JSP to Freemarker conversion

Consider the following use case: what does it take and, overall, how long does it take to convert the following JSPs

entando-widget-navigation__bar.jsp

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

<%--
  A 8-year-long effort, lovely brought to you by:

    - Marco Diana <m.diana@entando.com>
    - Eugenio Santoboni <e.santoboni@entando.com>
    - William Ghelfi <w.ghelfi@entando.com>
    - Andrea Dessì <a.dessi@entando.com>
--%>

<wp:headInfo type="JS" info="entando-misc-jquery/jquery-1.10.0.min.js" />
<wp:headInfo type="JS" info="entando-misc-bootstrap/bootstrap.min.js" />

<wp:currentPage param="code" var="currentPageCode" />
<c:set var="currentPageCode" value="${currentPageCode}" />
<c:set var="previousPage" value="${null}" />
<ul class="nav">
<wp:nav var="page">

<c:if test="${previousPage.code != null}">
    <c:set var="previousLevel" value="${previousPage.level}" />
    <c:set var="level" value="${page.level}" />
    <%@ include file="entando-widget-navigation_bar_include.jsp" %>
</c:if>

    <c:set var="previousPage" value="${page}" />
</wp:nav>

<c:if test="${previousPage != null}">
    <c:set var="previousLevel" value="${previousPage.level}" />
    <c:set var="level" value="${0}"  /> <%-- we are out, level is 0 --%>
    <%@ include file="entando-widget-navigation_bar_include.jsp" %>
    <c:if test="${previousLevel != 0}">
        <c:forEach begin="${0}" end="${previousLevel -1}"></ul></li></c:forEach>
    </c:if>
</c:if>
```

```
</ul>
```

and **entando-widget-navigation__bar__include.jsp**

```
<%@ taglib prefix="wp" uri="/aps-core" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<c:set var="liClass" value="" />
<c:set var="homeIcon" value="" />
<c:set var="caret" value="" />
<c:set var="ulClass" value=' class="dropdown-menu"' />
<c:set var="aClassAndData" value="" />
<c:set var="aURL" value="${previousPage.url}" />

<c:if test="${previousPage.voidPage}">
  <c:set var="aURL" value='#' />
</c:if>

<c:if test="${fn:containsIgnoreCase(previousPage.code, 'homepage')}">
    <c:set var="homeIcon"><i class="icon-home"></i>&#32;</c:set>
</c:if>

<c:if test="${previousPage.code == currentPageCode}">
    <c:set var="liClass" value=' class="active"' />
</c:if>

<c:if test="${previousLevel < level}">
    <c:set var="liClass" value=' class="dropdown"' />
    <c:if test="${previousPage.code == currentPageCode}">
        <c:set var="liClass" value=' class="dropdown active"' />
    </c:if>
    <c:if test="${previousPage.voidPage}">
        <c:set var="liClass" value=' class=" dropdown"' />
    </c:if>

    <c:if test="${previousLevel > 0}">
        <c:set var="liClass" value=' class="dropdown-submenu"' />
        <c:if test="${previousPage.code == currentPageCode}">
            <c:set var="liClass" value=' class="dropdown-submenu active"' />
        </c:if>

        <c:set var="ulClass" value=' class="dropdown-menu"' />
    </c:if>
    <c:set var="aClassAndData" value=' class="dropdown-toggle" data-toggle="dropdown"' />
    <c:if test="${previousLevel == 0}">
        <c:set var="caret"> <span class="caret"></span></c:set>
    </c:if>
</c:if>

<li<c:out value="${liClass}" escapeXml="false" />><a href="<c:out value="${aURL}" escapeXml=

<c:if test="${previousLevel == level}"></li></c:if>
<c:if test="${previousLevel < level}"><ul<c:out value="${ulClass}" escapeXml="false" />></c:
<c:if test="${previousLevel > level}">
    <c:forEach begin="${1}" end="${previousLevel - level}"></li></ul></c:forEach></li>
</c:if>
```

into the following Freemarker templates

