

## **Exercice9 : Configuration de son outil de développement**

Dans cet exercice, nous allons utiliser l'implémentation de Sun, nommée Sun JSF RI 1.2 (Mojarra). Les développements sont effectués avec Eclipse 3.5 (Ganymède) avec WTP et Apache Tomcat 6.0 (Catalina). Une version JSF 1.2 exige au moins Java 5.0, JSP 2.1 et Servlet 2.5 API's. Nous allons suivre les étapes pour installer cet ensemble afin de pouvoir effectuer les premiers développements.

Il y a d'autres IDE's comparables à Eclipse, tels que. Sun Netbeans, Sun Java Studio Creator, Oracle JDeveloper, IntelliJ IDEA, etc. Le choix d'Eclipse est fait car il est grandement configurable, personnalisable et possède davantage de plug-ins que ses voisins. En revanche c'est un goinfre de mémoire vive et de temps en temps il fait un crash nerveux.

Il y a d'autres serveurs d'application Web proches de Apache Tomcat, tels que. Sun Java Application Server (nommé Glassfish dans sa version actuelle), JBoss Application Server, Mortybay Jetty, Objectweb Jonas, etc. Le choix de Tomcat s'explique par sa taille petite, son chargement aisé, il est rapide et il est supporté par Eclipse par défaut avec son plug in WTP. Son seul désavantage est de ne pas supporter les EJBs.

### **Etape1 : téléchargement de Java**

Si ce n'est pas déjà fait, télécharger la dernière version de java jdk1.6.0\_07 ou plus sur :

<http://java.sun.com/javase/>

Installer cette version et initialiser la variable d'environnement `JAVA_HOME` au répertoire où est installé java.

### **Etape2 : téléchargement d'Eclipse**

Si ce n'est pas déjà fait, télécharger la dernière version d'Eclipse pour J2EE sur :

<http://www.eclipse.org/downloads/>

Sélectionner le lien : `Eclipse IDE for Java EE Developers` et installer la version Ganymède 3.4 en décompressant l'archive qui a été téléchargée.

### **Etape3 : téléchargement de tomcat**

Si ce n'est pas déjà fait, télécharger la dernière version du serveur tomcat 6.0 sur :

<http://tomcat.apache.org/download-60.cgi>

Chaque version de tomcat supporte une spécification différente de JSP, servlet. Nous allons commencer à développer avec JSF 1.2, aussi, il est essentiel d'utiliser une version de tomcat qui traite des JSP2.1 et des servlet 2.5. Sélectionner le lien : `Binary Distributions » Core click` et installer la version `apache-tomcat-6.0.18.zip` ou ultérieure en décompressant l'archive qui a été téléchargée.

Initialisez la variable d'environnement `TOMCAT_HOME` au répertoire où est installé tomcat.

#### **Etape4 : téléchargement JSTL1.2**

L'archive `servlet-api.jar` de tomcat ne contient pas l'API JSTL alors que cela est indispensable pour JSF. Si ce n'est pas déjà fait, télécharger la dernière version de la JSTL sur :

<http://download.java.net/maven/1/jstl/jars/>

Sélectionnez le lien : `jstl-1.2.jar` et déplacez le dans le répertoire des librairies de tomcat, nommé : `%TOMCAT_HOME%/lib`.

Dans le cas où il n'est pas possible d'ajouter des librairies dans le répertoire de tomcat, alors il faudra placer ce fichier dans tous les projets Web qui seront faits et plus particulièrement dans le répertoire local `WEB-INF/lib`.

#### **Etape5 : téléchargement de Mojarra 1.2**

Si ce n'est pas déjà fait, télécharger la dernière version de JSF de Sun, depuis la version 1.2\_07, elle porte le nom de Mojarra qui est plus joli que JSF RI (Reference Implementation) sur :

<https://javaserverfaces.dev.java.net/download.html>

Sélectionner le lien : `1.2_09 binary` et télécharger `mojarra-1.2_09-b02-FCS-binary.zip` en le décompressant dans le workspace.

#### **Etape6 : configuration d'Eclipse**

Démarrez Eclipse et le placer dans le workspace créé initialement.

Effectuez les premiers contrôles :

allez à Window » Preferences » Java » Installed JREs, validez que le jre utilisé est bien celui qui convient (celui qui a été installé ou une version ultérieure). Si ce n'est pas le cas, alors il faut le mettre à jour avec :

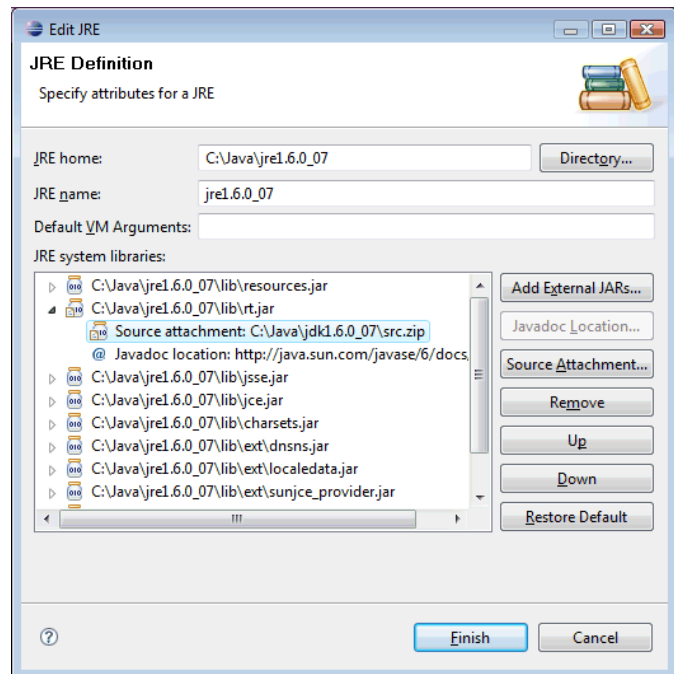
sélectionnez le jre utilisé et cliquez sur Edit.

sélectionnez

C:\Java\jre1.6.0\_07\lib\rt.jar et cliquez sur Source Attachment

cliquez sur External File et naviguez vers %JAVA\_HOME% puis sélectionnez le fichier src.zip, l'ajouter

Ainsi le source code de l'API java sera disponible dans la suite des développements sous Eclipse

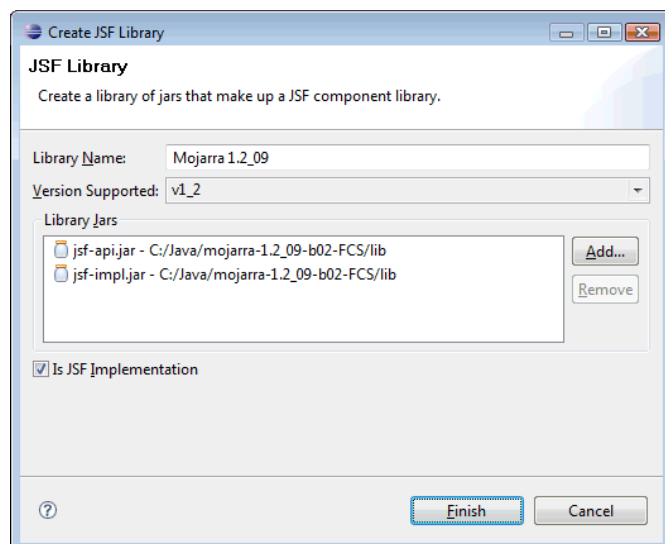


allez à Window » Preferences » Web » JavaServer Faces Tools » Libraries, et cliquez sur New afin de assigner de nouvelles librairies.

Il faut donner un nom unique tel que Mojarra 1.2\_09, initialiser le numéro de version à v1\_2.

En dessous de Library Jars, cliquez sur Add et naviguez vers l'emplacement où a été désarchiver Mojarra : mojarra-1.2\_09-b02-FCS\lib

Sélectionnez les archives jsf-api.jar et jsf-impl.jar et les ajouter, puis contrôlez qu'ils soient pris comme implémentation de référence.



allez à Window » Preferences »  
Web » JSP Files » Editor »  
Templates, et cliquez sur New.

