wanted to display the number as a percentage value, let me type in paragraph Percent formatNumber type=percent value=$ curly braces of no. Now let me display the date in a specific format. Paragraph Date - Pattern formatDate pattern= pattern to be used for displaying the date value=$ curly braces of now. Now let me add the code to display the date and time in different styles. Paragraph Date - Long and Short Time formatDate type=both dateStyle=long timeStyle=short value=$ curly braces of now. Paragraph Date - Medium and Long Time formatDate type=both dateStyle=medium timeStyle=long value=$ curly braces of now. If you wanted to display only date, then we can type in par graph Only Date formatDate type=date value=$ curly braces of now. And if you wanted to display only time then paragraph Time formatDate type=time value=$ curly braces of now. =>slides: Pg. 17

Now if we execute this page we can now observe how the date and numbers has been formatted in different styles. I have listed many JSTL formatting tags while explaining and used only the most frequently used formatting tags in the above demo. Feel free to explore the other formatting tags also. In the next section we shall understand how to use JSTL SQL tags.

# SQL Tags

=>slides: Pg. 18

In the last section we have understood how to use the JSTL formatting tags, now let us understand how to use JSTL SQL tags. JSTL SQL tag library provides the tags for interacting with the relational databases, such as Oracle, MySQL or SQL Server.

=>slides: Pg. 19

In order to include the JSTL SQL tag library within the JSP page, we need to use taglib prefix=SQL uri=http://java. sun. com/jsp/jstl/sql.

=>slides: Pg. 20

Now let us understand the various JSTL SQL tags. SetDataSource, it is used to create a simple data source. Query tag is used to execute the SQL query defined in its body or through the SQL attribute. Update tag is used to execute any valid SQL data manipulation language statements such as inset, update, or delete statements defined in its body or through the SQL attribute. Param tag is used to set a parameter in a SQL statement. DateParam tag is used to set a parameter in a SQL statement to the specified Java. util. date value. Transaction tag is used to provide a nested database action elements with a shared connection and used to execute all statements as one transaction.

=>slides: Pg. 21

Now let us understand how to work with JSTL SQL tag library practically. We have a simple JSP page. In order to work with JSTL SQL tags, first we need to add the taglib for the JSTL ssql tag. So let me type in taglib directive prefix=sql uri=http://java. sun. com/jsp/jstl/sql. Now within the body, first we need to define the data source. To do, let me type in setDataSource var=myDB driver=com. mysql. jdbc. Driver url=jdbc:mysql://localhost/psdemo user=root password=pwd123. Assume that we have a table with the name departments where I would like to add a record. To do, let me type in update dataSource=$ curly braces of myDB var=count, and within the update tag body let me write the insert statement to add the record. So let me type in insert into Departments values of 10, SALES, Hyderabad. Now let me write the code to retrieve the data also. To do, let me type in query dataSource=$ curly braces of myDB var=rsDepartments, and then within the query tag body let me add the select statement to retrieve the data. So let me type in select \* from Departments. In order to display the data, let me add the code to prepare the table. Now let me add a foreach tag to iterate within the rows retrieved, and let me display the data, foreach var=dept items=$ curly braces rsDepartments. rows. Now let me add the code to display the department number, department name, and location.

=>slides: Pg. 22

Now once we execute the page we can observe the record has been inserted and also the data has been displayed. In the next section we shall understand how to work with JSTL function tags.

# JSTL Functions

=>slides: Pg. 23

JSTL tags also includes a number of standard functions, most of which are common string manipulation functions.

=>slides: Pg. 24

In order to use the JSTL functions tags we need to include taglib directive prefix=fn uri=http://java. sun. com/jsp/jstl/function.

=>slides: Pg. 25

Now let us understand the various JSTL function tags. Contains tag is used to verify if an input string contains the specified substring or not. ContainsIgnoreCase tag is used to verify if an input string contains the specified substring in a case insensitivity. EndsWith tag is used to verify an input string ends with a specified suffix. IndexOf tag is used to return the index within a string of the first occurrence of a specified substring. Join tag is used to join all the elements in an array into a string. Length tag is used to return the number of items in a collection or the number of characters present in a string. Split tag is used to split a string into an array of substring. StartsWith tag is used to define an input string starts with a specified prefix. Substring tag is used to return a subset of a string. ToUpperCase tag is used to convert all of the characters of a strong to uppercase. ToLowerCase tag is used to convert all of the characters of a string to lowercase. Trim tag is used to remove all the white spaces from both ends of a string.

=>slides: Pg. 26

Now let us understand with a simple example how to use the JSTL function tags. In order to work with JSTL function tags, first we need to include the tag library. So let me add the code to include the tag library. Within the body tag, let me set the variable with the string set var=course value=welcome to JAVA EE: Java Server Pages. Now let me find out how many characters are present within the course variable. To verify, let me type in paragraph Total Characters $ curly braces length of course. Now I would like to verify if in the above course do we have Java within it. So let me type in if test=$ curly braces contains of course, Java paragraph is related to Java path. But since I have provided the value in lowercase, the condition will fail. So when we have a requirement to ignore the case while verifying the condition, then we can perform one of the following, either we can convert both the values into lower or uppercase or simply we can ignore the case while comparing with the support of test containsIgnoreCase of course, java.

