

MASTER INFORMATIQUE - SEMESTRE 5

Cours : Développement Web

Séquence 3: JavaServer Faces





Apache Tomcat 8
Version 8.5.24, Nov 27 2017



Partie 3: Composants JSF



Contenu

- Architectures Multi-tiers (Rappel)
- La plate-forme Java EE (Rappel)
- JavaServer Faces
 - Environnement de développement
 - Une séparation de la couches présentation des autres couches
 - Un mapping entre l'HTML et L'objet
 - JSF et Bases de données
 - Un ensemble de composants riches et utilisables
 - Une liaison simple entre les actions côte Client et côte serveur
 - Composants additionnels JSF: Primefaces
 - Combinaison de plusieurs composants pour aboutir à un composant plus complexe

Composants HTML de JSF

Tag	Description
h:form	Formulaire HTML
h:inputText	Champ de saisie de texte sur une ligne.
h:inputTextArea	Champ de saisie de texte sur plusieurs lignes.
h:inputSecret	Champ de saisie de mot de passe.
h:inputHidden	Champ caché
h:outputLabel	Label pour un autre composant pour son accessibilité.
h:outputLink	Ancre HTML.

Composants HTML de JSF

Tag	Description
h:outputFormat	Comme outputText, mais ajoute un filtre d'affichage.
h:outputText	Affiche du texte.
h:commandButton	Bouton de contrôle: submit, reset, ou button.
h:commandLink	Lien de contrôle qui fonctionne comme un h:commandButton.
h:message	Affiche le message le plus récent pour un composant.
h:messages	Affiche tous les messages.
h:graphicImage	Affiche une image.

Composants HTML de JSF

Tag	Description
h:selectOneListbox	Une liste de sélection, une seule sélection possible.
h:selectOneMenu	Une liste de sélection de menu, une seule sélection possible.
h:selectOneRadio	Met en place des radio buttons, une seule sélection possible.
h:selectBooleanCheckbox	Checkbox.
h:selectManyCheckbox	Met en place des checkboxes.
h:selectManyListbox	Sélection multiple.

Composants HTML de JSF

Tag	Description
h:selectManyListbox	Sélection multiple.
h:selectManyMenu	Sélection multiple de menu.
h:panelGrid	Tableau HTML.
h:panelGroup	Regroupement de plusieurs composants dans un seul.
h:dataTable	Un tableau paramétrable.
h:column	Colonne dans un h:dataTable

Espaces de noms pour référencer les composants visuels JSF

- JSF 2.2 a introduit une liste d'espaces de noms pour les composants JSF. Dans la liste ci-dessous, je montre l'ancien espace de noms avec le nouveau:

Librairie	URI
Composants Composites	http://xmlns.jcp.org/jsf/composite
Faces Core	http://xmlns.jcp.org/jsf/core
HTML	http://xmlns.jcp.org/jsf/html
JSTL Core	http://xmlns.jcp.org/jsp/jstl/core
Fonctions JSTL	http://xmlns.jcp.org/jsp/jstl/functions
Facelets Templating	http://xmlns.jcp.org/jsf/facelets
Pass Through Attributes	http://xmlns.jcp.org/jsf/passthrough
Pass Through Elements	http://xmlns.jcp.org/jsf

Expression Language (EL)

A set of standard tags used in Facelets pages to refer to Java EE components

- Depuis la version 2.0 des JSP, il est possible de placer à **n'importe quel endroit d'une page JSP** des expressions qui sont évaluées et remplacées par le résultat de leur évaluation : les **Expressions Languages (EL)** .
- Elles permettent de manipuler les données au sein d'une page JSP plus simplement qu'avec les **scriptlets Java** (`<% ... %>`) .
- Utilisées conjointement avec des bibliothèques de tags (telles que les JSTL que nous verront plus tard), elles permettent de se passer totalement des scriptlets.
- Une EL permet également d'accéder simplement aux **beans** des différents **scopes** de l'application web (**page, request, session et application**).

2

Expression Language (EL)

- La syntaxe d'une EL est de la forme suivante :

`${ expression }`

- La chaîne **expression** correspond à l'expression à interpréter.
- Une expression peut être composée de plusieurs **termes**
- séparés par des **opérateurs**.

- — Ex :
$$\${ (10 + 2) * 2 } \Rightarrow 24$$
$$\${ a \ \&\& \ b } \Rightarrow a \ ET \ b$$

Ou, comme nous avons vu pour les beans :

`${bean.property}`

Expression Language (EL)

- Les EL permettent d'accéder simplement aux propriétés (attributs) des objets ou des beans:
 - `${ object.property }` <!-- objet normal -->
 - `${object["index property"] }` <!-- map -->
 - `${object['index property'] }` <!-- map -->
 - `${object[index] }` <!-- tableau/List -->
- Note : Les chaînes de caractères peuvent être définies entre simples quotes ' ' ou double quotes " "
- Il est également possible de ne pas interpréter une EL en particulier en la protégeant avec un anti-slash : `\${ ceci ne sera pas interprété comme une EL }`

Managed beans

- Une classe java qui joue le rôle de **Contrôleur** (donc, font partie de l'UI)
- Généralement utilisé pour implanter le contrôle et contenir les données des **formulaires**
- Il exécute la logique métier (ou la délègue aux EJB, par exemple), gère la navigation entre les pages ainsi que l'état des données d'une page.
- Accès possible aux EJB par injection pour effectuer les traitements qui ne sont pas liés directement à l'interface utilisateur.

Managed Beans

- Un managed bean doit obligatoirement:
 - Avoir un constructeur sans argument
 - Des attributs privés auxquels on accède par les accesseurs et modificateurs
- Annotés avec @ManagedBean ou @Named pour les définir comme des beans managés
- **Portées (scope):**
 - @SessionScoped,
 - @RequestScoped,
 - @ApplicationScoped,
 - @ViewScoped



```
import javax.faces.bean.ManagedBean;
```

Exemple

```
PersonneManagedBean.java
1 package set.mbeans;
2 import javax.faces.bean.ManagedBean;
3
4 @ManagedBean
5 public class PersonneManagedBean {
6     private String nom;
7     private String prenom;
8
9     public PersonneManagedBean() {
10         // TODO Auto-generated constructor stub
11     }
12
13     public String getNom() {
14         return nom;
15     }
16
17     public void setNom(String nom) {
18         this.nom = nom;
19     }
20
21     public String getPrenom() {
22         return prenom;
23     }
24
25     public void setPrenom(String prenom) {
26         this.prenom = prenom;
27     }
28 }
```

Salutation JSF

http://localhost:8081/premierProjet/faces/index.xhtml

Salut, Moussa Diop

Exemple: création d'un managedBean nommé *PersonneManagedBean*

```
PersonneManagedBean.java
1 package set.mbeans;
2 import javax.faces.bean.ManagedBean;
3
4 @ManagedBean
5 public class PersonneManagedBean {
6     private String nom;
7     private String prenom;
8
9     public PersonneManagedBean() {
10         // TODO Auto-generated constructor stub
11     }
12
13     public String getNom() {
14         return nom;
15     }
16
17     public void setNom(String nom) {
18         this.nom = nom;
19     }
20
21     public String getPrenom() {
22         return prenom;
23     }
24
25     public void setPrenom(String prenom) {
26         this.prenom = prenom;
27     }
28 }
```

Exemple: *index.xhtml*

Attention:

- Nom de la classe **PersonneManagedBean**
- Nom du bean **personneManagedBean**

```
index.xhtml
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml"
4   xmlns:h="http://xmlns.jcp.org/jsf/html"
5   xmlns:a="http://xmlns.jcp.org/jsf/passthrough">
6 <h:head>
7   <title>Ma première application JSF</title>
8 </h:head>
9
10 <h:body>
11   <h:form>
12     <h:inputText id="nom" value="#{personneManagedBean.nom}"
13       a:placeholder="Votre nom..."/>
14     <h:inputText id="prenom" value="#{personneManagedBean.prenom}"
15       a:placeholder="Votre Prénom..."/>
16     <h:commandButton value="Valider" action="LaReponse"/>
17   </h:form>
18 </h:body>
19
20 </html>
```

JSF Expression Language est utilisé pour:

- Accéder aux propriétés d'une ManagedBean
- Aux fonctions logiques disponibles

Syntaxe : `#{<NomDuBean>.<attribut>}` `#{personneManagedBean.nom}`

Pour accéder à une propriété d'un Bean

```
<h:inputText id="nom" value="#{personneManagedBean.nom}"/>
```