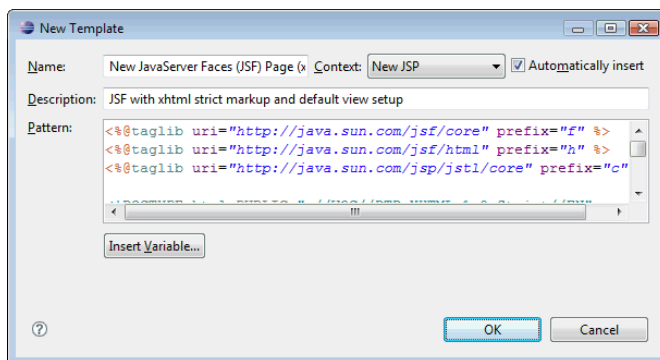
saisir un nom de template :

New JavaServer Faces (JSF)  
Page (xhtml strict)

Sélectionnez comme contexte New JSP  
et saisir une description:

JSF with xhtml strict markup  
and default view setup

Ensuite, copiez le pattern ci-dessous et  
validez sur OK.



```
<%@taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<f:view>
  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <title>Insert title here</title>
    </head>
    <body>
      ${cursor}
    </body>
  </html>
</f:view>
```

allez à Window » Preferences »  
Web » JavaServer Faces Tools  
» Validation, et

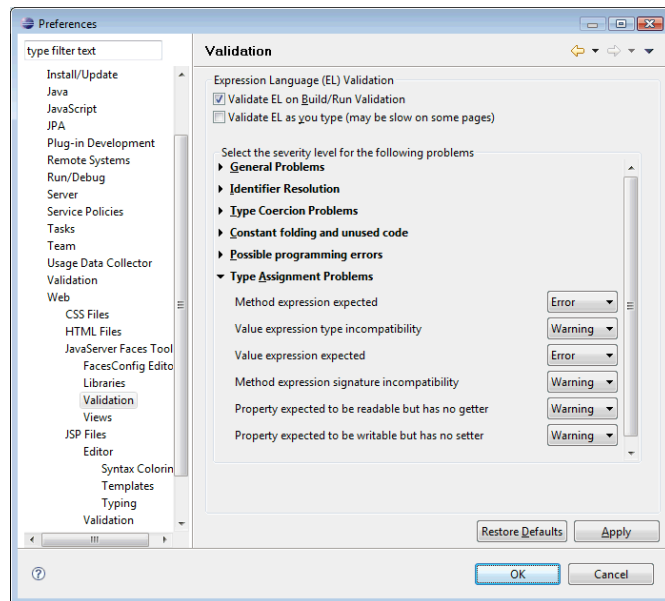
sous Type Assignment Problems  
placez :

Method expression signature  
incompatibility

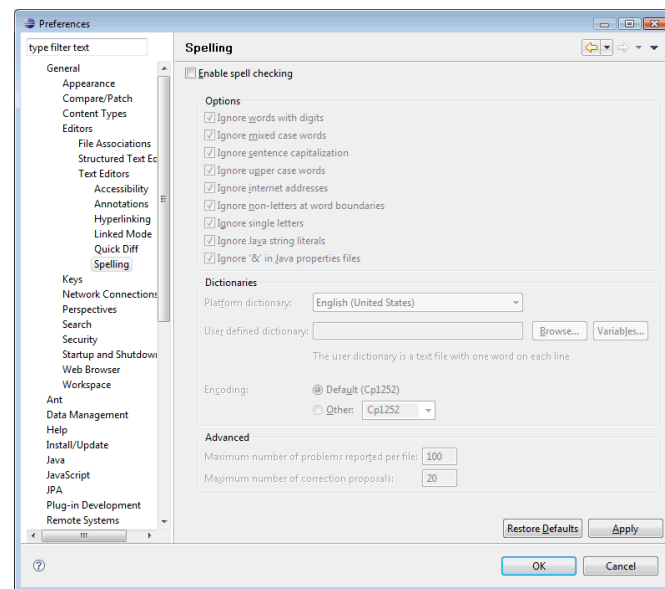
à Warning ou Ignore, si l'on veut que  
les actions JSF puissent retourner void,

sous Type Coercion Problems  
placez :

Unary operation number (et  
boolean) coercion problems à  
Warning ou Ignore, qui permet de  
supprimer des erreurs en utilisant des  
composants graphiques



allez à Window » Preferences »  
General » Editors » Text  
Editors » Spelling, et désactivez  
le contrôle afin de ne pas effectuer de  
contrôle orthographique des fichiers  
web.xml, faces-config.xml et  
autres.



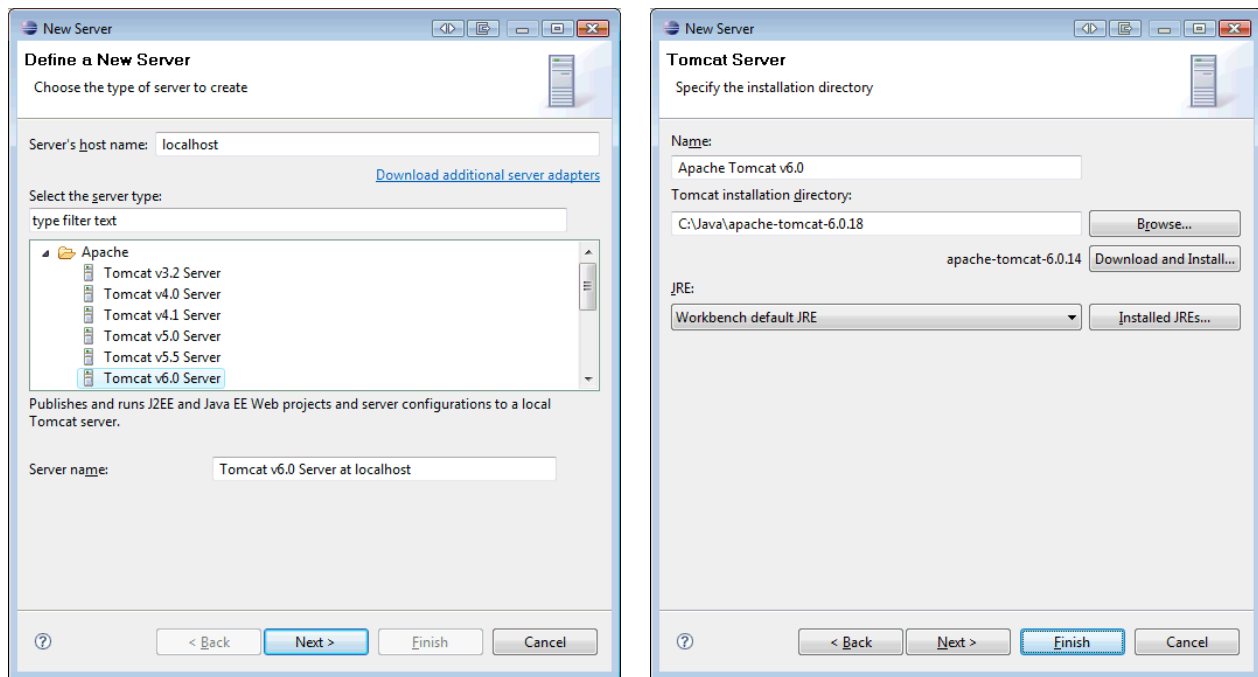
## Etape7 : configuration de tomcat

Dans Eclipse, il faut ouvrir le panneau server (Window » Show View » servers), créez un nouveau serveur (clic droit New » Server). Dans les différentes boites de dialogue, il faut sélectionner le type de serveur : Apache » Tomcat v6.0 Server.

Dans le panneau suivant, naviguer pour sélectionner le serveur tomcat qui a été installé précédemment :

apache-tomcat-6.0.18

Au final, tomcat est intégré dans Eclipse.



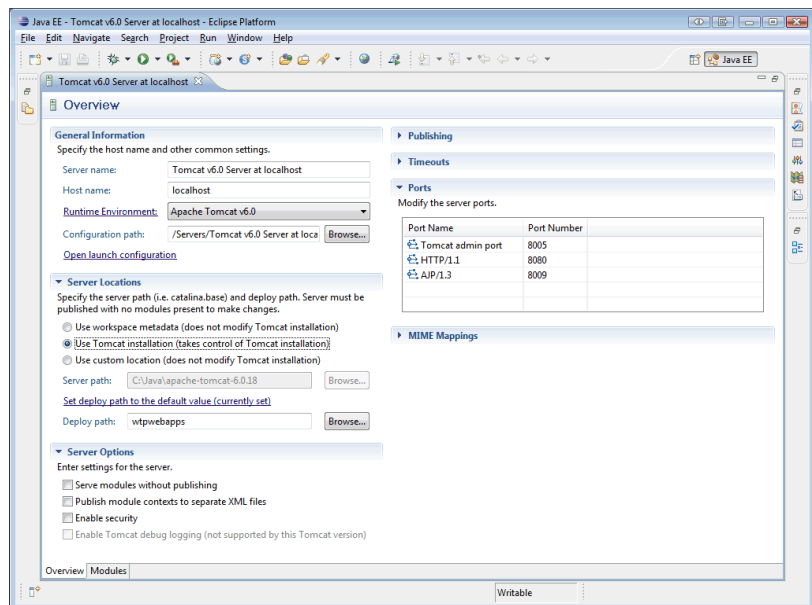
Dans le panneau Servers, un double clic sur le serveur tomcat permet d'accéder à la configuration de ce serveur.

Dans la colonne de gauche, sous Server Locations, sélectionnez Use Tomcat installation,

Ce panneau permet de configurer totalement tomcat et en particulier la page d'accueil de tomcat avec tomcat manager à l'URL :

<http://localhost:8080>

même lorsque l'on travaille sous Eclipse.



Dans la colonne de gauche sous General Information, sélectionnez Open launch configuration,

sélectionnez l'onglet Arguments puis ajoutez dans les arguments de la VM, l'argument en fin de ligne.

