****

***Contenu***

1. Introduction
2. Java Server Faces
   1. Framework MVC
   2. Structure d'une application JSF
   3. Facelets et composants
   4. La Servlet
   5. Les Managed-Beans
3. Primefaces
   1. Installation
   2. Les apports de Primefaces
   3. Quelques Composants
      1. Spinner Avec Ajax
      2. Calendar
      3. Slider
      4. Autocomplete
   4. Les Thèmes

**1- Introduction :**

De nos jours, le développement web a évolué d’une façon très rapide et est devenu plus facile à mettre en place une application web grâce aux frameworks java existants. Cependant ces frameworks connaissent un manque de composants, pour cela des projets ont été faits afin de faciliter le développement des interfaces web JSF. Et cela en créant des bibliothèques de composants plus riches et évolués.

Primefaces est une bibliothèque Open Source qui propose plusieurs dizaines de composants ajaxifiés permettant de construire des applications web ayant un aspect et un comportement analogues à ceux des applications de bureau. Primefaces s'appuie sur JSF2 d'où la nécessité de présenter d'abord ce framework. Primefaces propose de plus des composants spécifiques pour les mobiles.

**2- Java Server Faces :**

 JSF (Java Server Faces) est un framework MVC basé sur les composants, pour le développement d'applications Web. Il est construit sur l'API Servlet et fournit des composants sous forme de bibliothèques de balises.

1. **C’est quoi Framework MVC ?**

MVC est une architecture pour le développement de logiciels, elle n'est pas récente mais elle est extrêmement populaire et proposée par la plupart des langages/frameworks actuels.

MVC signifie Modèle, Vue, Contrôleur.

Il s'agit donc de diviser le logiciel en trois parties distinctes qui vont interagir entre elles selon un schéma précis :

- Le contrôleur gère les événements et supervise les deux autres parties. Il administre les communications entre le modèle et la vue.

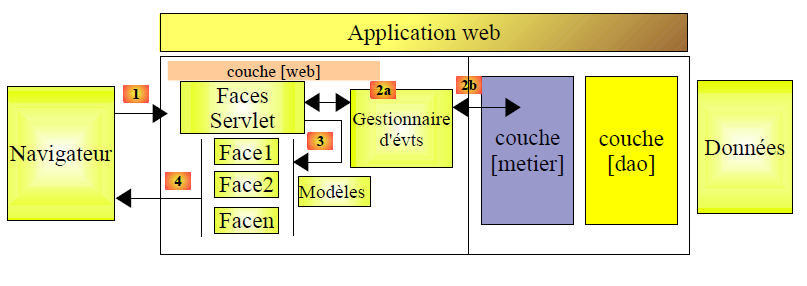
- Le modèle fournit des fonctions pour gérer les données. Ce modèle peut utiliser un SGBD (MySQL par exemple), des fichiers XML, ou autre ; cela est complètement transparent pour le contrôleur.

- La vue gère l'interface et se charge de l'affichage. Elle est concrétisée par les templates et les modèles d’affichage qui sont le plus souvent des fichiers xhtml et qui n'attendent que les données finales pour être présentés à l'utilisateur.

1. **Structure d’une application JSF :**

Comment se construit une application basée sur JSF:

* la vue est généralement assurée par des pages XHTML (on parle alors de Facelets) ;
* le modèle est assuré par des entités ou des JavaBeans ;
* le contrôleur est décomposé en deux éléments:
  + une unique servlet mère servant de point d'entrée à toute requête, la FacesServlet.
  + un JavaBean particulier, déclaré via une annotation et désigné par le terme managed-bean.
* le tout est mis en musique par des fichiers de configuration : web.xml et faces-config.xml.



1. **Facelets et composants :**

Avec JSF, de simples pages XHTML sont utilisées pour la création de vues dans une application Java EE, ces pages sont nommées des Facelets.

Une Facelet est un fichier XML pur qui ne peut contenir que des balises, et il est impossible d'y inclure des scriptlets : le Java est donc définitivement banni de la vue avec cette technologie.

