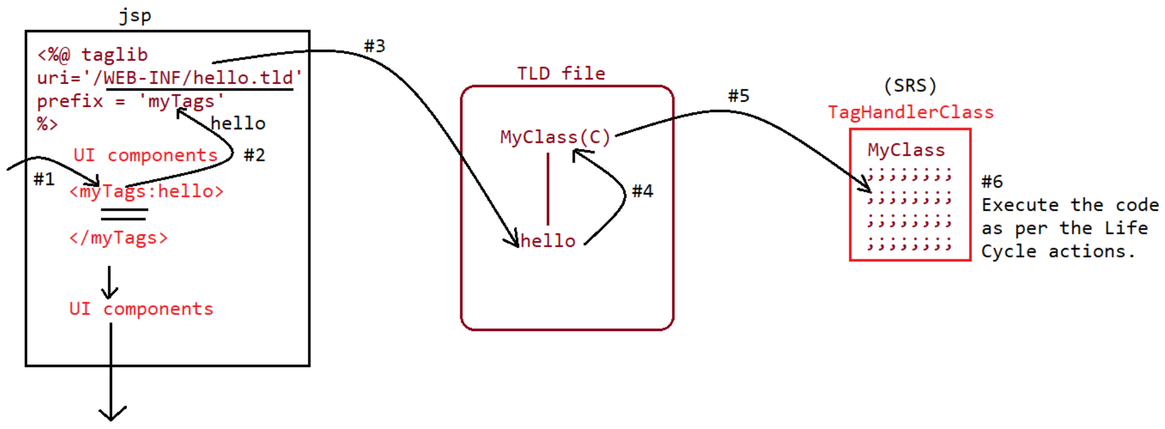
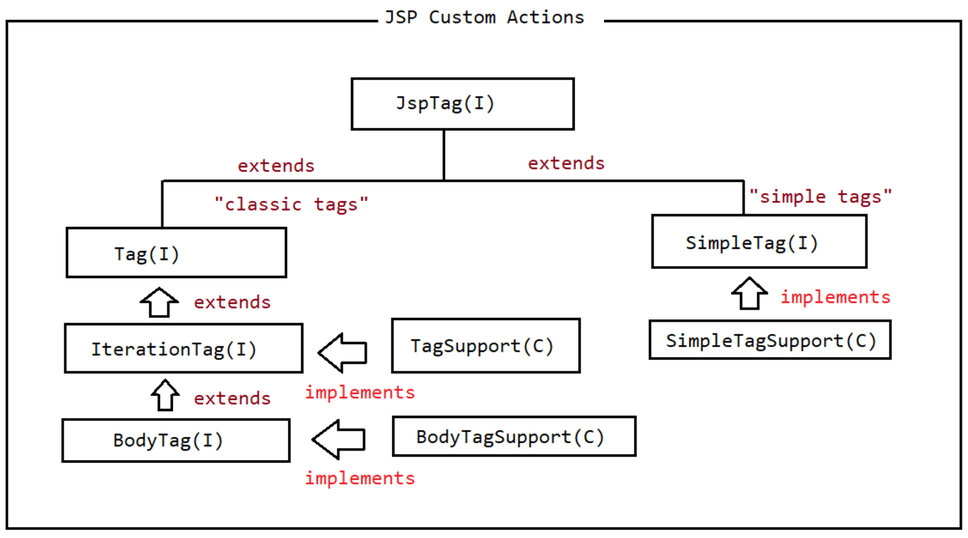
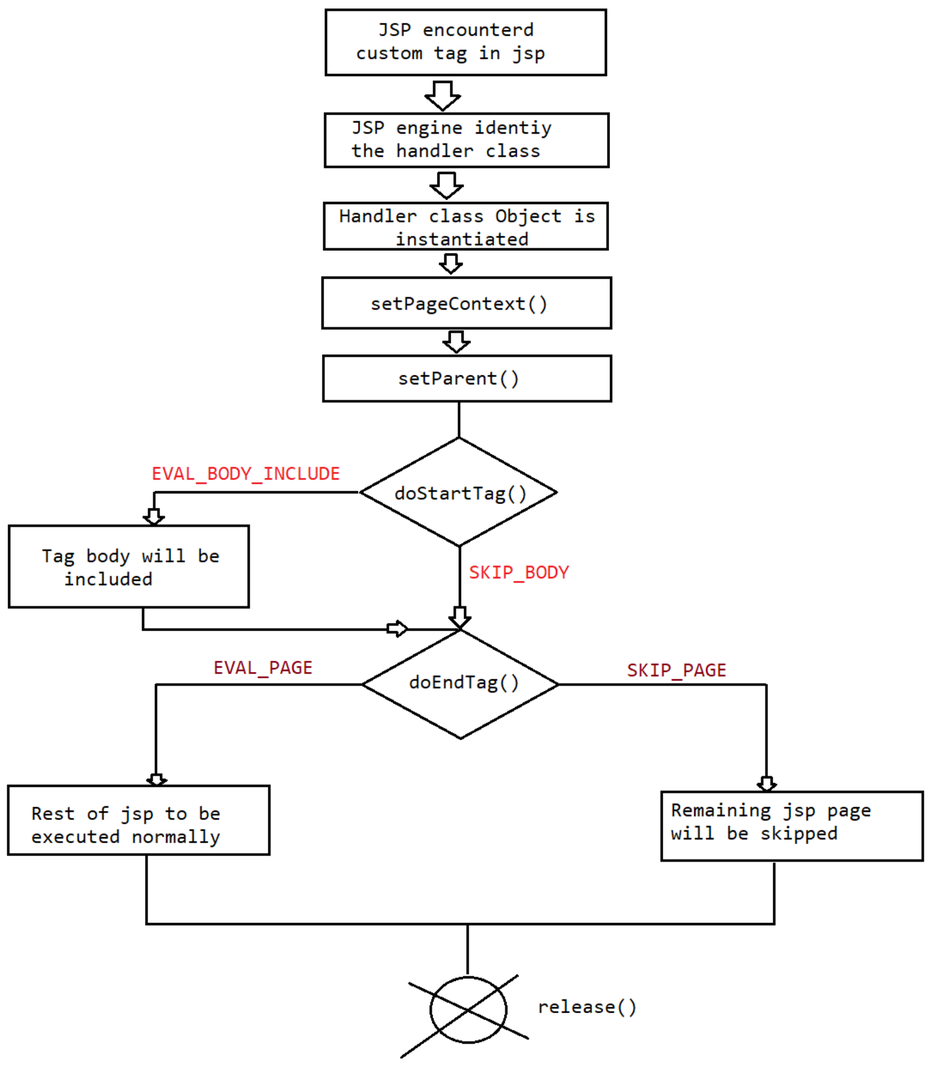




