# UNIT : 9

Reference :-Marty Hall, Larry Brown, "Core Servlets and JavaServer Pages Volume – 2", Pearson Education, 2nd ed.(2004)
Chapter :- 09 : JSP Standard Tag Library (JSTL)

## Installation of JSTL

- Apache Foundation's Jakarta Taglibs project provides a very popular implementation of JSTL

## c:out tag

- The c:out tag is very similar to JSP scripting expressions like <%= ... %> and standard JSP EL use like ${...}, but it has many advantages.
- unlike JSP EL, the c:out tag lets the JSP developer specify a default value to output if the resulting expression evaluates to null.
- It also has an attribute that allows the tag to automatically escape any XML type characters like <, >, &, ", '.

## c:out tag

- cross-site scripting (CSS) attack → if a malicious user inserts some HTML markup into an input field that is output to the screen unfiltered, which is known as a cross-site scripting (CSS) attack.
- The c:out tag avoids all these problems by providing the escapeXml attribute that is set to true by default.
- This feature means that the c:out tag converts those special characters into the HTML equivalent characters.
- For example, the character < would be converted to &lt;.

## c:out tag

- cross-site scripting (CSS) attack → if a malicious user inserts some HTML markup into an input field that is output to the screen unfiltered, which is known as a cross-site scripting (CSS) attack.
- The c:out tag avoids all these problems by providing the escapeXml attribute that is set to true by default.
- This feature means that the c:out tag converts those special characters into the HTML equivalent characters.
- For example, the character < would be converted to &lt;.

## c:forEach and c:forTokens Tags

- The c:forEach tag provides basic iteration, accepting many different collection types.
- It can also function as a counter-based for loop, specifying a begin, end, and a step index.
- It uses the var attribute to allow the JSP developer to specify a key with which to refer to the current element through the iteration.
- The c:forTokens tag functions much like c:forEach except it is designed to iterate over a string of tokens separated by delimiters.
- What is considered a delimiter is customized through its required delims attribute.

## c:forEach and c:forTokens Tags

- The c:forEach tag provides basic iteration, accepting many different collection types.
- It can also function as a counter-based for loop, specifying a begin, end, and a step index.
- It uses the var attribute to allow the JSP developer to specify a key with which to refer to the current element through the iteration.
- The c:forTokens tag functions much like c:forEach except it is designed to iterate over a string of tokens separated by delimiters.
- What is considered a delimiter is customized through its required delims attribute.

## c:if Tag

- The c:if tag is a simple conditional tag that evaluates its body if the supplied condition is true. The condition is evaluated through its required attribute test.

## c:choose Tag

- The c:choose tag is a conditional tag that establishes a context for mutually exclusive conditional operations, marked by the c:when and c:otherwise tags.
- These three tags work much in the same way as the standard Java switch-case-default statements.
- The c:choose tag itself does not have any attributes.
- Its sole purpose is to provide the context to the other two tags (i.e., c:when and c:otherwise).

## c:choose Tag

- The c:when tag functions in exactly the same way as the c:if tag.
- It has a test attribute that allows the JSP developer to specify a condition, which if evaluated to true, would signal to the container to evaluate the body of the c:when tag.
- If a particular c:when tag’s condition evaluates to true, no other c:when tags below it are evaluated and the processing jumps to the line after the closing c:choose tag.
- If the optional c:otherwise tag is specified and no c:when tag evaluates to true, the body of the c:otherwise tag is evaluated.

## c:choose Tag

- The c:when tag functions in exactly the same way as the c:if tag.
- It has a test attribute that allows the JSP developer to specify a condition, which if evaluated to true, would signal to the container to evaluate the body of the c:when tag.
- If a particular c:when tag’s condition evaluates to true, no other c:when tags below it are evaluated and the processing jumps to the line after the closing c:choose tag.
- If the optional c:otherwise tag is specified and no c:when tag evaluates to true, the body of the c:otherwise tag is evaluated.

## c:set and c:remove Tags

- The c:set tag is used either for setting a scoped attribute or updating and creating bean properties and map values. The following are some of the common forms in which the c:set tag is used.

- <c:set var="attributeName" value="someValue" />
- <c:set target="map" property="elementKey" value="elementValue" />
- <c:set target="bean" property="name">John Dow</c:set>

## c:set and c:remove Tags

- The c:set tag is used either for setting a scoped attribute or updating and creating bean properties and map values. The following are some of the common forms in which the c:set tag is used.

- <c:set var="attributeName" value="someValue" />
- <c:set target="map" property="elementKey" value="elementValue" />
- <c:set target="bean" property="name">John Dow</c:set>

## c:set and c:remove Tags

- The c:set tag provides two attributes, var and target, of which at least one is required.