```
<#assign c=JspTaglibs["http://java.sun.com/jsp/jstl/core"]>
<#assign wp=JspTaglibs["/aps-core"]>

<#--
  A 8-year-long effort, lovely brought to you by:

    - Marco Diana <m.diana@entando.com>
    - Eugenio Santoboni <e.santoboni@entando.com>
    - William Ghelfi <w.ghelfi@entando.com>
    - Andrea Dessì <a.dessi@entando.com>
-->

<@wp.headInfo type="JS" info="entando-misc-jquery/jquery-1.10.0.min.js" />
<@wp.headInfo type="JS" info="entando-misc-bootstrap/bootstrap.min.js" />

<@wp.currentPage param="code" var="currentPageCode" />
<@wp.freemarkerTemplateParameter var="currentPageCode" valueName="currentPageCode" />
<ul class="nav">
<@wp.nav var="page">

<#if (previousPage?? && previousPage.code??)>
    <#assign previousLevel=previousPage.level>
    <#assign level=page.level>
        <@wp.freemarkerTemplateParameter var="level" valueName="level" />
    <@wp.freemarkerTemplateParameter var="previousLevel" valueName="previousLevel" />
    <@wp.fragment code="entando-widget-navigation_bar_include" escapeXml=false />
</#if>

    <@wp.freemarkerTemplateParameter var="previousPage" valueName="page" />
```

```
</@wp.nav>

<#if (previousPage??)>
    <#assign previousLevel=previousPage.level>
        <#assign level=0>
    <@wp.freemarkerTemplateParameter var="level" valueName="level" />
    <@wp.freemarkerTemplateParameter var="previousLevel" valueName="previousLevel" />
    <@wp.fragment code="entando-widget-navigation_bar_include" escapeXml=false />

        <#if (previousLevel != 0)>
        <#list 0..(previousLevel - 1) as ignoreMe>
            </ul></li>
        </#list>

    </#if>
</#if>

</ul>
```

and

```
<#assign wp=JspTaglibs["/aps-core"]>
<#assign c=JspTaglibs["http://java.sun.com/jsp/jstl/core"]>


<#assign liClass="">
<#assign homeIcon="">
<#assign caret="">
<#assign ulClass=' class="dropdown-menu"'>
<#assign aClassAndData="">
<#assign aURL=previousPage.url>

<#if (previousPage.voidPage)>
      <#assign aURL='#' />
</#if>

<#if (previousPage.code?contains("homepage"))>
     <#assign homeIcon='<i class="icon-home"></i>&#32;'>
</#if>

<#if (previousPage.code == currentPageCode)>
     <#assign liClass=' class="active"'>
</#if>

<#if (previousLevel < level)>
```

```
    <#assign liClass=' class="dropdown"' >

    <#if (previousPage.code == currentPageCode)>
  <#assign liClass=' class="dropdown active"'>
    </#if>

    <#if previousPage.voidPage>
    <#assign liClass=' class=" dropdown"' >
    </#if>

    <#if (previousLevel > 0) >
    <#assign liClass=' class="dropdown-submenu"'>
    <#if (previousPage.code == currentPageCode)>
        <#assign liClass=' class="dropdown-submenu active"'>
        </#if>

    <#assign ulClass=' class="dropdown-menu"'>
    </#if>

    <#assign aClassAndData=' class="dropdown-toggle" data-toggle="dropdown"'>

    <#if (previousLevel == 0)>
    <#assign caret=' <span class="caret"></span>'>
    </#if>
</#if>

<li ${liClass} >
    <a href="${aURL}"  ${aClassAndData} >
                <!-- [ ${previousLevel} ] -->
                ${homeIcon}
                ${previousPage.title}
                ${caret}
    </a>

