1.9.1 Généralités

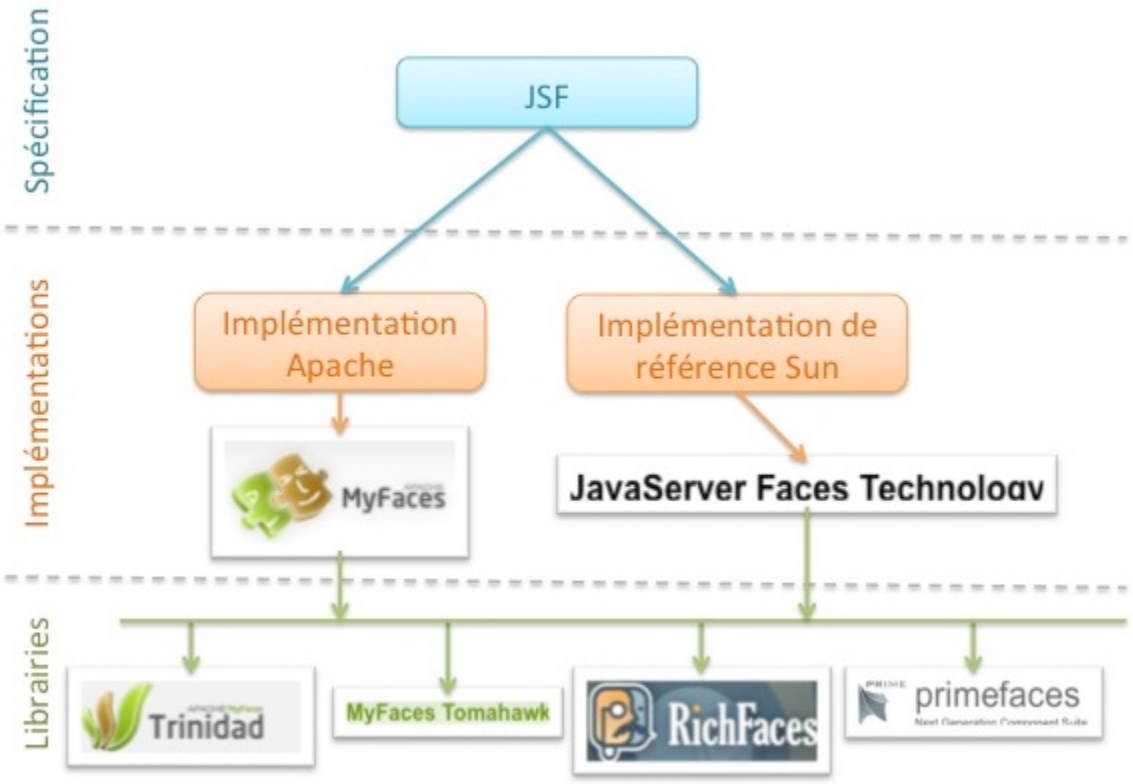
Relu

Relecture RB faite le 16/02/2011

Sommaire :

Introduction

Ce chapitre n'a pas la prétention d'être une formation à JSF. Pour plus d'informations, vous pouvez vous reporter à une documentation comme http://www.jmdoudoux.fr/java/dej/chap-jsf.htm. Ne sont abordés ici que quelques éléments.



Avec *esup-commons V2*, nous utilisons facelet (Cf. 1.9.2 Facelet). Les pages sont suffixées en .xhtml et sont écrites en XML, ce qui permet d'utiliser des espaces de noms et des commentaires XML.

JSF 1.2 ou JSF 2.0 ?

Esup-commons V2 permet d'utiliser JSF 1.2 ou JSF 2.0 !

JSF 2.0 apporte de nombreuses améliorations par rapport à JSF 1.2. De plus, en JSF 2.0 il sera, a priori, beaucoup plus facile de faire cohabiter différentes librairies de composants. D'ailleurs, les dernières librairies (notamment celles qui offrent des composants de haut niveau et permettent un fonctionnement en Ajax) évoluent beaucoup plus en JSF 2.0 qu'en JSF 1.2.

Malheureusement, JSF 2.0 n'est pas compatible avec le mode portlet ! Pour utiliser JSF 2.0 en mode portlet, il faut utiliser un portlet bridge non normalisé qui ne fonctionne qu'en mode portlet 2.0... alors que uPortal 3.2 ne supporte que le mode portlet 1.0 (uPortal 3.3 devrait supporter le mode portlet 2.0). De plus, certaines librairies sont encore en version beta pour JSF 2.0 (ex. de Trinidad à l'heure à laquelle nous écrivons cette page).

C'est la raison pour laquelle *esup-commons V2* propose d'utiliser JSF 1.2 ou JSF 2.0 suivant le contexte cible :

- JSF 1.2 pour le développement d'une portlet
- JSF 2.0 pour le développement d'une servlet (typiquement une application importante contenant beaucoup d'écrans et pour laquelle on souhaite avoir un bon niveau de productivité lors de l'écriture des pages)

Exemple de page

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:t="http://myfaces.apache.org/tomahawk"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:p="http://primefaces.prime.com.tr/ui">
<f:view locale="#{sessionController.locale}">
    <head>
        <!-- Commentaire -->
        <title><h:outputText value="#{applicationService.name}" /></title>
        <ui:repeat value="#{tagsConfigurator.stylesheets}" var="path">
            <t:stylesheet path="#{path}" />
        </ui:repeat>
        <ui:repeat value="#{tagsConfigurator.scripts}" var="path">
            <script type="text/javascript" src="#{path}" />
        </ui:repeat>
    </head>
    <body>
        <ui:include src="_include/_header.xhtml" />
        <div id="welcomepages">
            <div id="navigationHeader">
                <h:form id="nav_header" styleClass="app-form" />
            </div>
            <div>
                <div>
                </div>
                <div>
                    <t:htmlTag value="br" />
                    <ui:insert name="allContent" />
                </div>
                <div style="clear:both;" />
            </div>
            <div id="navigationFooter"
                class="fl-container-flex app-pagebar">
                <h:form id="nav_footer" styleClass="app-form" />
            </div>
        </div>
        <div>
            <h:outputText value="#{applicationService.name} " />
            <h:outputText value="v#{applicationService.version} - " />
            <h:outputText value="#{applicationService.copyright}" />
        </div>
    </body>
</f:view>