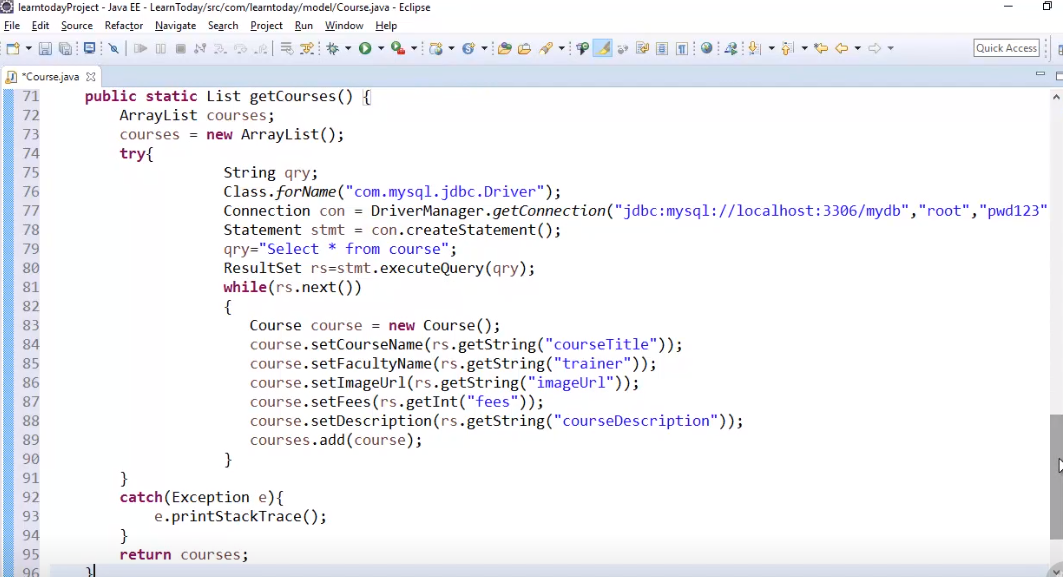
=>slides: Pg. 27

Now when we execute the above page we can observe the total characters present within the string. And also we can observe the message course is related to Java Path. I hope you have got a fair idea on using the JSTL tags within the JSP. Now before you continue to the next section, take time to write the code for displaying all the courses in the library for our virtual training company web application in the following style with the support of JSTL and Expression Language, else you can also follow me while developing the page in the next section and complete the task.

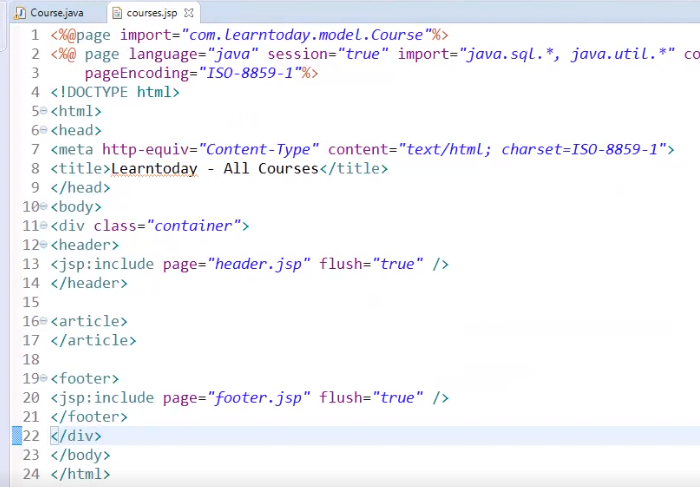
# Demo: Extending Virtual Training Company Application Using JSTL

=>slides: Pg. 28

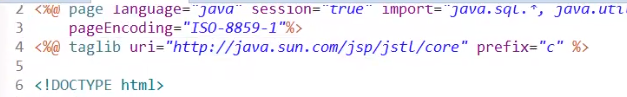
In the last module we have updated the virtual training company application to display the search results using Expression Language. Now let us extend our application to display all the courses with the support of JSTL and Expression Language. To retrieve the course details based on the course name, in the last I have returned the code directly within the JSP page and also I have informed that it is advisable not to write the code in that way. Now in order to display all the codes present in that library, let us follow the best practice. To start, let me open the course Java Bean and let me add the code to retrieve all the courses present within our library.



