



Support de cours Java server faces

INSTITUT DES NOUVELLES TECHNOLOGIES DE L'INFORMATION

Chapitre 1 : Java Server Faces

1. Qu'est-ce qu'un Framework ?
2. Introduction
3. Historique
4. Le model d'architecture MVC 2
5. Cycle de vie
6. Configuration JSF
7. Les convertisseurs et les Validateurs
- 7.1 Les convertisseurs
- 7.2. Les validateurs
8. Interaction avec JavaBeans
9. Managed Bean
10. Faclette (depuis version JSF 1.2)
11. Navigation
12. Les composants graphiques (HTML)
13. Conclusion
14. Aller plus loin

1. Qu'est-ce qu'un Framework ?

- Un outil qui facilite le travail et qui augmente la productivité du programmeur.
- Un Framework est un ensemble de classes et de mécanismes qui fournissent un ou plusieurs services aux applications qui s'appuient dessus.
- Un Framework peut être spécialisé, sur un langage particulier, une plateforme spécifique, un domaine particulier

JSF est un Framework de présentation disponible sur la plateforme JAVA EE

2. Introduction

Java Server Faces (JSF) est une technologie dont le but est de proposer un Framework qui facilite et standardise le développement d'applications web avec Java.

JSF est un Framework orienté composants Son développement a tenu en compte des différentes expériences acquises lors de l'utilisation des technologies standards pour le développement d'applications web Servlet, JSP, JSTL et de différents frameworks (Struts, Spring MVC).

L'objectif de JSF est de :

- fournir un standard JEE spécifié dans une JSR pour le développement des IHM web riches Maximiser la productivité des applications web
- Fournir des fonctionnalités récurrentes et avancées (Validations, Conversion, Ajax ...)
- Fournir des fonctionnalités récurrentes et avancées (Validations, Conversion, Ajax ...) Masquer la complexité
- Une séparation de la couche présentation des autres couches (MVC2)

Pourquoi JSF ?

JSF permet d'accroître la productivité des développeurs dans la conception IHM (Interface Homme Machine) d'applications Web exécute cotes serveurs grâce a la notion de composants graphique.