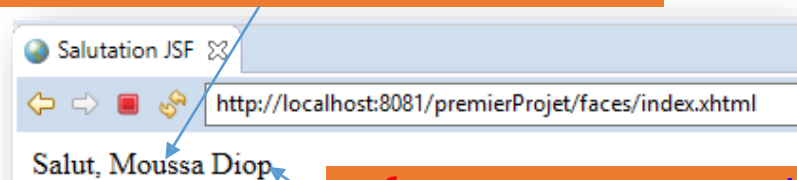
Quand le formulaire sera soumis JSF appellera la méthode
`personneManagedBean.setNom(value)`

Exemple: *laReponse.xhtml*

```
laReponse.xhtml
1 <html xmlns="http://www.w3.org/1999/xhtml"
2     xmlns:h="http://xmlns.jcp.org/jsf/html"
3     xmlns:a="http://xmlns.jcp.org/jsf/passthrough">
4 <h:head>
5     <title>Salutation JSF</title>
6 </h:head>
7
8 <h:body>
9     Salut, #{personneManagedBean.prenom} #{personneManagedBean.nom}
10 </h:body>
11
12 </html>
```

- Accéder aux données du Managed Bean
`#{personneManagedBean.nom}` i.e `personneManagedBean.getNom()`

`#{personneManagedBean.prenom}`



`#{personneManagedBean.nom}`

Les listes déroulantes JSF `<h:selectOneMenu>`

- Pour créer une liste déroulante sous JSF il faut utiliser le tag `<h:selectOneMenu>`
- Utiliser la librairie core `xmlns:f = http://java.sun.com/jsf/core` pour les items

```
<h:selectOneMenu value = "#{managedBean.attribut}">
    <f:selectItem itemValue = "1" itemLabel = "Item 1" />
    <f:selectItem itemValue = "2" itemLabel = "Item 2" />
</h:selectOneMenu>
```

Les listes déroulantes JSF <h:selectOneMenu>

Ma première application JSF

http://localhost:8081/premierProjet/faces/index.xhtml

Votre Nom...

Votre Prénom...

Dakar ▼

Valider

Ma première application JSF

http://localhost:8081/premierProjet/faces/index.xhtml

Votre Nom...

Votre Prénom...

Dakar
Thiès
Louga
Saint-Louis

```
PersonneManagedBean.java
1 package set.mbeans;
2 import javax.faces.bean.ManagedBean;
3
4 @ManagedBean
5 public class PersonneManagedBean {
6     private String nom;
7     private String prenom;
8     private String villeOrigine;
9
10    public String getVilleOrigine() {
11        return villeOrigine;
12    }
13
14    public void setVilleOrigine(String villeOrigine) {
15        this.villeOrigine = villeOrigine;
16    }
17
18    public PersonneManagedBean() {}
19
20
21    public String getNom() {}
22
23
24    public void setNom(String nom) {}
25
26
27    public String getPrenom() {}
28
29
30    public void setPrenom(String prenom) {}
31
32
33
34
35
36
37 }
```

Les listes déroulantes JSF <h:selectOneMenu>

Ma première application JSF

http://localhost:8081/premierProjet/faces/index.xhtml

Votre Nom...

Votre Prénom...

Dakar ▼

Valider

Ma première application JSF

http://localhost:8081/premierProjet/faces/index.xhtml

Votre Nom...

Votre Prénom...

Dakar
Thiès
Louga
Saint-Louis

```
index.xhtml
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:h="http://xmlns.jcp.org/jsf/html"
5       xmlns:a="http://xmlns.jcp.org/jsf/passthrough"
6       xmlns:f="http://java.sun.com/jsf/core">
7 <h:head>
8   <title>Ma première application JSF</title>
9 </h:head>
10
11 <h:body>
12   <h:form>
13     <h:inputText id="nom" value="#{personneManagedBean.nom}"
14       a:placeholder="Votre Nom..."/>
15     <br/><br/>
16     <h:inputText id="prenom" value="#{personneManagedBean.prenom}"
17       a:placeholder="Votre Prénom..."/>
18     <br/><br/>
19     <h:selectOneMenu value = "#{personneManagedBean.villeOrigine}">
20       <f:selectItem itemValue = "Dakar" itemLabel = "Dakar" />
21       <f:selectItem itemValue = "Thiès" itemLabel = "Thiès" />
22       <f:selectItem itemValue = "Louga" itemLabel = "Louga" />
23       <f:selectItem itemValue = "Saint-Louis" itemLabel = "Saint-Louis" />
24     </h:selectOneMenu>
25     <br/><br/>
26     <h:commandButton value="Valider" action="laReponse"/>
27   </h:form>
28 </h:body>
```

Les listes déroulantes JSF <h:selectOneMenu>

Ma première application JSF

http://localhost:8081/premierProjet/faces/index.xhtml

Diop

Moussa

Louga ▼

Valider

laReponse.xhtml

```
1 <html xmlns="http://www.w3.org/1999/xhtml"
2     xmlns:h="http://xmlns.jcp.org/jsf/html"
3     xmlns:a="http://xmlns.jcp.org/jsf/passthrough">
4 <h:head>
5   <title>Salutation JSF</title>
6 </h:head>
7
8 <h:body>
9   Salut, #{personneManagedBean.prenom} #{personneManagedBean.nom}
10  originaire de #{personneManagedBean.villeOrigine}
11 </h:body>
12
13 </html>
```

Salutation JSF

http://localhost:8081/premierProjet/faces/index.xhtml;jsessionid=3296E7779944EC141FA766A08B5BB660

Salut, Moussa Diop originaire de Louga

Les listes déroulantes JSF `<h:selectOneMenu>`: utiliser des données provenant du Managed bean

Mettre à jour le manager bean en créant une liste et le remplir dans le constructeur

```
private List<String> listeVilles;

public List<String> getListeVilles() {
    return listeVilles;
}

public void setListeVilles(List<String> listeVilles) {
    this.listeVilles = listeVilles;
}

public PersonneManagedBean() {
    listeVilles=new ArrayList<String>();
    listeVilles.add("Dakar");
    listeVilles.add("Thies");
    listeVilles.add("Louga");
    listeVilles.add("Saint-Louis");
    listeVilles.add("Podor");
}
```

Mettre à jour la liste au niveau de la vue

```
<h:selectOneMenu value = "#{personneManagedBean.villeOrigine}">
    <f:selectItems value = "#{personneManagedBean.listeVilles}" />
</h:selectOneMenu>
```

Les listes

- Éléments de sélections

Tag	Description
h:selectBooleanCheckbox	Checkbox.
h:selectManyCheckbox	Met en place des checkboxes.
h:selectManyListbox	Sélection multiple.
h:selectManyMenu	Sélection multiple de menu.
h:selectOneListbox	Une liste de selection, une seule sélection possible.
h:selectOneMenu	Une liste de sélection de menu, une seule sélection possible.
h:selectOneRadio	Met en place des radio buttons, une seule sélection possible.

Composants JSF: `<h:outputText>` et `<h:graphicImage>`

```
<h:outputText  
value="#{form.testString}" />
```

12345678901234567890

```
<h:outputText value="Number  
#{form.number}" />
```

Number 1000

```
<h:outputText value="<input type='text'  
value='hello' />" />
```

hello

```
<h:outputText escape="true"  
value="<input type='text'  
value='hello' />" />
```

```
<input type="text" value="hello">
```

```
<h:graphicImage value="/logo.jpg" />
```



<h:outputFormat> <f:param> Texte paramétré

```
<h:outputFormat value="{0} est agé de {1} ans">  
  <f:param value="Moussa"/>  
  <f:param value="25"/>  
</h:outputFormat>
```

<h:panelGroup>

- Regroupements

```
<h:panelGroup>
    <h:outputText value="une ligne" />
    <h:outputText value=" " />
    <h:outputText value="regroupée avec panelGroup"/>
</h:panelGroup>
```

<h:inputText> <h:inputSecret>

```
<h:inputText value="#{form.testString}" readonly="true"/>
```

```
<h:inputSecret value="#{form.passwd}" redisplay="true"/>
```

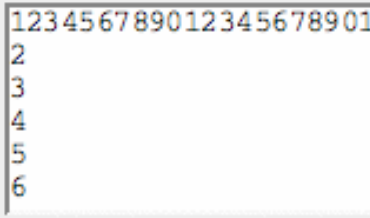
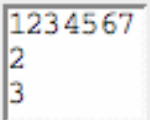

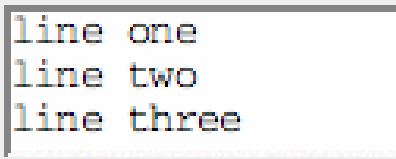
```
<h:inputSecret value="#{form.passwd}" redisplay="false"/>
```

```
<h:inputText value="inputText"  
             style="color: Yellow; background: Teal;"/>
```

```
<h:inputText value="1234567" size="5"/>
```

```
<h:inputText value="1234567890" maxlength="6" size="10"/>
```