</html>
```

Dans la page ci-dessus :

- la balise **<html>** est l'élément racine de la page. Elle contient notamment la définition des espaces de noms qui permettent de définir les composants JSF que l'on va utiliser. Ici :
    - **xmlns:f="http://java.sun.com/jsf/core"** et **xmlns:h="http://java.sun.com/jsf/html"** sont les 2 librairies de base de JSF
    - **xmlns:ui="http://java.sun.com/jsf/facelets"** est la librairie facelet (NB : en JSF 2.0, facelet est aussi une librairie de base de JSF alors qu'elle est optionnelle en JSF 1.2)
    - **xmlns:t="http://myfaces.apache.org/tomahawk"** est une librairie proposée par la fondation Apache qui propose quelques utilitaires (notamment de manipulation du HTML)
    - **xmlns😛="http://primefaces.prime.com.tr/ui"** est la librairie PrimeFaces qui va apporter des composants de haut niveau (interface riche et support de Ajax). Elle est donnée ici à titre d'exemple car, suivant les besoins, on utilisera telle ou telle librairie (RichFaces, Trinidad, OpenFaces, etc.)

- la balise **<f:view locale="#{sessionController.locale}">** est la balise JSF servant à inclure les autres balises JSF. La propriété *locale* permet de définir la langue qui sera utilisée pour les messages.
- la balise **<ui:repeat value="#{tagsConfigurator.stylesheets}" var="path">** permet de répéter n fois un même contenu. n est fonction du nombre d'entrées retournées par la méthode *getStylesheets* du bean *tagsConfigurator*. Chaque entrée est ensuite positionnée dans une variable (*path* ici) afin d'être réutilisée dans la boucle.

Syntaxe EL

JSF dispose d'un *Expressions Language* (*EL*) qui lui est propre et syntaxiquement différent de l'*EL* de *JSP* 2.0. Il est utilisable dans les attributs des *taglibs JSF*. Il permet d'accéder en lecture ou en écriture à des propriétés du contrôleur (dans la mesure où celui-ci implémente des getters/setters sur ces propriétés). Il permet aussi d'invoquer une action du contrôleur (méthode sans paramètre qui renvoie une chaîne).

La syntaxe de base est

```
#{beanName.propertyName}
```

ou

```
#{beanName.methodName}
```

Il est possible d'accéder à des objets récursivement, par exemple :

```
#{homeController.context.name}
```

Il est possible d'accéder à des éléments de tableaux ou de type **Map**, par exemple :

```
#{tableau[1]}
#{hash.key}
#{hash["key"]}
#{hash[keyvar]}
```

(dans ce dernier exemple **keyvar** est évalué avant de retrouver l'entrée dans la table **hash**). Certains tableaux associatifs sont définis par défaut : **param**, **h eader**, **cookie**, etc.

On peut également utiliser quelques opérateurs logiques (**and**, **or**, **not**, **empty**, ...) ainsi que la plupart des opérateurs mathématiques.

Navigation entre les pages

La navigation entre les pages de l'application est faite en *JSF* à l'aide de règles de navigation écrites en *XML*, définies dans *esup-commons* par convention dans le fichier **/webapp/WEB-INF/navigation-rules.xml**.

Cette approche permet de bien dissocier les contrôleurs de la navigation en elle-même. Quand on appelle une méthode d'un contrôleur, par exemple au moment de la validation d'un formulaire, cette méthode (*callback*) renvoie une simple chaîne de caractères qui sera utilisée par *JSF* pour trouver la règle de navigation correspondante dans le fichier **navigation-rules.xml**.

Voici ci-dessous un exemple de règle de navigation :

```
<navigation-rule>
  <display-name>administrators</display-name>
  <from-view-id>/stylesheets/administrators.jsp</from-view-id>
  <navigation-case>
    <from-outcome>addAdmin</from-outcome>
    <to-view-id>/stylesheets/administratorAdd.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>deleteAdmin</from-outcome>
    <to-view-id>
      /stylesheets/administratorDelete.jsp
    </to-view-id>
  </navigation-case>
</navigation-rule>
```

La balise **from-view-id** est facultative. Elle donne la vue (page) à partir de laquelle les règles de navigation vont s'appliquer. Si elle n'est pas présente, la règle est potentiellement applicable depuis toutes les pages de l'application.

Le noeud **navigation-case** permet de définir une règle de navigation :

- **from-outcome** est l'action de l'utilisateur. Cette action peut être définie « en dur » dans la page *JSP* (par ex. **<h:commandButton value=" suivant" action="next"/>**) ou bien être le résultat de l'exécution d'une méthode d'un bean contrôleur de l'application (par ex. <h:commandButton value="suivant" action="#{controller.value}"/>).
- **to-view-id** donne la page de destination.
- **<redirect />** peut être utilisé pour indiquer que l'affichage de la page de destination doit être fait sous forme d'une redirection *HTTP* (code *HTTP* 302). Cette balise doit être utilisée pour protéger l'utilisateur des effets de soumission multiple des formulaires à l'aide des boutons « Page précédente » et « Page suivante » des navigateurs.