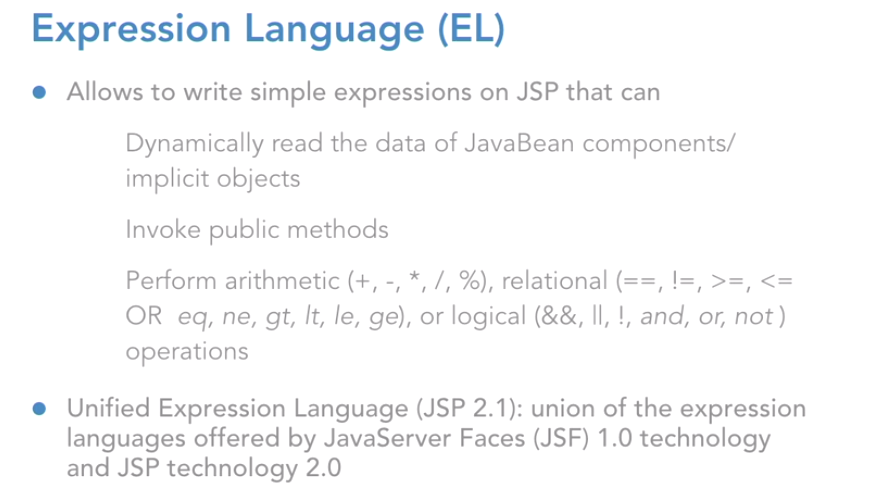


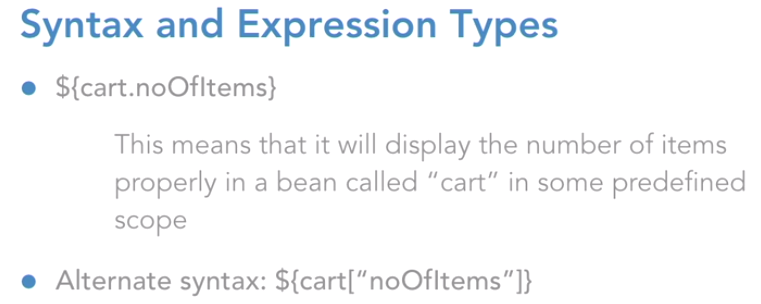
Had you not used JSP standard actions and had you stuck to expressions and scriptlets then, you would have to write down the code of request dot get attribute in a scriptlet to replace this line. But, now do you see that we don't have to write code at all and we can stick to a simple tag which will do the entire job same as what was being done by the scriptlet. So, now that we are ready with the JSP use bean standard action tag, let's go ahead and start displaying the data out of this.

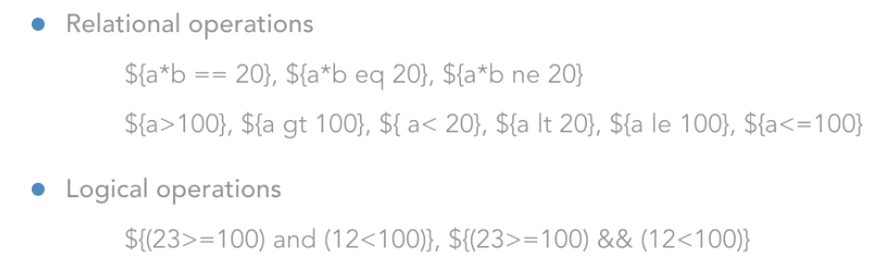
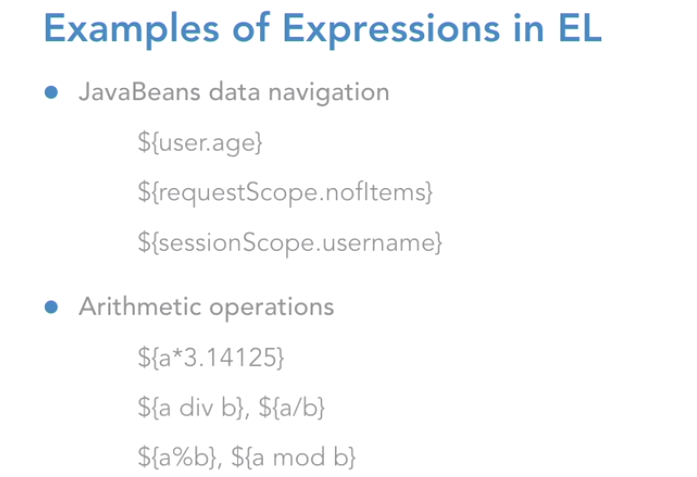


What's the difference between the include directive and the JSP include standard action. So, the difference essentially is that when you use the include directive then the included resource and the current resource, they're both going to be combined and then, it is going to be translated into a servlet file.