<#if (previousLevel == level)></li></#if>
<#if (previousLevel < level)>
    <ul ${ulClass}>
</#if>
<#if (previousLevel > level)>
     <#list 1..(previousLevel - level) as ignoreMe>
            </li></ul>
     </#list>
    </li>
</#if>
```

Let's start recalling that in Entando there is a frontend and a backoffice: JSPs continue to exist and remain unchanged for the backoffice since there's little to none interest in administrators to change the backoffice; this kind of things is usually made by the Entando team.

What should be (and actually is) customizable are the JSP of the portal: this includes the JSP of the widgets which offer their functionality either via standard JSP file or Internal servlet.

Good news first: *converting internal servlets is almost immediate using a few rules*; converting JSPs of the portal is not difficult but there are differences and caveats that must be kept clear in mind: that's the reason why we are using simple JSP files and not internal servlets in this tutorial. Another good news is that standard Entando tags continue to exist and are used in the same manner, the difference lays in that there is a slight different syntax to use them. What is going to change are the classical `<c:xyz />` tags and we'll show how (and why).

So we are not going to write the nth Freemarker tutorial but a simple step by step guide that aims to make as easy as possible to transform a standard Entando JSP to a standard Entando Freemarker *Fragment*; a fragment is the result of the conversion of a JSP into Freemarker.

Lets start from the beginning and proceed with order. Generally speaking it turned out that it's easier to start converting the JSP included (if any) and then converting the "main" one: in this case we would convert *entando-widget-navigation_bar_include.jsp* and then *entando-widget-navigation_bar.jsp*; however since there are a few observations to do we start our process from the entando-widget-navigation_bar.jsp itself.

It's quite natural to start from the top of the file.

Tags declaration

In a nutshell: a classic tag declarations must be substituted with one of the following:

```
<#assign c=JspTaglibs["http://java.sun.com/jsp/jstl/core"]>
<#assign s=JspTaglibs["/struts-tags"]>
<#assign wp=JspTaglibs["/aps-core"]>
<#assign wpsa=JspTaglibs["/apsadmin-core"]>
<#assign wpsf=JspTaglibs["/apsadmin-form"]>
```

Comments

A comment in Freemarker is always of the form `<#-- my strategic comment -->`

Tags utilization

The rule is rather simple: a tag prefix must be preceded by the '@' and the tag must be preceded by the dot '.' so that

```
<wp:headInfo type="JS" info="entando-misc-jquery/jquery-1.10.0.min.js"
/>
```

becomes

```
<@wp.headInfo type="JS" info="entando-misc-jquery/jquery-1.10.0.min.js"
/>
```

Setting variables #1

After a few trivial substitutions we came to the main reason we have started directly from the *entando-widget-navigation_bar.jsp* JSP.

Consider the following statements:

```
<c:set var="currentPageCode" value="${currentPageCode}" />
<c:set var="previousPage" value="${null}" />
```

currentPageCode is a varible used in the included file *entando-widget-navigation_bar_include.jsp*: it turned out that the `<c:set/>` (or `<@c.set />`) tag won't work as expected.
Normally we would use the Freemerker `<#assign>` tag but, again, this isn't going to work as expected in the included fragment.

We solve the problem using

```
<@wp.freemarkerTemplateParameter var="currentPageCode" valueName="currentPageCode"
/>
```

Basically we take the value of *currentPageCode* from the JSTL context and put it into the Freemarker one. Please note how we conveniently kept the same variable name, so that it will be later resoved as `${currentPageCode}` or directly as `currentPageCode`, depending on the case.

What about `<c:set var="previousPage" value="${null}" />`? The answer is simple: we *don't convert it.*

Freemarker will later complain, stopping the parsing of the template, if a reference to a null (or unknown variable) is made. *Always* check for null values. Be warned.
(Recommended read: Why is FreeMarker so picky about null-s and missing variables)

At this point we are halfway in the conversion process of the JSP, nothing difficult so far, isn't it?

Now let's have a look inside the `<wp:nav var="page">` tag: there are a few things that deserve our attention:

if-then-else clause

The rule is simple here: don't use `<c:if>`, don't use `<c:chose>` and similar. Freemarker has its own `<#if>` tag, which s used in this way:

```
<#if (expression) > [or <#if boolean_condition> ]
<#else>
</#if>
```

Please note how the *#else* tag is nested inside the if: there's no `</#else>`!

We must ensure that **previousPage** is not null. Freemarker does this in a fancy way, using the double question mark `??`. So

```
<c:if test="${previousPage.code != null}">
```

becomes

```
<#if (previousPage??  && previousPage.code??)>
```

Before checking the property **previousPage.code** against the null value we must be sure that **previousPage** itself is not null.

Setting variables #2

Assigning a (not-null) variable is quite easy and natural:

```
<c:set var="previousLevel" value="${previousPage.level}" />
<c:set var="level" value="${page.level}" />
```

are translated immediately into

```
<#assign previousLevel=previousPage.level>
<#assign level=page.level>
```

Note how there are neither `${}` surrounding the *previousPage.level* variable nor the quotations marks `""` for the variable names. No fuss here.
Since both *previousLevel* and *level* will be used in the included JSP (fragment) we use again the `<@wp.freemarkerTemplateParameter />` tags to pass the value.

Generally speaking we use the following syntax:

```
<#assign numeric_variable=11> <#-- No quotations marks! -->
<#assign bool_variable=true> <#-- No quotations marks! -->
<#assign string_variable="a_string">
```

Fragment inclusion

Invoking another fragment is easy:

```
<%@ include file="entando-widget-navigation_bar_include.jsp" %>
```

is immediately substituted with

```
<@wp.fragment code="entando-widget-navigation_bar_include" escapeXml=false
/>
```

entando-widget-navigation__bar__include is the code of the fragment which must exist in the database (we'll cover the details later).
If this fragment is missing or mispelled the proper message will appear in the portal (MY_FRAGMENT NOT FOUND or similar) *but the generation of the output won't be aborted.*

List #1

So far we have the knowledge needed to complete the work, with the solely exception of the List. The syntax for list is rather simple:

```
<#list my_list as current_item>
  ${current_item}
</#list>
```

but this case we also have to deal with indices:

```
<#list 0..(previousLevel - 1) as ignoreMe>
    </ul></li>
</#list>
```

does the trick. `ignoreMe` is the current value of the iteration, which is not needed in this case.

Completing the conversion

At this point we have all that is needed to convert the *entando-widget-navigation_bar_include.jsp*.

This is a fairly simple task, there are only two things which deserve our attention:

conditional statment:

`<#if (previousPage.code?contains("homepage"))> <#assign homeIcon='<i class="icon-home"></i>&#32;'> </#if>` which resulted from

```
<c:if test="${fn:containsIgnoreCase(previousPage.code, 'homepage')}">
  <c:set var="homeIcon"><i class="icon-home"></i>&#32;</c:set>
</c:if>
```

Freemarker has builtin methods for string manipulation so that we avoid including the *fn* tag `<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>`

printing values

```
<a href="${aURL}"  ${aClassAndData} >
  <!-- [ ${previousLevel} ] -->
    ${homeIcon}
    ${previousPage.title}
    ${caret}
</a>
```

is the much simpler and readable form of

```
<a href="<c:out value="${aURL}" escapeXml="false" />"
    <c:out value="${aClassAndData}" escapeXml="false" /> >
    <!-- [ <c:out value="${previousLevel}" /> ] -->
    <c:out value="${homeIcon}" escapeXml="false" />
    <c:out value="${previousPage.title}" />
    <c:out value="${caret}" escapeXml="false" />
</a>
```

we have omitted entirely the c:out tag (which should not be used)

General considerations

Parameters from the session

Use `Session.<PARAM_NAME>`

```
<#if (Session.currentUser != "guest")>
```

More complex list

This is a real working fragment

