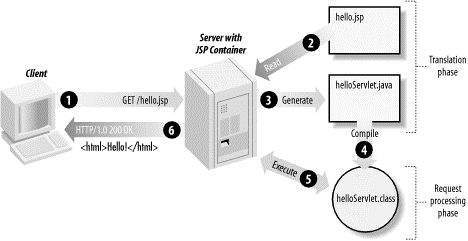
**Basic:** The web server needs a JSP engine that is container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. A JSP container works with the Web server to provide the runtime environment and other services a JSP needs.

**JSP Process:**



* As with a normal page, your browser sends an HTTP request to the web server.
* The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of .html.
* The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.
* The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
* A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response.
* The web server forwards the HTTP response to your browser in terms of static HTML content.
* Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.

**Note:** Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with other scripting languages (such as PHP) and therefore faster.

**JSP Lifecycle:**

The following are the paths followed by a JSP

* Compilation
* Initialization
* Execution
* Cleanup



JSP Compilation:

When a browser asks for a JSP, the JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page.

The compilation process involves three steps:

* Parsing the JSP.
* Turning the JSP into a servlet.
* Compiling the servlet.

JSP Initialization:

When a container loads a JSP it invokes the jspInit() method before servicing any requests. Typically initialization is performed only once and as with the servlet init method, you generally initialize database connections, open files, and create lookup tables in the jspInit method.

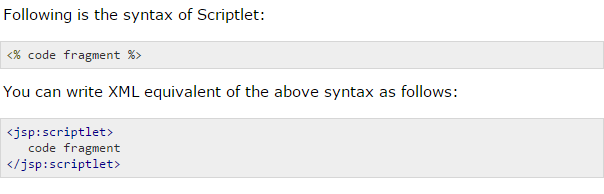
## JSP Execution:

* 1. This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed. Whenever a browser requests a JSP and the page has been loaded and initialized, the JSP engine invokes the **\_jspService()** method in the JSP. The \_jspService() method takes an **HttpServletRequest** and an**HttpServletResponse** as its parameters.
  2. The \_jspService() method of a JSP is invoked once per a request and is responsible for generating the response for that request and this method is also responsible for generating responses to all seven of the HTTP methods ie. GET, POST, DELETE etc.

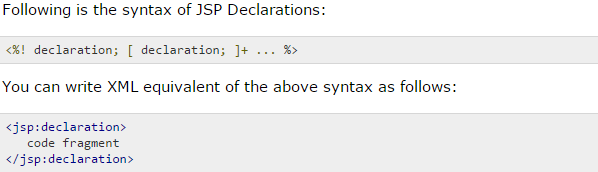
## JSP Cleanup:

The destruction phase of the JSP life cycle represents when a JSP is being removed from use by a container. The **jspDestroy()** method is the JSP equivalent of the destroy method for servlets. Override jspDestroy when you need to perform any cleanup, such as releasing database connections or closing open files.

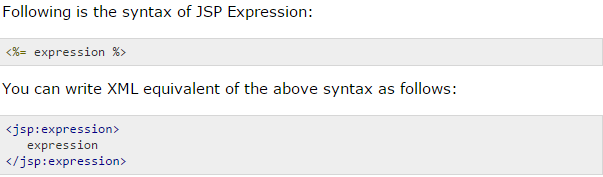
## The Scriptlet:



## JSP Declarations:



## JSP Expression:



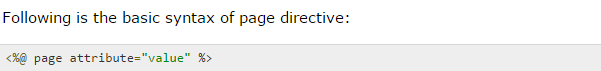
## JSP Comments:

## 

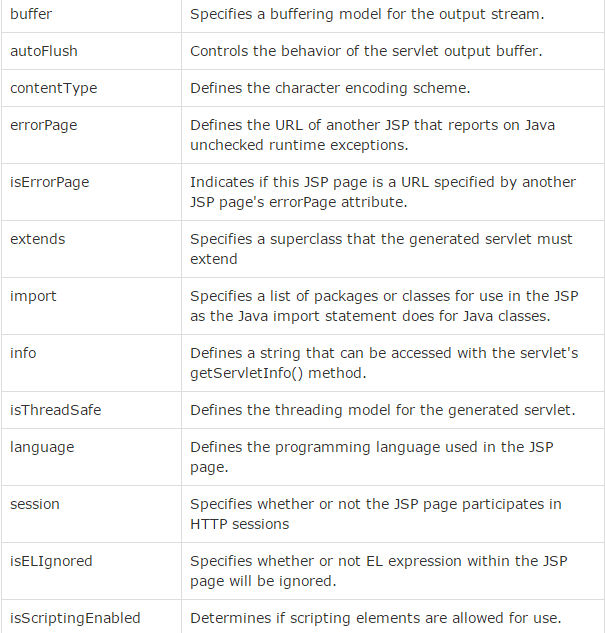
## JSP Directives:

## 

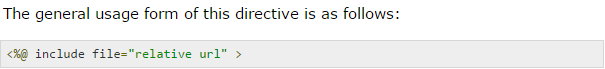
1. **page directive:**



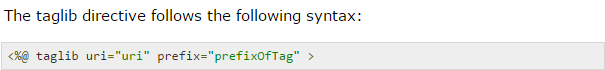
**There are following attributes associated with it.**



1. include directive:



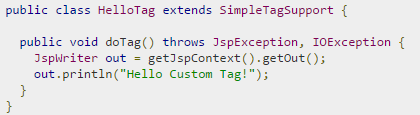
1. taglib directive:



Note: uri-Location of tag, prefix-prefix for use of our tag

Steps:

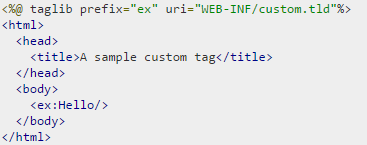
1. Custom tag without body.
2. Create a class than extends SimpleTagSupport.



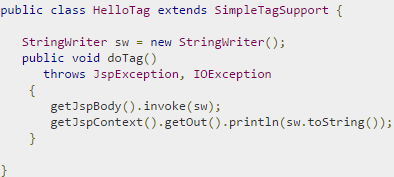
1. Create .tld file.



1. Use custom tag in our app.



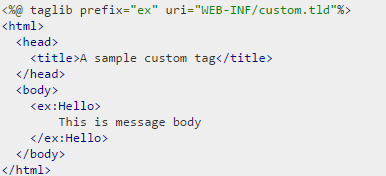
1. Custom tag with body.
2. Create a class than extends SimpleTagSupport.



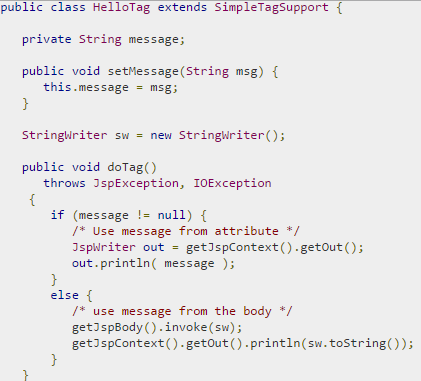
1. Create .tld file.



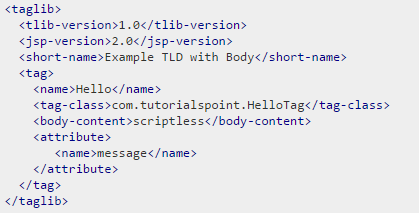
1. Use custom tag in our app.



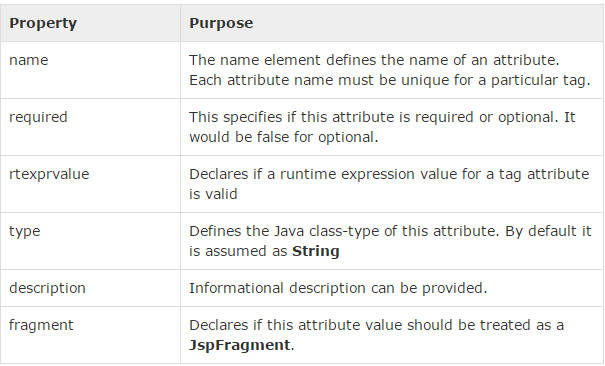
1. Custom tag with attribute.
2. Create a class than extends SimpleTagSupport.



1. Create .tld file.



Note: Other sub tags of <attribute>.



1. Use custom tag in our app.

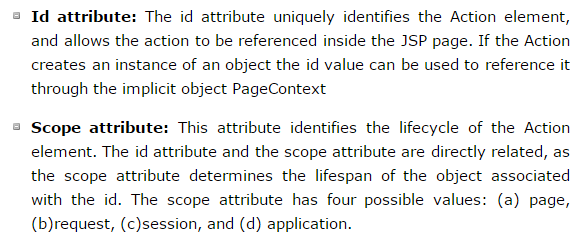


**JSP Actions:**

**Syntax:**

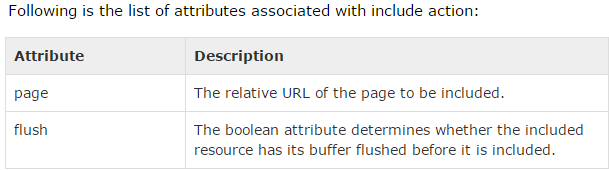


**Following two attributes are common to all the actions.**



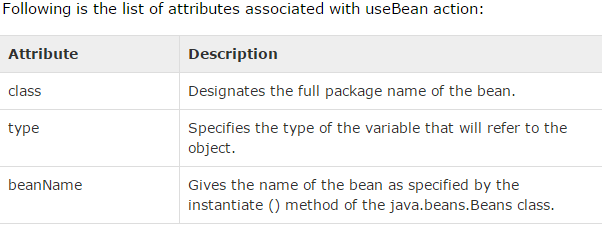
1. **<jsp:include> :** Unlike the **include** directive, which inserts the file at the time the JSP page is translated into a servlet, this action inserts the file at the time the page is requested.