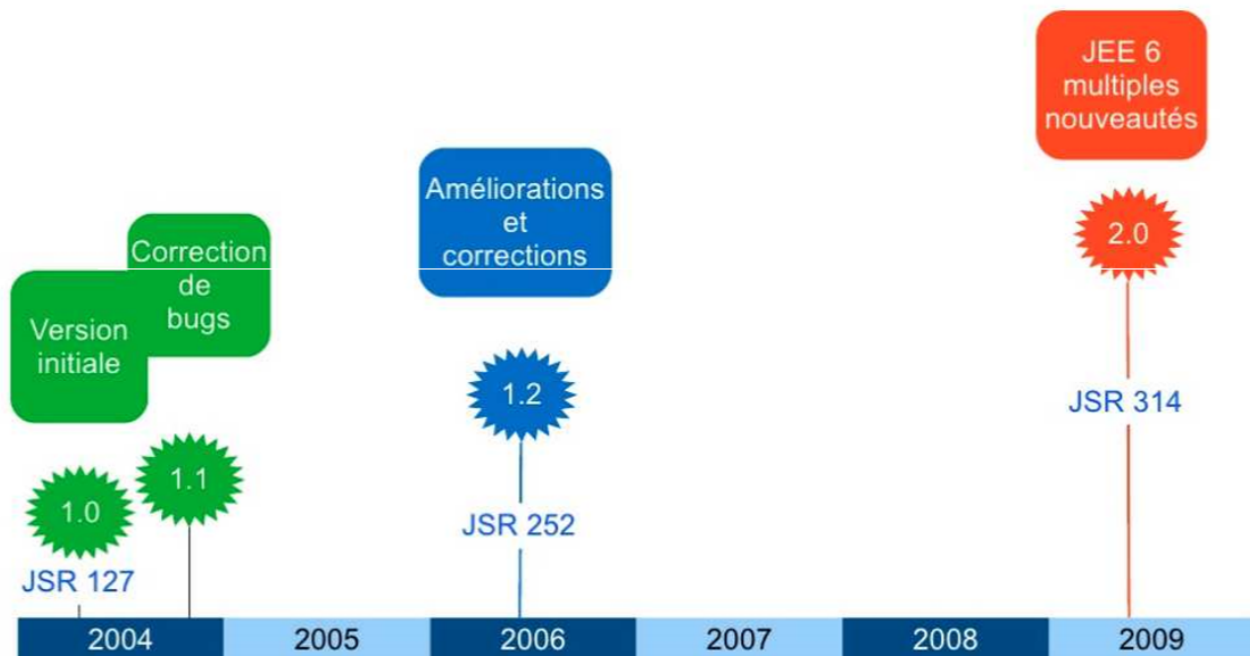
3. Historique

Le couple Servlet/jsp possède plusieurs limitations :

- la création de la balise personnalisée est assez lourde
- il n'existe pas de balises de haut niveau en JSP
- il n'est pas possible de récupérer facilement les entrées utilisateurs

Sun ont proposé JSF la première fois le 11 Mars 2004.

Plusieurs versions de JSF ont apparus, pour atteindre aujourd'hui la dernière version JSF 2.2 sorti en Avril 2013.



4. Le model d'architecture MVC 2

Le MVC 2 hérite des propriétés du model MVC sauf que le navigateur interagit avec un seul composant le contrôleur.

Il garantit l'unicité du point d'entrée.

Le fait de subdiviser l'architecture d'une application en trois couches :

Modèle :

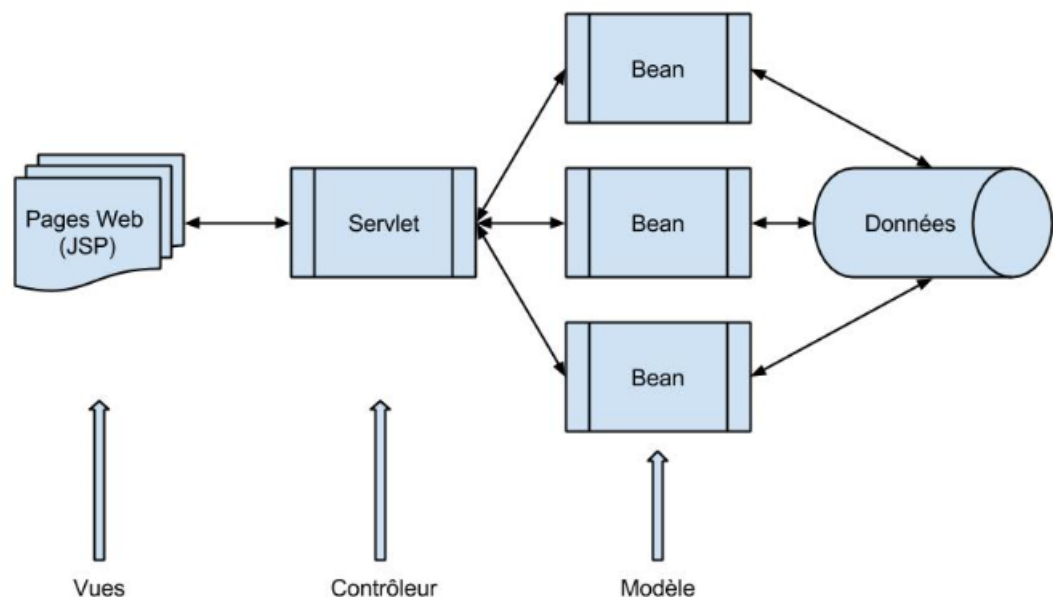
- Classes Java spécialisées qui synchronise les valeurs avec les composants UI,
- Accèdent au logique métier et gèrent la navigation entre les pages.

