El hadji Ibrahima DIAGO

Master 2

Institut Superieur d'Informatique

Java Server Faces

Définition

JavaServer Faces (abrégé en JSF) est un framework Java, pour le développement d'applications Web.

À l'inverse des autres frameworks **MVC** [i]traditionnels à base d'actions, JSF est basé sur la notion de **composants**[ii], comparable à celle de Swing ou SWT, où l'état d'un composant est enregistré lors du rendu de la page, pour être ensuite restauré au retour de la requête.

JSF est agnostique à la technologie de présentation. Il utilise Facelets (en) par défaut depuis la version 2.0, mais peut être utilisé avec d'autres technologies, comme JSP (qui était utilisé jusqu'à la version 1.2) ou **XUL**[iii].

MVC : Modèle-vue-contrôleur ou MVC est un motif d'architecture logicielle destiné aux interfaces graphiques lancé en 1978 et très populaire pour les applications web. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

Composants : La programmation orientée composant (POC) consiste à utiliser une approche modulaire de l'architecture d'un projet informatique, ce qui permet d'assurer au logiciel une meilleure lisibilité et une meilleure maintenance. Les développeurs, au lieu de créer un exécutable monolithique, se servent de briques réutilisables.

La POC n'est pas sans similitudes avec la POO, puisqu'elle revient à utiliser une approche objet, non pas au sein du code, mais au niveau de l'architecture générale du logiciel.

La POC est particulièrement pratique pour le travail en équipe et permet d'industrialiser la création de logiciels.

XUL : XML-based User interface Language (abréviation XUL) est un langage de description d'interfaces graphiques fondé sur XMLcréé dans le cadre du projet Mozilla. XUL comprend un ensemble de balises permettant de définir des boutons, des listes, des menus, ou encore des zones d'édition ; en résumé, tous les éléments d'une véritable interface utilisateur. Un tel langage facilite le travail du développeur et permet d'écrire l'interface d'une application aussi aisément qu'une page web.

Les versions de JSF

- **Version JSF 1.0** sortie le 11 mars 2004.
- **Version JSF 1.1** sortie le 27 mai 2004
- **Version JSF 1.2** sortie le 11 mai 2006
- **Version JSF 2.0** sortie le 28 juin 2009
- **Version JSF 2.1** sortie le 22 octobre 2010
- **Version JSF 2.2** sortie en avril 2013

Statut actuel

- JSF 2.2 est la norme d'interface utilisateur pour Java EE 7 La dernière version majeure de JSF est 2.2. Cette version a eu lieu le 21 mai 2013.
  - L'implémentation exécutable de JSF 2.2 est incluse dans **GlassFish 4.0** .
  - La spécification lisible par l'homme peut être téléchargée à partir de *http://jcp.org/*
  - La spécification binaire est disponible chez maven central à ces coordonnées.

```
<dependency>
  <groupId>javax.faces</groupId>
  <artifactId>javax.faces-api</artifactId>
  <version>2.2</version>
  <scope>provided</scope>
</dependency>
```

- JSF 2.3 est la norme d'interface utilisateur désignée pour Java EE 8 . Il est allé en finale le 17 avril 2017.
  - Bien que JSF 2.3 soit complet, Java EE 8 est encore en développement. Les implémentations exécutables des jalons JSF 2.3 ainsi que d'autres versions sont disponibles dans le dépôt javax.faces . Cette version est incluse dans **GlssFish 5 Builds** .
  - La spécification lisible par l'homme peut être téléchargée à partir de http://jcp.org/
  - L'API est disponible chez Maven Central à ces coordonnées.

```
<dependency>
  <groupId>javax.faces</groupId>
  <artifactId>javax.faces-api</artifactId>
  <version>2.3</version>
  <scope>provided</scope>
</dependency
```

  - La mise en œuvre est également disponible à ces coordonnées

```
<dependency>
  <groupId>org.glassfish</groupId>
  <artifactId>javax.faces</artifactId>
  <version>2.3.0</version>
  <scope>provided</scope>
</dependency>
```

Les constituants de JSF

JSF est constitué principalement de:

- Un ensemble d'APIs pour la représentation et la gestion des composants, de leur état, des évènements, de la validation des entrées et la conversion des sorties, l'internationalisation et l'accessibilité ainsi que la navigation inter-vues
- Deux jeux de composants standards (affichage de texte, saisie de texte, tables, zone à cocher, etc.) : html et core
- Deux bibliothèques de balises JSP (une pour chaque jeu de composants) pour permettre l'utilisation des JSPs pour la construction de vues JSF
- Un modèle évènementiel côté serveur
- **Les Managed-Beans** : qui forment la couche contrôle de JSF
- Unified Expression Language (abrégé en EL) ou langage d'expressions unifié pour JSF et JSP 2.0. Il permet de lier les composants aux managed-beans

Les composants de JSF

Tous les frameworks Java qui permettent de construire des applications web fournissent un jeu de composants, construits au-dessus des éléments standard HTML. JSF ne fait pas exception à cette règle, et propose des dizaines de tels composants, du plus simple au plus complexe.

Le développement d'applications réelles en JSF requiert la connaissance de ce jeu de composants (composants core et composants HTML). Il peut aussi s'acquérir au fur et à mesure du développement de telles applications.