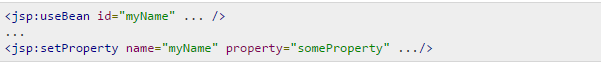


1. **<jsp:useBean> :** It first searches for an existing object utilizing the id and scope variables. If an object is not found, it then tries to create the specified object. Once a bean class is loaded, you can use **jsp:setProperty** and**jsp:getProperty** actions to modify and retrieve bean properties.



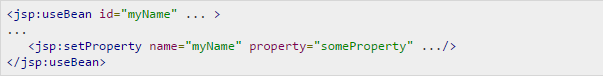


1. **<jsp:setProperty>:** There are two basic ways to use the setProperty action.
2. You can use jsp:setProperty after, but outside of, a jsp:useBean element, as below:

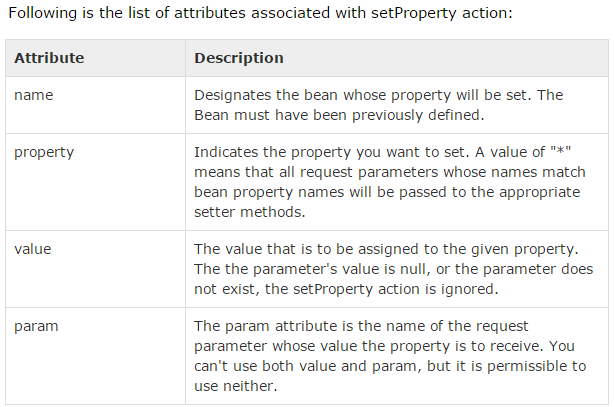


In this case, the jsp:setProperty is executed regardless of whether a new bean was instantiated or an existing bean was found.

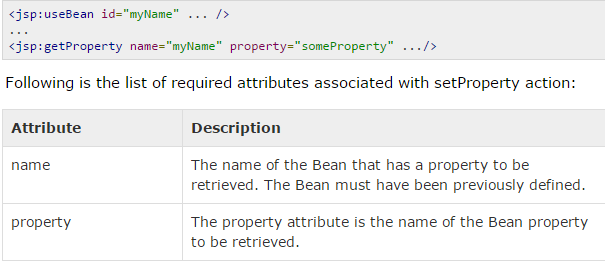
1. jsp:setProperty can appear is inside the body of a jsp:useBean element, as below:



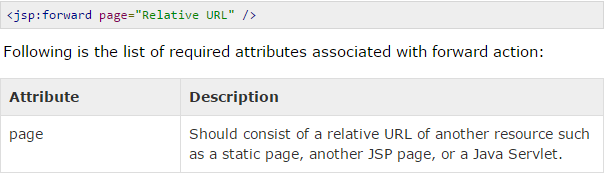
Here, the jsp:setProperty is executed only if a new object was instantiated, not if an existing one was found.



1. **<jsp:getProperty>:**The **getProperty** action is used to retrieve the value of a given property and converts it to a string, and finally inserts it into the output.



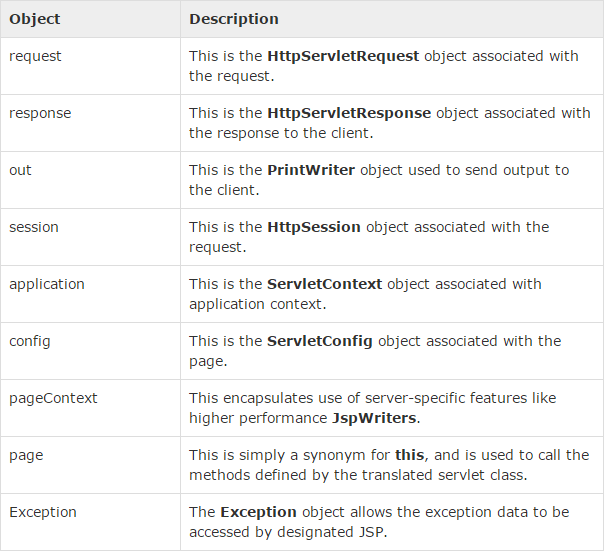
1. **<jsp:forward>:**The **forward** action terminates the action of the current page and forwards the request to another resource such as a static page, another JSP page, or a Java Servlet.



1. **<jsp:element>,<jsp:attribute>,<jsp:body> :** The <jsp:element>, lt;jsp:attribute> and <jsp:body> actions are used to define XML elements dynamically. The word dynamically is important, because it means that the XML elements can be generated at request time rather than statically at compile time.



**JSP – Implicit objects**



* How to handle exception by specify error page in web.xml: we don’t need to mention error page in all jsp’s.