<h:inputTextarea>

<code><h:inputTextarea rows="5"/></code>	
<code><h:inputTextarea cols="5"/></code>	
<code><h:inputTextarea value="123456789012345" rows="3" cols="10"/></code>	
<code><h:inputTextarea value="#{form.dataInRows}" rows="2" cols="15"/></code>	

<h:commandButton> <h:commandLink>

Navigation JSF

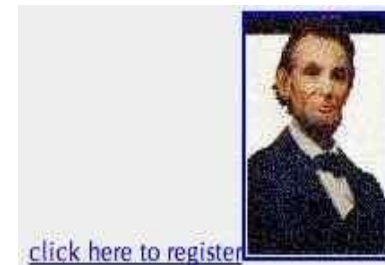
```
<h:commandButton value="submit"  
type="submit"/>
```

submit

```
<h:commandLink>  
  <h:outputText value="register"/>  
</h:commandLink>
```

register

```
<h:commandLink>  
  <h:outputText value="{msgs.link}"/>  
  <h:graphicImage value="/regis.jpg"/>  
</h:commandLink>
```



<h:outputLink>

Liens html simples

```
<h:outputLink value="#introduction">
  <h:outputText value="Introduction"
style="font-style: italic"/>
</h:outputLink>
```

[Introduction](#)

```
<h:outputLink value="#toc"
  title="Go to the table of contents">
  <f:verbatim>
    <h2>Table of Contents</h2>
  </f:verbatim>
</h:outputLink>
```

Table of Contents

Validation de la saisie dans les formulaires JSF

- Les données du formulaire sont injectées dans le bean
- validation automatique ; affichage de messages d'erreurs si la validation n'est pas correcte
- le formulaire est réaffiché s'il n'est pas valide.

Validation de la saisie dans les formulaires

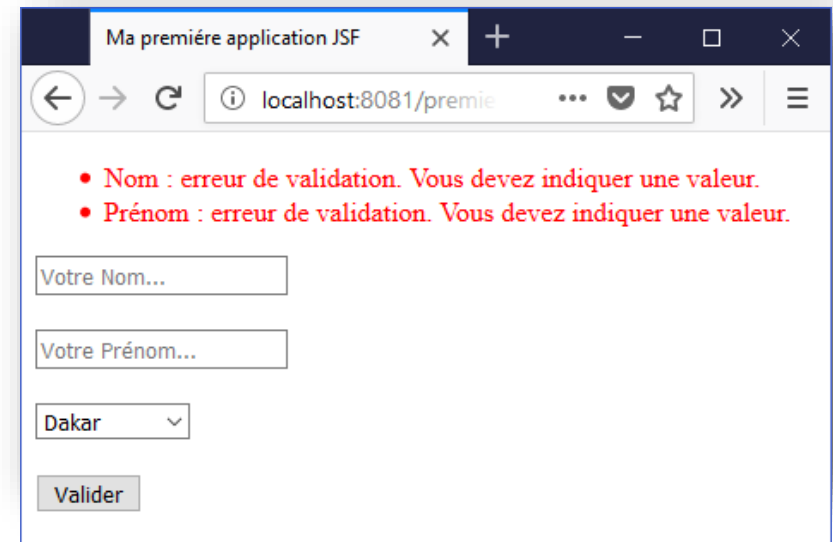
Classe validateur	Tags	Fonctions
BeanValidator	validateBean	Enregistre un validateur de bean pour le composant.
DoubleRangeValidator	validateDoubleRange	Vérifie si la valeur locale d'un composant est comprise dans une certaine plage. La valeur doit être à virgule flottante ou convertible en virgule flottante.
LengthValidator	validateLength	Vérifie si la longueur de la valeur locale d'un composant se situe dans une certaine plage. La valeur doit être un java.lang.String.
LongRangeValidator	validateLongRange	Vérifie si la valeur locale d'un composant est comprise dans une certaine plage. La valeur doit être n'importe quel type numérique ou String pouvant être converti en long.
RegexValidator	validateRegEx	Vérifie si la valeur locale d'un composant correspond à une expression régulière du package java.util.regex.
RequiredValidator	validateRequired	Garantit que la valeur locale n'est pas vide sur un composant javax.faces.component.EditableValueHolder

Validation de la saisie dans les formulaires JSF: champ obligatoire avec `required="true ou false"` et `<h:messages/>`

- Les données du formulaire sont injectées dans le bean
- validation automatique ; affichage de messages d'erreurs si la validation n'est pas correcte
- le formulaire est réaffiché s'il n'est pas valide.

```
6 <h:head>
7   <title>Ma première application JSF</title>
8 </h:head>
9   .error{color:red}
10  </style>
11 </h:head>
```

```
<h:messages styleClass="error"/>
<h:inputText id="nom" value="#{personneManagedBean.nom}"
  a:placeholder="Votre Nom..."
  label="Nom"
  required="true"/>
```



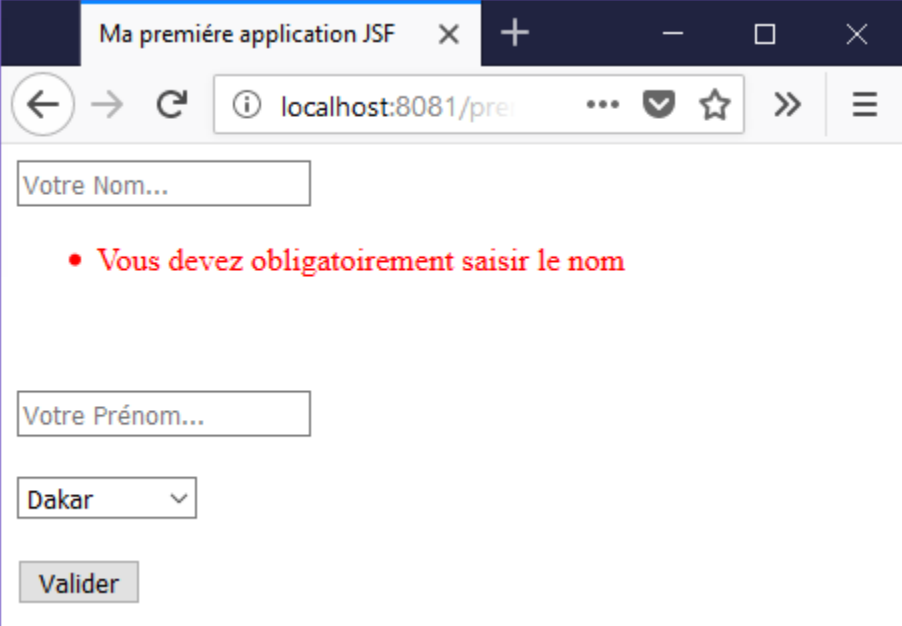
The screenshot shows a web browser window titled "Ma première application JSF" at the URL "localhost:8081/premie". The page displays two red error messages:

- Nom : erreur de validation. Vous devez indiquer une valeur.
- Prénom : erreur de validation. Vous devez indiquer une valeur.

Below the messages, there are two text input fields labeled "Votre Nom..." and "Votre Prénom...", a dropdown menu with "Dakar" selected, and a "Valider" button.

Validation de la saisie dans les formulaires JSF: champ obligatoire avec `required="true ou false"` et `<h:messages/>` personnaliser les messages d'erreurs

```
<h:inputText id="nom" value="#{personneManagedBean.nom}"  
  a:placeholder="Votre Nom..."  
  label="Nom"  
  required="true"  
  requiredMessage="Vous devez obligatoirement saisir le nom"/>  
<h:messages for="nom" styleClass="error"/>
```



The screenshot shows a web browser window titled "Ma première application JSF". The address bar indicates the URL is "localhost:8081/pre". The page contains a form with three input fields: "Votre Nom...", "Votre Prénom...", and a dropdown menu currently showing "Dakar". Below the "Votre Nom..." field, there is a red error message: "• Vous devez obligatoirement saisir le nom". At the bottom of the form is a "Valider" button.

Conversion JSF des données saisies

Class	Converter ID
BigDecimalConverter	javax.faces.BigDecimal
BigIntegerConverter	javax.faces.BigInteger
BooleanConverter	javax.faces.Boolean
ByteConverter	javax.faces.Byte
CharacterConverter	javax.faces.Character
DateTimeConverter	javax.faces.Datetime
DoubleConverter	javax.faces.Double
EnumConverter	javax.faces.Enum
FloatConverter	javax.faces.Float
IntegerConverter	javax.faces.Integer
LongConverter	javax.faces.Long
NumberConverter	javax.faces.Number
ShortConverter	javax.faces.Short

Conversion et validation des données saisies:

<f:convertNumber>

- Numérique `<f:convertNumber type="number"/>`

En interne `int getPoids() : 12`

Affichage 12

- Valeur monétaire

`<h:outputText value="#{convert.prix}">`

`<f:convertNumber type="currency"
currencyCode="XOF"/>`

`</h:outputText>`

En interne `float getPrix() : 1.23f`

Affichage 1,23 **XOF**

<f:convertNumber>

- *Pourcentage* <f:convertNumber type="percent"/>
En interne float getRatio() : 0.5f
Affichage 0,5 %
- *Entier* nombre maximum de chiffres
<f:convertNumber integerOnly="true"
maxIntegerDigits="2"/>
En interne float getPrix() : 1234.56f
Affichage 34.56
- <f:convertNumber pattern="#.#"/>

<f:convertNumber>

The locale attribute sets the language, country, and variant for formatting locale-sensitive data such as numbers and dates. If not specified, the Locale returned by FacesContext.getViewRoot().getLocale() will be used. Valid expressions must evaluate to a java.util.Locale.

