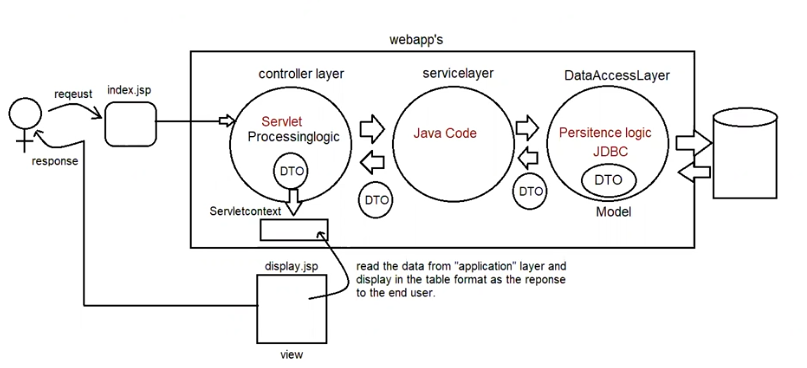
2. <jsp:include>=> include request Dispatching mechanism.

<jsp:forward>=> forward request Dispatching mechanism.

Eg: ForewardIncludeActionsJsp



Eg:ParamJspActions

<jsp:param name = " value=''"> => To add new values to request object and send it to the respective page we use

<jsp:param name = " value="/>

deprecated jsp actions are

<jsp:plugin>

<jsp:fallback>

<jsp:params>

Customactions

===========

These are the actions which are developed by developers as per their application requirements.

In jsp two types of tags are available

a. standardactions -> predefined tags known to container

b. customactions -> inform explicitly to the container.

To prepare custom tags in Jsp pages we use the following syntax.

<prefix\_name:tag\_name>

;;;;;;;;

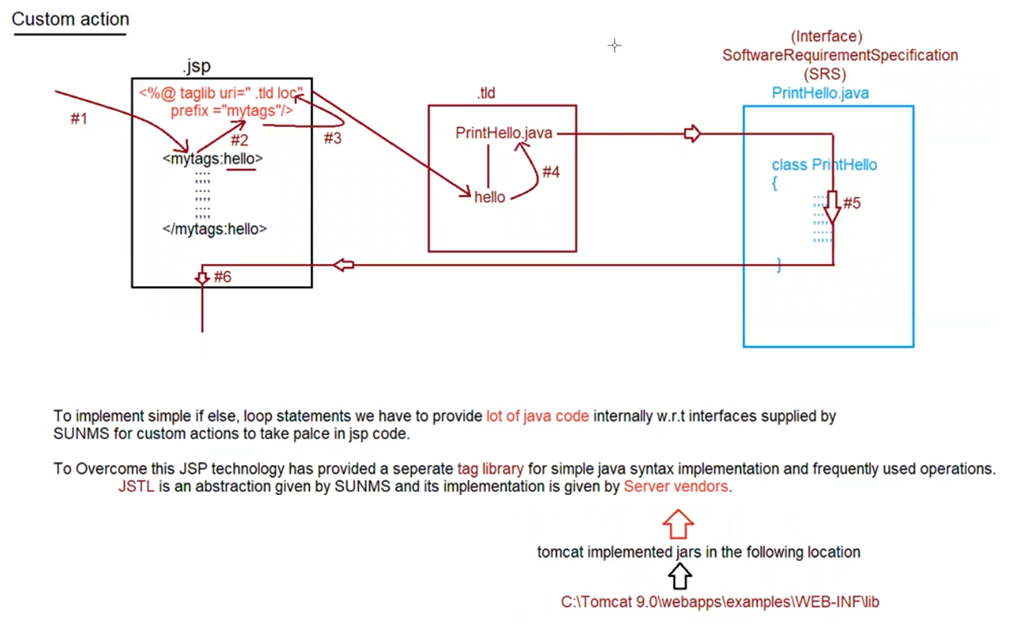
;;;;;;;;