1. Composants core

Tag	Description
`f:view`	Crée une vue au premier plan.
`f:subview`	Crée une sous-vue d'une vue.
`f:facet`	Ajoute une facette à un composant.
`f:attribute`	Ajoute un attribut (clé/valeur) à un composant.
`f:param`	Crée un composant de paramètre
`f:actionListener`	Ajoute un écouteur à un composant.
`f:valueChangeListener`	Ajoute un écouteur de valeur changée à un composant.

f:converter	Ajoute une conversion arbitraire à un composant.
f:convertDateTime	Ajoute une conversion date/heure à un composant.
f:convertNumber	Ajoute une conversion de nombre à un composant.
f:validator	Ajoute une validateur à un composant.
f:validateDoubleRange	Valide un intervalle de double pour la valeur du composant.
f:validateLength	Valide la taille de la valeur du composant.
f:validateLongRange	Valide un intervalle de long pour la valeur du composant.
f:loadBundle	Charge un ensemble de ressources, et enregistre les propriétés dans une *Map*.
f:selectitems	Spécifie les choix pour les *h:selectMany**.
f:selectitem	Spécifie les choix pour les *h:selectOne**.
f:verbatim	Ajoute des balises dans une page JSF.

2. Les composants HTML

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.
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:grapicImage	Affiche une image.
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.
h:selectBooleanCheckbox	*Checkbox*.
h:selectManyCheckbox	Met en place des *checkboxes*.
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`

Composants additionnels

Les deux jeux de composants standards de JSF (core et HTML) s'avèrent trop limités et insuffisants pour le développement d'applications d'entreprise. Il est possible dès lors d'utiliser des jeux de composants additionnels qui offrent de nouveaux composants plus riches.

On peut citer par exemple:

Primefaces : PrimeFaces est une bibliothèque de composants d'interface utilisateur (UI) open source pour applications JavaServer Faces (JSF), créée par PrimeTek Turkey. Le développement initial de PrimeFaces a débuté fin 2008. [4] Le prédécesseur de PrimeFaces est la bibliothèque YUI4JSF un ensemble de composants JSF basés sur la bibliothèque YUI JavaScript. YUI4JSF a été annulé en faveur de PrimeFaces début 2009.

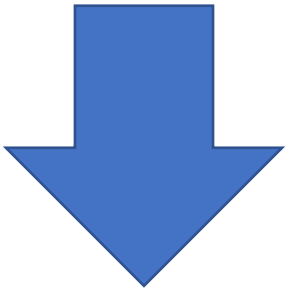
Depuis sa sortie, PrimeFaces a été fortement soutenu par Oracle, en particulier dans le monde NetBeans . La PrimeFaces 6.1 sortie en avril 2017.

ICEfaces : ICEfaces est un kit de développement logiciel open source qui étend JavaServer Faces (JSF) en utilisant Ajax . Il est utilisé pour construire des applications Internet riches (RIA) en utilisant le langage de programmation Java . Avec ICEfaces, le codage de l'interaction et Ajax côté client est programmé en Java, plutôt qu'en JavaScript , ou avec des plug-ins .jBoss RichFaces et Ajax4JSF, un jeu de composants open-source supportant Ajax (End of Life schedulded)

Apache MyFaces : Apache MyFaces est un projet Apache Software Foundation qui crée et maintient une implémentation JavaServer Faces open-source, ainsi que plusieurs bibliothèques de composants JSF pouvant être déployées sur l'implémentation principale. Le projet est divisé en plusieurs sous-projets: Tomahawk, Trinidad, Tobago, Portlet Bridge etc…