Au niveau du format et du rendu, c'est le langage XHTML qui est le plus utilisé pour créer des Facelets (XHTML étant une version de la syntaxe HTML conforme au standard XML). Le framework JSF est markup-agnostique : cela signifie qu'il peut s'adapter à n'importe quel langage, à partir du moment où ce langage respecte la structure décrite par le standard XML. Ainsi, il est également possible de construire une Facelet avec du langage XML pur, par exemple pour créer un flux RSS.

Au niveau de l'aspect extérieur du fichier, une Facelet porte en général une de ces trois extensions : .jsf, .xhtml ou .faces. Dans l'absolu, c'est par défaut l'extension .xml qui pourrait être utilisée.

1. **La Servlet:**

Toutes les requêtes vers des pages « JSF » sont interceptées par une unique servlet, la FacesServlet, définie dans le fichier **web.xml** de l’application Web.

<servlet>

<servlet-name>Faces Servlet</servlet-name>

<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

1. **Les Managed-Beans**

Les managed beans font l’interface entre les pages JSF et le reste de l’application.

Un managed-bean fournit du code Java pour l’interface graphique. Il peut effectuer les traitements liés directement à l’interface utilisateur ; il fait appels à des EJB ou des classes Java ordinaires pour effectuer les autres traitements.

* le bean contient deux annotations spécifiques à JSF :
  + @ManagedBean : permet de préciser au serveur que ce bean est géré par JSF. Cela signifie simplement que JSF va utiliser ce bean en tant que modèle associé à une ou plusieurs vues
  + L’une des portées suivantes :

@RequestScoped : On crée une nouvelle instance du bean pour chaque requête.

@SessionScoped : On crée une instance du bean et elle durera le temps de la session. Le bean doit être Sérialisable. Utile par exemple pour gérer le statut « connecté/non connecté » d’un formulaire login/password.

@ApplicationScoped : Met le bean dans « l’application », l’instance sera partagée par tous les utilisateurs de toutes les sessions.

@ViewScoped : La même instance est utilisée aussi souvent que le même utilisateur reste sur la même page, même s’il fait un refresh (reload) de la page ! Le bean doit être sérialisable. Convient bien pour les pages JSF faisant des appels Ajax (une requête ajax = une requête HTTP=> une instance si on est en RequestScoped)

**3- Primefaces :**

L'utilisation des composants Primefaces permet de créer des interfaces web plus conviviales grâce au nombreux composants de cette bibliothèque et plus fluides grâce à la technologie AJAX qu'elle utilise nativement. On parle alors d'interfaces riches ou RIA (Rich Internet Application).

* + 1. **Installation :**

Pour pouvoir utiliser les PrimeFaces il faut disposer au minimum d’un Runtime JAVA 5 et d’un framework JSF 2.0.

Primefaces a un seul jar appelé primefaces-{version}.jar. Il peut être téléchargé de deux façons :

* soit manuellement du site <http://www.primefaces.org/downloads.html> puis importer le JAR au projet JSF
* soit via Maven en utilisant les paramètres suivants:

<repository>

<id>prime-repo</id>

<name>PrimeFaces Maven Repository</name>

<url>http://repository.primefaces.org</url>

<layout>default</layout>

</repository>

<dependency>

<groupId>org.primefaces</groupId>

<artifactId>primefaces</artifactId>

<version>2.2</version>

</dependency>

Les pages JSF étaient construites avec trois bibliothèques de balises :

1. <html xmlns="http://www.w3.org/1999/xhtml"

2. xmlns:h="http://java.sun.com/jsf/html"

3. xmlns:f="http://java.sun.com/jsf/core"

4. xmlns:ui="http://java.sun.com/jsf/facelets">

• ligne 2 : les balises <h:x> de l'espace de noms [http://java.sun.com/jsf/html] qui correspondent aux balises HTML,

• ligne 3 : les balises <f:y> de l'espace de noms [http://java.sun.com/jsf/core] qui correspondent aux balises JSF,

• ligne 4 : les balises <ui:z> de l'espace de noms [http://java.sun.com/jsf/facelets] qui correspondent aux balises des facelets.

Pour utiliser les composants Primefaces, nous allons ajouter une quatrième bibliothèque de balises.