-  
`Dorg.apache.el.parser.COERCE_TO_ZERO=false`

Cela permet de lever un erreur dans l'implémentation du langage EL dans tomcat 6.0.

Sauvegardez ces configurations

Il ne reste plus qu'à démarrer le serveur tomcat (bouton droit sur le serveur dans le panneau servers).

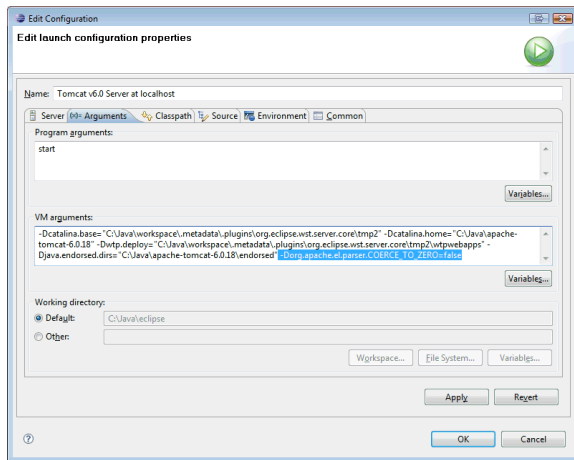
Lorsque le serveur est lancé, ouvrir un navigateur, puis tester l'URL :  
<http://localhost:8080>

Si il y a la moindre erreur, il faut accéder à la configuration du serveur tomcat, en particulier sur le bon numéro de port.

Dans Eclipse, il est possible de lire les logs de tomcat depuis le panneau console.

Dans tous les cas, il est possible de stopper le serveur et de le redémarrer si cela est utile.

## Etape8 : configuration d'un projet Web JSF sous Eclipse



<b>Administration</b>
<a href="#">Status</a>
<a href="#">Tomcat Manager</a>
<b>Documentation</b>
<a href="#">Release Notes</a>
<a href="#">Change Log</a>
<a href="#">Tomcat Documentation</a>
<b>Tomcat Online</b>
<a href="#">Home Page</a>
<a href="#">FAQ</a>
<a href="#">Bug Database</a>
<a href="#">Open Bugs</a>
<a href="#">Users Mailing List</a>
<a href="#">Developers Mailing List</a>
<a href="#">IRC</a>
<b>Miscellaneous</b>
<a href="#">Servlets Examples</a>
<a href="#">JSP Examples</a>
<a href="#">Sun's Java Server Pages Site</a>
<a href="#">Sun's Servlet Site</a>



If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:  
`$CATALINA_HOME/webapps/ROOT/index.html`

where "\$CATALINA\_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the [Tomcat Documentation](#) for more detailed setup and administration information than is found in the INSTALL file.

**NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager".** Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- [users@tomcat.apache.org](mailto:users@tomcat.apache.org) for general questions related to configuring and using Tomcat
- [dev@tomcat.apache.org](mailto:dev@tomcat.apache.org) for developers working on Tomcat

Thanks for using Tomcat!

Powered by



Copyright © 1999-2008 Apache Software Foundation  
All Rights Reserved

Pour créer un nouveau projet, il suffit de sélectionner File » New » Dynamic Web Project.

Il suffit alors de fournir un nom de projet :

ProjetJSF

Puis contrôlez que Target Runtime est initialisé à Apache Tomcat v6.0 et Dynamic Web Module version est initialisé à 2.5.

Sous Configuration sélectionnez JavaServer Faces v1.2 project.

Dans le panneau suivant, il est demandé de choisir le nom de contexte, il est recommandé de le choisir en minuscule :

projetjsf

Les aptitudes JSF tels que les librairies JSF peuvent de nouveau être chargées à ce stade.

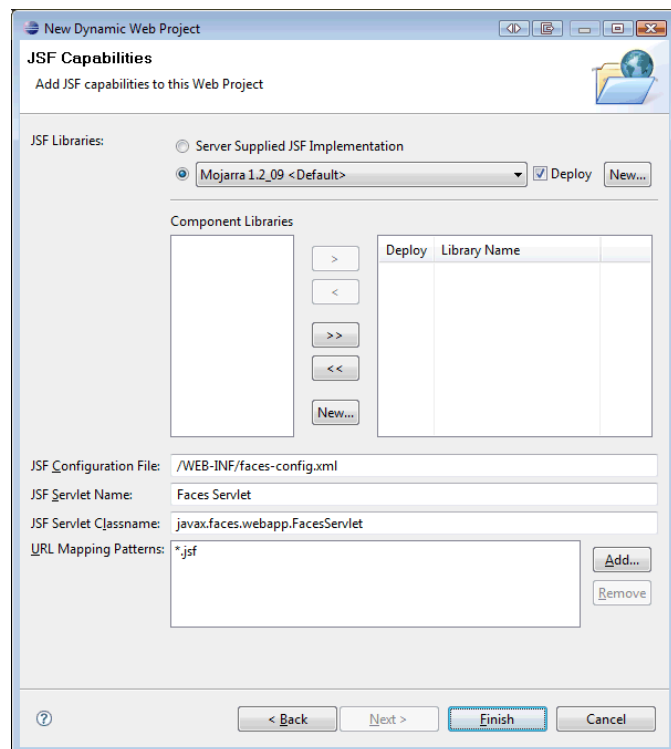
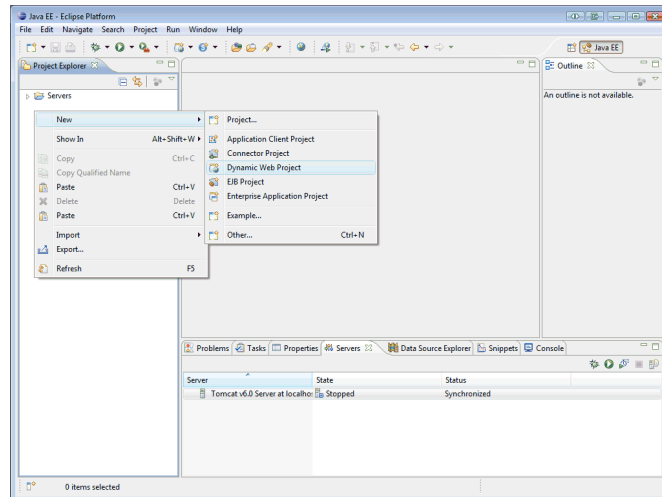
Dans JSF Libraries, sélectionnez Mojarra 1.2\_09 <Default> (qui sont définies dans les préférences).

*Optionnel :*

Dans URL Mapping Patterns, sélectionnez la chaîne /faces/\* et supprimez la.

Ajoutez un nouveau pattern ayant pour valeur \*.jsf

Cliquez Finish.



Pendant la création de ce projet, il sera possible d'ajouter des schémas XML lorsque la plate forme de développement le nécessitera.

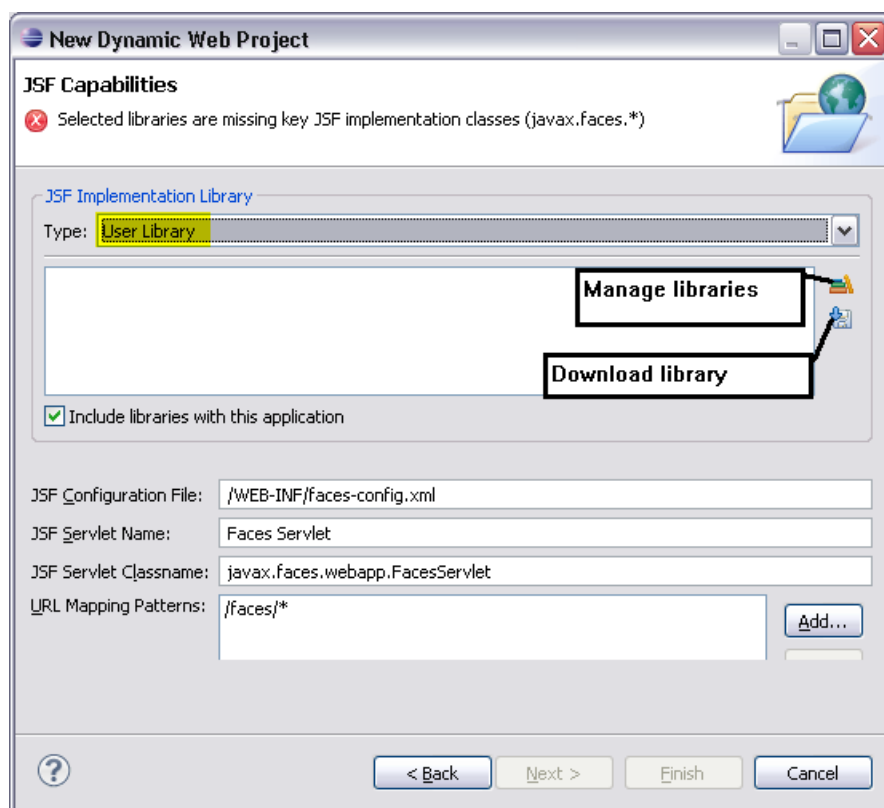


En fin de création un répertoire est créé pour le projet, il contiendra l'ensemble des sources et autres artefacts.