RCFaces : un jeu de composants très riche AJAX et open-source

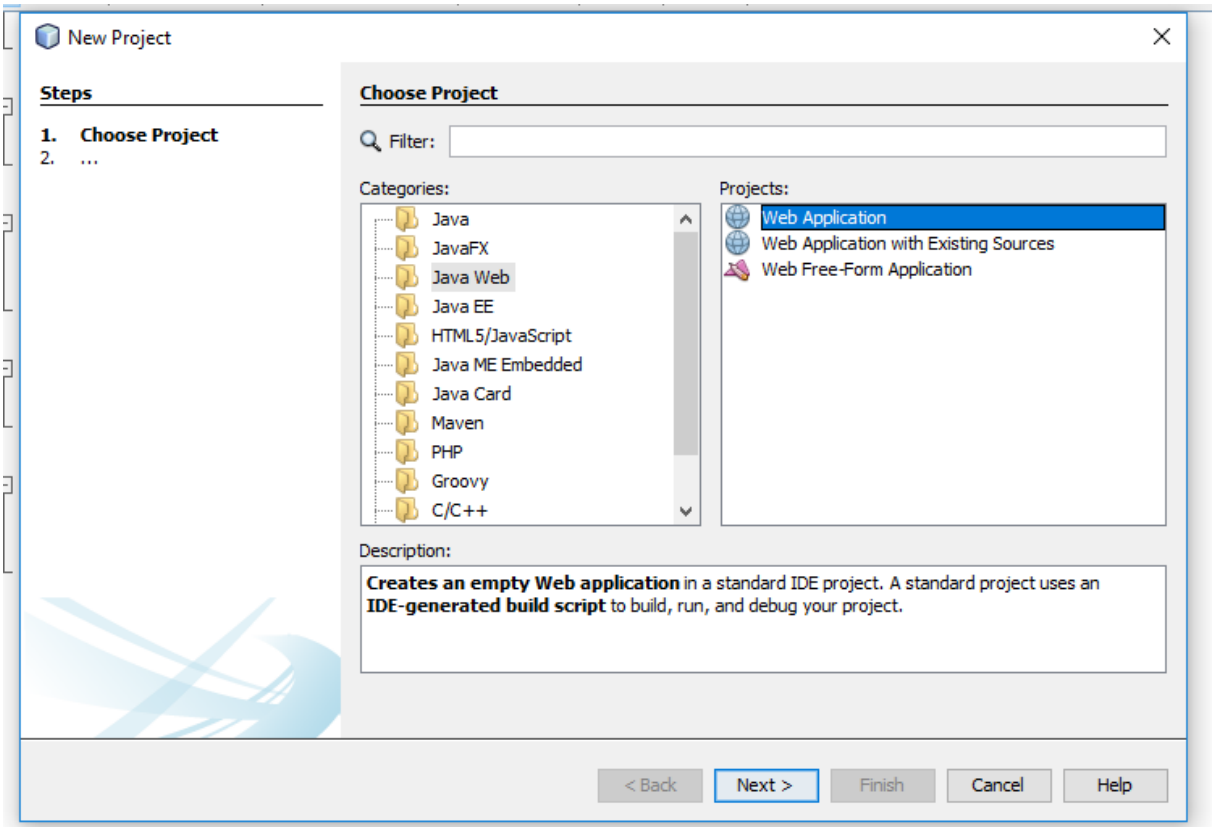
CAS PRATIQUE

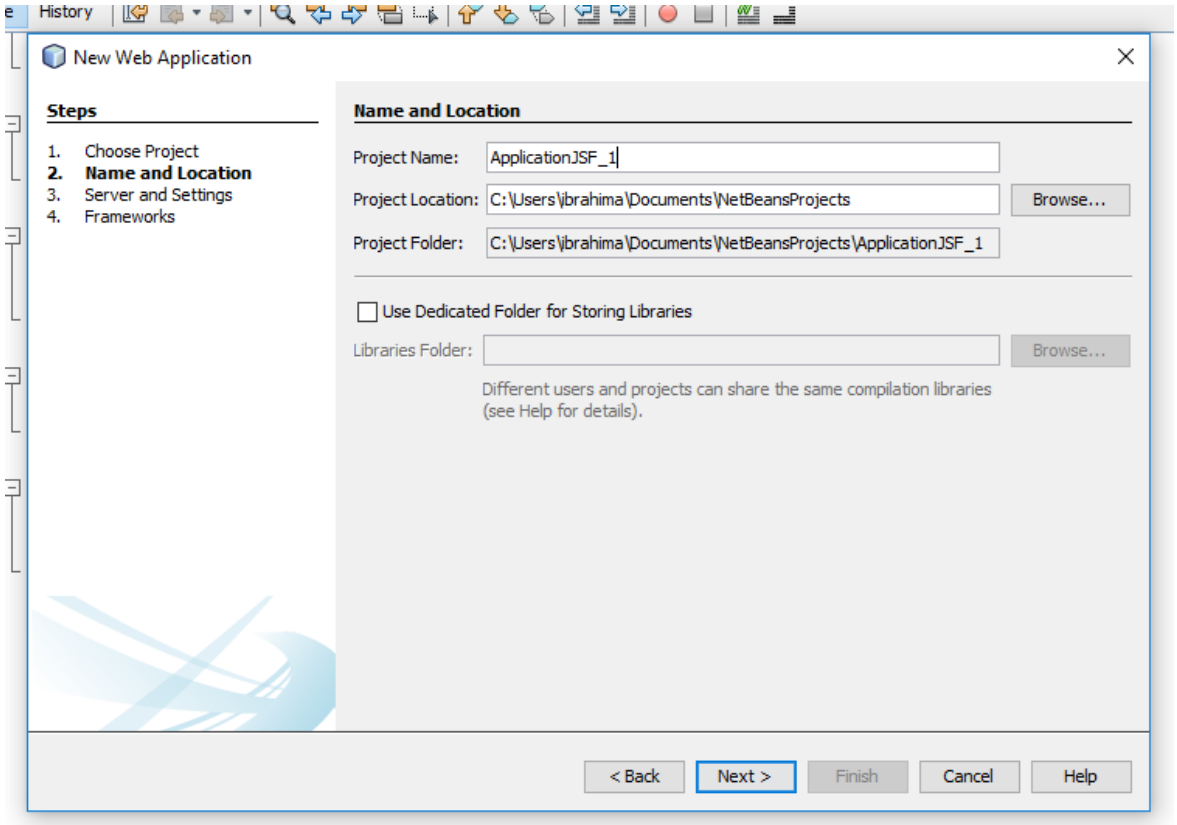


Création d'un projet JSF sous Netbeans

