3. Working with Directives

Contents

[1.Introduction 1](#_Toc10768)

[2. @page Directive 1](#_Toc16307)

[3. Demo: @page Directive 1](#_Toc22470)

[4. @include Directive 1](#_Toc10442)

[5. @taglib Directive 2](#_Toc15734)

[6. Summary 2](#_Toc11777)

# 1.Introduction

=>slides: Pg. 1

In this module we shall understand how to work with JSP directives. To start with, let us understand what is JSP directive?

=>slides: Pg. 2

The directives of JSP are used to provide some meaningful information about the current JSP page to the JSP container which will be required during the translation phase. The directives will not contain any code that will be a part of the execution. But a JSP directive affects the overall structure of the servlet that results from the JSP page. In order to write a JSP directive, we need to use an angular braces, a percentile, and an at symbol.

=>slides: Pg. 3

There are three types of directives supported by JSP, page directive, include directive, and taglib directive. In this module, I will be explaining in detail about the page directive and include directive and I will provide a basic overview on taglib directive, and I will explain taglib directive in detail at the later part of the course while discussing JSP tag library and custom tags in our course.

=>slides: Pg. 4

One exciting news is that from this module we shall start developing a mini application for a virtual training company called as Learn Today, and we shall enhance the application in every module, and if you follow the instructions and if you start developing the application along with me, by the end of this course you will have developed a web application using JSP by exploring the concepts of JSP in detail. Now let us start with understanding page directive.

# @page Directive

=>slides: Pg. 5

The first directive to understand in JSP is the page directive. If we recollect, in every JSP page we can observe with directive. Now let us understand the page directive in detail.

=>slides: Pg. 6

Page directive provides a means for setting attributes that affect how the page is interpreted an executed. The attributes of the page directive define page specific properties such as character encoding, the content type of the page, response and whether this page should have the implicit session object, exception handling details, and language to be used, et cetera.

=>slides: Pg. 7

The syntax for defining a page directive will be less than, percentile, @ page attribute equal to value, attribute equal to value.

=>slides: Pg. 8

Page directive supports the following attributes.

=>slides: Pg. 9

The attributes can be specified in any order and more than one page directives can be specified in a compilation unit. If multiple page directives are used, then they cannot specify the same attribute more than once. But one attribute is an exception and it can be used any number of times and that attribute does import.

=>slides: Pg. 10

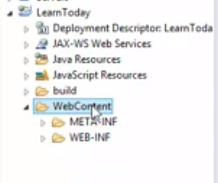
Now let us understand the usage of the page directive attributes. Language, this attribute will specify the language to be used for the scripting of JSP page. The default value for this attribute will be Java. For example, page directive language equal to Java. Import, this attribute will specify the classes and the interfaces of the packages that has to be specified in the generated servlet. This is the only attribute which can be specified multiple times and this is the only attribute which can contain multiple values separated by comma. For example, page directive import equal to Java. sql. \*, import equal to Java. util. \* or we can also define the above statement by writing as page directive import equal to Java. sql. \*, Java. util. \*. Info, this attribute will specify the information or the description about the JSP page. The information specified in the info attribute can be accessed with the help of get servlet info method, for example, page directive info equal to This is the home page for the web site. Extends, this attribute will specify the name of the class which has to be inherited by the generated servlet. Syntax will be page directive extends equal to packageName. className. For example, page directive extends equal to com. Pluralsight. demos. ContentType, this attribute is used to specify the main type that has to be used by the JSP page. The main type will provide the information of the kind of data that will be sent to the client as a part of the response. For example, page directive contentType equal to text/HTML. IsELIgnored, this attribute is used to specify whether the current JSP supports the expression language or not. The default value of the isELIgnored will be false, that is, by default the current JSP will support the expression language and if we want the current JSP to ignore the expression language, then we need to explicitly specify the value as true. For example, page directive isELIgnored equal to true. =>slides: Pg. 11

