7. Simplifying JSP Development with Expression Language

Contents

[1. Introduction 1](#_Toc1682)

[2. EL Fundamentals 1](#_Toc19388)

[3. Basic Operators 2](#_Toc7151)

[4. Implicit Objects 3](#_Toc31922)

[5. Demo: Simplifying JSP Development with Expression Language 4](#_Toc5381)

[6. Summary 4](#_Toc20520)

# 1. Introduction

=>slides: Pg. 1

In this module we shall understand one of the important and powerful features of JSP for simplifying the JSP development with the support of Expression Language.

=>slides: Pg. 2

We shall first understand what is Expression language and its advantages. Then we shall understand the basic operators, functions, and the implicit objects supported by the Expression Language. Once we get a fair idea on Expression Language, we shall use this feature to update our virtual training company web application and understand how the Expression Language simplifies the JSP development. Now let us get started.

# EL Fundamentals

=>slides: Pg. 3

The first question we get in our minds is when we have scriptlets, then why do we need Expression Language? In the initial days of JSP, many developers used to write lots of code within the scriptlets. =>slides: Pg. 4

We know that we can write any valid Java code within the scriptlet, which means that we can also write complete database code inside the scriptlet to perform the various CRUD operations.

=>slides: Pg. 5

That is exactly what we don't want to do. We want our JSP code to be simple and maintainable, and should be used for providing the user interface and Expression Language helps in achieving in this. =>slides: Pg. 6

Expression Language is introduced from JSP 2. 0 and helps in accessing the data from JavaBean, Request, Response, Session, Application objects easily. Expression Language allows us to create both arithmetic and logical expression. Expression Language is also a null friendly language. That is, if a given attribute is not available then the Expression Language will not throw an exception, instead it returns a null value.

=>slides: Pg. 7

The basic Expression Language syntax will be a $ followed by an opening curly brace and a closing curly brace and the JSP engine will understand that anything present within the curly braces are considered as an Expression Language code.

=>slides: Pg. 8

Now let us understand the advantages of JSP Expression Language. It can provide more information and can access the stored object easily by using a simple expression shortened notation for Bean properties accessing and modifying Bean properties will be easy using the Expression Language tags, and also using the dot notation. If you can recollect in the previous module, while explaining the get property action, I mentioned that we can't access the nested Bean object property details, but using Expression Language with the support of dot notation, we can access the nested property easily. It provides an easy access to collection elements such as arrays, collections and maps, et cetera. Provides useful set of simple math and conditional operators to manipulate and test the objects within the Expression Language expressions, supports automatic type conversion. EL removes the need of type conversions. In most of the cases, EL automatically performs type conversions.

=>slides: Pg. 9

One important point we need to remember, that is if we happen to have a template takes plain old HTML or text in our JSP that included something that looked like EL expression then we are in big trouble. If we couldn't tell the JSP container to just ignore anything that appears to be EL and instead treat it like any other unprocessed text, EL is enabled by default. So if you want EL looking things and our JSP has to be ignored, then we need to set explicitly either through our deployment descriptor element or within the page directive.

=>slides: Pg. 10

In order to enable or disable EL in the deployment descriptor within the jsp:config tag body, we need to add a tag el-ignored. The value for that should be either true or false

=>slides: Pg. 11

and in order to enable or disable the expression language at the page directive, then within the page directive we need to set isELIgnored= true or false. If there is a conflict between the ELIgnored setting in the deployment descriptor and the isELIgnored page directive attribute, then the page directive always wins. In the next clip we shall understand how to use basic operators in Expression Language.

# Basic Operators

=>slides: Pg. 12

In the previous modules I have informed that we should not do calculations and write logics within the JSP page. As we know that a JSP is the view and the view's job is to render the response, and it is not the responsibility of the view to make any big and important decisions or to perform big processing, and we know that if we require any real functionality, then it should be the job of the controller and model, and for lesser functionality we can use JSTL and custom tags, which we shall understand in the next two modules. But sometimes if we require to perform a small arithmetic operation or a simple Boolean test, then using the Expression Language will be very handy.