- ar
- ar_AE
- ar_BH
- ar_DZ
- ar_EG
- ar_IQ
- ar_JO
- ar_KW
- ar_LB
- ar_LY
- ar_MA
- ar_OM
- ar_QA
- ar_SA
- ar_SD
- ar_SY
- ar_TN
- ar_YE
- be
- be_BY
- bg
- bg_BG
- ca
- ca_ES
- cs
- cs_CZ
- da
- da_DK
- de
- de_AT
- de_CH
- de_DE
- de_LU
- el
- el_GR
- en
- en_AU
- en_CA
- en_GB
- en_IE
- en_IN
- en_NZ
- en_US
- en_ZA
- es
- es_AR
- es_BO
- es_CL
- es_CO
- es_CR
- es_DO
- es_EC
- es_ES
- es_GT
- es_HN
- es_MX
- es_NI
- es_PA
- es_PE
- es_PR
- es_PY
- es_SV
- es_UY
- es_VE
- et
- et_EE
- fi
- fi_FI
- fr
- fr_BE
- fr_CA
- fr_CH
- fr_FR
- fr_LU
- hi_IN
- hr
- hr_HR
- hu
- hu_HU
- is
- is_IS
- it
- it_CH
- it_IT
- iw
- iw_IL
- ja
- ja_JP
- ko
- ko_KR
- lt
- lt_LT
- lv
- lv_LV
- mk
- mk_MK
- nl
- nl_BE
- nl_NL
- no
- no_NO
- no_NO_NY
- pl
- pl_PL
- pt
- pt_BR
- pt_PT
- ro
- ro_RO
- ru
- ru_RU
- sk
- sk_SK
- sl
- sl_SI
- sq
- sq_AL
- sv
- sv_SE
- th
- th_TH
- th_TH_TH
- tr
- tr_TR
- uk
- uk_UA
- vi
- vi_VN
- zh
- zh_CN
- zh_HK
- zh_TW

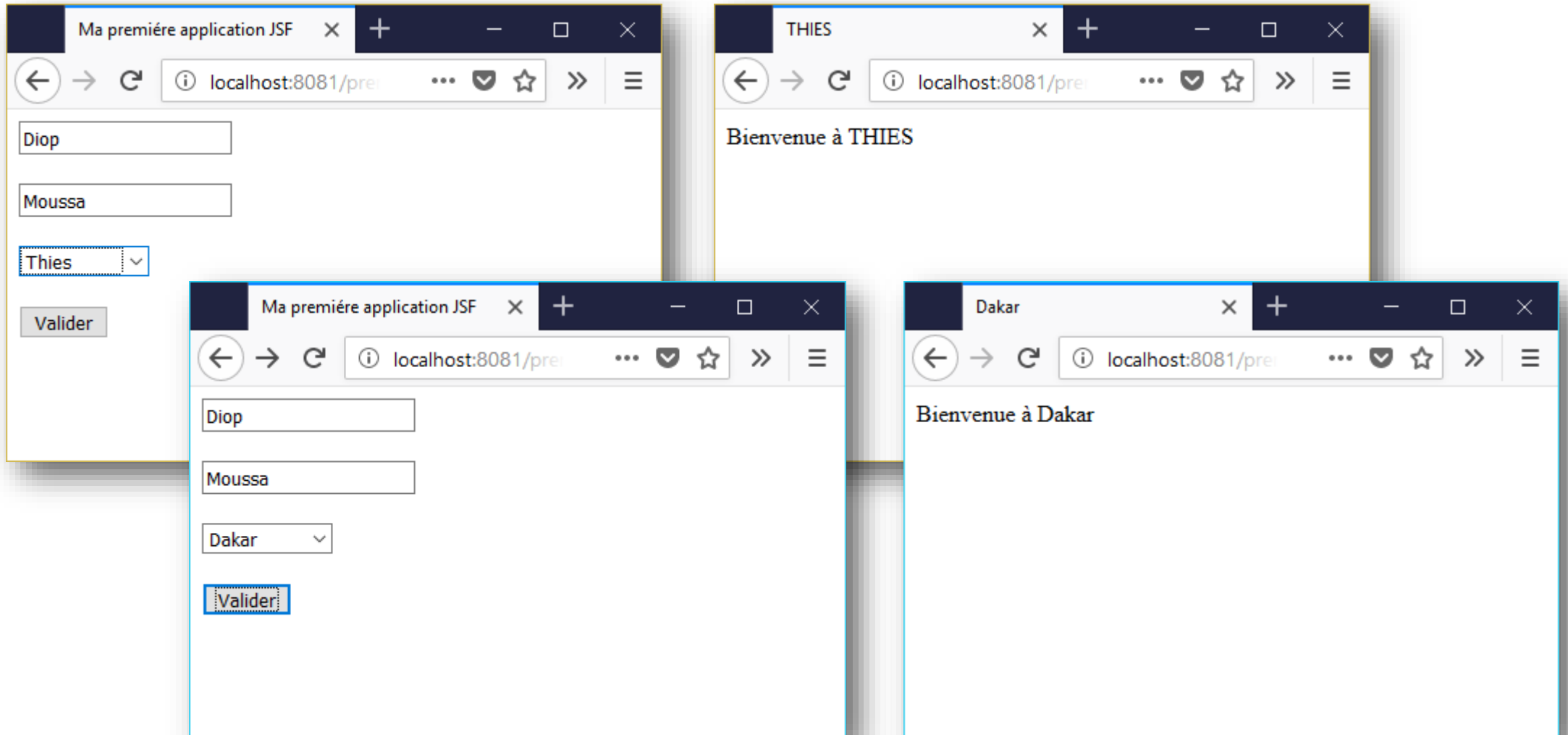
Tag Attributes

S.N.	Attribute & Description
1	type number (default), currency , or percent
2	pattern Formatting pattern, as defined in java.text.DecimalFormat
3	maxFractionDigits Maximum number of digits in the fractional part
4	minFractionDigits Minimum number of digits in the fractional part
5	maxIntegerDigits Maximum number of digits in the integer part
6	minIntegerDigits Minimum number of digits in the integer part
7	integerOnly True if only the integer part is parsed (default: false)
8	groupingUsed True if grouping separators are used (default: true)
9	locale Locale whose preferences are to be used for parsing and formatting
10	currencyCode ISO 4217 currency code to use when converting currency values
11	currencySymbol Currency symbol to use when converting currency values

Navigation en JSF: deux types que sont implicite et explicite

- **Implicite:** Les actions peuvent renvoyer une String, qui dit quelle est la prochaine vue. On peut aussi, dans un lien de la JSF, donner le nom d'une vue au lieu d'une action. **Voir les exemples passées !!!**
- **Explicite:** La chaîne renvoyée par l'action ou le lien peut être le nom d'une vue, ou un résultat abstrait (comme « success » ou « error ») qui sera utilisé par le fichier faces-config.xml pour décider de la prochaine vue.