Vue: (pages web) :

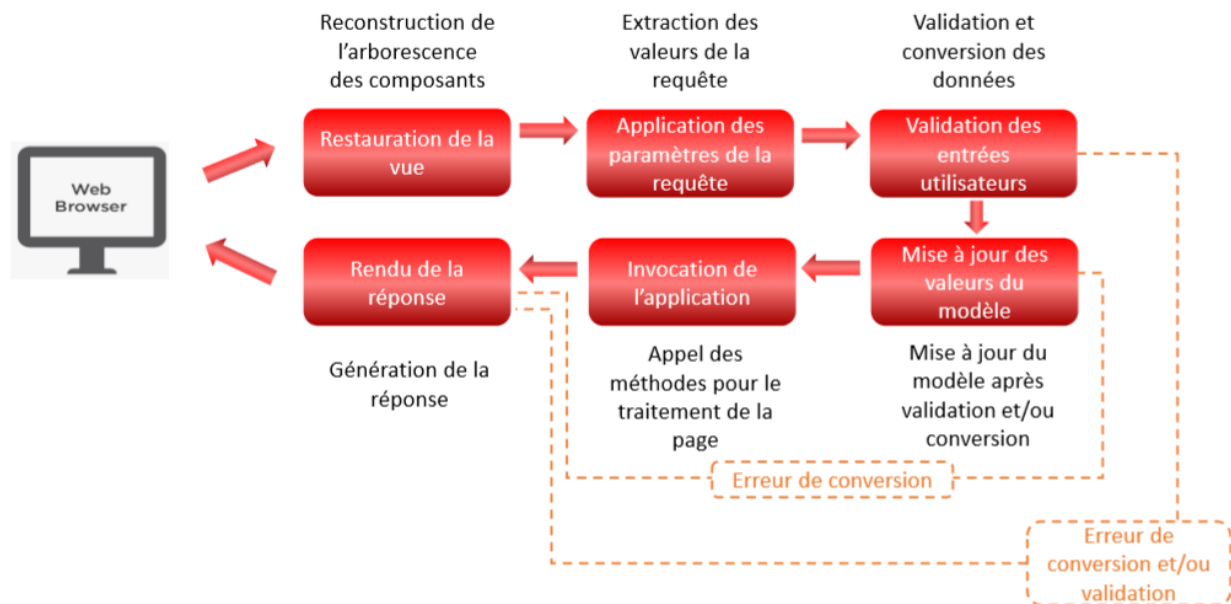
- JSF peut générer plusieurs formats de réponse (comme WML pour les dispositifs mobiles) ou (JSP, XHTML)

Contrôleur (Faces Servlet) :

- Servlet principale de l'application qui sert de contrôleur.
- Toutes les requêtes de l'utilisateur passent systématiquement par elle, qui les examine et appelle les différentes actions correspondantes.



5. Cycle de vie



ApplyRequestValue (restauration de la vue) :

Dans cette étape, les valeurs des données sont extraites de la requête HTTP pour chaque composant et sont stockées dans leur composant respectif dans le FaceContext. Composant respectif dans le FaceContext.

Durant cette phase des opérations de conversions sont réalisées pour permettre de transformer les valeurs stockées sous forme de chaîne de caractères dans la requête http en un type utilisé pour le stockage des données.

Performvalidations (Application des paramètre de la requête) :

Une fois les données extraites et converties, il est possible de procéder à leur validation en appliquant les validateurs enregistrés auprès de chaque composant.

Les éventuelles erreurs de conversions sont stockées dans le FaceContext.

Dans ce cas, l'étape suivante est directement «RenderResponse» pour permettre de réafficher la page avec les valeurs saisies et afficher les erreurs

SynchronizeModel ou update model values (mis a jours des valeurs du modèle) :

Cette étape permet de stocker dans les composants du FaceContext leur valeur locale validée respective.

Les éventuelles erreurs de conversions sont stockées dans le FaceContext.

Dans ce cas, l'étape suivante est directement «RenderResponse» pour permettre de réafficher la page avec les valeurs saisies et afficher les erreurs

InvokeApplication Logic (invocation de l'application):

Dans cette étape, le ou les événements émis dans la page sont traités.

Cette phase doit permettre de déterminer quelle sera la page résultat qui sera renvoyée dans la réponse en utilisant les règles de navigation définie réponse en utilisant les règles de navigation définie dans l'application.