1. <html xmlns="http://www.w3.org/1999/xhtml"

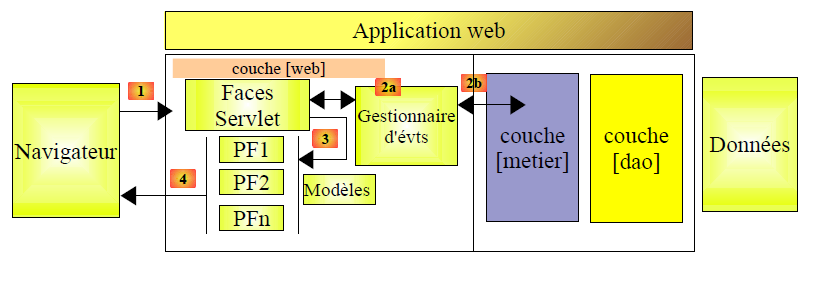
2. ...

3. xmlns:p="http://primefaces.org/ui">

• ligne 3 : les balises <p:z> de l'espace de noms [http://primefaces.org/ui] correspondent aux composants Primefaces.

C'est la seule modification qui va apparaître. Elle apparaît donc dans les vues. Les gestionnaires d'événements et les modèles restent ce qu'ils étaient avec JSF.

L’architecture JSF deviendra l’architecture PF (primefaces) suivantes :



* + 1. **Les apports de Primefaces:**

Cette bibliothèque est très riche en composants (plus de 100 composants). Ces composants sont personnalisables et faciles à utiliser et intégrer, compatibles avec d’autre bibliothèques, et à base de jQuery.

Cette bibliothèque contient aussi plus de 30 thèmes prédéfinis qui permettent la personnalisation du look de l’interface du projet web

* + 1. **Quelques composants :**

****

Spinner avec Ajax:

Le composant « Spinner » qui n'existe pas de base en JSF, permet une saisie d'un nombre entier soit directement dans la zone de saisie du spinner ou bien en utilisant les flèches pour augmenter / diminuer le nombre saisi.



3

2

1

le code suivant met en oeuvre une action AJAX en utilisant spinner :

1. <h:outputLabel for="ajaxspinner" value="Ajax Spinner: " />

2. <p:outputPanel>

3. <p:spinner id="ajaxspinner" value="#{spinnerController.number}">

4. <p:ajax update="ajaxspinnervalue" process="@this" />

5. </p:spinner>

6. <h:outputText id="ajaxspinnervalue" value="#{spinnerController.number}"/>

7. </p:outputPanel>

ligne 1 : affiche le text [1]. Est en même temps un libellé pour le composant d'*id=ajaxspinner* (attribut for). Ce composant est celui de la ligne 3 (attribut id)

lignes 3-5 : affichent le composant [2]. Ce composant est un composant de saisie / affichage associé au modèle #{spinnerController.number} (attribut value),

ligne 6 : affiche le composant [3]. Ce composant est un composant d'affichage lié au modèle #{spinnerController.number} (attribut value),

ligne 4 : la balise <p:ajax> ajoute un comportement AJAX au *spinner*. A chaque fois que celui-ci change de valeur, un POST de cette valeur (attribut process="@this") est fait au modèle #{spinnerController.number}. Ceci fait, une mise à jour de la page est faite (attribut update). Cet attribut a pour valeur l'id d'un composant de la page, ici celui de la ligne 6.

Le composant cible de l'attribut update est alors mis à jour avec le modèle. Celui-ci est de nouveau #{spinnerController.number}, donc la valeur du *spinner*. Ainsi la zone [3] suit les saisies de la zone [2].

Calendar :

La balise <p:calendar> permet de sélectionner une date à partir d'un calendrier. Cette balise admet différents attributs



Slider:

La balise <p:slider> permet de saisir un entier en faisant glisser un curseur le long d'une barre :