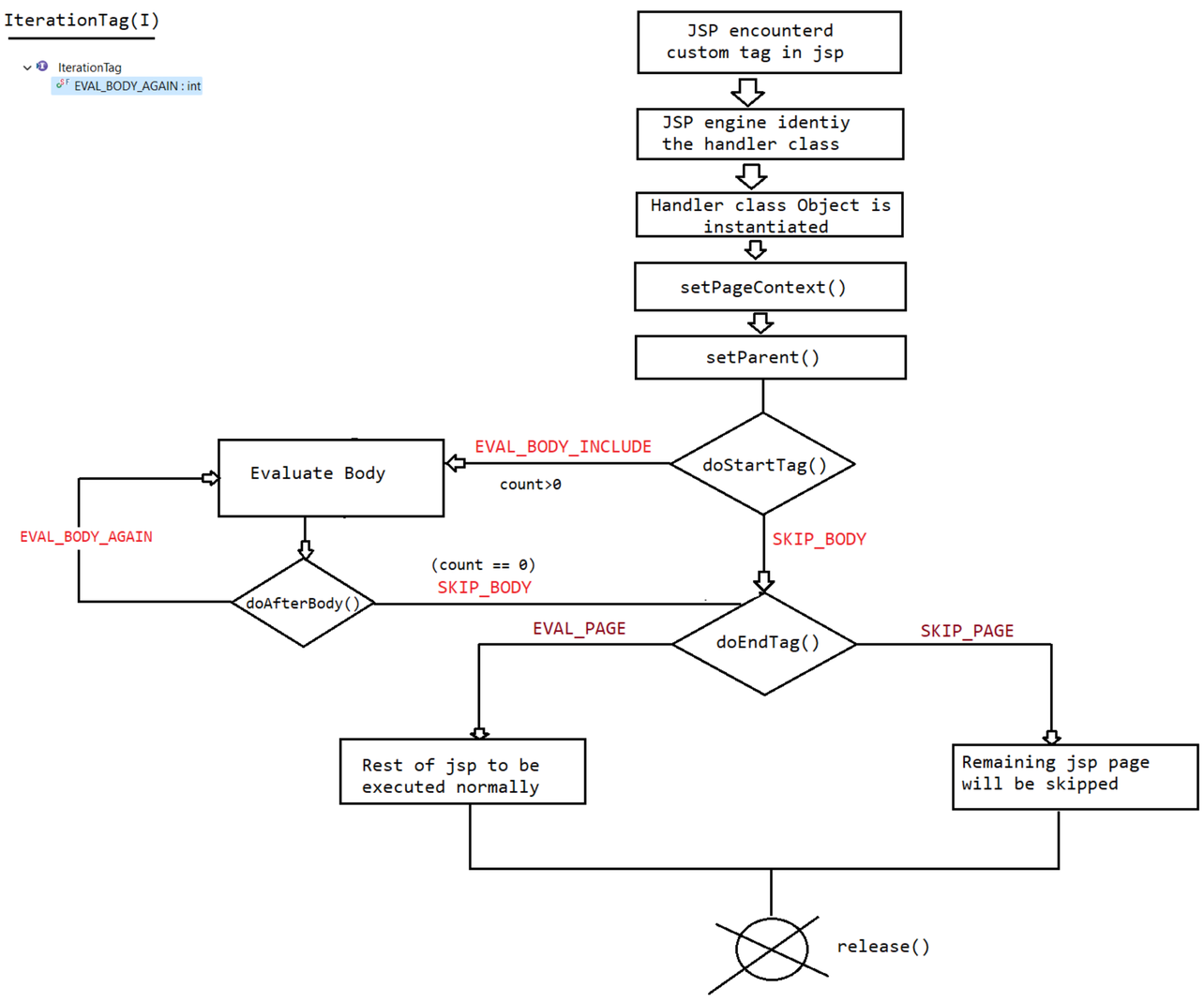
Tag(I)