L'arborescence des composants de cette page est créée

RenderResponse (rendu de la réponse):

Cette étape se charge de créer le rendu de la page de la réponse.

6. Configuration JSF

Pour intégrer JSF il faut penser à ajouter les maven dependencies dans le Pom.xml

```
<dependency>
  <groupId>com.sun.faces</groupId>
  <artifactId>jsf-api</artifactId>
  <version>2.2.1</version>
</dependency>
<dependency>
  <groupId>com.sun.faces</groupId>
  <artifactId>jsf-impl</artifactId>
  <version>2.2.1</version>
</dependency>
```

Pour configurer JSF il faut penser aux deux principaux fichiers **web.xml** et **faces.-config.xml**

Le **web.xml** est :

- Le premier fichier descripteur de toute application web J2EE.
- Le fichier web.xml contenu dans le répertoire WEB-INF.
- Décrit les 3 principaux éléments :

- La page d'accueil de l'application : welcome-file
- La servlet mère du JSF : servlet-class
- Associer les vues portant l'extension .xhtml à la FacesServlet : url-pattern.

Web.xml : Version 3.0 pour Tomcat7 ou JEE 6

```
<web-app ...>
  <context-param>
    <param-name>com.sun.faces.validateXml</param-name>
    <param-value>true</param-value>
  </context-param>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.faces</url-pattern>
  </servlet-mapping>
</web-app>
```

Context
Production par défaut

Faces Servlet

URL mapping

Plusieurs formes :
*.jsf
*.faces
*.xhtml

Le Faces-config.xml

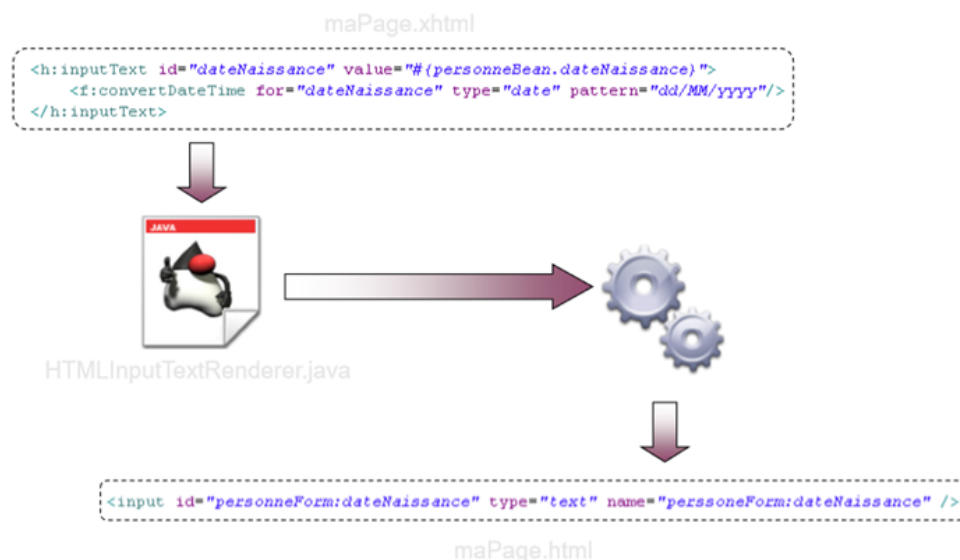
C'est le fichier gérant la logique de l'application web s'appelle par défaut faces-config.xml :

- Il est placé dans le répertoire WEB-INF au même niveau que web.xml.
- La balise de départ est <faces-config>.
- Il décrit essentiellement six principaux éléments :
 - <managed-bean> : Définit les managed Bean.
 - <navigation-rule> : Définit les règles de navigation.
 - <message-bundle> : Définit les ressources éventuelles.
 - <resource-bundle> : Définit la configuration de la localisation.
 - <validator> : Définit la configuration des validateurs.
 - <converter> : Définit la configuration des convertisseurs.

Le fichier de configuration est un fichier XML décrit par une DTD.