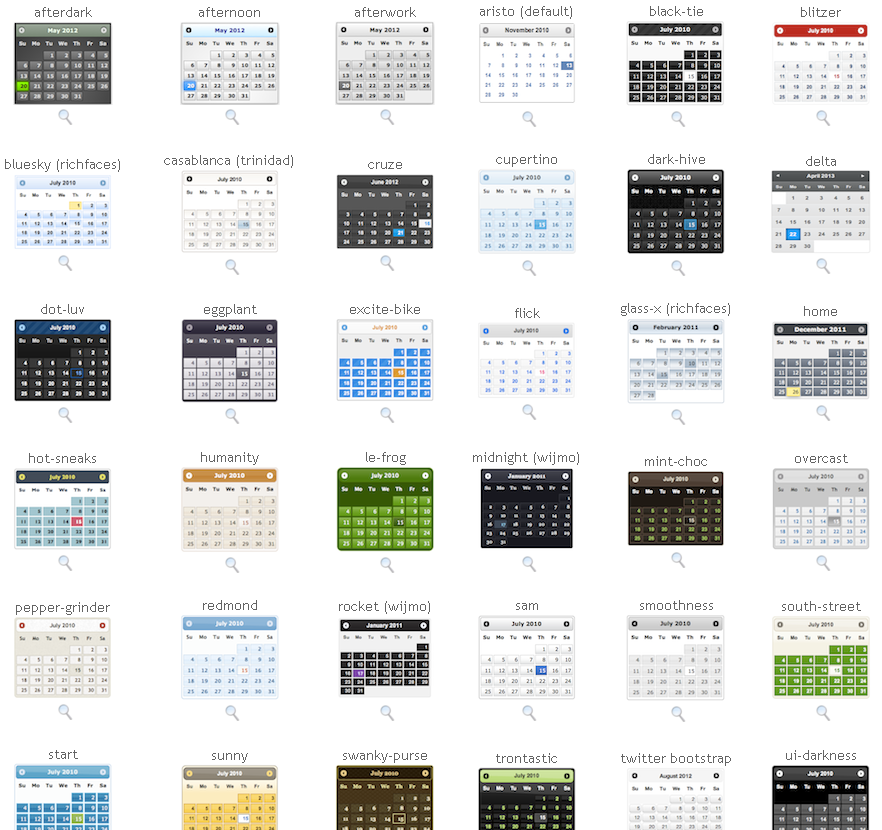
Autocomplete :

La balise <p:autoComplete> consiste à taper les premiers caractères de la saisie. Des propositions apparaissent alors dans une liste déroulante. On peut sélectionner l'une d'elles. On utilise ce composant à la place des listes déroulantes lorsque celles-ci ont un contenu trop important. Supposons que l'on veuille proposer une liste déroulante des villes de France. Cela fait plusieurs milliers de villes. Si on laisse l'utilisateur taper les trois premiers caractères de la ville, on peut alors lui proposer une liste réduite des villes commençant par ceux-ci.



* + 1. **Les Thèmes :**

Dans la bibliothèque Primefaces il y a plus de 30 thèmes prédéfinis



### L’application d’un thème sur un projet PrimeFaces est très facile, il suffit de télécharger le jar du thème, ajouter le au classpath et configurer PrimeFaces pour l’utiliser.

### 1) Vous pouvez télécharger le thème manuellement ou via maven.

### Téléchargement : Les Thèmes sont regroupés sous forme de fichiers jar disponibles au repositoire Primefaces http://repository.primefaces.org/org/primefaces/themes/

### Via Maven:

1. **<dependency>**
2. **<groupId>**org.primefaces.themes**</groupId>**
3. **<artifactId>**cupertino**</artifactId>**
4. **<version>**1.0.10**</version>**
5. **</dependency>**

Tous les thèmes sont disponibles dans un package "all-in-one".

1. **<dependency>**
2. **<groupId>**org.primefaces.themes**</groupId>**
3. **<artifactId>**all-themes**</artifactId>**
4. **<version>**1.0.10**</version>**
5. **</dependency>**

2) Ensuite et dernière étape, la configuration du paramètre primefaces.THEME avec le nom du thème

1. **<context-param>**
2. **<param-name>**primefaces.THEME**</param-name>**
3. **<param-value>**bootstrap**</param-value>**
4. **</context-param>**

Primefaces est aussi intégré avec un CSS framework ThemeRoller qui permet de créer un thème personnalisé au lieu d’utiliser ceux prédéfinis, c’est le moyen le plus facile qui fournit un outil visuel pour créer des thèmes facilement, sans nécessiter des connaissances sur CSS.