- SKIP_BODY : int
- EVAL_BODY_INCLUDE : int
- SKIP_PAGE : int
- EVAL_PAGE : int



IterationTag(I)

- IterationTag
  - EVAL_BODY_AGAIN : int



JSP(main theme is to avoid java code as much as possible)
==========
Standard actions ===> Predefined actions provided by jsp technology.
Custom actions
   These are the jsp actions which are prepared by the developers as per their application requirement.

syntax: <prefix_Name: tagName>

            </prefix_Name>

To work with custom actions we need to use
      a.jsp taglib directive
      b.TLD file(Taglibrary descriptor)
      c. TagHandler class
                        refer: ***.png

TagHandler class
==========
In custom tags preparation, the main purpose of Tag Handler class is to define the basic functionality for the
custom tags.
      To define the Taghandler class in custom tags preparation JSP API has provided a predefined library in the form
       of javax.servlet.jsp.tagext.* package.
                                    refer: JspCustomAction-01,JspCustomAction-02

Note:
1. To avoid java code inside jsp, we use custom actions but writing a complex logic using custom actions is tough.
2. To avoid writing the custom actions for commonly use java code there is an API defined by SUNMS in the form
     of "JSTL-API(Java Standard Tag Library)".
3. Using JSTL 90% of java code can be avoided in JSP.
4. Remaining 10% of java code can also be avoided using "EL" inside jsp.
5. JSP => EL + JSTL + HTML code. [so no java code inside jsp]

EL and scopes in JSP

Scopes
   In servlets we have 3 scopes
            a. Request Scope
            b. Session Scope
            c. Application Scope

  In Jsp addition to this 3 scopes we have
            a. request scope(request implicit object)
            b. Session scope(session implicit object)
            c. application scope(context implicit object)
            d. page scope.(page implicit object)

The order of scope is
      pagescope<requestscope<sessionscope<applicationscope

What is the difference b/w pageContext.getAttribute("key") vs pageContext.findAttribute("key")?

pageContext.getAttribute(k) => It will search only in page scope,if found returns value otherwise returns null.
pageContext.getAttribute(k,scope) => It will search in the specified scope if found return value otherwise it
returns null.
pageContext.findAttribute(k) => It will search in pagescope,requestscope,sessionscope,applicationscope if found
in any scope particular value will be returned othewise it would return null value.

refer: JspScopeApp

Expression Language
==========
The main intention of EL is to avoid java code inside jsp

eg:
To print the value of the reques parameter we use
The username is :: ${param.user} ====> request.getParameter("user")
To print the value of any attribute in any scope
The attribute value is :: ${attributeName}===> pageContext.getAttribute("attributeName",1)
To take the values from context object which is configured in web.xml file we can use
initParam -> context.getInitParameter('key')

web.xml

```
<web-app>
  <context-param>
      <param-name>user</param-name>
      <param-value>root</param-value>
  </context-param>
  <context-param>
      <param-name>pwd</param-name>
      <param-value>root123</param-value>
  </context-param>
</web-app>
```

index.jsp
=======
```
<h1>UserName is :: ${initParam.user }</h1>
<h1>Password is   :: ${initParam.pwd }</h1>
<h1>URL is             :: ${initParam.url }</h1>
```

Note: if we are using any variable directly in EL syntax, then that variable should be in some scope otherwise it
the expression won't be evaulated by JSPEngine.

To search for a particular attribute in the particular scope we can use
a. ${sessionScope.attributeName} ====> pageContext.getAttribute("",3)
b. ${requestScope.attributeName} ====> pageContext.getAttribute("",2)
c. ${applicationScope.attributeName}===> pageContext.getAttribute("",4)

Note: Between JSP and EL the only common implicit object available is "pageContext"
In EL session,request implicit objects are not directly available.

refer: JSPELApp