=>slides: Pg. 13

So with that point of view, we shall understand the most useful Expression Language arithmetic, logical, and relational operators. EL supports the following arithmetic operators, plus for addition, minus for subtraction, asterisk for multiplication, slash and also we can use div for division. And one important point we need to remember in Expression Language is we can divide a number by 0 in Expression Language. We get the result as infinity, but not an error. Percent or mod is used to return the remainder. EL supports the following logical operators, double ampersand or we can use and to represent the logical and double pipe or we can use or to represent the logical or exclamation or we can use not to represent the logical not operator. EL also supports the relational operators, double equal to and eq can be used to represent equals, not equal to and any can be used to represent not equals, less than and lt can be used to represent less than, greater than and gt can be used to represent greater than, less than equal to and le can be used to represent less than or equal to, greater than equal to and ge can be used to represent greater than or equal to. Along with the operators, EL also supports some reserved words such as true, false, null, empty, and there are some reserved words which have been reserved for the future such as instance of null. Let us understand how the EL operators can be used within the JSP page.

=>slides: Pg. 14

For example, within the JSP page, now let me perform a small calculation paragraph, what is 2 + 3 : $ curly braces expression as 2+3. Now I wanted to perform a small comparison. So let me type in paragraph Is 2 greater than 3 $ curly brace expression as 2 greater than 3. Now I wanted to divide a number by 0. So let me type in paragraph What is 10 / 0 : $ curly brace, and let me type in the expression as 10 div 0. Now let us understand a small calculation to find out the temperature in Fahrenheit. So I can just define a paragraph 2C = $ curly braces, in order to convert the Celsius to Fahrenheit, we need to perform a simple calculation within the braces, 9. 0 divided by 5. 0 multiplies, we wanted to convert to Celsius, so 2. 0 + 32. 0 degrees F. Now let us understand slightly more complex example to represent an if condition using Expression Language. Let us verify if 6 is an even number or odd. To do, let me type in a paragraph 6 is an $ curly braces, we know that if any number is divided by 2 and if the result equals to 0, then it is considered as an even number, else it is treated as an odd number. So to verify, we can type in 6 mod 2 == 0, that is our condition,?. If the condition is true, then it is even. So let me type in within double quotes even :, if the condition is false, then it should be an odd number. So let me type in double quotes odd number.

=>slides: Pg. 15

If we execute the above code, the result will be what is 2+3 : 5. Similarly, 2 > 3 : false, and we get 10 divided by 0 is infinity and the temperature 2C in Fahrenheit, and 6 is an even number. In the next clip we shall understand implicit objects provided by Expression Language to write the code efficiently.

# Implicit Objects

=>slides: Pg. 16

Expression Language supports some important implicit objects which are different from the JSP implicit objects. These objects can be used in any expression as if they are variables.

=>slides: Pg. 17

Now let us understand the implicit objects provided by the Expression Language. PageContext object, this object gives us the access to the pageContext JSP object which is used to read the data from the request session config and application. For example, $ curly braces pageContext. request. queryString. This expression will return the incoming query string of a request. Header object is used to map the request header information. For example, $ curly braces header of user-agent will return the browser's user-agent detail. HeaderValue, this object will map the request header names with the array of values. We have already used JSP built-in objects such as request session and application objects in the earlier modules. Expression Language will provide these objects as scope objects. There are four possibilities for the scope, these are all accessed by accessing some of the implicit objects provided by the Expression Language. PageScope object maps to the attributes that are only available to JSP page. RequestScope object maps to the attributes that are available to request context. SessionScope object maps to the attributes that are available to session context, and applicationScope object maps to the attributes that are available to application context. Param object gives us the access to the parameter values normal available through the requestor. get parameter. ParamValues object gives us access to the parameter values normally available through the requestor. get parameter values. InitParam object is used to get the context initParam's values. We can't use this object for accessing the servlet initParameter values. Cookie object is used to get the cookie value within the JSP page.

=>slides: Pg. 18

For example, to access the Java Bean property, we can use $ curly braces course. courseName. In this case we haven't provided the scope, but if you wanted to explicitly provide the scope, then we need to write the expression as $ curly braces scope. course. courseName. For example, requestScope. course. courseName. before we continue further, we need to understand some basic rules. Scope can be implicit or explicit. That is, if the scope is not specified within the Expression Language expression, then the JSP engine will search for all the scopes from the most local page to the most general application. Dot notation is used to separate scope if specified object and property. When dot notation is used, then final element must specify a Java Bean property or the map key.