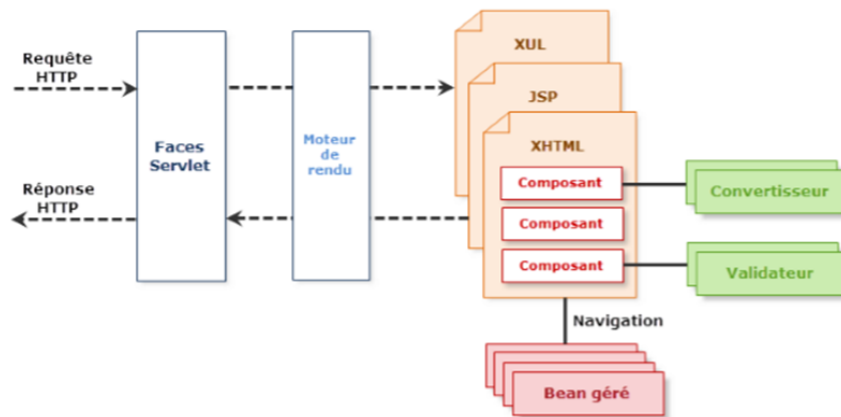


7. Les convertisseurs et les Validateurs



Le moteur de rendu décode la requête de l'utilisateur pour initialiser les valeurs du composant et encode la réponse pour créer une représentation que le client pourra comprendre et afficher.

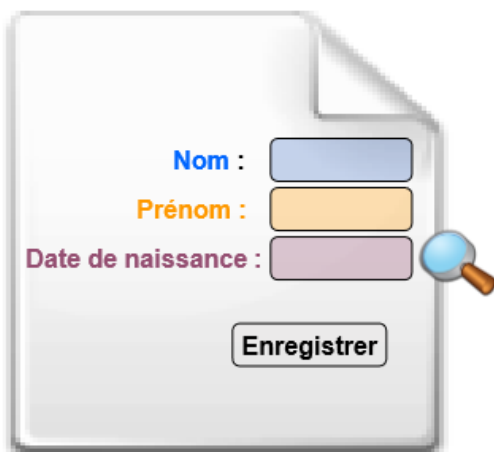
Les Renderers sont des classes java qui récupérant les attributs des composants et transcrivant le composant en fonction du format souhaité en HTML,WML...



le protocole HTTP est un protocole uniquement textuel, donc nous aurons besoin de valider les champs de saisie textuelle et les convertir.

7.1 Les convertisseurs

JSF propose des convertisseurs Ont comme rôle de gérer la conversion des données échangées entre IHM et le Managed bean :



Nom :

Prénom :

Date de naissance :

```

/**
 * Nom de l'utilisateur
 */
private String nom;

/**
 * Prénom de l'utilisateur
 */
private String prenom;

/**
 * Date de naissance de l'utilisateur
 */
private Date dateNaissance;
  
```

```

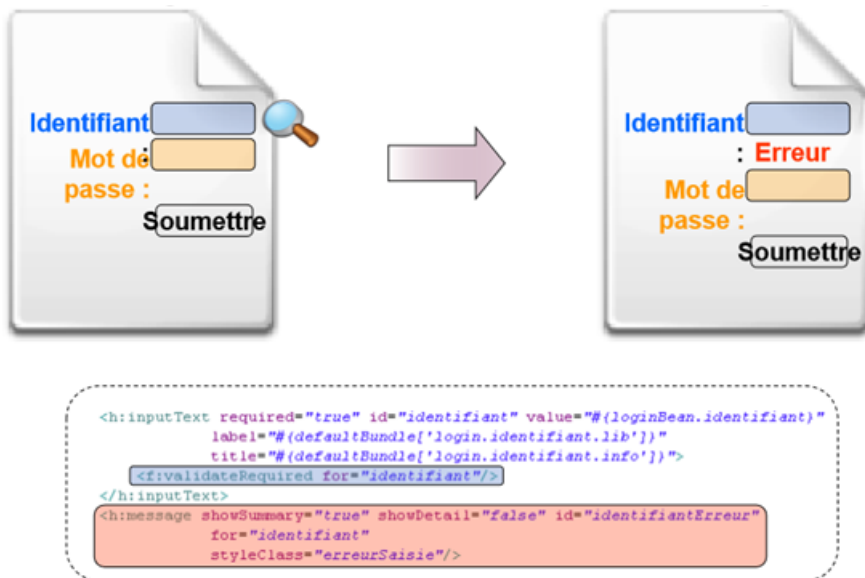
<h:inputText id="dateNaissance" value="#{personneBean.dateNaissance}">
  <f:convertDateTime for="dateNaissance" type="date" pattern="dd/MM/yyyy"/>
</h:inputText>
  