### Exercice7 : Utilisez l'outil de développement pour commencer le projet

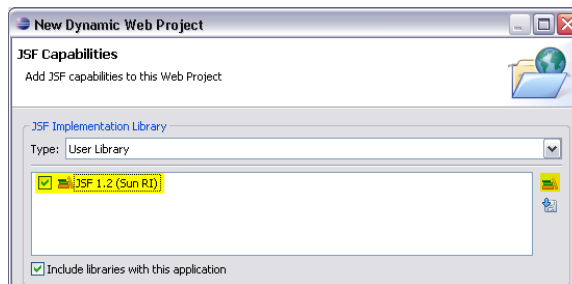
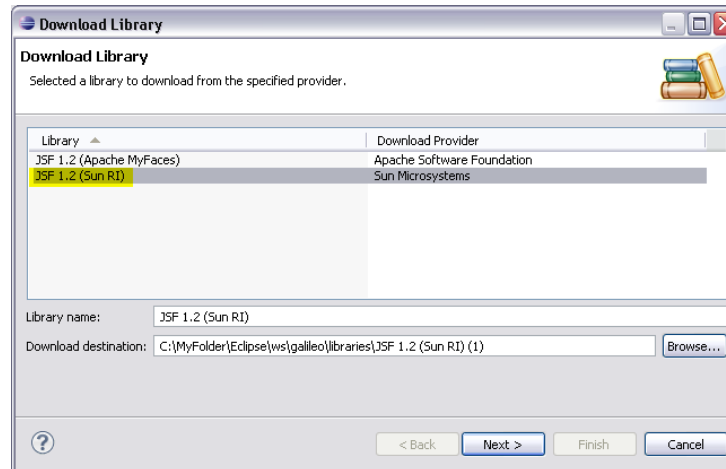
Le but de ce premier programme est de faire une interface de connexion qui sera ensuite enrichi.

*Remarque* : avec une version plus récente d'Eclipse, les boites de dialogues évoluent un peu et lors de la création, il est possible de configurer les librairies différemment.



Dans ce cas, la sélection de « Download library » permet de télécharger les librairies JSF utiles pour les ajouter automatiquement au build-path et au class-path. Il suffit alors de sélectionner la boîte à cocher pour

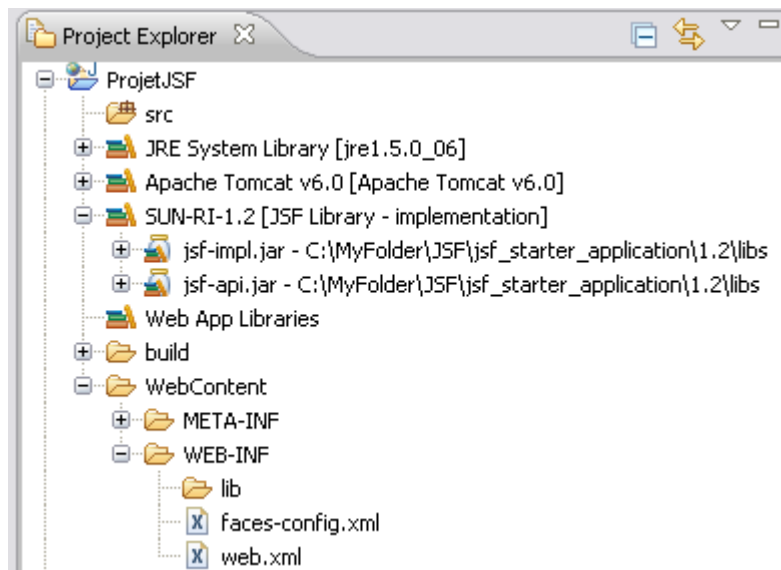
Prendre en compte cette nouvelle librairie utilisateur au sein du projet.



Il est alors utile de refaire la même démarche que l'on souhaiterait ajouter aux projets de ce type.

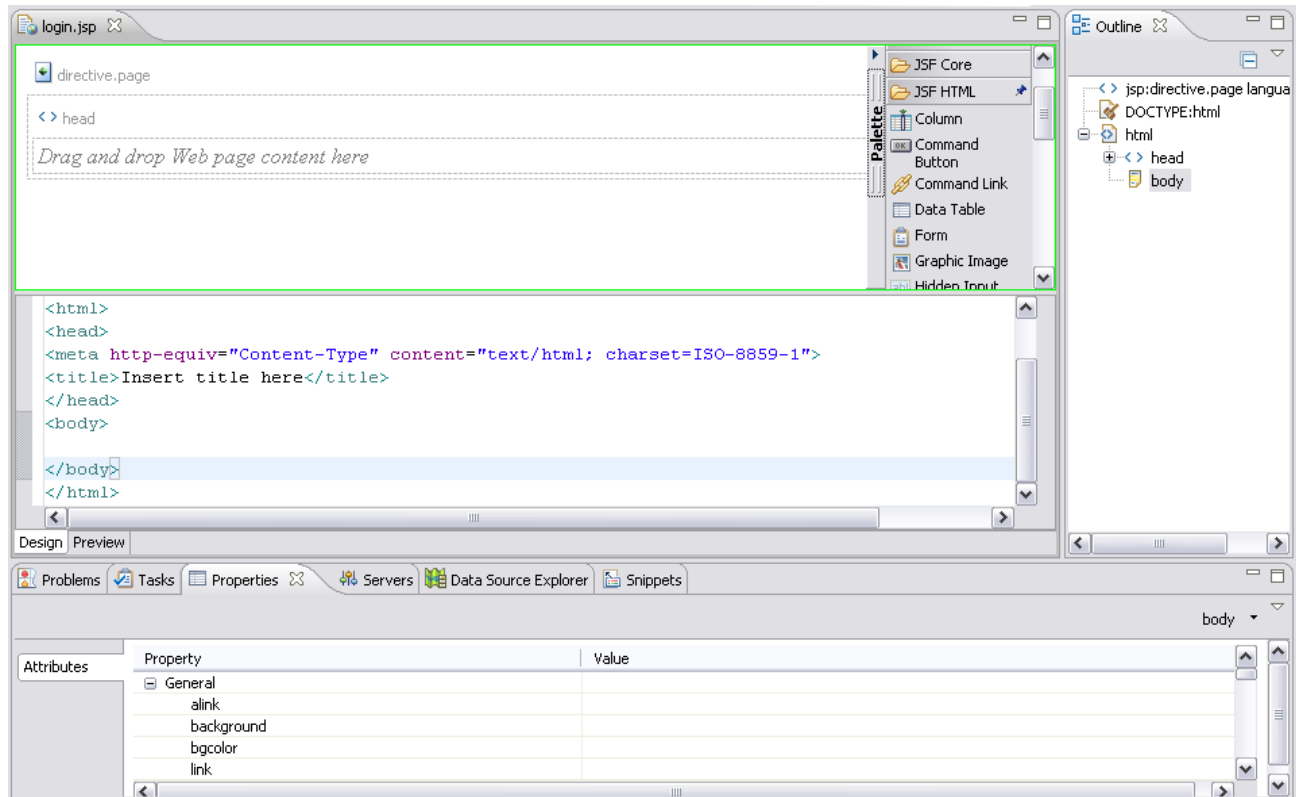
### Etape1

L'application JSF a été créée. Notez que le fichier `web.xml` a été mis à jour avec la servlet Faces et le servlet-mapping, un template de fichier de configuration JSF (`faces-config.xml`) a été créé, et le build path a été mis à jour avec les jars de JSF.



### Etape2: un première page

Il est demandé de créer une première page JSP nommé login.jsp à partir du répertoire Web Content de cette nouvelle application. Dans la page Select Templates du wizard, sélectionnez New JSP (html) template. puis Finish. La page s'ouvrira alors dans un éditeur de page Web assez classique



Ouvrir la vue Properties. Un clic droit dans la fenêtre de conception et depuis le menu contextuel, sélectionnez Show->Properties. La boîte de dialogue "Reading Properties" se masquera.

### Ajoutez un CommandButton au canvas

Dans la palette, sélectionnez la section JSP HTML pour afficher la liste des composants.