However, when you use a JSP include standard action then, the current JSP will be translated separately and this one, header JSP will be translated again separately. And then, at run time, those responses will be combined, and they'll be rendered to the user. But, for the include directive, the translation itself involves the combination and then entirely that thing is going to be translated to the servlet file. So, you are going to use JSP include in your projects. If you think that you want to execute some code in the included resource. Dynamic execution if it's required in your included resource. Stick to JSP include standard action. However, if you're just trying to include static resources, like for example header dot JSP in this case is static.







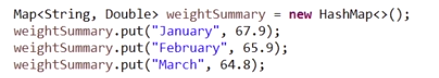
Now you must be wondering how it understood that, it must pick up this user object from the request scope.

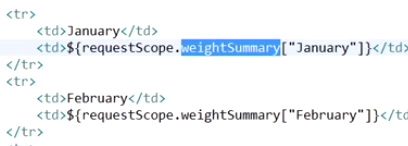
Because when we specified the JSP standard actions mechanism we are explicitly mention that you're supposed to pick up this object from the request scope. However, here, in expression language, you don't have to do that. Because expression language is smart enough. What it does, is, if you have not mentioned any scope here in your expression itself, then it is going to search this object by the key user, by the name user, in those four scopes. The page scope, the request, the session, and then the application, in that order.

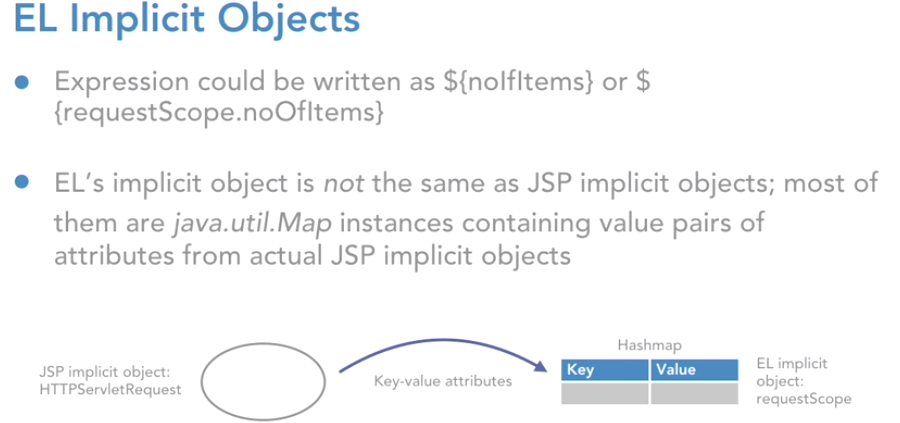


You must also be wondering as to where is this request scope name coming from? How did I write request scope? That is because, if you look at a JSP implicit objects, we have request as the implicit object for the request. Right? But in expression language, the implicit object, which can help you access the data of the request object, the name for it is request scope