```

7.2. Les validateurs

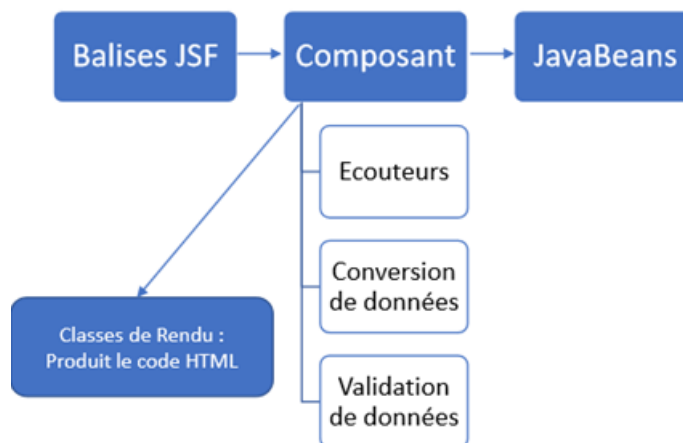
Egalement JSF propose des validateurs qui vous permettent de Vérifier la validité des donnes converties.

Il est applicable sur l'ensemble des composants de saisies pour valider la présence, la longueur de saisie

Exemple :



8. Interaction avec JavaBeans



JSF : un framework orienté 'composant'

Les JavaBeans représentent l'état des éléments à afficher et permettent d'interagir avec les couches plus basses de l'application.

Un système de rendu (Render) qui permet de redéfinir comment sera affiché (traduit en code HTML) une balise JSF

Un système d'événement et d'écouteurs pour effectuer des traitements en fonction des actions des utilisateurs

Un model de conversion qui permet de convertir des types de données

Un système de validation qui permet de vérifier les entrées utilisateurs.

9. Managed Bean

C'est une classe **javaBean** gérer par JSF.

Rappelons qu'un Bean est une classe java respectant un ensemble de directives :
Un constructeur public sans argument

Les propriétés d'un Bean sont accessible au travers de méthode **getXXX** et **setXXX** proton le nom de la propriété

Elle peut etre definit a partir du fichier de configuration faces-config.xml en utilisant :

<manged-bean-name> // définit un nom qui servira d'étiquette quand le bean sera exploité dans les pages jsp/xhtml.

<managed-bean-class> // déclare le nom de la classe dt type package.class

<managed-bean-scope> //precise le type de scope utilise pour le Bean (none , application ,session,request)

```
<managed-bean>
  <description> User Bean </description>
  <managed-bean-name>userBean</managed-bean-name>
  <managed-bean-class>com.adaming.fr.managedBean.UserBean</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Ou par **@managetBean** à pâtir de la version 2.0.

Annotation

```
@ManagedBean(name="userBean")
@SessionScoped
public class UserBean {...}
```

Elle permet de faire le lien entre les interfaces utilisateur et le code métier de l'application.

Scopes (xml)	Annotation	Description
request	@RequestScoped	On crée une nouvelle instance du bean pour chaque requête.
session	@SessionScoped	On crée une instance du bean et elle durera le temps de la session. Le bean doit être Sérialisable.
application	@ApplicationScoped	Met le bean dans « l'application », l'instance sera partagée par tous les utilisateurs de toutes les sessions.
view	@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 ! A été conçu spécialement pour les pages JSF faisant des appels Ajax.
aucun	@NoneScoped	Le bean est instancié mais pas placé dans un Scope. Utile pour des beans qui sont utilisés par d'autres beans qui sont eux dans un autre Scope.
custom	@CustomScoped(value=« #{uneMap} »)	Le bean est placé dans la HashMap et le développeur gère son cycle de vie.

Injection de dépendance dans jsf :

Il existe deux méthodes possibles soit par XML (faces-config.xml) soit par annotation.

Imaginant qu'en a deux managedBeans UserBean et AdresseBean qui se trouvent sous le package com.fr.managedBean.

Notre objectif est d'injecter l'instance de AdresseBen dans une instance de ManagedBean.

Injection par xml(faces-config.xml) :