Navigation en JSF: mode explicite



Navigation en JSF: mode explicite

- Modifier le code du bouton de commande du fichier index.xhtml

```
<h:commandButton value="Valider"  
action="#{personneManagedBean.navigationExplicite()}" />
```

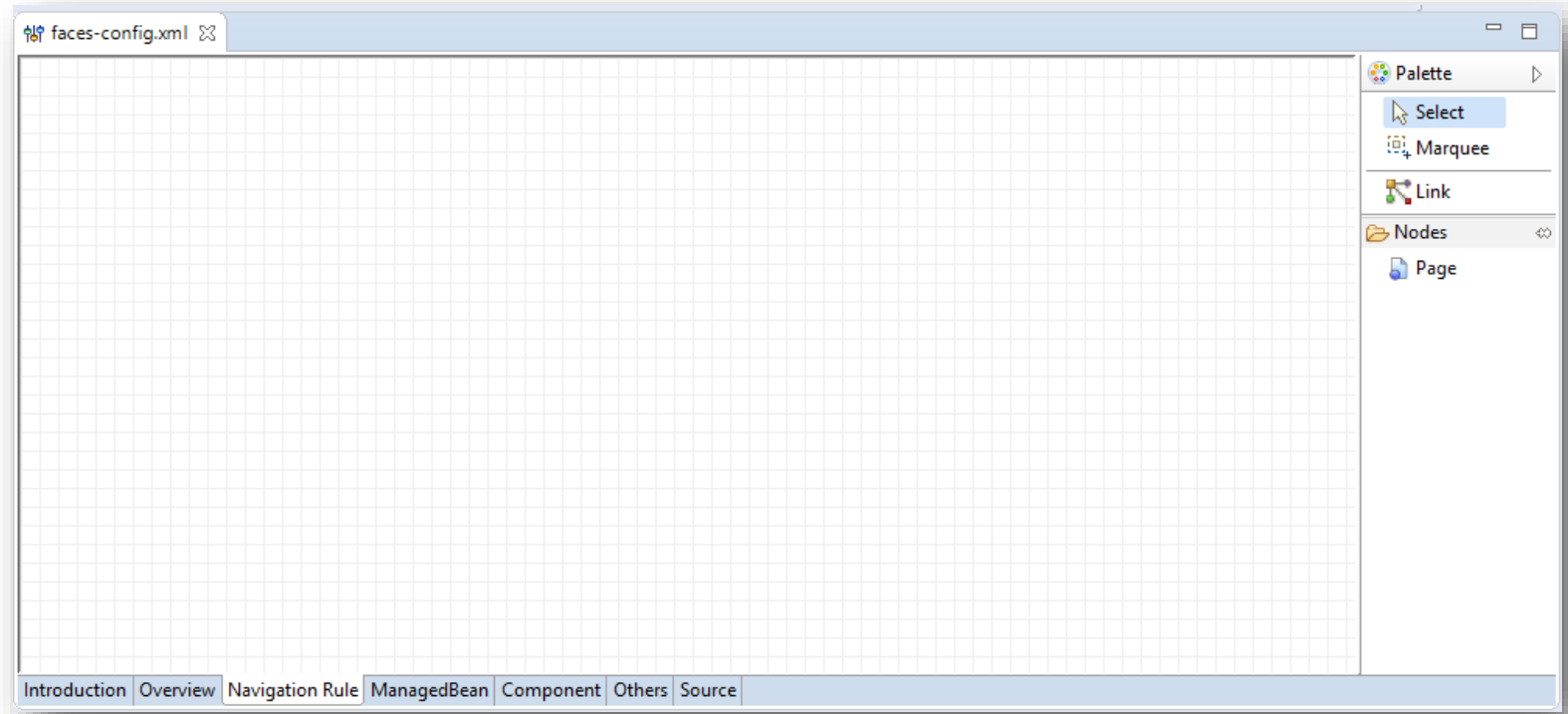
```
public String navigationExplicite() {  
    if(villeOrigine.equals("Dakar"))  
        return "dakar";  
    else if(villeOrigine.equals("Thies"))  
        return "thies";  
    return "laReponse";  
}
```

```
thies.xhtml  
1 <!DOCTYPE html>  
2 <html xmlns="http://www.w3.org/1999/xhtml"  
3     xmlns:h="http://xmlns.jcp.org/jsf/html"  
4     xmlns:a="http://xmlns.jcp.org/jsf/passthrough"  
5     xmlns:f="http://java.sun.com/jsf/core">  
6 <h:head>  
7     <title>THIES</title>  
8 </h:head>  
9  
10 <body>  
11     Bienvenue à THIES  
12 </body>  
13  
14 </html>
```

```
dakar.xhtml  
1 <!DOCTYPE html>  
2 <html xmlns="http://www.w3.org/1999/xhtml"  
3     xmlns:h="http://xmlns.jcp.org/jsf/html"  
4     xmlns:a="http://xmlns.jcp.org/jsf/passthrough"  
5     xmlns:f="http://java.sun.com/jsf/core">  
6 <h:head>  
7     <title>Dakar</title>  
8 </h:head>  
9  
10 <body>  
11     Bienvenue à Dakar  
12 </body>  
13  
14 </html>
```

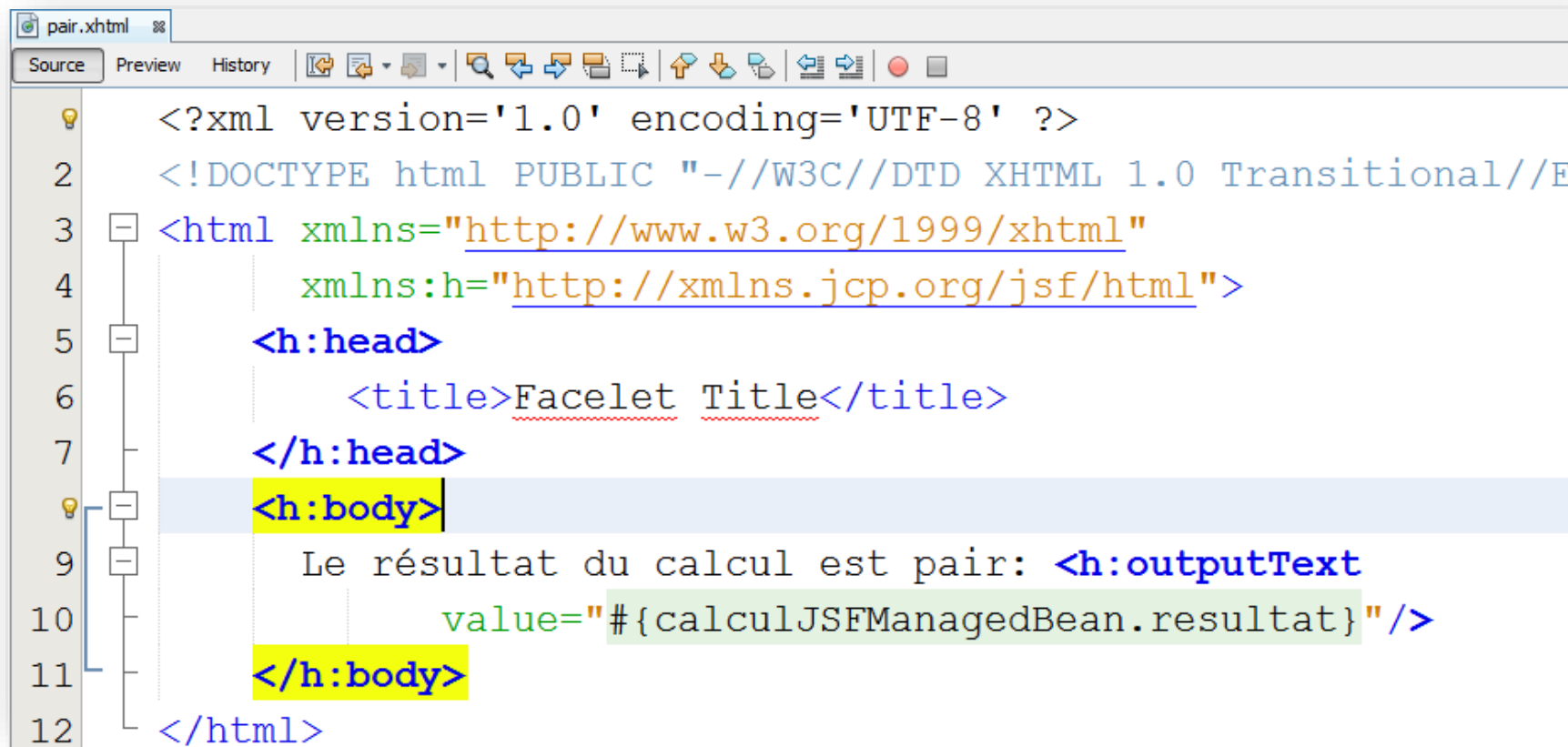
Navigation Explicite: séparer la navigation du résultat des actions