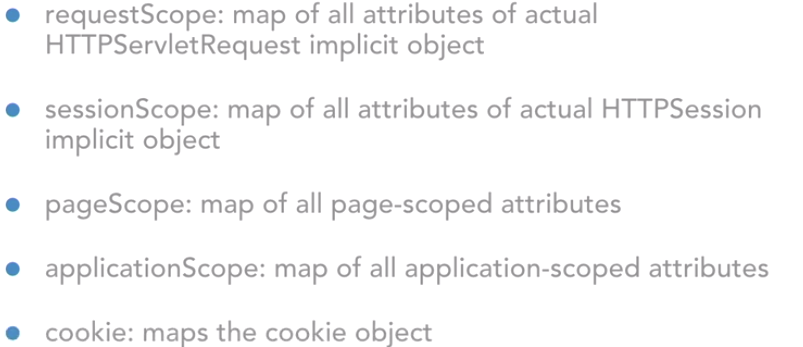
**How to access a Map in Jsp:**

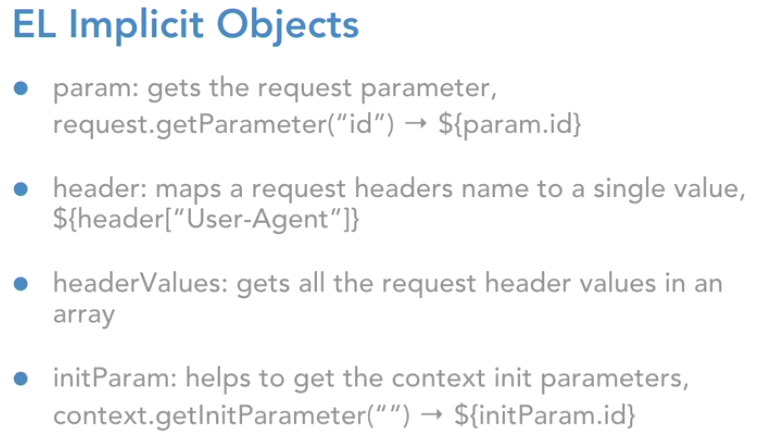


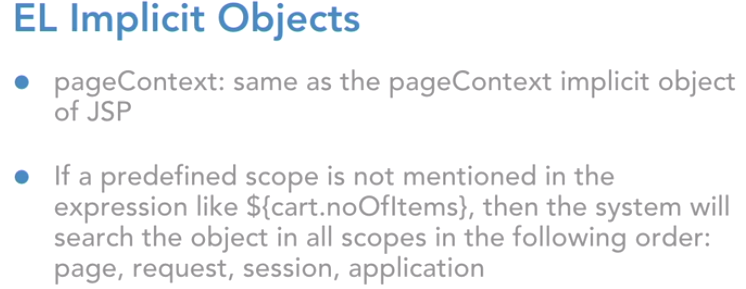


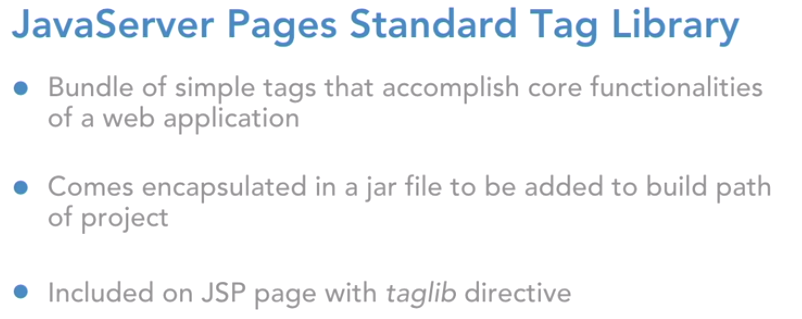


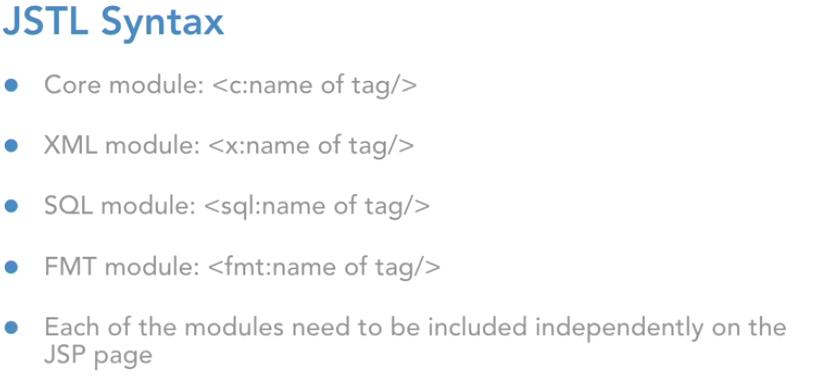
Let's understand that. When you talk about EL's implicit objects, please understand that they are not the same as our jsp implicit objects. There were nine implicit objects in jsp that we had talked about, and those are the ones that are of specific type. HTTP servlet request, HTTP servlet response, then you had servlet context, servlet config, then you had HTTP session etc, right? But when you talk about expression language, the implicit objects are not the same as you have in jsp.