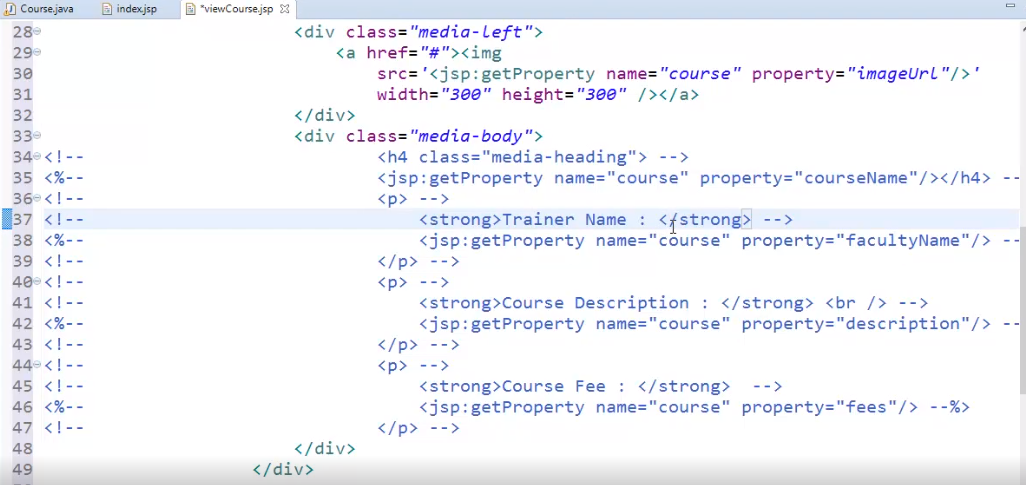
=>slides: Pg. 19

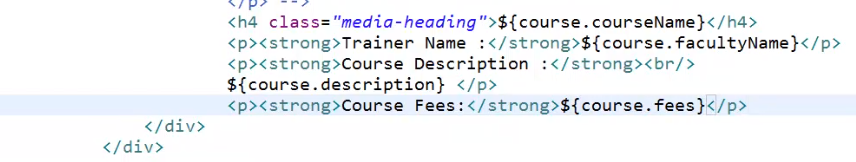
Expression Language also supports the nested properties. In the previous module while discussing about the get property action, I was discussing about the person Bean and the Car property. So if we wanted to access the car name based on that scenario, we can use $ curly braces requestScope. person. car. carName. We can also use Expression Language to read the values from the collections where a collection is an object that groups multiple elements into a single unit. In order to read the values from the collection, we can use $ curly braces collection name of index or expression. In the next clip we shall understand how to simplify the JSP development with the support of Expression Language.

# Demo: Simplifying JSP Development with Expression Language

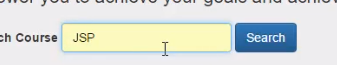
=>slides: Pg. 20

In the last module of our application we have created find course and view course JSP pages where view course page is used to display the course details based on the user input. Now let us update the viewCourse page using the Expression Language.



I have already opened the viewCourse page and also commented the lines used for displaying the data. 

Within the div class media-body, let me type in h4 class = media-heading $ curly braces of course. courseName. Now let me display the trainer name, so let me type in paragraph strong Trainer Name $ curly braces of course. facultyName. Then I wanted to display the course description. So let me type in paragraph strong Course Description. Let me add a break rule to display the description in next line. So let me type in br, then I want to display the value of Course Description, so let me type in $ curly braces of course. description. And finally, I wanted to display the fees. So let me type in paragraph strong Course Fees $ curly braces of course. fees.



Let me save the file and execute the application. Within the find course, let me type in JSP and click on search button.



We can observe the same result, but the coding is quite simple and easy.

# Summary

=>slides: Pg. 21

In this module we have understood the importance of Expression Language and how it can simplify the JSP development, but the effective utilization of Expression Language can be achieved with the support of JSP standard tag libraries, so in the next module we shall understand what is JSP standard tag library and how we can use the JSTL with the support of Expression Language and then we shall update our virtual training company web application by encapsulating the code functionality using JSP standard tag library to display all the courses in our library. Very much excited to move on yet to an interesting and a powerful concept of JSP. So see you in the next module.

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