- Ouvrir le fichier JSF Faces Configuration
- Ne pas changer
- le nom du fichier



Navigation Explicite selon que le résultat soit pair ou impair

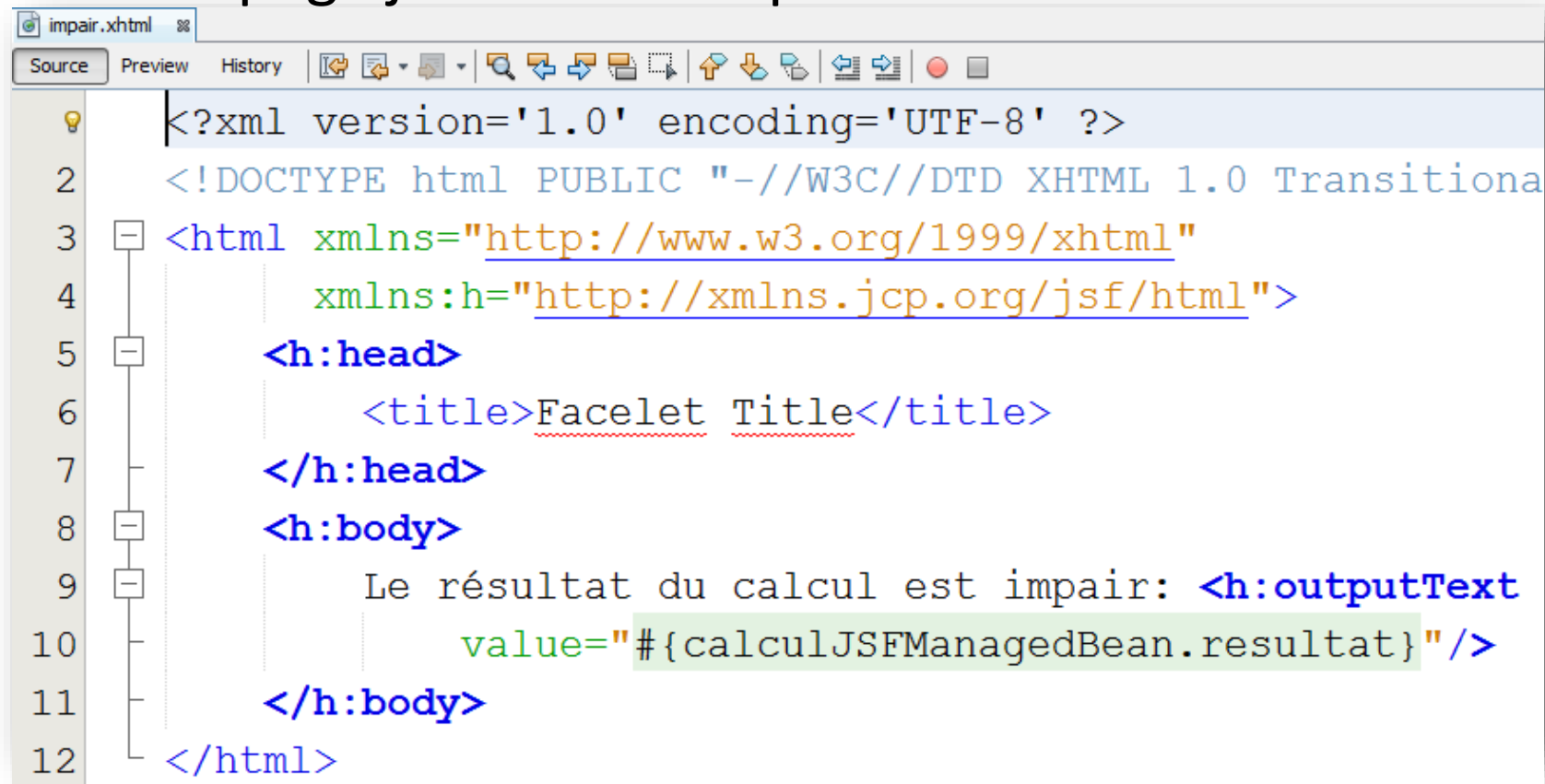
- Rajouter une page jsf nommé pair.xhtml



```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Facelet Title</title>
  </h:head>
  <h:body>
    Le résultat du calcul est pair: <h:outputText
      value="#{calculJSFManagedBean.resultat}"/>
  </h:body>
</html>
```

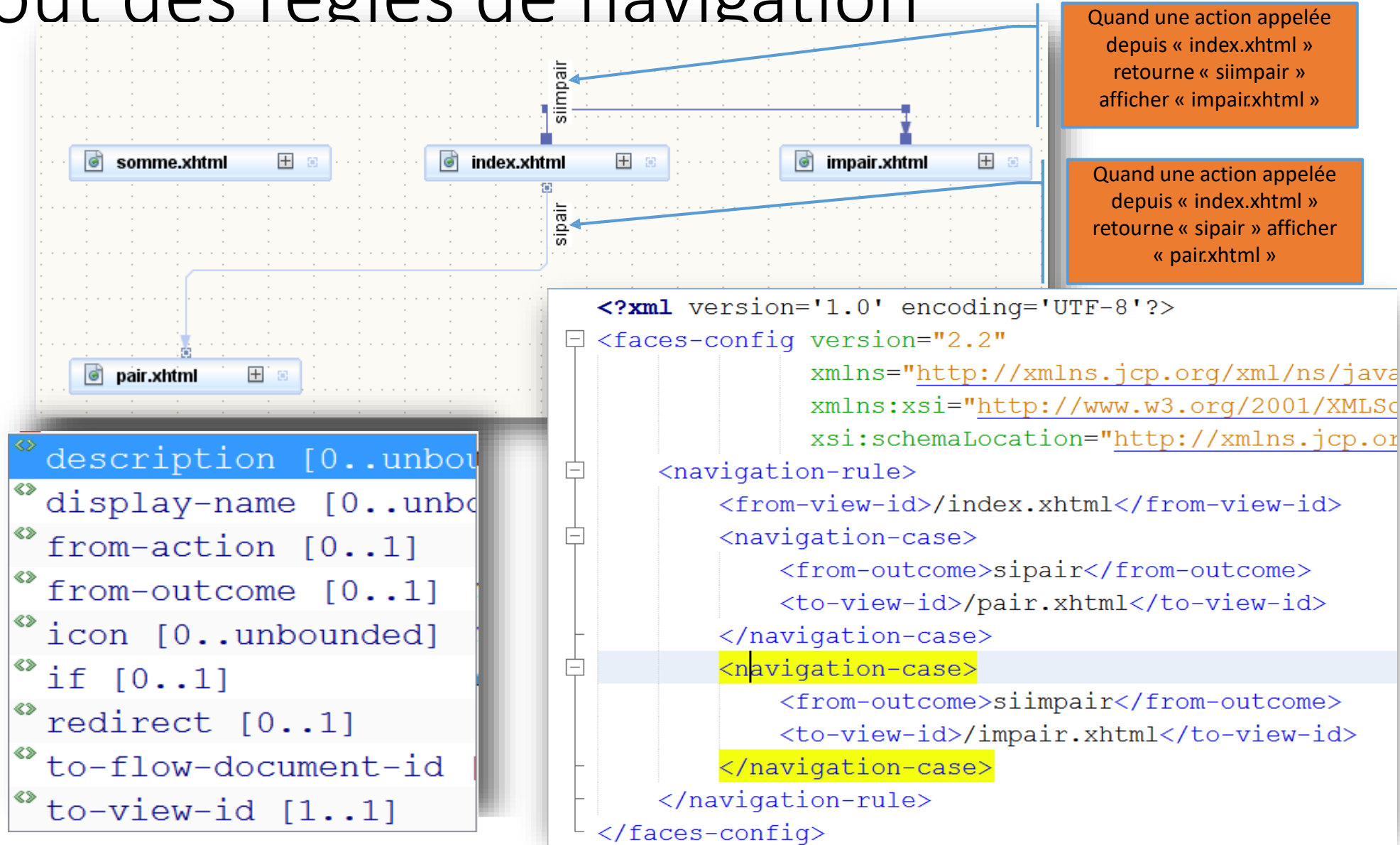
Navigation Explicite selon que le résultat soit pair ou impair

- Rajouter une page jsf nommé impair.xhtml



```
1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitiona
3 <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:h="http://xmlns.jcp.org/jsf/html">
5   <h:head>
6       <title>Facelet Title</title>
7   </h:head>
8   <h:body>
9       Le résultat du calcul est impair: <h:outputText
10          value="#{calculJSFManagedBean.resultat}"/>
11   </h:body>
12 </html>
```

Ajout des règles de navigation



Modifions maintenant la fonction somme du ManagedBean

```
public String somme() {  
    resultat = op1 + op2;  
    if (resultat % 2 == 0) {  
        return "sipair";  
    } else {  
        return "siimpair";  
    }  
}
```

Portées des beans

- Ce que l'on appelle portée d'un bean, dans le contexte des applications web, représente la durée de vie de ce bean en fonction de opérations menées à bien par l'utilisateur.
- [@Dependant](#) : le bean est utilisé uniquement pour un composant. Si plusieurs références sont faites dans la même page, il s'agit d'un bean différent à chaque fois
- [@RequestScoped](#) : le bean « vit » le temps de l'exécution de la requête (donc du retour du résultat à l'utilisateur)
- [@SessionScoped](#) : le bean vit le temps d'une session.

Portées des beans

- [@ViewScoped](#) : le bean vit le temps qu'une même vue est affichée, c'est-à-dire qu'aucune action effectuée ne change de page (ne retourne une valeur non nulle)
- [@ConversationScoped](#) : le bean vit le temps d'une « conversation ». Permet d'écrire des « wizards » (suites de dialogues).
- [@ApplicationScoped](#) : bean partagé au niveau de l'application.