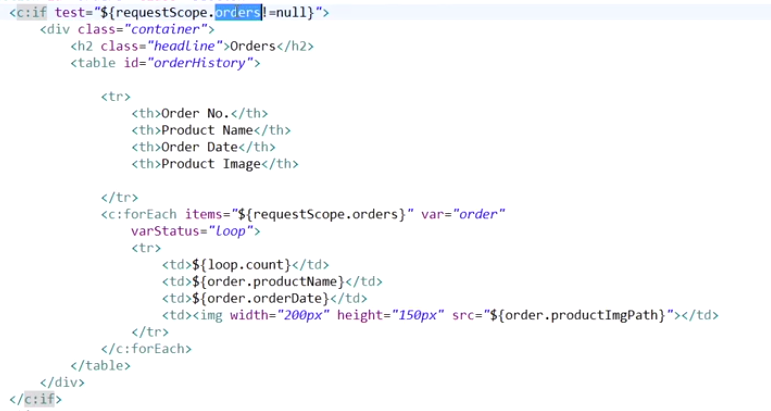


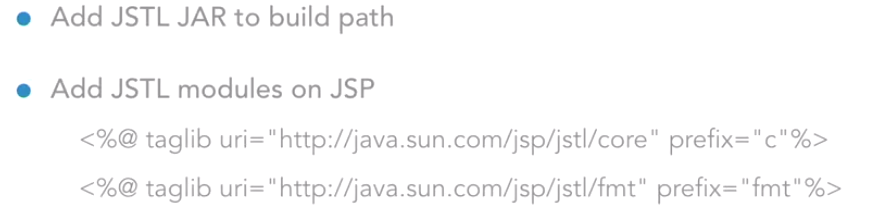


How to use the tag in Jsp:

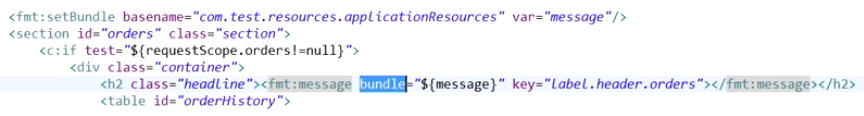


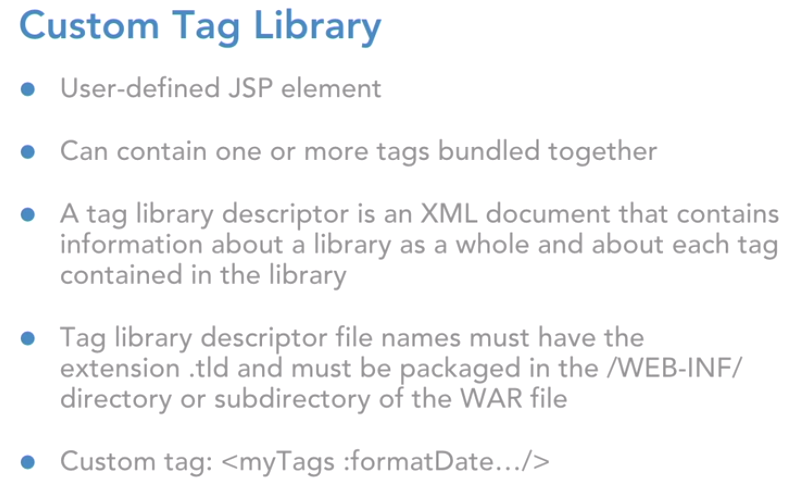
Iterating the list using jstl in Jsp:

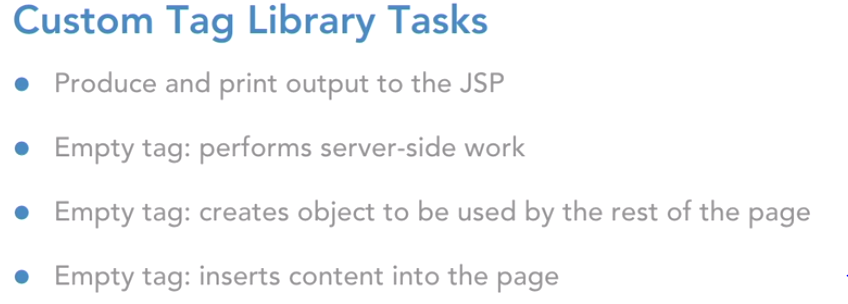


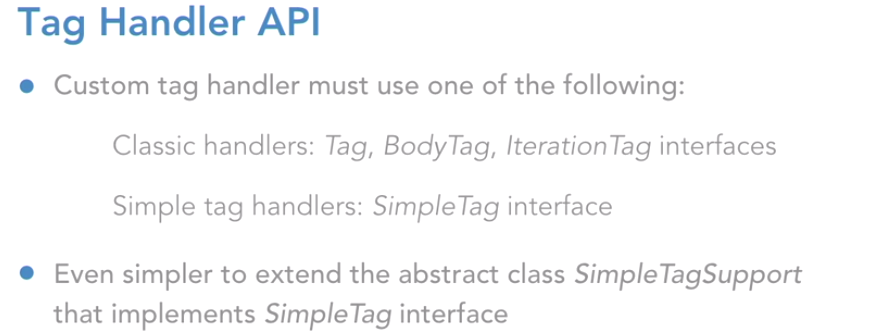


Internalization concept:

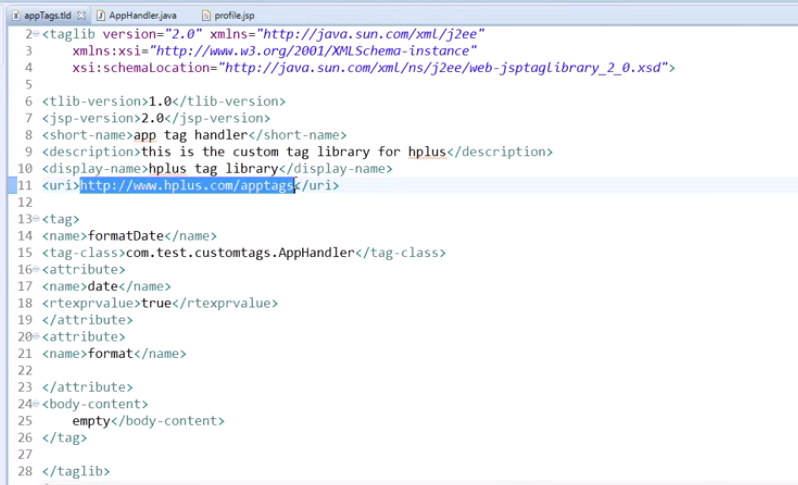




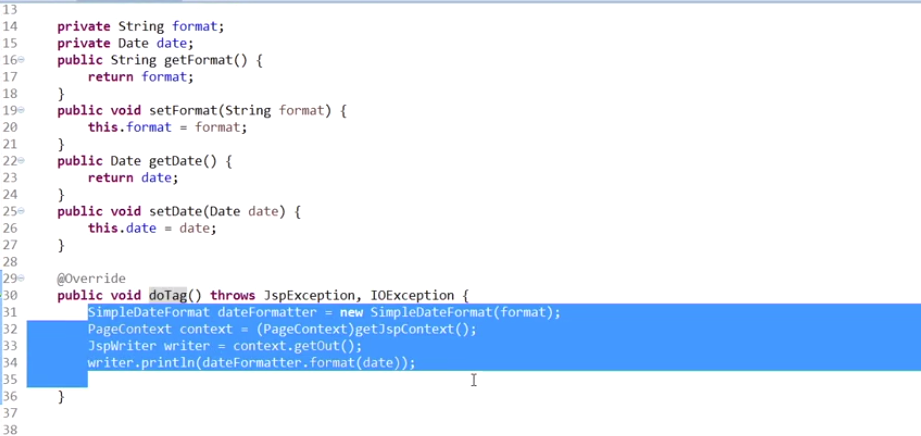




1. **Create a tld file**



1. **Create the Tag handler class**



1. Use the custom tag in the jsp file:



