

JSF et Facelets

par Jawher Moussa ([Accueil](#))

Date de publication : 5 Novembre 2007

Dernière mise à jour : 6 Novembre 2007

Cet article a pour objectif de vous présenter la technologie de présentation **Facelets** ainsi que de vous guider dans les tâches de configuration et d'utilisation de cette technologie.

I - Présentation de Facelets.....	3
II - La configuration.....	4
III - La création de vues avec facelets.....	5
IV - Le templating.....	6
V - La création de composants.....	12
V-A - Créer la page xhtml du composant.....	12
V-B - Création du taglib.xml.....	12
V-C - Déclaration dans web.xml.....	13
V-D - Utilisation du composant.....	13
V-E - Remarques.....	14
VI - Autre fonctionnalités de Facelets.....	15
VI-A - Débogage.....	15
VI-B - Inclusion.....	15
VI-C - Répétition.....	15
VI-D - Autres.....	15
VII - Conclusion.....	16
VIII - Remerciements.....	17

I - Présentation de Facelets

Comme  **JSP**, **Facelets** est une technologie de présentation pour le développement d'applications web en *Java*.

Une page *JSP* est transformée en une *Servlet* qui possède un cycle de vie différent de celui de *JSF*, ce qui peut être source de confusion et de problèmes. A l'inverse, *Facelets* est spécifiquement développé pour *JSF* et est plus performant et léger.

Facelets introduit aussi des fonctionnalités au-delà de celles offertes par le *JSP*, comme par exemple un système de *templating* ou encore la possibilité de créer des composants personnalisés sans écrire la moindre ligne de code *Java*.


II - La configuration

L'installation de Facelets nécessite :

- 1 Télécharger la distribution *Facelets* [ici](#).
- 2 Copier le fichier ***jsf-facelets.jar*** dans le dossier ***WEB-INF/lib*** de l'application.
- 3 Ajouter les paramètres d'initialisation de *Facelets* dans ***web.xml***.
- 4 Ajouter ***FaceletesViewHandler*** dans ***faces-config.xml***.


Ajout des paramètres d'initialisation :

Dans le fichier de configuration de l'application web, le fichier ***web.xml***, il faut ajouter un élément ***<context-param>*** pour spécifier l'extension par défaut des vues à traiter par *Facelets*.

Dans ce cas de figure, l'extension utilisée est le  ***xhtml***.

LIST. 1 - Configuration de web.xml

```
<web-app>
:
  <context-param>
<param-name>javax.faces.DEFAULT_SUFFIX</param-name>
<param-value>.xhtml</param-value>
</context-param>
:
:
</web-app>
```

 Notez cependant que les pages *xhtml* doivent toujours être référencées par le suffixe ***".jsf"*** ou encore le préfixe ***"faces/"*** dans les liens pour qu'elles soient traitées par la *FacesServlet*.

Ajouter *FaceletesViewHandler* dans *faces-config.xml* :

Il s'agit d'indiquer à JSF d'utiliser *Facelets* comme gestionnaire de vues : ***"View Handler"***. Un ***"View Handler"*** est un plugin JSF qui traite les deux phases *"rendu réponse"* et *"restaurer vue"*.

Pour ce faire, il suffit d'ajouter l'élément ***<view-handler>*** dans ***faces-config.xml***, le fichier de configuration de JSF.

LIST. 2 - Configuration de faces-config.xml

```
<faces-config>
:
  <application>
    <view-handler>com.sun.facelets.FaceletViewHandler</view-handler>
  </application>
</faces-config>
```

III - La création de vues avec facelets

Facelets est basé sur *xml*, c'est pour cette raison que les vues sous *facelets* sont des pages *xhtml* (ou encore *jsp*) et qu'elles doivent impérativement respecter la structure d'un document *xml*.

Considérons la page suivante :

LIST. 3 - une page xhtml

```
<!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:h="http://java.sun.com/jsf/html">
<head>
  <title>Ma premiere page XHTML avec facelets</title>
</head>
<body>
  <f:view>
    <h:outputText value="Hello World" />
  </f:view>
</body>
</html>
```

Cette page se contente d'afficher le message "*Hello World*" mais elle met en valeur les bases du *xhtml*. Notez surtout la façon dont on importe une bibliothèque de composants dans *facelets*, avec l'attribut *xmlns* dans la balise *html*. Par exemple, cette page importe le jeu de composants "*core*" avec le préfixe *f* avec ***xmlns:f="http://java.sun.com/jsf/core"***



Bien que cette page n'utilise aucun des composants de facelets, elle sera traitée et transformée par Facelets du moment qu'elle porte l'extension xhtml (cf LIST. 1).

Pour le reste, on travaille comme dans une *JSP*, mais n'oubliez pas de respecter rigoureusement la norme *xml*.

IV - Le templating

Le *templating* consiste à factoriser la structure commune d'un ensemble de pages et à l'extraire dans une nouvelle page, appelée *template* ou modèle. Les autres pages utilisent alors le *template* comme structure et y injectent leur contenu spécifique.

Le templating offre plusieurs avantages :

- Uniformiser la structure des pages
- Simplifier la mise à jour : une modification dans le *template* se propage automatiquement dans toutes les pages qui l'utilisent.
- Gain en productivité : moins de code à écrire : une page ne contient que ce qui lui est propre.

Pour illustrer le concept du *templating*, nous allons partir d'un exemple concret. Considérons un ensemble de pages du site www.developpez.com :

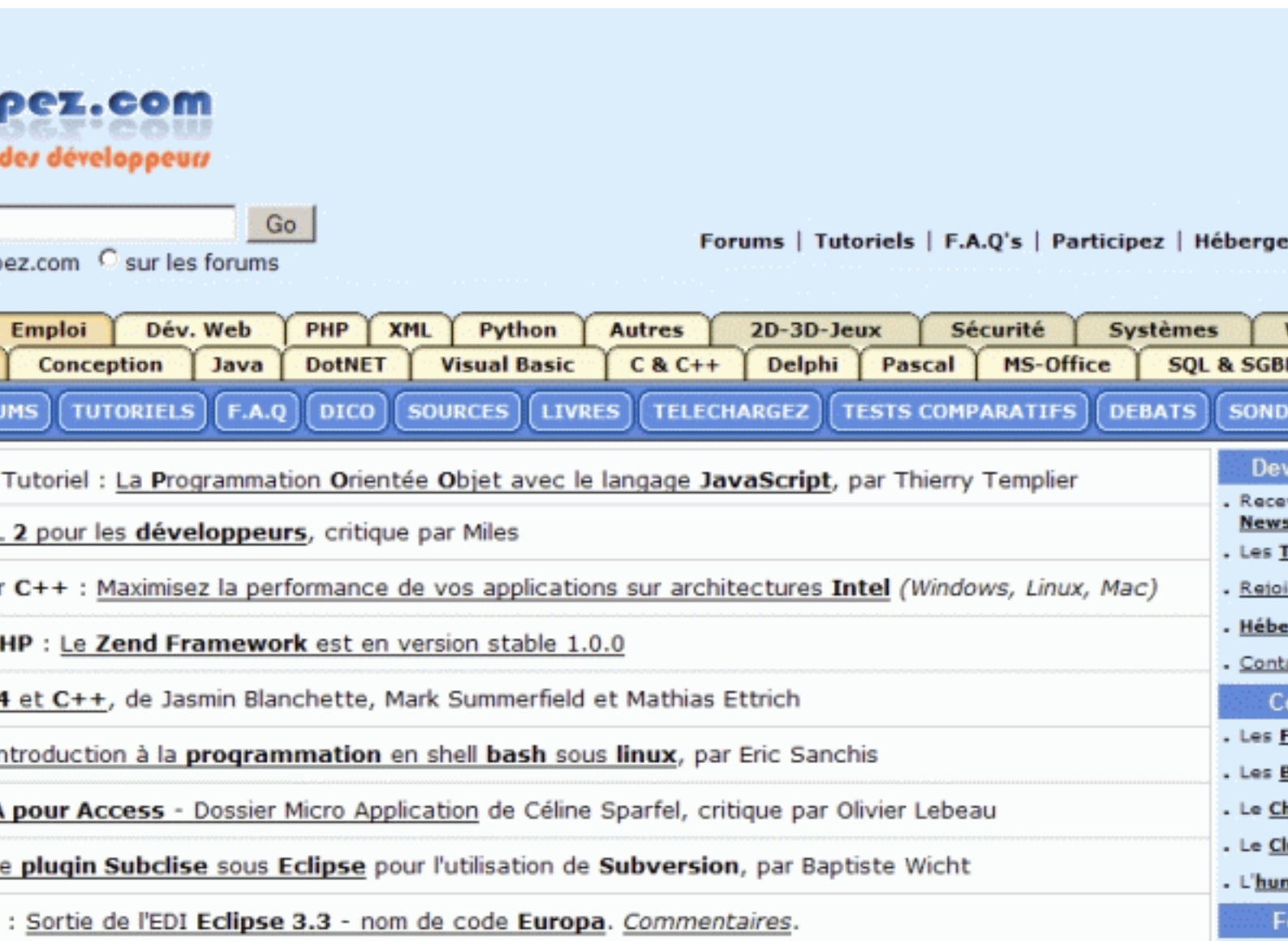


FIG. 1 - La section accueil de Developpez.com



Toutes
les
FAQ's
Java
1007 Q/R

Cette page recense toutes les **F.A.Q** (Foires Aux Questions) relatives à **Java** du site **Developpez.com**

L'équipe Java vous souhaite une bonne lecture.

F.A.Q Général Java	F.A.Q Java XML	F.A.Q Java GUI	F.A.Q Java ME
------------------------------------	--------------------------------	--------------------------------	-------------------------------

FIG. 2 - La section Java/FAQs de Developpez.com

A partir de ces deux pages, on peut extraire la structure globale d'une page dans le site de Developpez.com qui est représentée dans la figure FIG. 3.

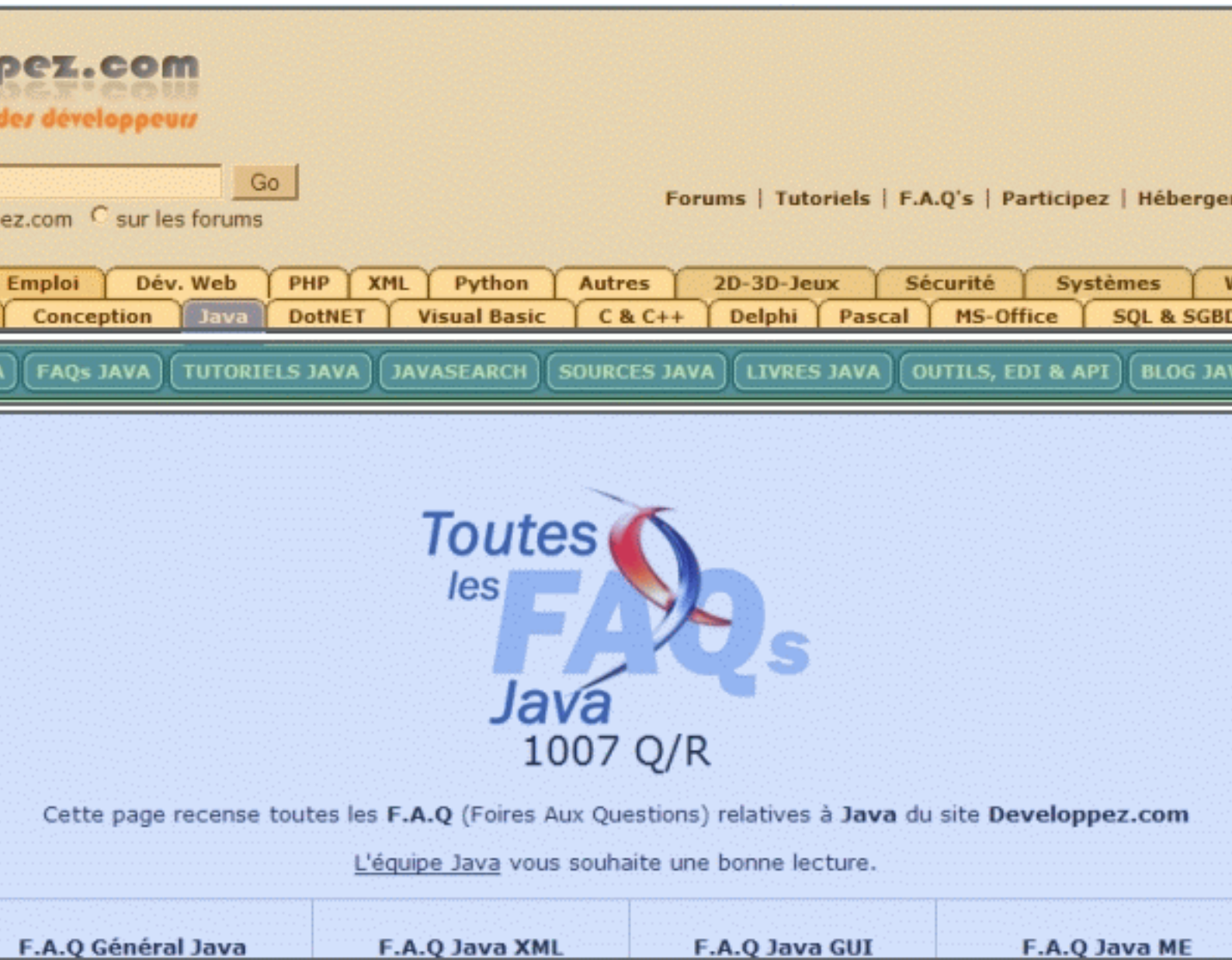


FIG. 3 - Un template possible pour Developpez.com

On distingue dans le template présenté dans la figure FIG. 3 :

- L'en-tête (situé en haut, coloré en orangé et numéroté (1) dans la figure) : contient le logo, le formulaire de recherche et le menu principal.
- Le menu secondaire (situé au milieu, coloré en vert et numéroté (2) dans la figure) : contient un menu spécifique à la section en cours.
- Le contenu (situé en bas, coloré en bleu et numéroté (3) dans la figure) : accueille le contenu de la page courante.

Seul l'en-tête est commun à toutes les pages. Le menu secondaire et le contenu sont spécifiques à la page courante.

On peut donc construire ce *template* avec *facelets* comme ceci :

LIST. 4 - le template `template.xhtml`

```
<!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:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:t="http://myfaces.apache.org/tomahawk">

<head>
  <title><ui:insert name="titre" /></title>
</head>

<body>
  <f:view>
    <p>
      <ui:insert name="entete">
        <h:outputText value="entete generique" />
      </ui:insert>
    </p>

    <div>
      <ui:insert name="menu" />
    </div>

    <p>
      <ui:insert name="contenu" />
    </p>
  </f:view>
</body>
</html>
```

Dans *facelets*, un *template* est une page *xhtml* ordinaire qui définit la structure du document avec des emplacements spécifiques où les pages qui utilisent ce *template* (pages clientes) inséreront leur contenu. Pour définir un tel emplacement, on utilise le tag **<ui:insert>** avec comme paramètre le nom logique de cet emplacement : en-tête, menu, etc. Le tag **<ui:insert>** comme tous les autres *tags* préfixés par " ui " sont fournis par *facelets*.

En général, le tag **<ui:insert>** a un corps vide puisqu'il sera remplacé par le contenu fourni par la page cliente. Si par contre ce corps n'est pas vide, alors son contenu sera affiché lorsque la page cliente ne spécifie pas son propre contenu. Par exemple, dans le *template* précédent, on a défini un message dans l'emplacement réservé à l'en-tête qui sera affiché si les pages clientes ne spécifient pas leur propre en-tête.

Une fois le *template* défini, les pages peuvent l'utiliser de cette façon :

LIST. 5 - La page cliente `java-faq.xhtml`

```
<!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:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html">

<ui:composition template="/template.xhtml">
  <ui:define name="titre">FAQ Java</ui:define>

  <ui:define name="entete">
    <h:outputText value="Entete Java" />
  </ui:define>

  <ui:define name="menu">
    <h:outputText value="FAQ" />
    <h:outputText value="Tutoriels" />
    <h:outputText value="Forums" />
  </ui:define>

  <ui:define name="contenu">
```

LIST. 5 - La page cliente java-faq.xhtml

```
<h:outputText value="Les FAQs Java" >
</ui:define>
</ui:composition>
</html>
```

On commence par spécifier que l'on utilise un *template* avec la balise **<ui:composition>** qui prend comme paramètre le chemin vers le fichier contenant le *template*. Pour définir les différents blocs qui seront injectés dans le *template*, on utilise la balise **<ui:define>**. Cette balise prend comme paramètre le nom logique du bloc correspondant dans le *template* et contient dans son corps le contenu à injecter dans le *template*.

 *Notez l'absence des balises <head> et <body> du moment qu'elles sont définies dans le template.*


Par exemple, lors de l'affichage de **faq-java.xhtml**, *facelets* effectue les opérations suivantes sur **template.xhtml** :


- Remplace **<ui:insert name= "title">** par le "FAQ Java".
- Remplace **<ui:insert name= "entete">** par le "Entete Java".
- Remplace **<ui:insert name= "menu">** par le "FAQ Tutoriels Forums".
- Remplace **<ui:insert name= "contenu">** par le "Les FAQs Java".