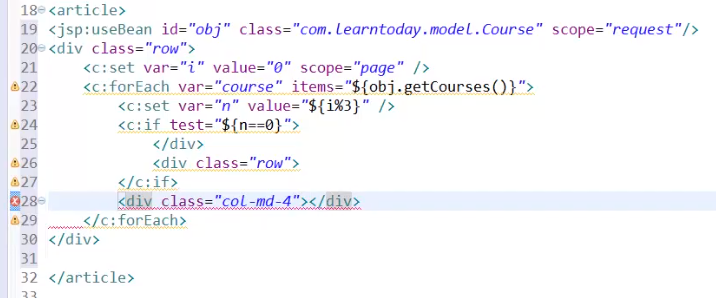
Let us have a quick view what is written within this method. I have defined an ArrayList object of courses and we have some statements which establish the database connection and a select statement to retrieve all the courses, and the result we have collected within the ResultSet rs. I have used a while loop to iterate all the courses and each course object I have added to the courses collection, which I have written. Once we have prepared our Java Bean,



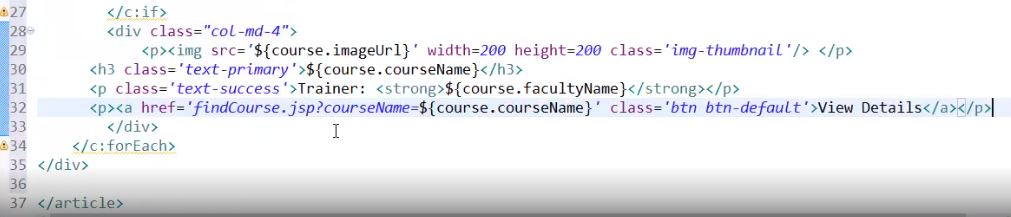
now let me add the courses JSP page and also update the header and footer sections. Now let me first add the taglib directive to support the JSTL core functionalities.



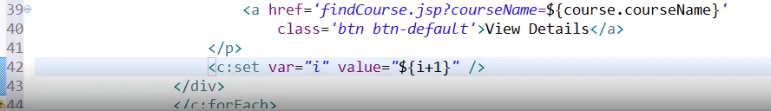
So let me type in directive taglib uri=http://java. sun. com/jsp/jstl/core prefix=c.



Now within the article section, let me add the useBean action tag to refer the course Java Bean. Now let me add a div row, so let me type in div class=row. I would like to define a variable i with the value 0 using the pages scope, so let me type in c:set var=i value=0 scope=page. I require a loop to iterate within the courses present within the library. So let me type in c:forEach var=course items=$ curly braces object. getCourses. I wanted to divide each row into three equal columns such that the details of three courses will be displayed per row. To do let me type in c:set var=n value=$ curly braces i modulus 3. If test=$ curly braces n==0 let me close the div. And now let me open a new row div class=row. In order to divide each grid row into three columns, we require a div tag with the class called md-4. So let me type in div class=col-md-4.



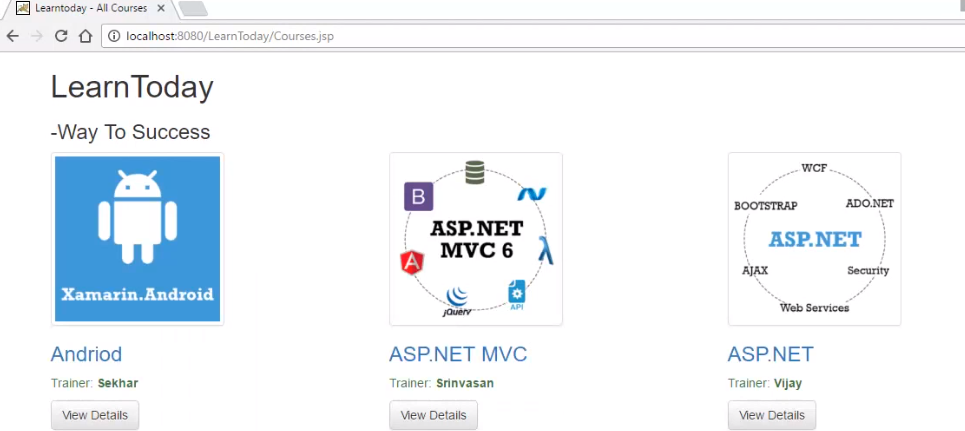
Now within the div tag let me add the code to display course imageUrl, courseName, facultyName and a link to view the course details. I have applied some bootstrap classes to beautify the look and feel.



Let me increment the value for the variable i after the course is displayed. So let me type c:set var=i value=$ curly braces i+1. We can observe there is no Java code present within this page.



Now let me execute the application and let me click on View All Courses.



We can observe the list of all the courses present in the library is formatted properly and displayed very efficiently with the support of JSTL and Expression Language. Now for the people who likes to accept the challenge, in the sixth module within the findCourse. jsp page I have returned the Java code directly, try to update that page with the support of Java Bean, JSTL, and Expression Language, and avoid the Java code within it.

# Summary

=>slides: Pg. 29

In this module we have understood how to work with JSTL and also observe the code JSTL tags, formatting tags, SQL tags, and JSTL functions. In the next module we shall understand how to implement Session Handling with Cookies.

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