Remarques sur la portée des beans

- [@RequestScoped](#) : temps d'une requête... bien comprendre quand démarre et quand s'arrête la requête
- [@ViewScoped](#) : durée de vie complètement différente de [@RequestScoped](#).
 - finit quand on quitte la page (@RequestScoped « vit » jusqu'à l'affichage de la page résultat)
 - survit par contre à des requêtes faites en restant sur la même page
 - bien adapté à l'utilisation d'Ajax.

Cycle de vie

- Comme tous les composants managés, les managed beans ont un cycle de vie. On peut définir deux callbacks sur ce cycle de vie, en annotant deux méthodes :
 - [@PostConstruct](#) : cette méthode sera invoquée une fois le bean construit, mais avant son utilisation par le moteur JSF ;
 - [@PreDestroy](#) : cette méthode sera appelée juste avant la destruction du bean.
- Comme d'habitude, la méthode appelée une fois le bean construit peut être utilisée pour préparer des connexions à des sources de données, entre autres.

Affichage des données avec les listes et tableaux JSF

- Créer un managed bean et la page JSF pour afficher la liste
- Collection autorisée doit être un des types suivant
 - Un tableau (ex `String[]`, `Integer[]`, ...)
 - `java.util.List`
 - `java.sql.ResultSet`
 - `java.servlet.jsp.jstl.sql.Result`
 - `javax.faces.model.DataModel`

Affichage des données avec les listes et tableaux JSF

`xmlns:ui="http://xmlns.jcp.org/jsf/facelets"`

- Soit la liste des ville précédemment déclarée et initialisée dans le manager bean *personneManagedBean*

```
listeVilles=new ArrayList<String>();  
listeVilles.add("Dakar");  
listeVilles.add("Thies");  
listeVilles.add("Louga");  
listeVilles.add("Saint-Louis");  
listeVilles.add("Podor");
```

- Pour afficher la liste dans la page JSF

```
<ui:repeat var="ville" value="#{personneManagedBean.listeVilles}">  
    #{ville}  
</ui:repeat>
```

Affichage des données avec les listes et tableaux

JSF: `<h:panelGrid>`

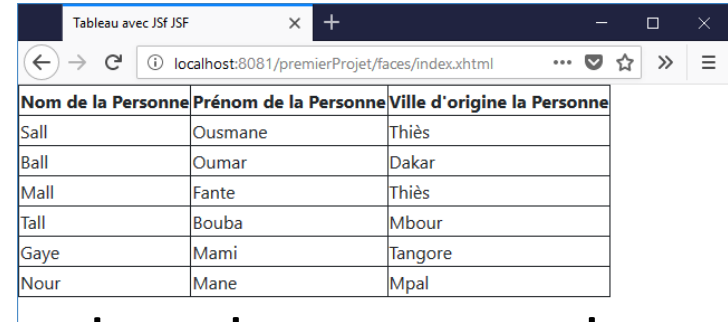
```
<h:panelGrid columns="4">
  <f:facet name="header">
    <h:outputText value="Tableau avec des nombres" />
  </f:facet>
  <h:outputText value="1" />
  <h:outputText value="2" />
  <h:outputText value="3" />
  <h:outputText value="4" />
  <h:outputText value="5" />
  <h:outputText value="6" />
  <h:outputText value="7" />
  <f:facet name="footer">
    <h:outputText value="Texte en bas du panel" />
  </f:facet>
</h:panelGrid>
```

Tableau avec des nombres

1	2	3	4
5	6	7	

Texte en bas du panel

Affichage des données avec les listes et tableaux JSF: `</h:dataTable>`



Nom de la Personne	Prénom de la Personne	Ville d'origine la Personne
Sall	Ousmane	Thiès
Ball	Oumar	Dakar
Mall	Fante	Thiès
Tall	Bouba	Mbour
Gaye	Mami	Tangore
Nour	Mane	Mpal

- Soit l'entité Personne suivant

```
Personne.java
1 package set.mbeans;
2
3 import java.util.List;
4
5 public class Personne{
6     private String nom;
7     private String prenom;
8     private String villeOrigine;
9     private List<String> listeVilles;
10
11     public List<String> getListeVilles() {}
12
13
14
15     public void setListeVilles(List<String> listeVilles) {}
16
17
18
19     public String getVilleOrigine() {}
20
21
22
23     public void setVilleOrigine(String villeOrigine) {}
24
25
26
27     public String getNom() {}
28
29
30
31     public void setNom(String nom) {}
32
33
34
35     public String getPrenom() {}
36
37
38
39     public void setPrenom(String prenom) {}
40
41
42
43     public Personne(String nom, String prenom, String villeOrigine) {
44         this.nom = nom;
45         this.prenom = prenom;
46         this.villeOrigine = villeOrigine;
47     }
48 }
```

Soit son intégration dans le manager bean

```
private List<Personne> listePersonnes=new ArrayList<Personne>(  
    Arrays.asList(  
        new Personne("Sall", "Ousmane", "Thiès"),  
        new Personne("Ball", "Oumar", "Dakar"),  
        new Personne("Mall", "Fante", "Thiès"),  
        new Personne("Tall", "Bouba", "Mbour"),  
        new Personne("Gaye", "Mami", "Tangore"),  
        new Personne("Nour", "Mane", "Mpal")));  
  
public List<Personne> getListePersonnes() {  
    return listePersonnes;  
}  
  
public void setListePersonnes(List<Personne> listePersonnes) {  
    this.listePersonnes = listePersonnes;  
}
```

Affichage des données avec les listes et tableaux JSF: `</h:dataTable>`

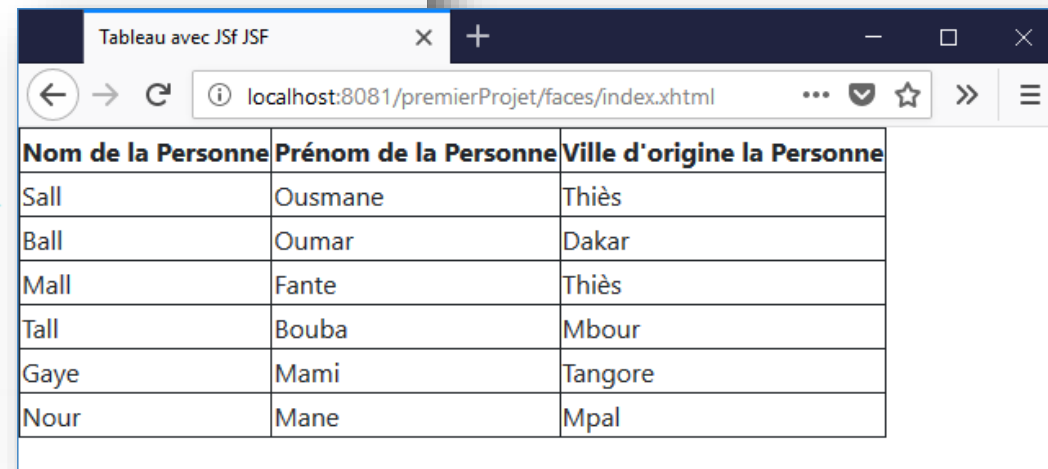
- Pour afficher la collection :

Ne pas oublier d'inclure l'espace de noms:

`xmlns:f="http://xmlns.jcp.org/jsf/core"`

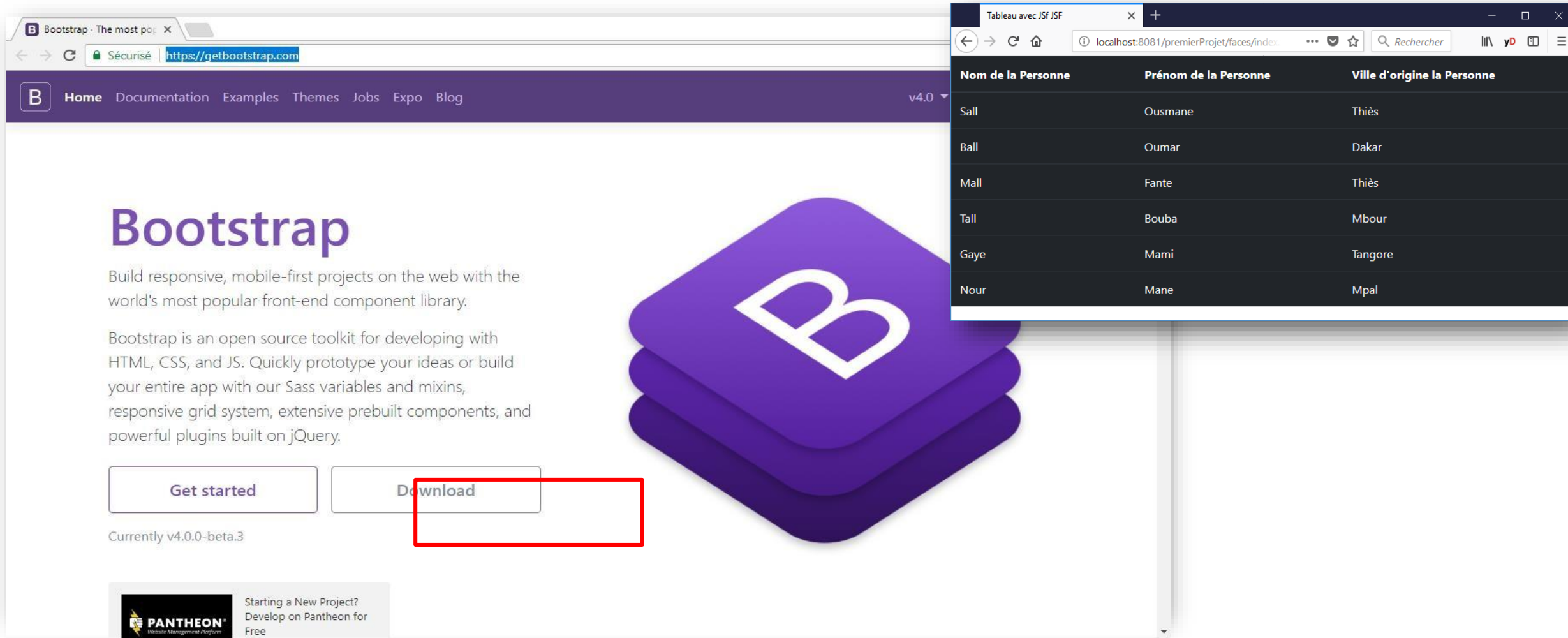
```
<h:dataTable var="pers" value="#{personneManagedBean.listePersonnes}" border="1">
  <h:column>
    <!-- Titre de la colonne -->
    <f:facet name="header"> Nom de la Personne</f:facet>
    <!-- Valeur de chaque ligne -->
    #{pers.nom}
  </h:column>
  <h:column>
    <!-- Titre de la colonne -->
    <f:facet name="header"> Prénom de la Personne</f:facet>
    <!-- Valeur de chaque ligne -->
    #{pers.prenom}
  </h:column>
  <h:column>
    <!-- Titre de la colonne -->
    <f:facet name="header"> Nom de la Personne</f:facet>
    <!-- Valeur de chaque ligne -->
    #{pers.villeOrigine}
  </h:column>
</h:dataTable>
```

- `@value` représente une collection sur la quelle l'itération sur les lignes du tableau porte
- `@var` contient le nom de la variable courante d'itération



Nom de la Personne	Prénom de la Personne	Ville d'origine la Personne
Sall	Ousmane	Thiès
Ball	Oumar	Dakar
Mall	Fante	Thiès
Tall	Bouba	Mbour
Gaye	Mami	Tangore
Nour	Mane	Mpal

Utiliser Bootstrap pour le design du tableau



The image shows two browser windows. The left window displays the Bootstrap v4.0 website at <https://getbootstrap.com>. The right window shows a web application titled 'Tableau avec JSF JSF' running on localhost:8081/premierProjet/faces/index, which features a table with three columns: 'Nom de la Personne', 'Prénom de la Personne', and 'Ville d'origine la Personne'.

Bootstrap Website Content:

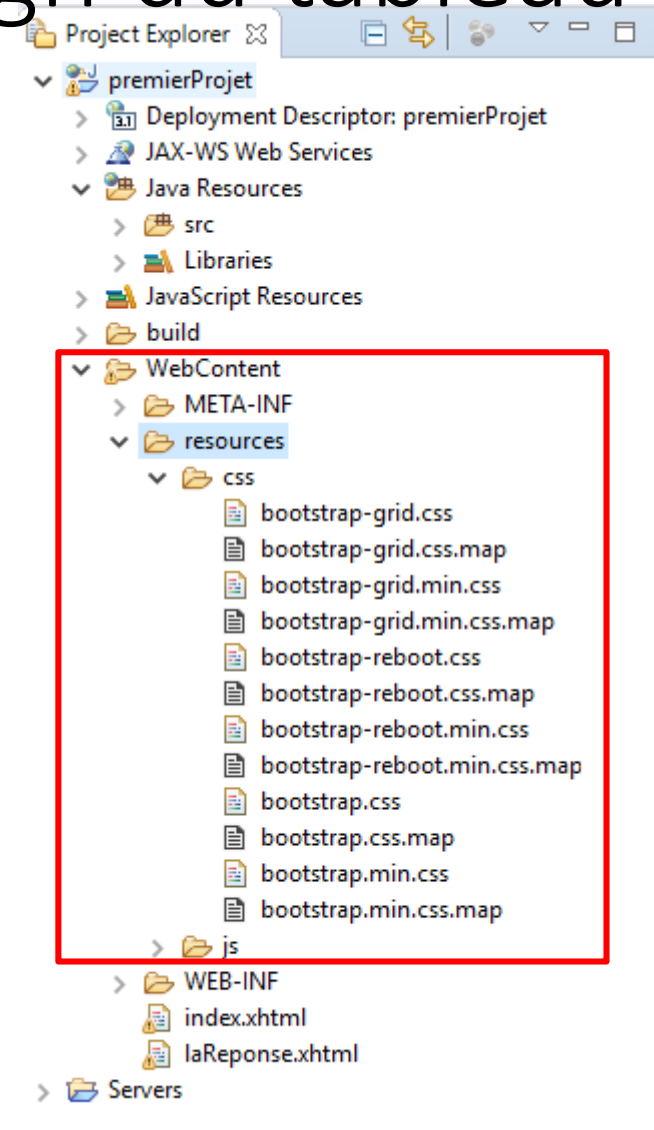
- Navigation: Home, Documentation, Examples, Themes, Jobs, Expo, Blog
- Version: v4.0
- Header: Bootstrap
- Text: Build responsive, mobile-first projects on the web with the world's most popular front-end component library.
- Text: Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.
- Buttons: Get started, Download
- Text: Currently v4.0.0-beta.3
- Footer: PANTHEON Website Management Platform, Starting a New Project? Develop on Pantheon for Free

Tableau avec JSF JSF Content:

Nom de la Personne	Prénom de la Personne	Ville d'origine la Personne
Sall	Ousmane	Thiès
Ball	Oumar	Dakar
Mall	Fante	Thiès
Tall	Bouba	Mbour
Gaye	Mami	Tangore
Nour	Mane	Mpal

Utiliser Bootstrap pour le design du tableau

- Créer un dossier nommé **resources** dans WebContent
- Coller les dossiers css et js de bootstrap



Utiliser Bootstrap pour le design du tableau

```
laReponse.xhtml
1 <html xmlns="http://www.w3.org/1999/xhtml" xmlns:h="http://xmlns.jcp.org/jsf/html"
2     xmlns:ui="http://xmlns.jcp.org/jsf/facelets" xmlns:f="http://xmlns.jcp.org/jsf/core">
3 <h:head>
4     <title>Tableau avec JSF JSF</title>
5     <h:outputStylesheet library="css" name="bootstrap.css"/>
6 </h:head>
7
8 <h:body>
9     <h:dataTable var="pers" value="#{personneManagedBean.listePersonnes}"
10     styleClass="table table-hover table-dark">
11         <h:column>
12             <!-- Titre de la colonne -->
13             <f:facet name="header"> Nom de la Personne</f:facet>
14             <!-- Valeur de chaque ligne -->
15             #{pers.nom}
16         </h:column>
17         <h:column>
18             <!-- Titre de la colonne -->
19             <f:facet name="header"> Prénom de la Personne</f:facet>
20             <!-- Valeur de chaque ligne -->
21             #{pers.prenom}
22         </h:column>
23         <h:column>
24             <!-- Titre de la colonne -->
25             <f:facet name="header"> Nom de la Personne</f:facet>
26             <!-- Valeur de chaque ligne -->
27             #{pers.villeOrigine}
28         </h:column>
29     </h:dataTable>
30 </h:body>
```

Nom du dossier contenant le fichier de style bootstrap.css

Tableau avec JSF JSF		
Nom de la Personne	Prénom de la Personne	Ville d'origine la Personne
Sall	Ousmane	Thiès
Ball	Oumar	Dakar
Mall	Fante	Thiès
Tall	Bouba	Mbour
Gaye	Mami	Tangore
Nour	Mane	Mpal

Webography

- https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
- Tutorial Java EE 7, section Servlets:
<https://docs.oracle.com/javaee/7/index.html>
- <http://www.servletworld.com/> : nombreux tutoriaux et exemples
- <http://www.kodejava.org/browse/8.html> : idem, nombreux exemples
- Google
- Youtube