```
<managed-bean>
  <managed-bean-name>addressBean</managed-bean-name>
  <managed-bean-class>com.adaming.fr.managedBean.AddressBean</managed-bean-class>
</managed-bean>
<managed-bean>
  <managed-bean-name>userBean</managed-bean-name>
  <managed-bean-class>com.adaming.fr.managedBean.UserBean</managed-bean-class>
  <managed-property>
    <property-name>address</property-name>
    <property-class>com.adaming.fr.managedBean.AddressBean</property-class>
    <value>#{addressBean}</value>
  </managed-property>
</managed-bean>
```

Jsف réalise l'injection de dépendance via l'utilisation de l'EL **# {}** dans faces.config.xml.

Injection par annotation :

```
@ManagedBean
@SessionScoped
public class AddressBean implements Serializable { ... }

@ManagedBean(name="userBean")
@SessionScoped
public class UserBean implements Serializable {

    @ManagedProperty(value="#{AddressBean}")
    private AddressBean address;

    ...
}
```

Unified Expression Language:

JSF propose une syntaxe basée sur des expressions qui facilitent l'utilisation des Valeurs d'un Bean.

Ces expressions doivent être **délimitées** par **#{ et }**.

Une expression est composée du nom du Bean suivi du nom de la propriété désirée séparés par un point.

```
<h:inputText id="name" value="#{userBean.name}"/>
```

Les **setters** sont automatiquement appelés par JSF.

NB : la syntaxe utilisée par JSF est proche mais différente de celle proposée par JSTL. JSF utilise les délimiteurs **#{ }** et JSTL utilise le délimiteur **\${ }**

10. Facette (depuis version JSF 1.2)

C'est une technologie de présentation pour le développement web en Java.

Facets est spécifiquement développé pour JSF, se sont des pages .xhtml et contenant des balises propres à JSF, chargées respectivement d'afficher des données, formulaire de Saisie ...

```
<!DOCTYPE html>
<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:ui="http://java.sun.com/jsf/faces">
    ...
</html>
```

Dans une Facette, une bibliothèque de balises est incluse via l'ajout d'un attribut **XMLNS** à la balise **<html>** qui ouvre le corps de la page. Il s'agit d'un namespace XML.

11. Navigation

En JSF il y'a deux types de navigation (la navigation statique et la navigation dynamique)

La navigation statique : définie au moment de l'écriture de l'application

```
<h:CommandButton value="Valider" value="pageSuivante" />
```

Lorsque l'utilisateur clique sur le lien HTML correspondant alors l'outcome pageSuivante est renvoyé au gestionnaire de navigation de JSF.

Le gestionnaire cherche une règle de navigation applicable au contexte courant :

- Par défaut si pagesSuivante.xhtml existe alors cette page est renvoyée
- si une correspondance est trouvée alors la page suivante est affichée
- si non la page courante est recharger

```
<navigation-rule>
  <from-view-id>/index.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>pageSuivante</from-outcome>
    <to-view-id>/succes.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

La navigation Dynamique : définie par l'état de l'application au moment de l'exécution (en fait, par la valeur retournée par une méthode)

```
<h:CommandButton value="Valider" value="#{nomBean.nomMethode}" />
```

```
<navigation-rule>
  <from-view-id>/index.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>succes</from-outcome>
    <to-view-id>/succes.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>echec</from-outcome>
    <to-view-id>/echec.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