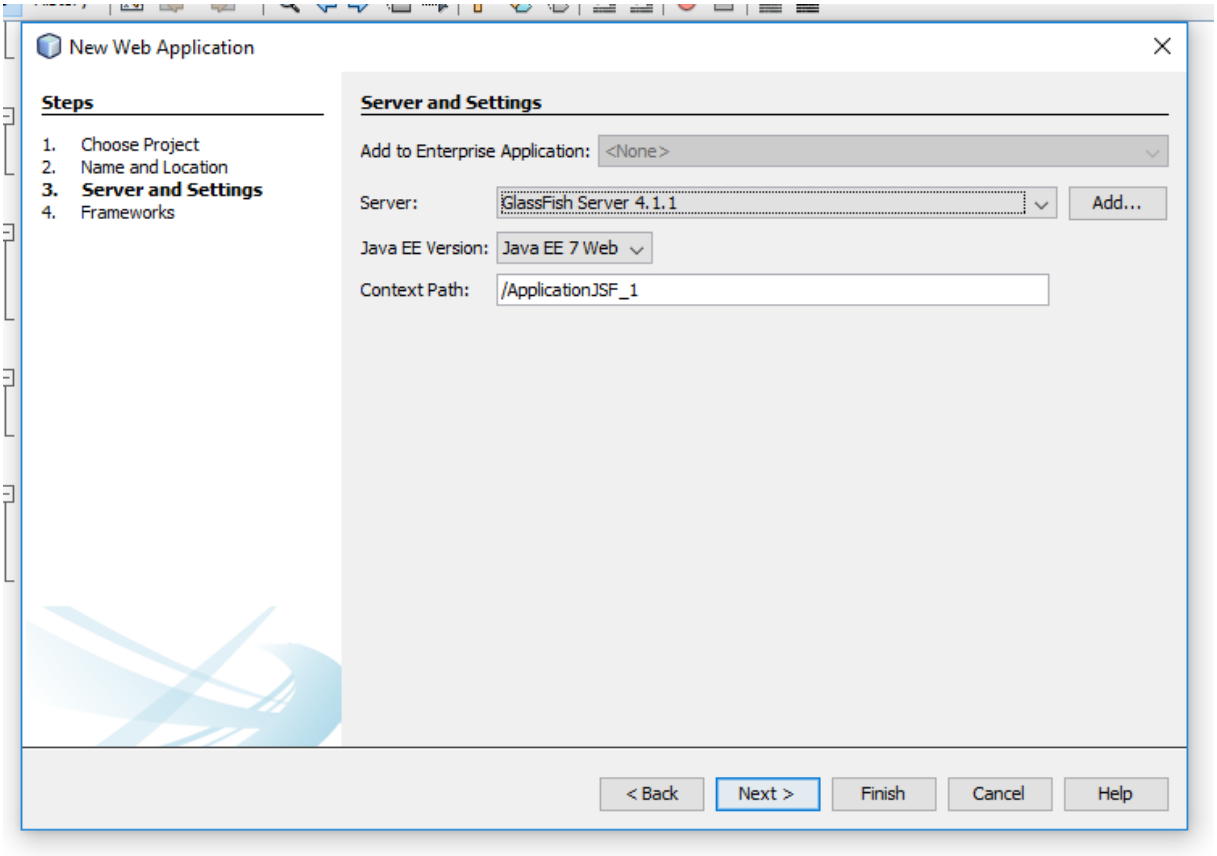
Environnement : GlassFish 4.1.1 – JavaServer Faces 2.2 - Java EE 7

Un projet JSF est une application web classique. La création du pojet Netbeans suit donc celle d'un tel projet.