IsThreadSafe, by default servlet engines load a single instance of a servlet and use a pool of threads to service individual requests. This means two or more threads can be executing the same servlet method simultaneously. If the servlet has instance variables and if no provision is made to synchronize access, the threads can collide and interfere with each other's access to the variables. The servlet API provides a way around this using the single thread model interface. This interface has no methods. It simply marks a servlet as requiring a dedicated thread for each instance of the servlet. IsThreadSafe attribute of the page directive provides a means for causing single thread model to be associated with a JSP page. If you specify isThreadSafe equal to true, you are asserting that you will take care of any possible thread conflicts so the JSP container can safely dispatch multiple requests to the servlet simultaneously. For example, page directive isThreadSafe equal to false, if the value is false, then the JSP container generates a servlet that implements single thread model. The default value for the isThreadSafe attribute is true. PageEncoding, this attribute will specify the JSP container, how the JSP has to be translated, and by using which character set the translation must be done. For example, page directive pageEncoding equal to UTF-8. Session, this attribute will provide the information to the JSP container that the current JSP will use the session object. The default value of this attribute is true. If the current page doesn't require the session object, then we need to explicitly set the value as false. For example, page directive session equal to false. Buffer, if this attribute is set, then the data will be stored within the buffer instead of sending the data directly to the client. Once the buffer is filled, then the data will be sent to the client, the default size of the buffer is 8kb. For example, page directive buffer equal to 10kb. AutoFlush, this attribute can be used for clearing the buffer automatically. The default value of this attribute is true, which implies that when the buffer is filled, it will be cleared automatically. If you don't want the buffer to be flushed or automatically, then we need to specify the value as false. If we want to flush or clear the buffer manually, then we should use our. flush method. There are two more attributes available for the page directive, errorPage and isErrorPage, which I will be explaining in the later modules of this course while discussing exception handling in JSP.

# Demo: @page Directive

=>slides: Pg. 12

I already informed that we will start with web application development from this module and slowly we will extend the application by adding new features by exploring the concepts of JSP, such that by the end of this course we shall complete the application by understanding the usage of JSP concepts in a proper way.

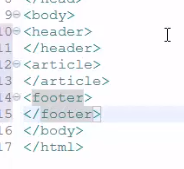


To start, let us create the home page for our application. I have taken the liberty to create a new dynamic web project with the name Learn Today. Now let me right click on the WebContent folder and click on New and select JSP File. Let us provide a meaningful name for our home page.



So let me type in index. jsp and then click on Finish button. We can observe a JSP file based on predefined template. If you observe the first line, we can notice the page directive tag, which provides us the information about the page such as this page is created using Java language and the content type of this page is an HTML content, and also it is providing the information about the character set as well as page encoding standards used. We should add additional attributes to the page directive in the later modules. I would like to use the HTML5 for our application development.



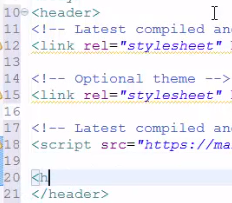
Let me update the title as LearnToday- Way to Success. Now let us start the design for our home page. 

Within the body, let me type in the header where we shall place the header to be displayed in this page and then let me type in the article tag where the main content of the page we shall add, and finally let me type in the footer tag where we shall display the copyright details. For our application, instead of creating a custom stylesheet, we shall use the bootstrap stylesheets. Just to inform for the people who are not aware of what is bootstrap, boot, remember that, bootstrap is the most popular HTML, CSS and JavaScript framework for developing responsive mobile first projects on the web. For our application development, knowledge of bootstrap is not mandatory. Whenever I use a bootstrap class in our design, I will explain what is the usage of that class. I would like to use the bootstrap CDN for our application. CDN stands for content delivery network, which is used to deliver the web content to a user based on the geographic locations of the user, the origin of the web page and the content delivery server. Using CDN, performance of the web application increases. Let me flip to the browser.

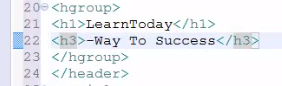


I have already typed in the URL of the bootstrap CDN, that is http://getbootstrap. com/getting-started. 

Click on Copy at Bootstrap CDN block to copy the bootstrap stylesheets and JavaScript links.



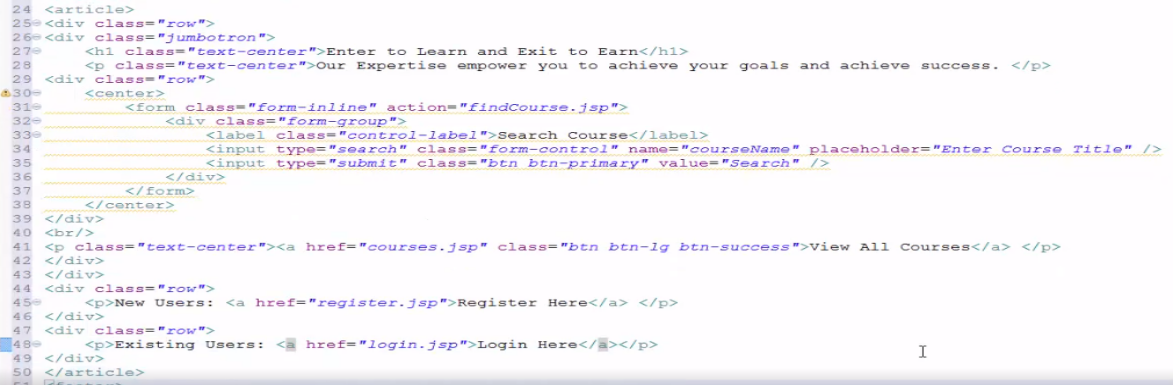
Let me flip to Eclipse and paste the links at the header. Now let us add the header details.