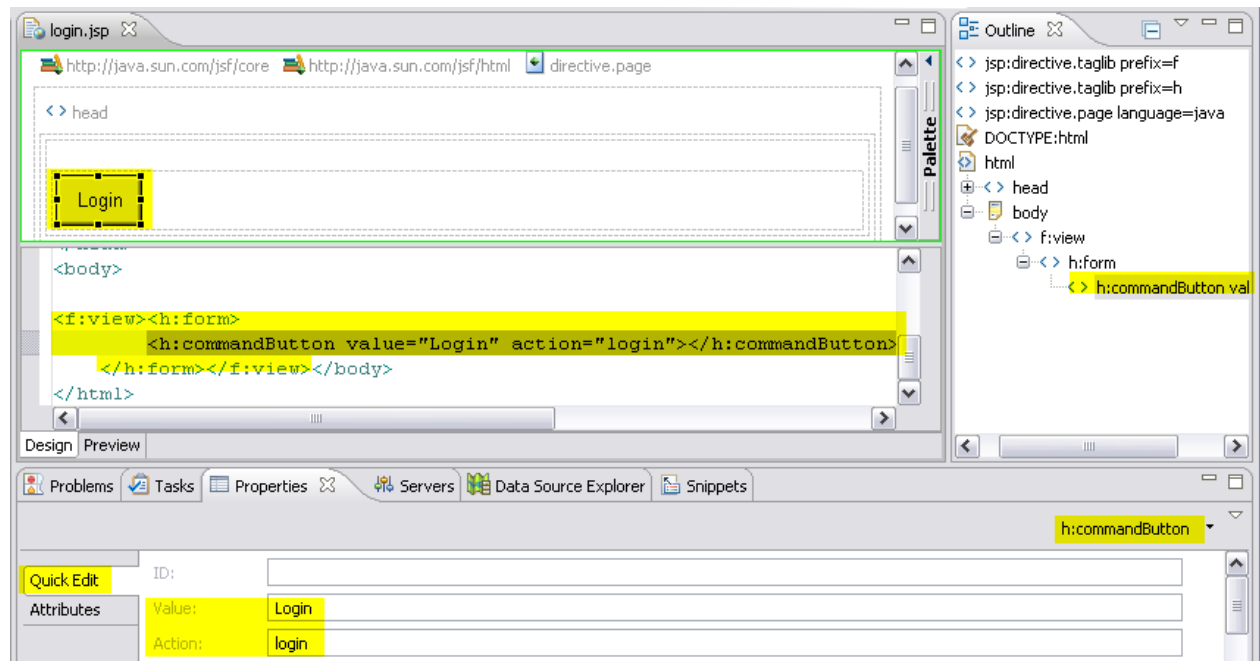
Par drag-and-drop sélectionnez le CommandButton vers le canvas.

Notez que l'éditeur place le CommandButton dans des balises <f:view> et <h:form>.

Dans la vue properties, sélectionnez Quick Edit.

Initialisez l'attribute Value à Login.

Initialisez l'attribut Action à login



## Ajoutez un PanelGrid

Dans la palette, déplacez un PanelGrid vers le canvas. Il y a un feedback différent en fonction des drop sites.

Posez le Panel Grid avant sélectionnez le CommandButton, mais dans la balise Form. Le Panel Grid est créé avec un composant prédéfinie pour une sortie texte: OutputText.

## Modifier les composants dans le PanelGrid

sélectionnez l'élément OutputText Item2 et supprimez le

ajoutez un InputText (Text Input de la palette) après Item1 et avant Item3

supprimez l'élément OutputText, Item 4

ajoutez un InputSecret (Secret Input de la palette) après Item3

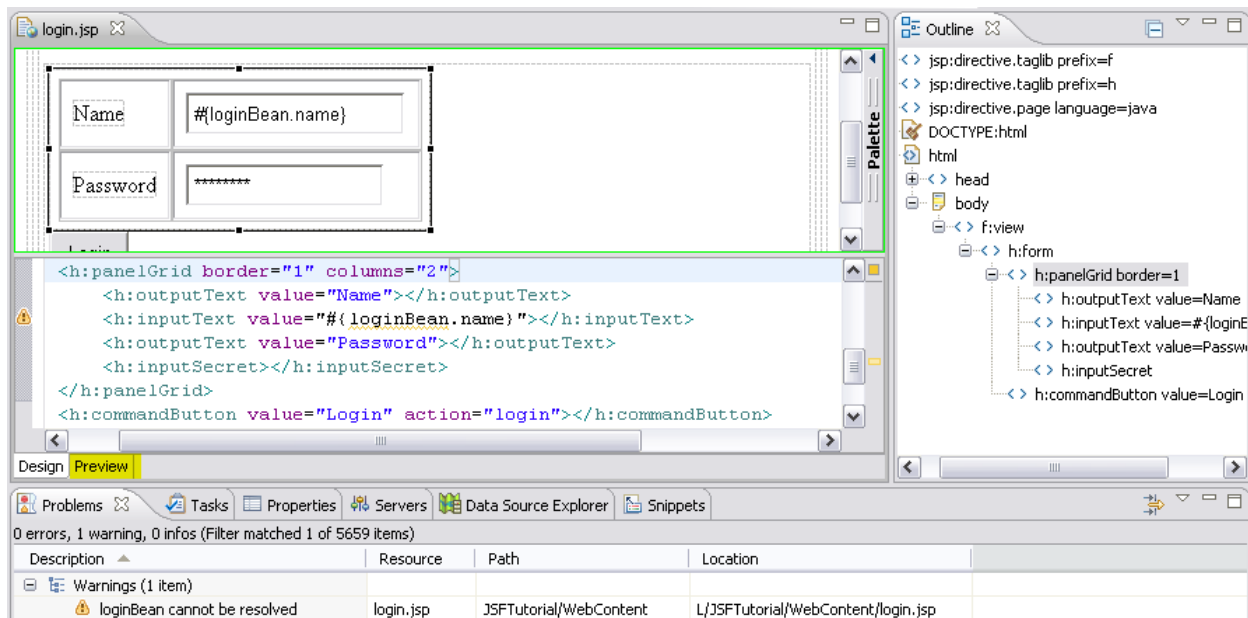
sélectionnez l'élément OutputText, Item1. Changez sa valeur dans la vue Source à Name

sélectionnez l'élément OutputText, Item3. Changez sa valeur dans la vue Source à Password

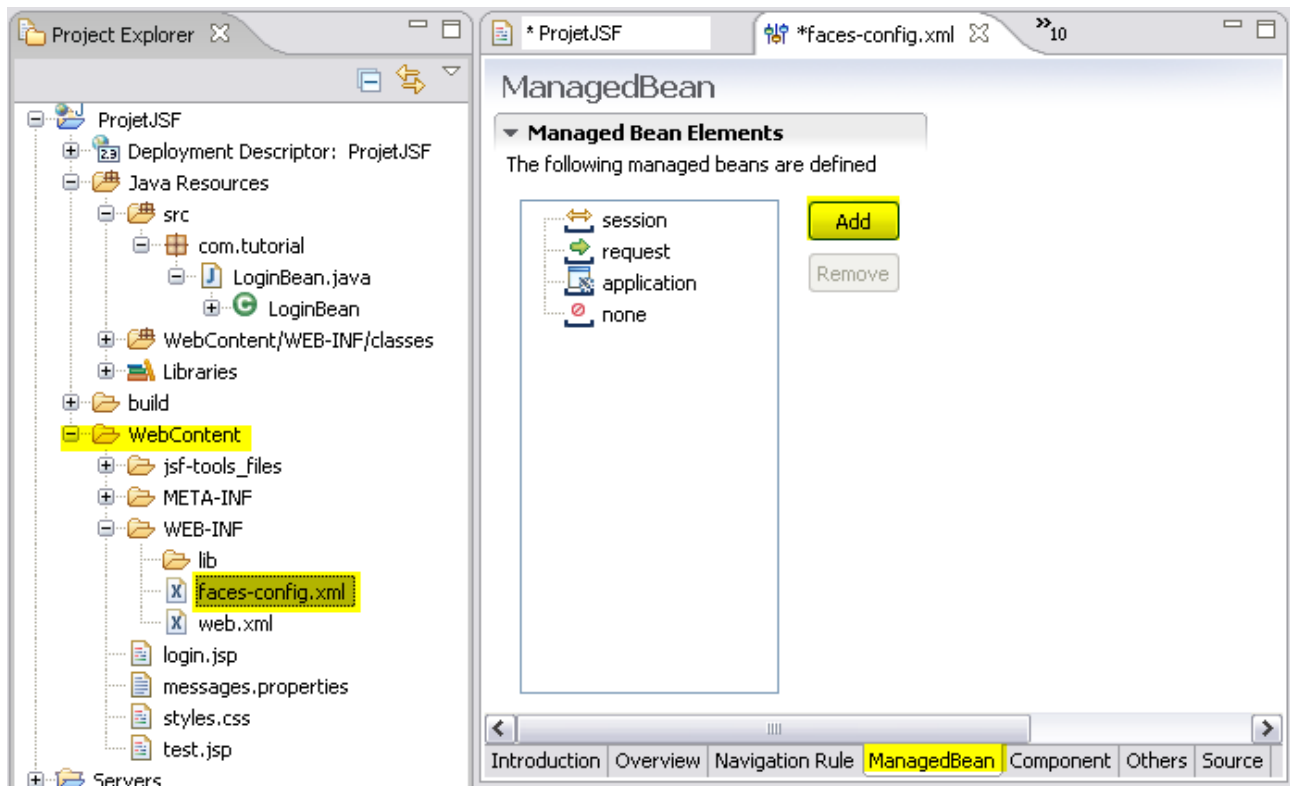
sélectionnez l'onglet Preview pour voir le rendu de la page dans un browser

sélectionnez l'élément InputText tag suivant Name. Dans la vue Property, initialisez l'attribut value à `#{loginBean.name}`.

Sauvez la page. L'éditeur indiquera que la variable, loginBean n'est pas connue.

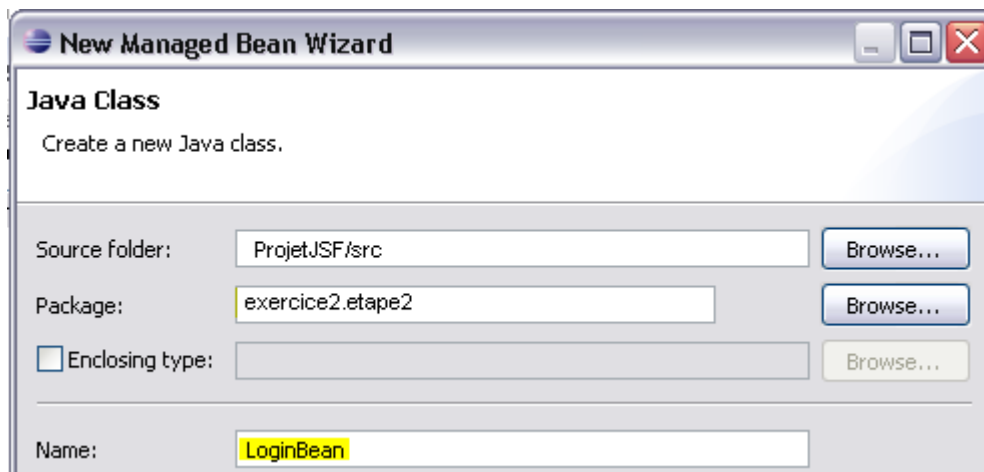


Dans Project Explorer, ouvrir le noeud, ProjetJSF->WebContent. ouvrir faces-config.xml . Cela lancera un éditeur dédié : Faces Configuration. Sélectionnez l'onglet ManagedBean.



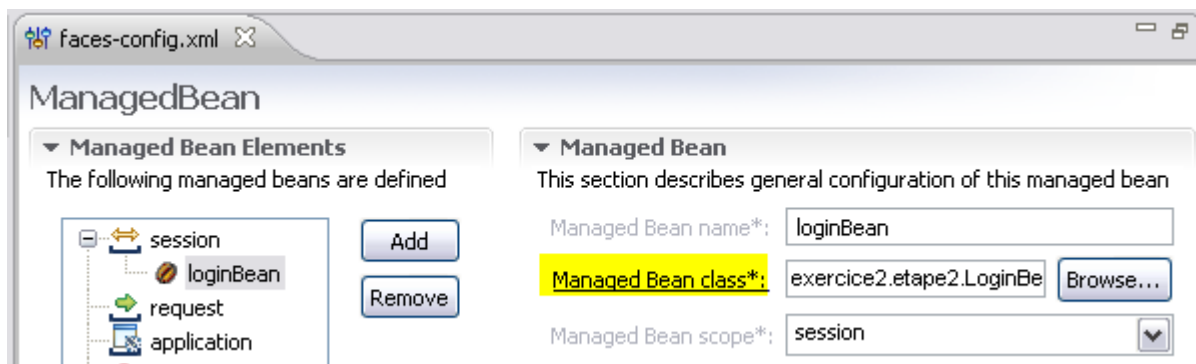
Un clic sur le bouton Add permet de déclencher la création d'un Managed Bean. Sélectionnez l'option, Create a new Java class.

Il faut fournir un nom de package, `exercice2.etape2` et le nom de la classe `LoginBean`.

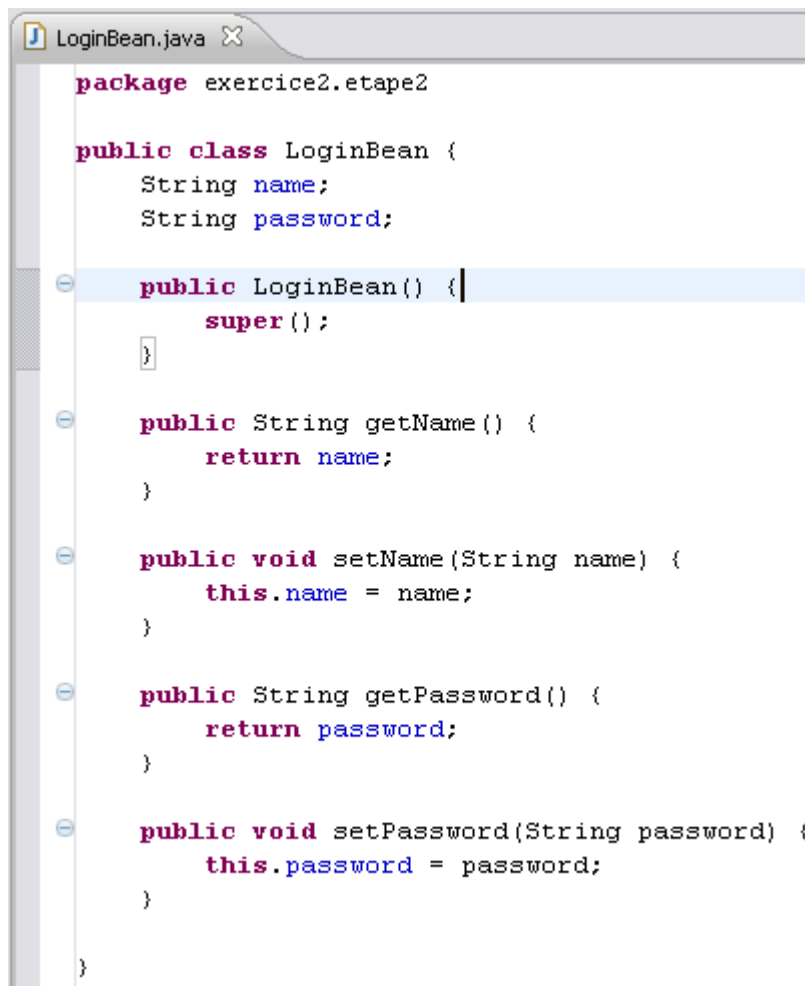


Cela entraîne la création d'une nouvelle classe et l'enregistre comme un composant géré dans le fichier de configuration de JSF. Sauvez dans l'éditeur `Faces Configuration`.

Pour éditer la classe Java, il suffit de sélectionner l'hyperlink, `ManagedBean` dans le panneau `Managed bean`. L'éditeur de code Java traditionnel est lancé.



Constituez la classe comme un `JavaBean`:



```
package exercice2.etape2

public class LoginBean {
    String name;
    String password;

    public LoginBean() {
        super();
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPassword() {
        return password;
    }

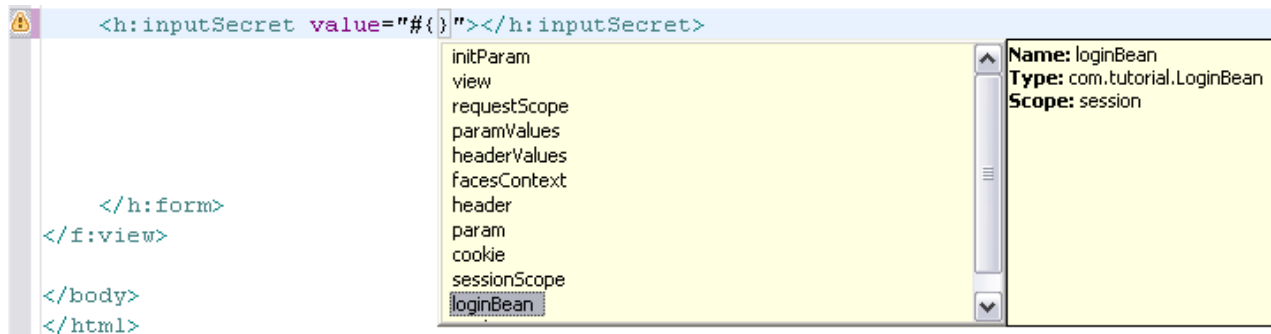
    public void setPassword(String password) {
        this.password = password;
    }
}
```

Dans la vue **Project Explorer**, un clic droit sur la page JSP, `login.jsp` et un menu contextuel apparaît pour sélectionner l'option **Validate**. Notez que la page devrait être considérée comme valide.

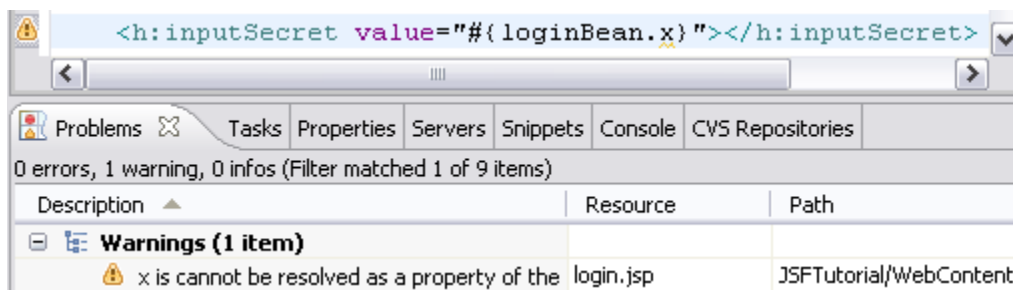
Dans le source de la page affichée dans l'éditeur de page Web, ajoutez la balise JSF,

```
<h:inputSecret value="#{}"/></h:inputSecret>.
```

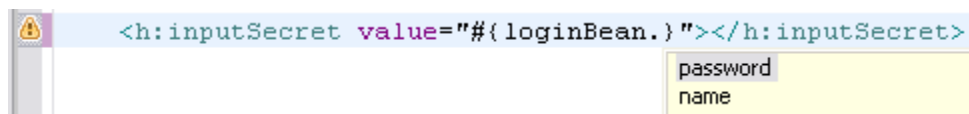
Avec le curseur dans les accolades, tapez **Ctrl+spacebar**. Vous verrez apparaître un pop-up avec la liste de tous les objets implicites plus les managed bean définis précédemment. Sélectionnez le managed bean, `loginBean`.



Entrez le nom de la propriété, soit x, qui n'est pas défini dans le managed bean sauvez les changements. L'éditeur affichera les erreurs en cours.



Supprimez la propriété invalide. tapez Ctrl+spacebar après le '.' dans le nom du bean. Un pop-up menu apparaîtra avec la liste des propriétés possibles dans le managed bean. Sélectionnez password depuis ce menu.



### Etape3 : un premier comportement

Créez la classe Java, `exercice2.etape2.validatePassword` qui implémente l'interface `Validator`.



```

package exercice2.etape2

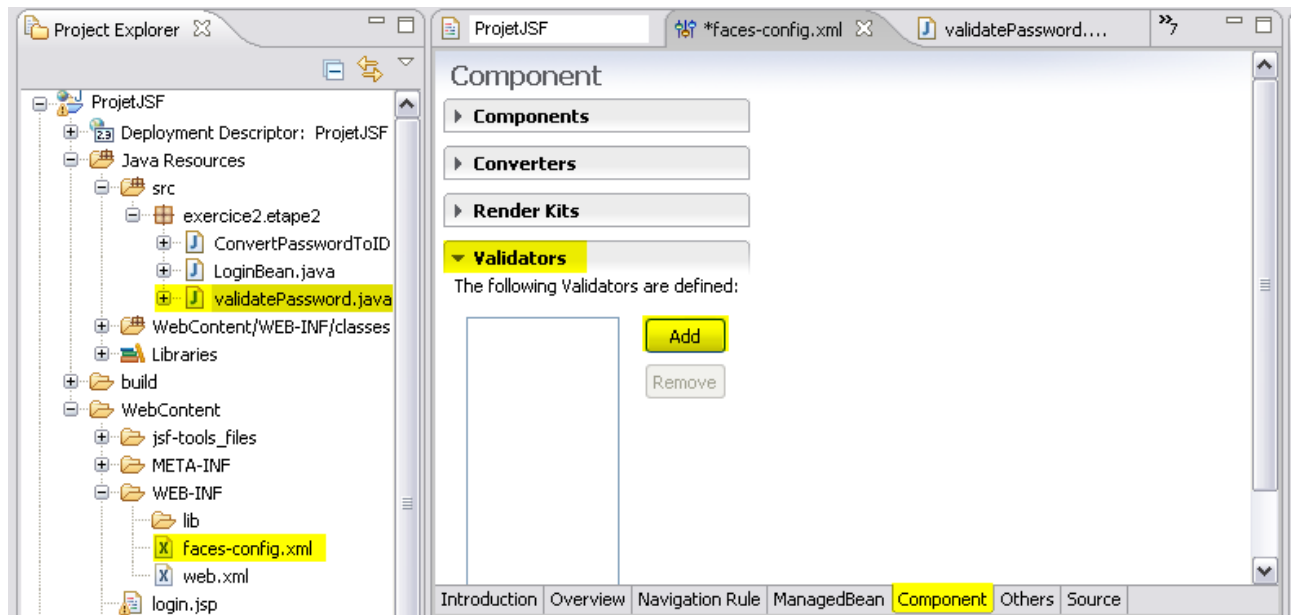
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

public class validatePassword implements Validator {

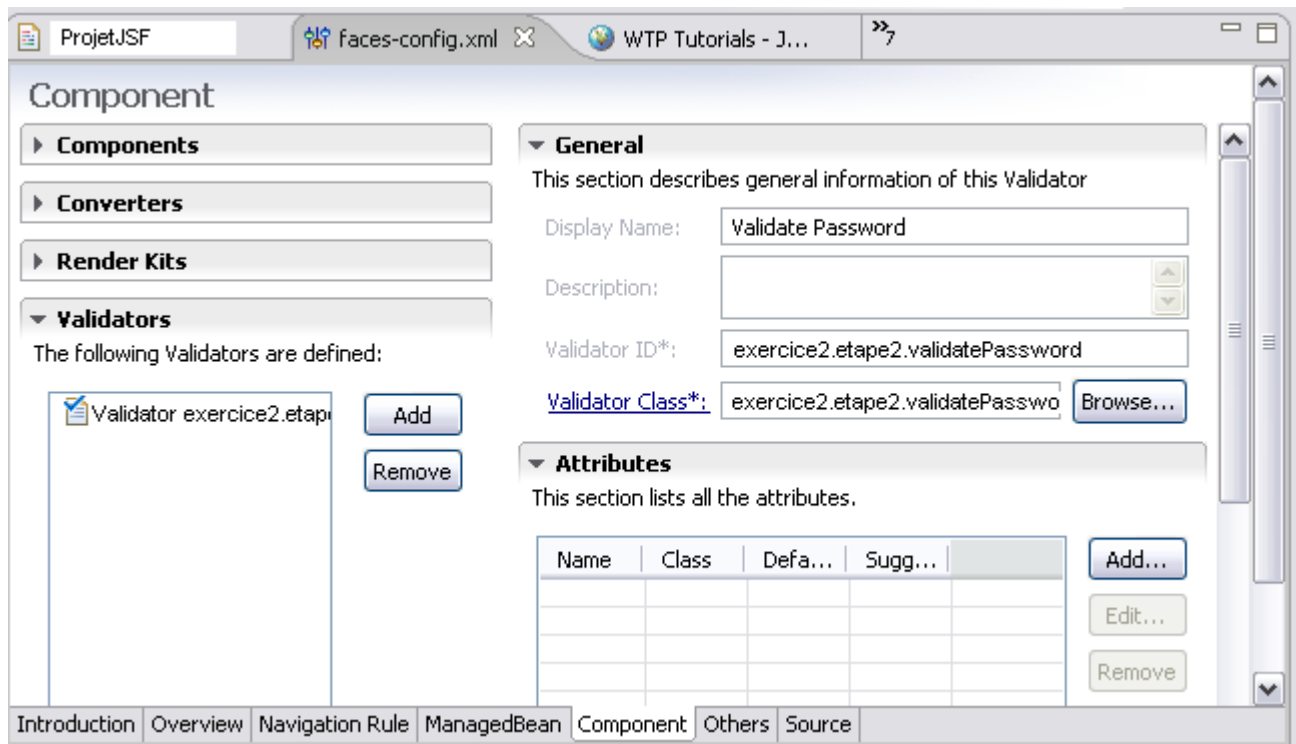
    public void validate(FacesContext arg0, UIComponent arg1, Object arg2)
        throws ValidatorException {
        // TODO Auto-generated method stub
    }
}

```

Dans la vue Project Explorer, ouvrir le noeud, ProjetJSF->WebContent. Editez le fichier faces-config.xml. Sélectionnez l'onglet Component. Allez à la section Validators.



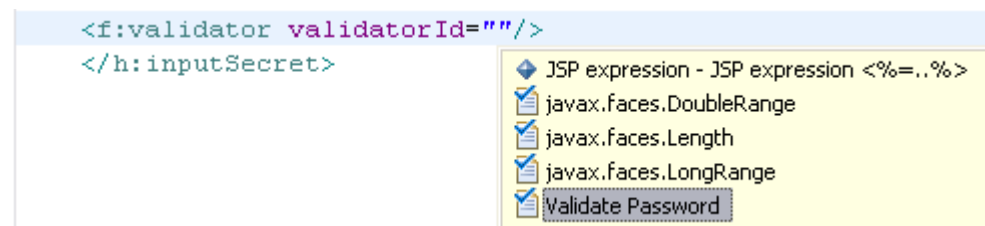
Le bouton Add permet d'ajouter des validateurs. Sélectionnez le bouton Browse associé à la section Validator et sélectionnez la classe exercice2.etape2.validatePassword. Sauvez les changements.



Ajoutez la balise JSF,

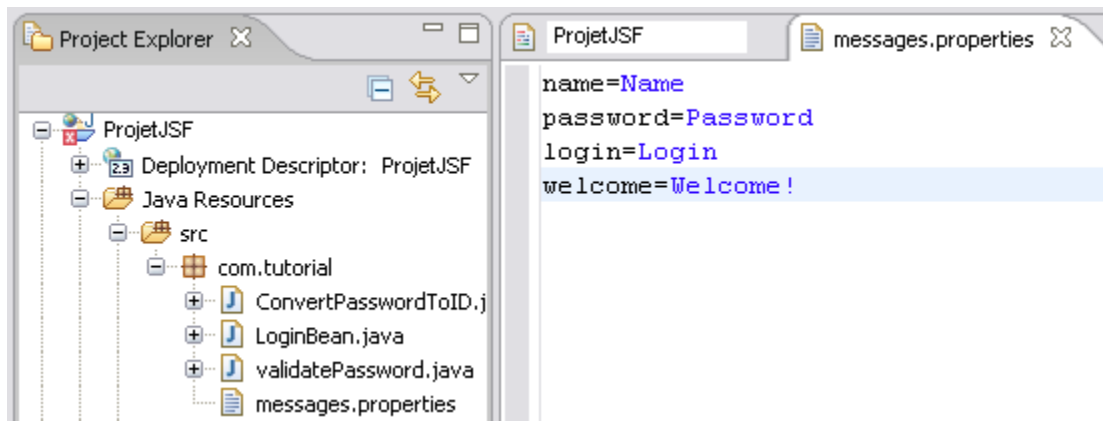
```
<f:validator id=""></f:validator>
```

Avec le curseur dans les double-quotes, tapez Ctrl+spacebar . Un menu pop-up apparaît avec la liste des validateurs standard plus ceux enregistrés dans le fichier faces-config.xml. Sélectionnez Validate Password.



#### Etape4 : definitions les informations de connexion

Il est essentiel dès le début du projet de déplacer les informations fixes dans une fichier externe ou resource bundle. Pour le faire, ajoutez un fichier nommé, messages.properties dans le répertoire exercice2.etape2 . Puis définir les propriétés à l'intérieur afin que chaque ligne soit autonome.



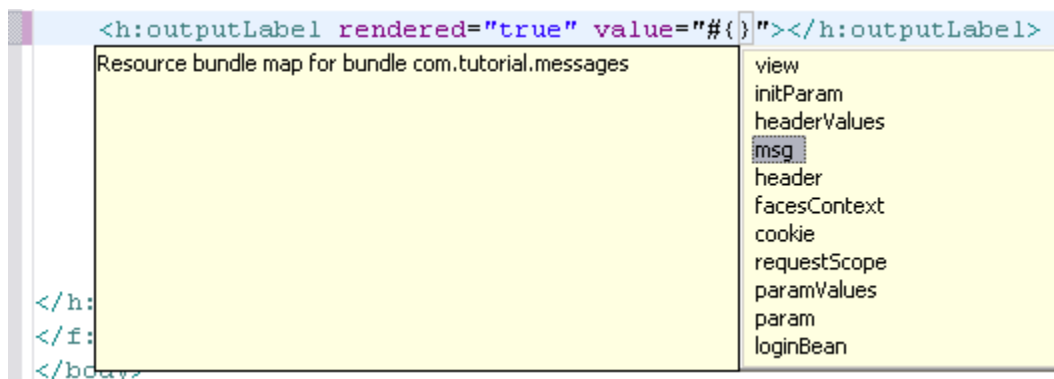
Ajoutez la balise JSF,

loadBundle.

Il est possible de faire un glisser déposer de cette balise depuis la palette JSF Core section.

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=
<title> Projet JSP .</title>
<f:loadBundle basename="exercice2.etape2.messages" var="msg"/>
</head>
```

Supprimez la chaîne de caractères, Name dans l'attribut value de la balise outputLabel. Positionnez le curseur dans les accolades et tapez Ctrl+spacebar . Un menu pop-up apparait avec la liste des variables msg qui ont été définies dans la balise loadBundle. Sélectionnez le.



Entrez un point après msg and et tapez Ctrl+spacebar. Un menu pop-up apparait avec la liste des clés définies dans le fichier messages.properties. Sélectionnez name.

```
<h:outputLabel rendered="true" value="#{msg.}"></h:outputLabel>
<h:messages layout="table"></h:messages>
<h:inputText value="#{loginBean.name}" tabin
<h:inputSecret value="#{loginBean.password.c
```

password
login
name

Répétez ce travail sur l'ensemble de la page.

```
<%@taglib uri="http://java.sun.com/jsf/core" prefix="f"%><%@taglib
    uri="http://java.sun.com/jsf/html" prefix="h"%><%@ page language="java"
    contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title> Projet JSF </title>
</head>
<body>

<f:view>
    <f:loadBundle basename="exercice2.etape2.messages" var="msg" />
    <h:messages></h:messages>
    <h:form>
        <h:panelGrid border="1" columns="2">
            <h:outputText value="#{msg.name}"></h:outputText>
            <h:inputText value="#{loginBean.name}"></h:inputText>
            <h:outputText value="#{msg.password}"></h:outputText>
            <h:inputSecret value="#{loginBean.password}">
            <f:validator validatorId="exercice2.etape2.ValidatePassword"/>
            </h:inputSecret>
        </h:panelGrid>
        <h:commandButton value="Login" action="login"></h:commandButton>
    </h:form>
</f:view>
</body>
</html>
```

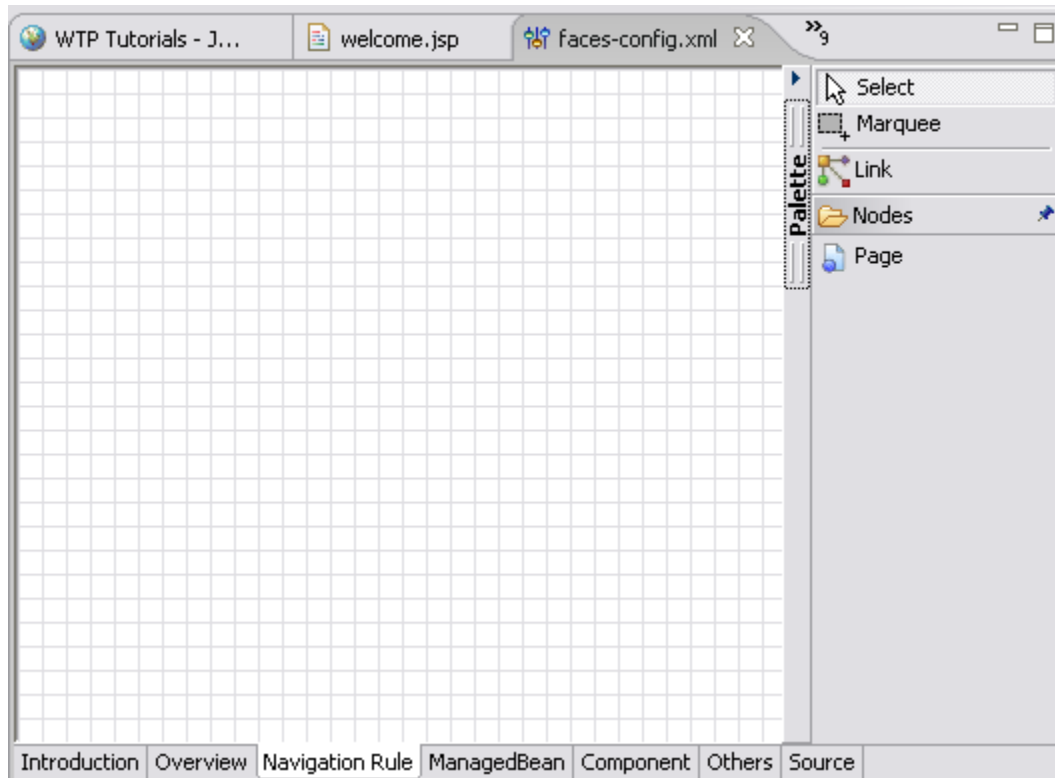
## Etape5 : faire la navigation

Créez une nouvelle page JSF, welcome.jsp comme ci-dessous.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome</title>
<f:loadBundle basename="exercice2.etape2.messages" var="msg"/>
</head>
<body>
<f:view>
    <h:form>
        <h:outputLabel value="#{msg.welcome} #{loginBean.name}"></h:outputLabel>
    </h:form>
</f:view>
</body>
</html>
```

Editeur de ressources dédiées JSF

Double-cliquez sur le fichier `faces-config.xml`. Basculez sur l'onglet Navigation.



Ajoutez les pages `login.jsp` et `welcome.jsp` dans l'onglet Navigation. Sélectionnez depuis la palette l'élément `Page` et glissez le sur l'onglet Navigation. Choisir la page dans la boîte de dialogue.

Connectez les 2 pages. Cliquez sur l'élément `Link` dans la Palette, Sélectionnez la page `login` et dessiner une ligne vers la page `welcome`. Sélectionnez la ligne dans l'onglet Navigation et dans la vue `Property` initialisez la valeur de `from-outcome` à `login`.