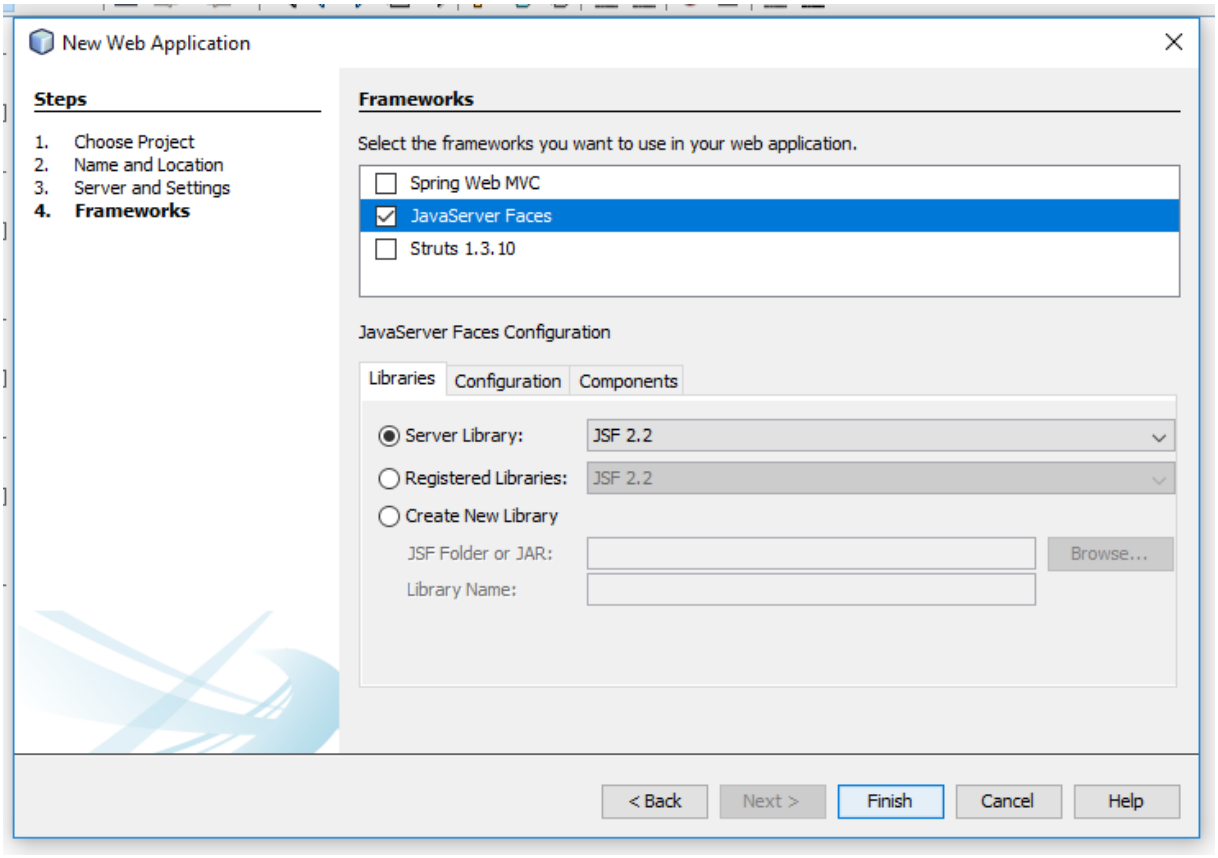




On prendra garde à créer un projet de type JEE7, sous Glassfish v4.1.1.



Enfin la dernière étape de création de ce projet nous permet de sélectionner JSF parmi les frameworks que nous allons utiliser.

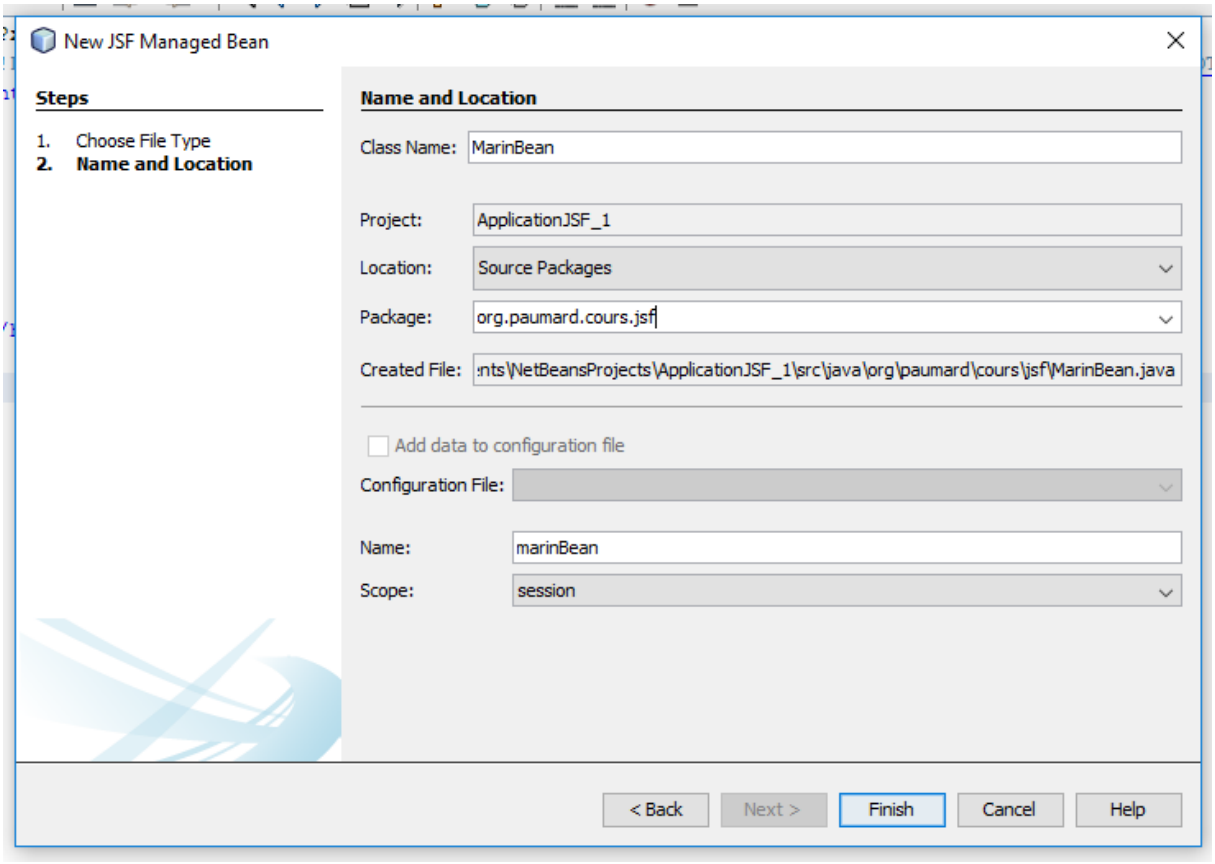


La structure finale de notre projet est la suivante. Elle ne diffère en rien d'un classique projet servlet / JSP.

Création d'un bean applicatif

Ce projet créé par Netbeans ne fait pas grand chose d'autre que de mettre en place une unique page XHTML. Nous allons créer un premier formulaire, adossé à un bean de notre vue.

Créons tout d'abord un bean MarinBean dans le dossier Source packages de notre projet Netbeans. Voici le code de ce bean.



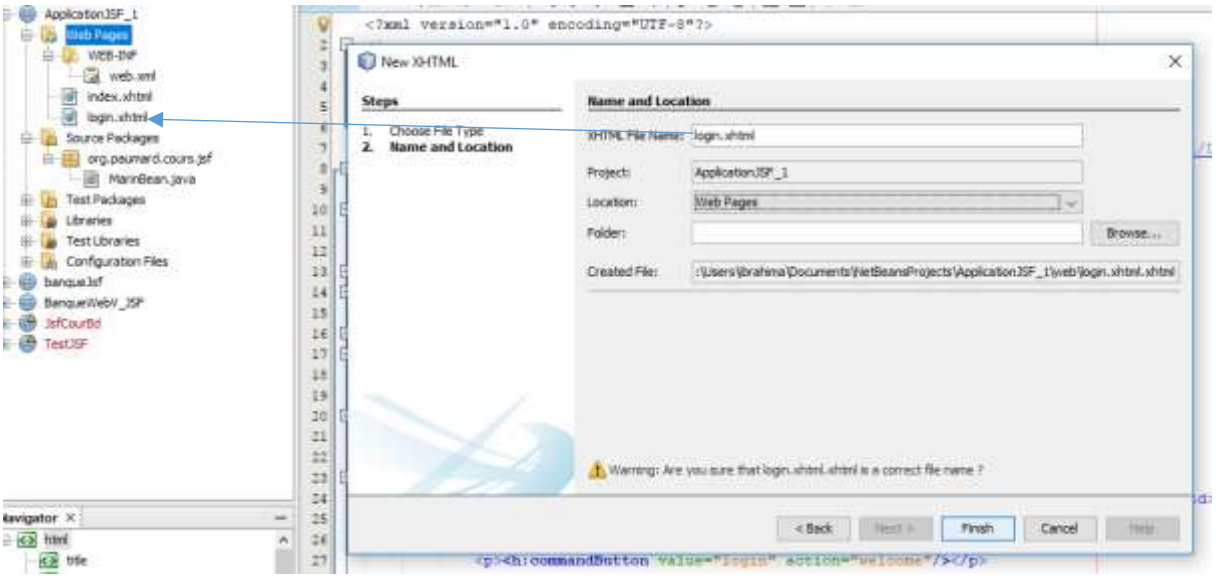
Exemple 1. Bean MarinBean pour notre vue

```
Source  History
    @ManagedBean(name="marin")
17  @SessionScoped
18  public class MarinBean implements Serializable{
19      private String nom, prenom, passwd;
20
21      public String getNom() {
22          return nom;
23      }
24
25      public void setNom(String nom) {
26          this.nom = nom;
27      }
28
29      public String getPrenom() {
30          return prenom;
31      }
32
33      public void setPrenom(String prenom) {
34          this.prenom = prenom;
35      }
36
37      public String getPasswd() {
38          return passwd;
39      }
40
41      public void setPasswd(String passwd) {
42          this.passwd = passwd;
43      }
44      public MarinBean() {
45      }
46
47  }
```

Notons que ce bean est spécifique à notre application web, ce n'est pas un bean du modèle, que nous aurions pu annoter, par exemple avec des annotations JPA. On remarquera la présence de deux annotations sur ce bean :

- @SessionScoped : indique que les instances de ce bean sont associées à la session de l'utilisateur.
- @ManagedBean(name="marin") : indique que ce bean est géré par le moteur JSF, et désigné dans l'application sous le nom marin.

Création d'un formulaire




```
<?xml version="1.0" encoding="UTF-8"?>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitio
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Connexion</title>
    </h:head>
    <h:body>
        <h:form>
            <h1>Entrez votre nom et votre mot de passe</h1>
            <table>
                <tr>
                    <td>Nom : </td><td><h:inputText value="#{marin.nom}"/></td>
                </tr>
                <tr>
                    <td>Prénom : </td><td><h:inputText value="#{marin.prenom}"/></td>
                </tr>
                <tr>
                    <td>Mot de passe : </td><td><h:inputSecret value="#{marin.passwd}"/></td>
                </tr>
            </table>
            <p><h:commandButton value="login" action="welcome"/></p>
        </h:form>
    </h:body>
</html>
```

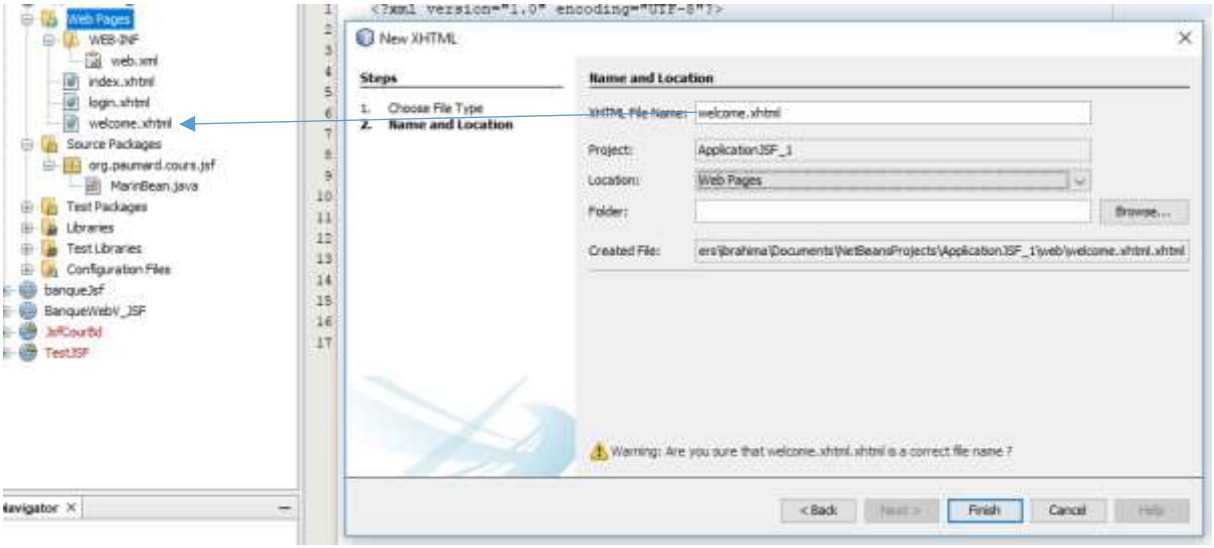
Il s'agit d'un fichier XHTML classique, comme dans les cours sur les servlets / JSP. Ce formulaire est structuré dans une table HTML classique. On remarque l'utilisation de deux choses dans ce formulaire :

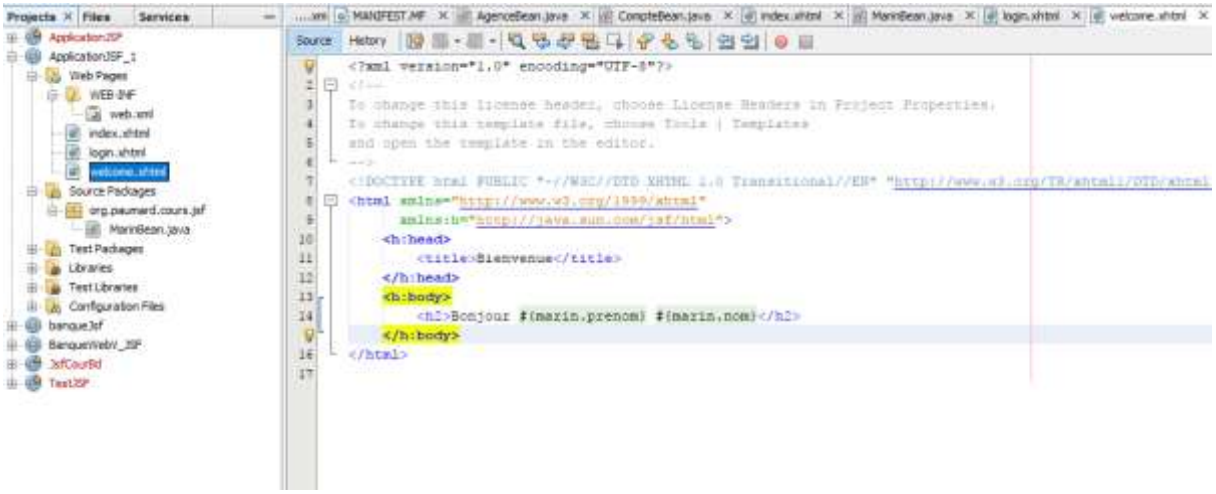
- Des tags de la bibliothèque JSF : inputText, inputSecret et commandButton ;
- Des références au bean marin et à ses propriétés sous la forme #{marin.nom}.

L'utilisation de ces tags et des références au bean marin permet de faire la liaison directe avec une instance de ce bean. Le fait de soumettre ce formulaire va automatiquement créer une instance de Marin, peupler ses champs avec les valeurs entrées dans le formulaire, et attacher cette instance à la session.

Création d'une page de bienvenue

L'action de ce formulaire est welcome. Cette action peut correspondre à plusieurs choses dans notre application. Ici, nous allons créer une simple page welcome.xhtml, qui sera le point de redirection de notre action.





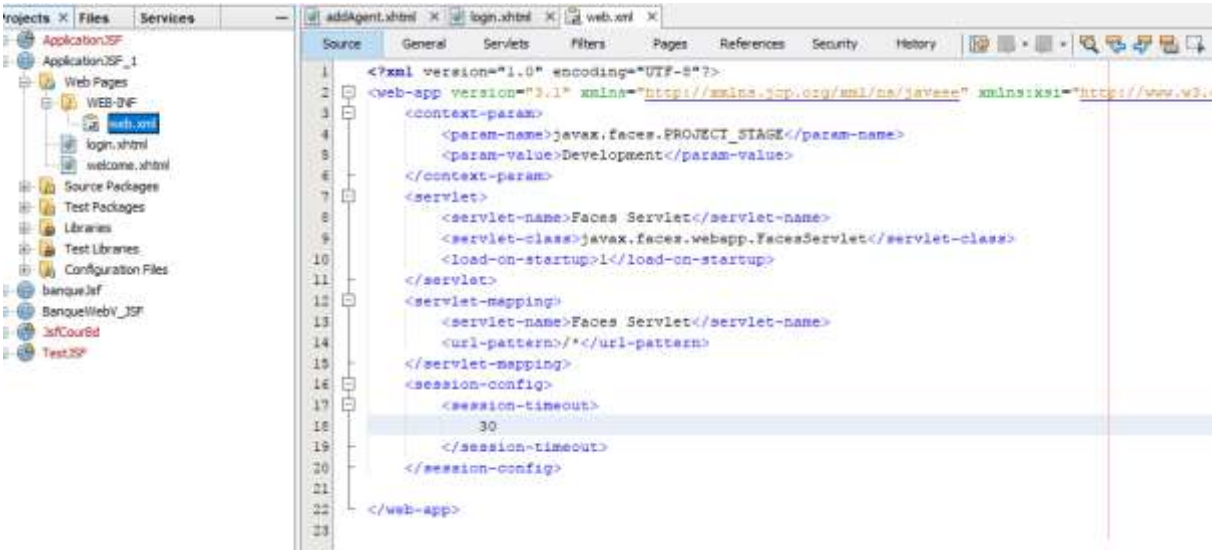
On remarque à nouveau dans cette page l'utilisation de tags JSF, et du bean marin.

Configuration de l'application

Pour que nos pages soient correctement traitées par JSF, il nous faut encore configurer la servlet JSF. Cette unique servlet technique prend en charge l'ensemble des requêtes des applications JSF, et a en charge la création des beans et leurs liaisons avec les formulaire ou pages d'affichage, de même que la navigation entre ces pages.

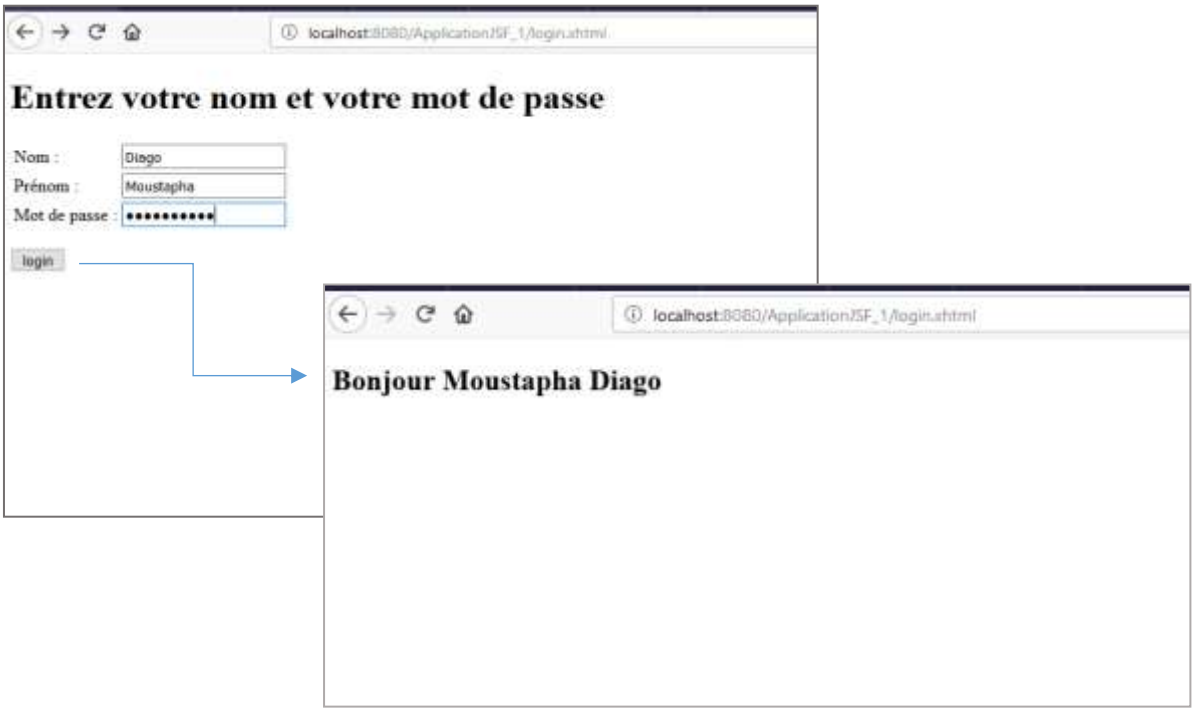
Pour l'activer, il faut ajouter ces éléments au fichier web.xml de notre application.

Cette servlet technique a pour classe javax.faces.webapp.FacesServlet et doit intercepter toutes les requêtes de notre application. On remarquera aussi le paramètre javax.faces.PROJECT_STAGE positionné à Development.



Fonctionnement de l'application

On pourra tester cette application en exécutant le fichier login.xhtml sous Netbeans.



FIN TUTO

[i] Modèle-vue-contrôleur ou MVC est un motif d'architecture logicielle destiné aux interfaces graphiques lancé en 1978 et très populaire pour les applications web. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

[ii] **Composants** : La programmation orientée composant (POC) consiste à utiliser une approche modulaire de l'architecture d'un projet informatique, ce qui permet d'assurer au logiciel une meilleure lisibilité et une meilleure maintenance. Les développeurs, au lieu de créer un exécutable monolithique, se servent de briques réutilisables.

La POC n'est pas sans similitudes avec la POO, puisqu'elle revient à utiliser une approche objet, non pas au sein du code, mais au niveau de l'architecture générale du logiciel.

La POC est particulièrement pratique pour le travail en équipe et permet d'industrialiser la création de logiciels.

[iii] XML-based User interface Language (abréviation XUL) est un langage de description d'interfaces graphiques fondé sur XMLcréé dans le cadre du projet Mozilla. XUL comprend un ensemble de balises permettant de définir des boutons, des listes, des menus, ou encore des zones d'édition ; en résumé, tous les éléments d'une véritable interface utilisateur. Un tel langage facilite