La page résultante de ce traitement est alors :

LIST. 6 - La page résultat

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title> FAQ Java </title>
</head>
<body>
<p>Entete Java</p>
<div>FAQ Tutoriels Forums</div>
<p>Les FAQs Java</p>
</body>
</html>
```

 *Notez que le résultat ne contient que des balises standards html compréhensibles pour le navigateur et que toutes les balises spécifiques à facelets ont été interprétées et n'apparaissent plus dans le résultat.*

 *Il est parfois intéressant de pouvoir passer des paramètres de la page cliente au template qu'elle utilise. Ceci peut se faire avec le tag **<ui:param>** qui prend en paramètre le nom du paramètre et sa valeur. Le paramètre en question est alors accessible dans le template avec la syntaxe des JSF EL (Expression Language), c'est à dire avec l'expression ***#{nomDuParametre}***.*

Exemples :

Dans la page cliente

```
<ui:composition template="/template.xhtml">
<ui:param name="active" value="3" />
:
:
```

Dans la page cliente

Dans le template

```
<h:outputText value="#{active}"/>
```

V - La création de composants

Facelets permet de développer des composants *JSF* personnels et ce sans avoir à écrire la moindre ligne de code en *Java*. Ceci se fait en réalisant des compositions de composants existants.

La création d'un composant avec facelets se déroule comme ceci :

- 1 Création de la page *xhtml* qui définit le composant.
- 2 Création d'un descripteur de bibliothèque de composants (fichier *.taglib.xml*).
- 3 Déclaration du fichier *taglib.xml* dans *web.xml*.

V-A - Créer la page *xhtml* du composant

Un composant est une composition, il est donc défini en utilisant `<ui:composition>`.

LIST. 7 - La définition du composant

```
<!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:ui="http://java.sun.com/jsf/facelets"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html">

  <ui:composition>
    <h:outputText value="#{titre}" styleClass="titre" />
    <h:inputText value="#{valeur}" styleClass="zoneTexte" />
  </ui:composition>
</html>
```

Les paramètres `#{titre}` et `#{valeur}` seront passés en paramètre lors de l'appel de ce composant. Vous êtes libre de choisir les noms des paramètres lors de la définition du composant.

Les fichiers de définition des composants vont généralement dans un sous dossier de *WEB-INF*. Dans cet exemple, on suppose que le fichier définissant ce composant (*zoneDeTexte.xhtml*) est situé dans *WEB-INF/composants*.

V-B - Création du *taglib.xml*

C'est un fichier *xml* qui décrit une bibliothèque de composants : Il définit son *namespace* ainsi que la liste des composants qu'il contient.

LIST. 8 - Le fichier *monJeu.taglib.xml*

```
<?xml version="1.0"?>
<!DOCTYPE facelet-taglib PUBLIC
  "-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN"
  "facelet-taglib_1_0.dtd">
<facelet-taglib>
  <namespace>http://www.developpez.com/jsfEtFacelets/monJeu</namespace>
  <tag>
    <tag-name>zoneDeTexte</tag-name>
    <source>composants/zoneDeTexte.xhtml</source>
  </tag>
</facelet-taglib>
```

Comme le montre ce fichier (LIST. 8), le *namespace* est indiqué dans l'élément `<namespace>`, et pour chaque composant dans le jeu de composants, on ajoute un élément `<tag>` avec le nom du composant et le fichier *xhtml* qui contient sa définition.


Le fichier *taglib.xml* doit être dans ou dans un sous dossier de *WEB-INF*. Dans ce cas, on suppose qu'il est situé dans *WEB-INF*.


V-C - Déclaration dans web.xml

Il s'agit ici d'indiquer à facelets l'emplacement du ou des fichiers **taglib.xml** et ce en ajoutant un élément **<context-param>** dans **web.xml** avec **facelets.LIBRARIES** comme nom de paramètre et le chemin vers le **taglib.xml** comme valeur. Ce chemin est relatif au dossier **web** de l'application.

LIST. 9 - La déclaration d'un taglib.xml dans web.xml