```
<@wp.info key="langs" var="langsVar" />
<@wp.freemarkerTemplateParameter var="langsListVar" valueName="langsVar" removeOnEndTag=true
<#list langsListVar as curLangVar>
  <#if (curLangVar.code == currentLangVar)>class="active" </#if>
    <a href=<@wp.url lang="${curLangVar.code}" paramRepeat=true />">
        <@wp.i18n key="ESLC_LANG_${curLangVar.code}" />
    </a>
</#list>
</@wp.freemarkerTemplateParameter>
```

- first we obtain the list of the system languages which is put in *langsVar*
- we place the list in the freemarker context with the name *langsListVar*, but we could have retained the same name. *Important: the tag is not closed*
- declare the list whose current item is named *curLangVar*.
- process the current item in the list, feeding the url tag. Important: within the `#if` condition we use `curLangVar.code` without quotation mark, but inside the `<wp:url>` tag we placed a `${curLangVar.code}` because the tag needs the literal value as input.
- close the list (pretty obvious)
- close the freemarkerTemplateParameter deleting the variable `langsListVar` from the context: otherwise the `removeOnEndTag=true` is missing.

Boolean and integers do not need quotation marks

Consider the following example:

```
<wpsf:password
  useTabindexAutoIncrement="true"
    name="password"
    id="userprofile-new-password" />
```

and the corresponding template `<@wpsf.password useTabindexAutoIncrement=true name="password" id="userprofile-new-password" />` `useTabindexAutoIncrement=true` has no quotation marks; otherwise the literal “true” would be used, causing an error.
The same applies for integers.

Pay attention to this:

```
<@wpsf.radio
  useTabindexAutoIncrement=true
    name="%{#attributeTracer.getFormFieldName(#attribute)}"
    id="%{#attribute_id + ''-true''}"
    value="true"
    checked="%{#attribute.value == true}"
    cssClass="radio" />
```

useTabindexAutoIncrement again does not need the quotation marks, *but ‘value’ does*, since the literal is needed as string input for the tag.

Setting variables #3

Another smart way to assign variables in internal servlets is the following:

```
<#assign ideaText>
  <@s.property value='ideaText'/>
</#assign>
```

Parameters from the request #1

```
${RequestParameters['myParameter']}
${RequestParameters.myParameter} <#-- the same as above -->
```

Again, this is handy in internal servlets:

```
<@s.property value="%{#parameters.myParameter}" />
```

Pager

Have a close look at the following snippet:

```
<@wp.pager listName="currentList" objectName="groupIdea" pagerIdFromFrame=true max=5 pagerId


<@wp.freemarkerTemplateParameter var="group" valueName="groupIdea"  >
<@wp.fragment code="default_pagerBlock" escapeXml=false />

<#list currentList as ideaId>

<#if (ideaId_index >= groupIdea.begin) && (ideaId_index <= groupIdea.end)>
```

```
<!-- iteration loop stuff goes here  -->

</#if>
</@wp.pager>
```

The *if* surrounding the inner iteration loop makes sure that the loop is executed only when it is really needed. To get the index for the comparision is sufficient to append *_index* to the desired variable, in this case *ideaId*

New result for core actions.

The careful reader might have already wondered how a fragment can be invoked as a result from the action.

Again, the answer is rather simple:

```
<action name="edit">
  <result type="dispatcher">/WEB-INF/aps/jsp/internalservlet/user/editCurrentUser.jsp</resul
  <interceptor-ref name="entandoFrontEndFreeStack"></interceptor-ref>
</action>
```

turns into

```
<action name="edit">
        <result type="guiFragment">
            <param name="code">user-profile_is_edit</param>
            <param name="jspLocation">/WEB-INF/aps/jsp/internalservlet/user/editCurrentUser.
        </result>
        <interceptor-ref name="entandoFrontEndFreeStack"></interceptor-ref>
</action>
```

returning from a successful action the system will search for the `user-profile_is_edit` in the database (for the curious readers: table **guifragment** in the ***Port** db) If the fragment is not found then the fallback is used and the default `/WEB-INF/aps/jsp/internalservlet/user/editCurrentUser.jsp` is invoked.