- However, these attributes are not allowed to appear at the same time.

- The var attribute is used to specify the name of an attribute to set. This attribute is set in the scope specified by the optional scope attribute. If the scope attribute is not specified, it defaults to page scope.

## c:set and c:remove Tags

- The target attribute must be an expression that evaluates to some object.

- If the object implements the Map interface, the property attribute of the c:set tag is treated as a key with which to create a new map entry with the value specified.

- If the target evaluates to a JavaBean, the value of the c:set tag is passed to the property of the target JavaBean specified by the property attribute.

## c:set and c:remove Tags

- When the value of the c:set tag evaluates to null, the effect is no different then invoking someScopeReference.setAttribute(null), where some-ScopeReference is one of request, response, session, and so on, which essentially removes the attribute from that scope.

- For convenience, JSTL provides an explicit c:remove tag. The c:remove tag allows the JSP author to specify the name of the attribute to remove using the required var attribute and the scope to remove it from using the optional scope attribute.

- Unlike the c:set tag, if the scope attribute is not specified, the c:remove tag will remove the attribute from all scopes.

## c:import tag

- To include content from the same container.
- Use the static inclusion by employing the include directive, which includes the referenced content at page translation time.

- Use the dynamic inclusion by employing the jsp:include standard action, which includes the referenced content at request time.

- if the content we want to include resides on a different server, The include directive as well as the jsp:include standard action can't help us there, but the c:import JSTL tag can.

## c:import tag

- This tag can include the content, pointed to by the required attribute url, even if the content resides on a different Web server.

- The default behavior of c:import is to output the included content into the including page at the place where c:import appears.

- if the optional var and the likewise optional scope attributes are specified, the included content can be saved as a scoped attribute instead. If the scope attribute is omitted, it defaults to the page scope.

## c:import tag

- The c:import tag can just as easily import content from the same container. In such cases, you can use the c:param tag in the exact same way as you would use the jsp:param with the jsp:include standard action.

## c:redirect tag

- If its required url attribute specifies an absolute URL, the c:redirect tag acts just like the sendRedirectURL method of the HttpServletResponse class.
- This results in the browser making an additional request to the new URL, and in fact is no different than if the user were to type the URL by hand.
- If, however, the URL specified in the url attribute is a relative URL, a dynamic forward occurs, which is equivalent to the forward method of the RequestDispatcher class. In this case, the browser address bar still shows the original URL and not the URL of the forwarded to page.

## c:redirect tag

- If its required url attribute specifies an absolute URL, the c:redirect tag acts just like the sendRedirectURL method of the HttpServletResponse class.
- This results in the browser making an additional request to the new URL, and in fact is no different than if the user were to type the URL by hand.
- If, however, the URL specified in the url attribute is a relative URL, a dynamic forward occurs, which is equivalent to the forward method of the RequestDispatcher class. In this case, the browser address bar still shows the original URL and not the URL of the forwarded to page.

## c:url & c:param tag

- One task that you cannot escape with automatic session tracking alone: links in your pages that point back to other pages in your own Web application.
- Automatic session tracking will work whether the client has cookies enabled or not.
- In case the cookies are disabled the session tracking is done through URL rewriting; that is, the container reads the session ID straight from the URL without using a session cookie.

## c:url & c:param tag

- The container will not automatically append the session ID to the URLs that are sent back to the client inside your JSP pages.
- Every URL within your application needs to be encoded if it is to retain the session ID information. This is usually done with the encodeURL method of the HttpServletResponse class.
- Just like the c:import tag, the c:url tag is able to hold off on outputting the URL string if the optional var attribute is present. In such a case, the resulting URL gets stored in a scoped attribute.

## c:url & c:param tag

- The scope can be specified by the optional scope attribute. If omitted, the scope defaults to page scope, as usual.
- <c:url value="myurl?fullName=${fullNameParam}"/>
- what if the fullNameParam turns out to be John Dow, with the space character in it? Spaces are illegal inside URLs, so we need to encode the parameters as well.
- quite easily done with one or more c:param tags nested inside the c:url tag.

## c:catch tag

- The c:catch tag acts like the try/catch construct in Java, except in the case of c:catch, there is no try.
- If you surround part of the page that may throw an exception with the c:catch tag, and an exception occurs, the page will still render up until the point where the exception occurred and then continue rendering from the c:catch end tag on.
- You can store the thrown exception in a page scoped attribute specified by the optional var attribute.

## c:catch tag

- Exception handling is not the job of the JSP page Exceptions should be handled by the business logic before the control ever gets to the JSP page.
- *Avoid using the c:catch tag for anything other than debugging or experimentation. Exception handling should be part of the business logic code, not the JSP page.*