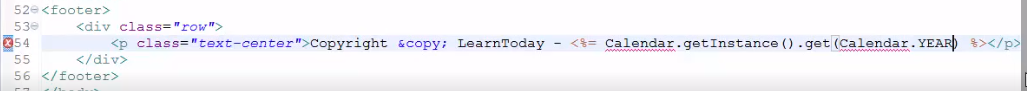
So let me type in hgroup, then let me type in the header h1 LearnToday, and then let me add a tag line h3- Way To Success.

=>slides: Pg. 13

Let us once again observe the home page of our finished application so that we can add the main content to be displayed on the home page. We can notice a block. Within that we can observe some codes along with the search option to find a course and a provision to view all courses. Then we can also observe some links to register for new users and a login link to sign in for existing users.



So let me type in div class=row, then to provide the block let me add another div with the class jumbotron where jumbotron is a class which is lightweight, flexible, component that can be used to showcase key content of the page. Then let me add a header tag, h1 class=text-center, so that the text will be displayed center, and let me type in Enter to Learn and Exit to Earn, and then a paragraph which I wanted to display again in center. So let me type in class=text-center and let me type in some message to encourage the trainees to get registered with that site. So let me type in Our Expertise empower you to achieve your goals and achieve success. Let me add a new row, div class=row, and then I want the form to be displayed in center, so let me type in center. If you're aware of bootstrap grid system, feel free to use the classes to provide the content in center. Now let me add the form tag to display the form elements in length to find the course. So let me type in form class=form-inline, action=findCourse. jsp. Then I want all the form elements to be grouped together, so let me type in div class=form-group, then let me add a label class=control-label Search Course, and then I need an input element to accept the course name. So let me type in input type=search, class=form-control, name=courseName, placeholder=Enter Course Title, and then I wanted to have a button to submit the data. So let me type in input type=submit, class=btn btn-primary, value=Search. Now let me add a break tag so that I can provide the link to view all courses. So let me type in break, then a paragraph class equal to text-center, href=courses. jsp, class=btn btn-lg btn-success, View All Courses where btn is used to represent the button, btn-lg represents a large button, btn-primary is used to display the button in blue color and the btn-success is used to display the button in green. Along with the block content we require some links to register a new user and to provide the login option for existing users. So let me type in div class=row, then a paragraph, New Users: a href=register. jsp Register Here. And then once again, let me add another div tag, class=row, a paragraph, Existing Users, a href=login. jsp Login Here. We will be creating the register. jsp and login. jsp pages in the next module.



Finally, within the folder tag, let me add the copyright details, so let me type in div class=row and then a paragraph, class=text-center Copyright &copy; LearnToday, and let me use an expression Calendar. getInstance. get of Calendar. YEAR. We can observe there is an error stating Calendar cannot be result. The reason for that error is the JSP page is not able to recognize the calendar. So we need to provide the information to the JSP page with the support of the page directive. So let me update the page directive using the import statement.



So within the page directive let me add import=Java. util. \*. Now we can observe the header has been result. Let me save the file and let me execute the page to view our application home page. Great. We have an awesome home page, but if we observe the page we might require the same header and footer content in all the pages. If you follow the previous page, then we need to perform lots of duplication in every page. So in the next clip, let us understand how to include the contents of one JSP page in another JSP page.

# @include Directive

=>slides: Pg. 14

As we have got a fair idea on the page directive, now let us understand the next important directive that we will be using frequently in any JSP development that is include directive.

=>slides: Pg. 15

Include directive will specify the JSP container to read the specify file and to merge its contents into the JSP source code currently being parsed at the translation phase.

=>slides: Pg. 16

In order to use the include directive, we need to use <%@ include file=filename. For example, <%@ include file=header. jsp.

=>slides: Pg. 17

Not only the include directive, not just the contents, its main advantage is that it is powerful. The included code can contain JSP constructs such as the field definitions and the content type settings that effect the main page as a whole.

=>slides: Pg. 18

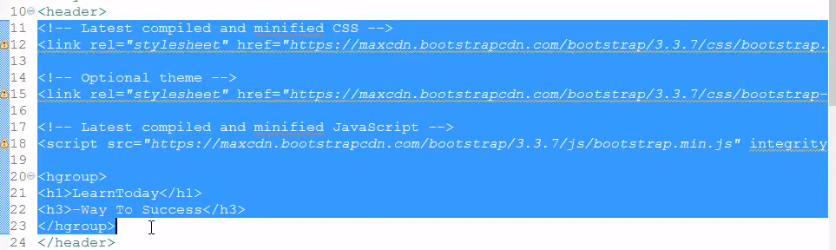
For example, suppose if we have a file codesnippet. jsp, which contains the following line of code, declaration int accessCount=0. Now within the main page we can include the code snippet. jsp using include directive file=codesnippet. jsp. So this line will define the accessCount variable and within the main page we can access the accessCount using expression accessCount++, and when we execute the page we get the result as 1.

=>slides: Pg. 19

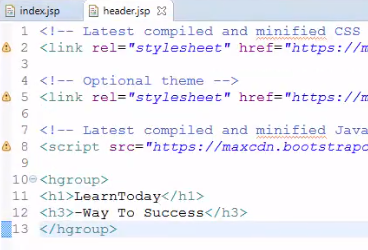
Now let us understand practically how to use the include directive in our application. If you recollect, I informed that the header and the footer remains the same for all the pages what we develop in our application. So instead of we duplicate the content, let us create the header and the footer JSP pages and let us include those pages in other JSP pages what we develop.



So let me right click on the WebContent folder and let me click on New and select JSP File, and let me provide a meaningful name, for example, header. jsp, and let me click on Finish.



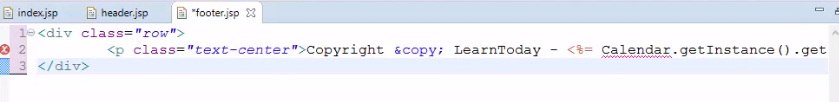
Now let me copy the content present within the header section of index. jsp



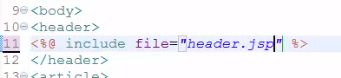
and let me paste that content and let us save that file, and once again, let me right click on the WebContent folder and click on New, JSP File



and let me provide the file name as footer. jsp, and let me click on Finish button.



Now let us copy the content present within the footer section of index. jsp and let me paste within the footer. jsp. Now as the content which is required for the header and the footer is available in other JSP pages, we can remove that content from the index. jsp page and we can include those JSP pages.



So let me first remove the content from the header section and let me type in include directive file=header. jsp, and then let me remove the content from the footer tag and



let me type in include directive file=footer. jsp. Now to wrap the contents within a containers so that we can have a proper padding across the site,



let me place the entire body within a div tag



and let me set the class as container. Now let me save the file and let us execute the page. We can observe the same result, but the content of the header and the footer is coming from different pages.

# @taglib Directive

=>slides: Pg. 20

JSP API allows us to define custom JSP pack library that looks like HTML or XML tags where a tag library is a set of user defined tags that implement custom behavior.

=>slides: Pg. 21

The tag library directive declares that our JSP page uses a set of custom tags, identifies the location of the library, and provides a means for identifying the custom tags in our JSP page.

=>slides: Pg. 22

The syntax for defining the tag library directive will be <%@taglib uri=uriofTagLibrary prefix=prefixofTag where the URI is a unique identifier in the tag library descriptor that is TLD. It is a unique name for the tag library. The TLD describes and the prefix is appended to the custom tag name. Each library used in a page needs its own tag library directive with a unique prefix. We can write XML equivalent of the above syntax as follows: jsp:directive. taglib uri=uri prefix=prefixofTag. We shall understand the usage of tag library directive in detail at the later modules of our course.

# Summary

=>slides: Pg. 23

In this module we have understood one of the important features of JSP, that is working with the JSP directives. We had observed how to use page directive and also include directive. The taglib directive I said we will understand its usage in the later modules of this course, but remember taglib directive is also very powerful directive, but this is too early for us to understand taglib directive. Since without knowing how to work with tag library, it will be difficult for us to understand the usage of taglib directory. Hope you got a fair idea on the usage of JSP directives.

=>slides: Pg. 24

Now in the next module we shall enhance our virtual training company, Learn Today application further with login and registration features using the request and server content through built-in objects. See you in the next module.

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