```
<context-param>
  <param-name>facelets.LIBRARIES</param-name>
  <param-value>/WEB-INF/monJeu.taglib.xml</param-value>
</context-param>
```

 Pour spécifier plusieurs fichiers **taglib.xml**, il faut le faire dans un seul **<context-param>** en séparant les noms des fichiers par un ";".

 Si le ou les fichiers **taglib.xml** en question sont dans un fichier **jar** qui se trouve dans le dossier **WEB-INF/lib** de l'application, alors il n'est plus nécessaire de les déclarer dans le **web.xml**.


V-D - Utilisation du composant


Reste enfin à utiliser le composant dans une vue. Pour ce faire, on doit importer le jeu de composants en question via son namespace en lui associant un préfixe et utiliser les composants qu'il contient.

Par exemple :

LIST. 10 - Une page utilisant le composant créé.

```
<!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:h="http://java.sun.com/jsf/html"
  xmlns:mj="http://www.developpez.com/jsfEtFacelets/monJeu">
<head>
  <title>Ma premiere page XHTML avec facelets</title>
</head>
<body>
  <f:view>
    <mj:zoneDeTexte titre="Hello World" valeur="#{unManagedBean.unChamp}" />
  </f:view>
</body>
</html>
```

 Veuillez noter comment on spécifie les paramètres d'un composant : on utilise des attributs ayant le même nom que ceux des paramètres utilisés dans la définition du composant.

 Pour pouvoir passer à un composant un bloc (un ou plusieurs composants) au lieu d'une valeur simple, on spécifie où ce bloc sera inséré dans le composant avec le tag **<ui:insert/>** sans paramètre et on déclare ce bloc comme corps du tag du composant.

Par exemple :

LIST. 11 - Composant qui accepte un bloc comme corps.

```
<ui:composition>
  <h:panelGrid columns="1" width="100%" border="1">
    <f:facet name="header">
      <h:outputText value="#{titre}"></h:outputText>
    </f:facet>
  </h:panelGrid>
</ui:composition>
```

LIST. 11 - Composant qui accepte un bloc comme corps.

```
</f:facet>
<ui:insert /> <!-- C'est ici que se fera l'insertion -->
</h:panelGrid>
</ui:composition>
```

LIST. 12 - La page qui passe un bloc à un composant.

```
<misc:group titre="Group1">
  <h:outputText value="un message" />
  <h:inputText value="une valeur" />
</misc:group/>
```

V-E - Remarques

La méthode présentée ici, bien qu'elle permet de mettre en place rapidement et facilement des composants réutilisables, souffre néanmoins de quelques limitations comme par exemple le fait qu'elle ne permet que de composer des composants déjà existants et non pas la création d'autres totalement nouveaux.

VI - Autre fonctionnalités de Facelets


VI-A - Débogage

Facelets offre un composant qui permet de voir l'arbre des composants (UI Tree) généré par JSF et ce via le composant `<ui:debug />`. Une fois ce composant ajouté à la page, il suffit d'appuyer sur **CTRL + SHIFT + D** une fois la page exécutée. Ce raccourci peut être configuré avec l'attribut `hotkey="J"` par exemple.

VI-B - Inclusion

Il est possible d'inclure d'autres pages *xhtml* dans un document *facelets* via le composant `<ui:include src="chemin" />`

VI-C - Répétition

Facelets offre un composant capable d'itérer sur une liste ou un tableau, tout comme le for de  **JSTL**, qui est le composant `<ui:repeat value="liste" var="un nom" />`

VI-D - Autres

Facelets offre quelques variantes à `<ui:composition />` qui sont :

- **component**: similaire à composition mais ajoute un *UIComponent* à l'arbre des composants *JSF*.
- **decorate**: similaire à composition mais garde aussi le contenu qui l'entoure.
- **fragment**: c'est à component ce qu'est decorate à composition.

VII - Conclusion

Pour conclure, j'espère que cet article vous a été utile et que **facelets** vous permettra d'augmenter votre productivité en utilisant les fonctionnalités qu'il offre comme le *templating* ou la création de composants.

Pour un complément d'informations, je vous invite à consulter la documentation officielle de *facelets* disponible ici :

 <https://facelets.dev.java.net/nonav/docs/dev/docbook.html>

VIII - Remerciements

Mes remerciements vont à :

- **Ricky81** pour ses encouragements
- **Romaintaz** pour sa relecture
- **Keulkeul** pour sa relecture
- **RideKick** pour sa relecture
- **Elitost** pour sa relecture