```
public String nomMethode() {
  if (this.ajouter)
    return "succes";
  else return "echec";
}
```

12. Les composants graphiques (HTML)

- Les balises personnalisées décrites dans HTML s'occupent de la description de l'interface graphique d'une JSF.
- La bibliothèque HTML propose 25 composants qui sont classifiables en quatre catégories

Composants de saisies notés I (pour input)

Composants de sorties notés O (pour output)

Composants de commandes notés C (pour commandes)

Composants de regroupement notés R (pour regrouper)

- Composants de saisies
 - ☐ inputHidden
 - ☐ inputSecret
 - ☐ inputText
 - ☐ inputTextArea
 - ☐ selectBooleanCheckbox
 - ☐ selectManyCheckbox
 - ☐ selectManyListbox
 - ☐ selectManyMenu
 - ☐ selectOneListbox
 - ☐ selectOneMenu
 - ☐ selectOneRadio

Label — User Name: Duke — Text Field

Password: — Password Field

Comments: A user can enter text across multiple lines. — Text Area

News —

News
Sports
Music
Java
Web

☐ News

☐ Sports

☒ Music

☒ Java

☐ Web

Select ProductName

☒ The number of this product is 10000

☐ The number of this product is 10001

- Composants de sorties
 - ☐ Column
 - ☐ messages
 - ☐ dataTable
 - ☐ outputText
 - ☐ outputFormat
 - ☐ outputLink
 - ☐ graphicImage
- Composants de commandes
 - ☐ commandButton
 - ☐ commandLink
- Composants de regroupements
 - ☐ panelGroup
 - ☐ form
 - ☐ panelGrid, panelGroup

h:panelGrid example

A screenshot of a web form titled "Login" enclosed in a panelGrid. It contains two input fields: "Username" and "Password", and a "Submit" button at the bottom.

JSF 2 panelGrid example

A screenshot of a JSF 2 panelGrid example. It shows a form with a text input labeled "Enter a number : 0" and a "Submit" button. Below the form, three red arrows point to the columns of the panelGrid, labeled "Column 1", "Column 2", and "Column 3, reserve for error message".

JSF 2 dataTable example

Order No	Product Name	Price	Quantity	Action
A0001	Intel CPU	700.00	1	Delete
A0002	Harddisk 10TB	500.00	2	Delete
A0003	Dell Laptop	11600.00	8	Delete
A0004	Samsung LCD	5200.00	3	Delete
A0005	A4Tech Mouse	100.00	10	Delete

12. Les composants graphiques (CORE)

- Les balises personnalisées décrites dans CORE s'occupent d'ajouter des fonctionnalités aux composants JSF.

Tag	Description	Version
f:actionListener	Ajout un listener pour une action sur un composant.	
f:ajax	Ajouter à un composant pour fournir des capacités Ajax de manière standard.	JSF 2.0
f:attribute	Ajouter un attribut à un composant	
f:convertDateTime	Ajouter un convertisseur de type DateTime à un composant	
f:convertNumber	Ajouter un convertisseur de type Number à un composant	
f:param	Ajouter un paramètre à un composant	
f:facet	Permet de définir un élément particulier d'un composant	
f:selectItem	Permet de sélectionner un élément dans un composant à choix.	

13. Conclusion

L'implémentation de JSF nous offre plusieurs avantages tels que :

- Architecture MVC pour séparer l'interface utilisateur, la couche de persistance et les processus métier, utilisant la notion d'événement.
- La conversion et validation des données.
- Automatisation de l'affichage des messages d'erreur en cas de problèmes de conversion ou de validation.
- Fournit des composants standards simples pour l'interface utilisateur.
- Internationalisation.

14. Aller plus loin

Les deux jeux de composants standards de JSF s'avèrent trop limités et insuffisants pour le développement d'applications d'entreprise.

Il est possible d'utiliser des jeux de composants additionnels qui offrent de nouveaux composants plus riches.

- Primefaces : un jeu de composants open-source supportant Ajax, JSF 2.X
<https://www.primefaces.org/showcase/>
- Richfaces : un jeu de composants open-source supportant Ajax, devenu depuis la version 3 une simple copie d'anciens composants de Primefaces.
- ICEFaces : et Ajax4JSF, un jeu de composants open-source supportant Ajax.
- RCFaces : un jeu de composants très riche AJAX et open-source