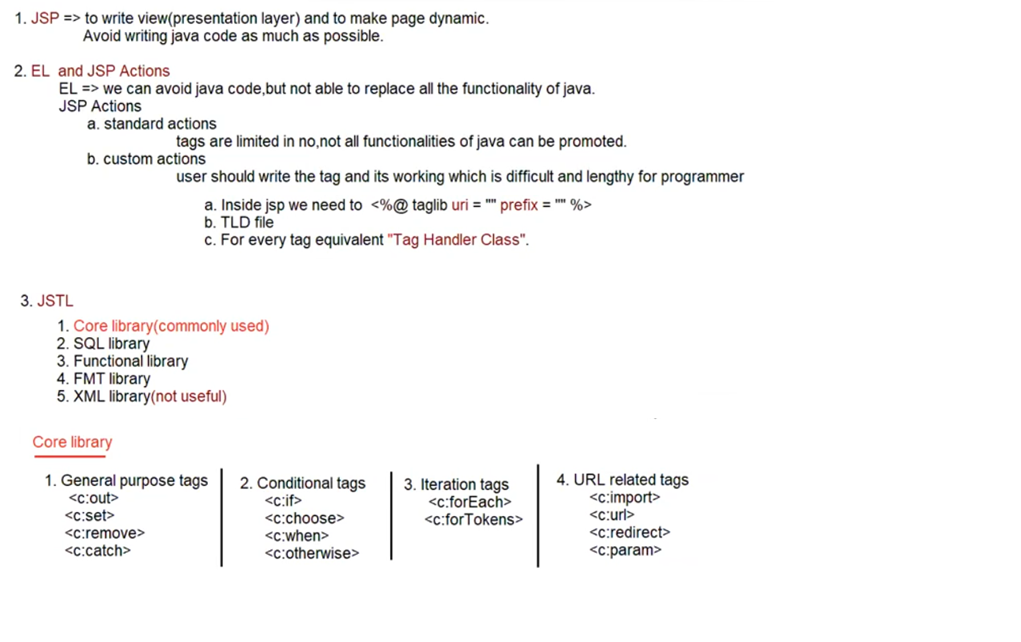
;;;;;;;;

</prefix\_name:tag\_name>

If we want to design custom tags in our jsp application, then we use the following 3 elements

a. jsp page with taglib directive





Core Library:

===========

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<c:out>

It is used for writing Template text data and expression to the JSP.

c=> prefixName

out => tagName

Eg: CustomActionsJsp

2. <c:set>

We can use to set attributes in any scope and to set map and bean properties also.

<c:set var="x" value="10" scope="request"/>

<c:set var="y" value="20" scope="request"/>

<c:set var="sum" value="${x+y}" scope="session"/>

<h1 style='color:red; text-align:center;'>

The result is :: <c:out value="${sum}"/>

3. <c:remove>

To remove attributes in the specified scope we can use this tag.

if the scope is not specified for removing, by default it will search in

a. page scope

b. request scope

c. session scope

d. application scope

Eg: SetRemoveAttributeJstl

4.

<c:catch var="">

//risky code

</c:catch>

If any exception occurs, then that exception object is collected inside var attribute vaiable which is page scope.I

if any exception is raised inside risky code, then this tag suppress that exception and rest of the jsp wil be executed

normally.

<c:if test= "">

//hanlding code

</c:if>

Eg: ExceptionalHandlingJstl

Conditional Tags

==============

1. <c:if>

It is used to implement core java if statement

<c:if test="" scope="" var="">

//body of if

</c:if>

if the condition evaluates to true only then body of if will be executed.

Eg: ConditionalTagsIfJstl

2. <c:choose>, <c:when> and <c:otherwise>

We can use these tags for implementing if else and switch statements.

Implementing if-else

===================

<c:choose>

<c:when test = "condition">

ACTION-1

</c:when>

<c:otherwise>

ACTION-2

</c:otherwise>

</c:choose>

if condition evaluates to tree then ACTION-1 otherwise ACTION-2

Implementing switch

=================

<c:choose>

<c:when test = "test\_condition1">

ACTION-1

</c:when>

<c:when test = "test\_condition2">

ACTION-2

</c:when>

<c:when test = "test\_condition2">

ACTION-3

</c:when>

<c:when test = "test\_conditionN">

ACTION-N

</c:when>

<c:otherwise>

Default Action

</c:otherwise>

</c:choose>

Note:

1. <c:when> tag explicitlly contains break statement, so no chance of fall through in switch.

2. <c:otherwise> should always be last case only

3. <c:choose> should compulsorily contain one <c:when> tag, but <c:otherwise> is optional.

Eg: ChooseWhenOtherWiseJstl

Iteration tags

============

1. <c:forEach begin="" end="" step="">

It would ressamble general purpose for loop.

default value of step is "1", it gets incremented automatially.

The loop body will be executed w.r.t "begin <= end".

2. <c:forTokens>

It is a specialized version of forEach to perform StringTokenization based on some delimitor.

Syntax:

<c:forTokens items ="" delims="" var="' begin="" end="" step="">

//body

</c:forTokens>

Note:

<c:forTokens> items attribute should be string only.

<c:forEach> items attributes can be String,Collection object,Map etc.

Eg: IterationTagJstl

Note:

<c:forTokens> items attribute should be string only.

<c:forEach> items attributes can be String,Collection object,Map etc.

URL related tags

============

1. <c:import>

we can use this tag for importing the response of the other pages in the current page response at the time of request processing.(ie dynamic include)

Eg: UrlRelatedTags

Eg: UrlRelatedTags2

2. <c:redirect>

This is similar to sendRedirect() method of ServletResponse.

Eg: UrlTagsRedirect

Eg: UrlTagsUrl