JAVA WEB PROGRAMMING

Model View Controller - JSF

En este documento analizaremos el desarrollo de una pequeña aplicación JavaServer Face. JSF un otro framework MVC incorporado a J2ee 1.4 que amenaza con derrocar a Struts.

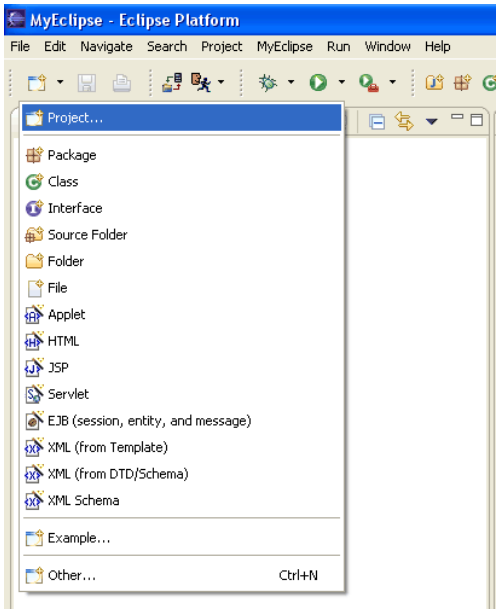
Con JSF podemos relacionar una página JSP con un Bean Java. El bean es el responsable de mantener la información del formulario de la página y de procesar esa información cuando el usuario la submitee.

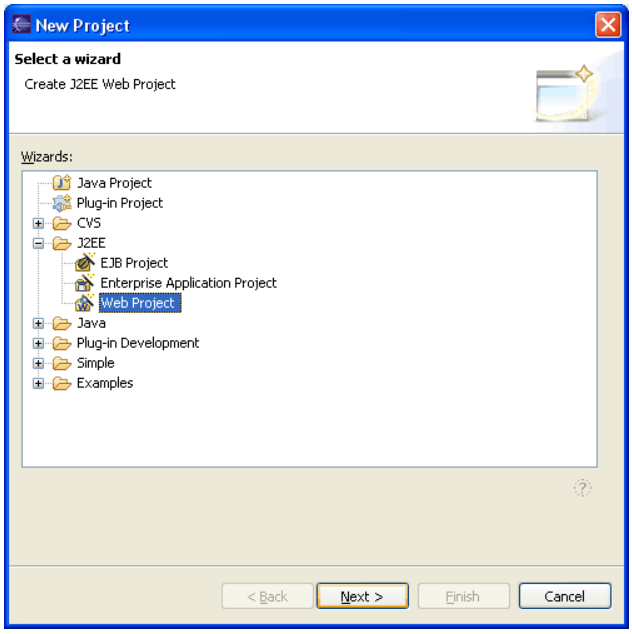
El formulario debe contener "**action commands**" que no son mas que botones submit. Cada action command esta relacionado con un método del bean que JSF invocará cuando el usuario presione el botón.

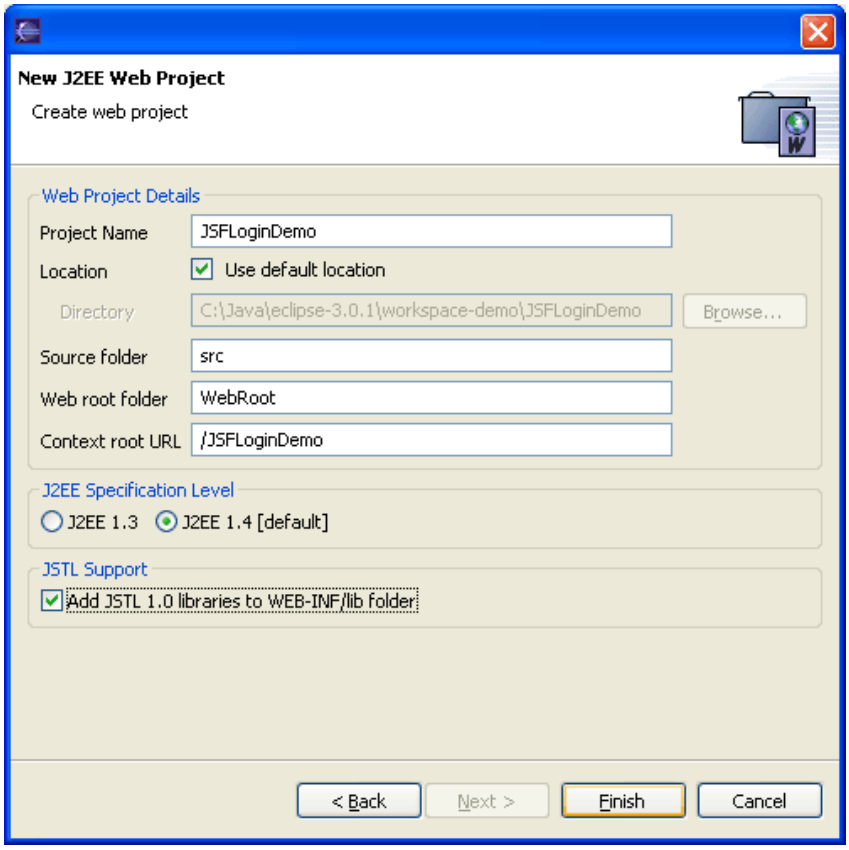
Así, cada formulario puede tener varios botones (action command) cada uno relacionado a un método del bean. Estos métodos devuelven un String. "ok", "error", etc. Luego, en un archivo (faces-config.xml) se define para cada página, según el retorno del método del bean que se haya invocado cual es la siguiente página que se debe mostrar.

Por último, JSF también permite levantar las etiquetas de la página JSP desde un archivo de propiedades. Así, al no hardcodear textos en la página es muy fácil su internacionalización.

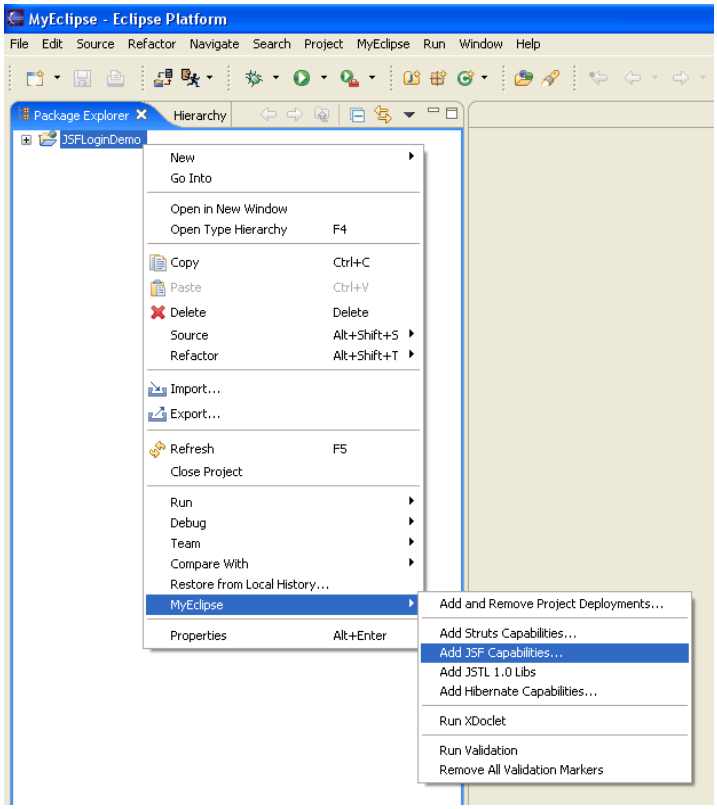
Creación y configuración de un nuevo proyecto web para utilizar JSF

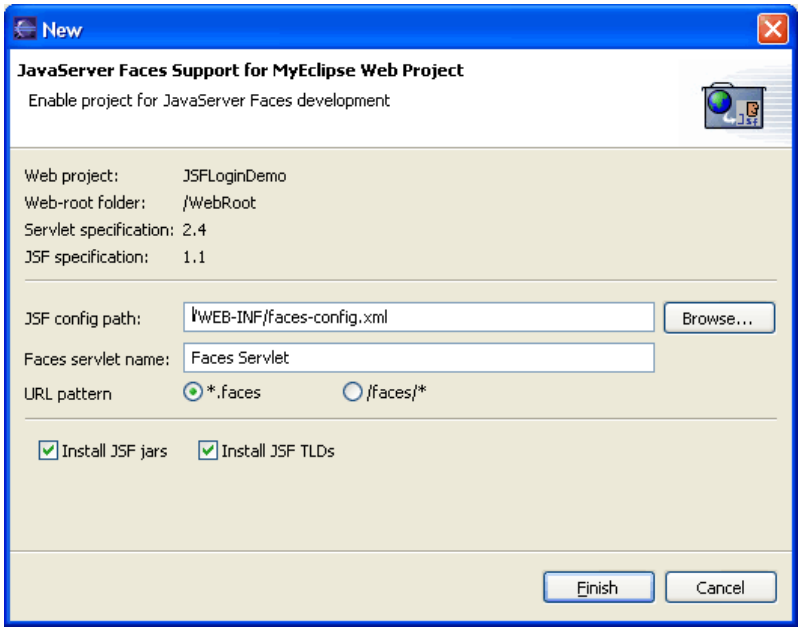






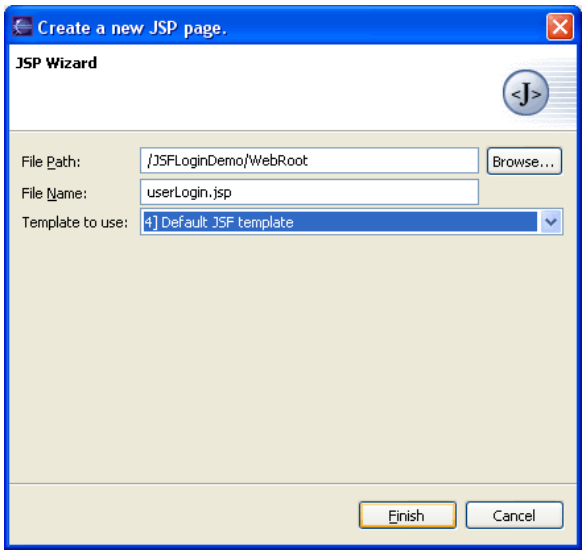
Agregando capacidades de JSF al proyecto



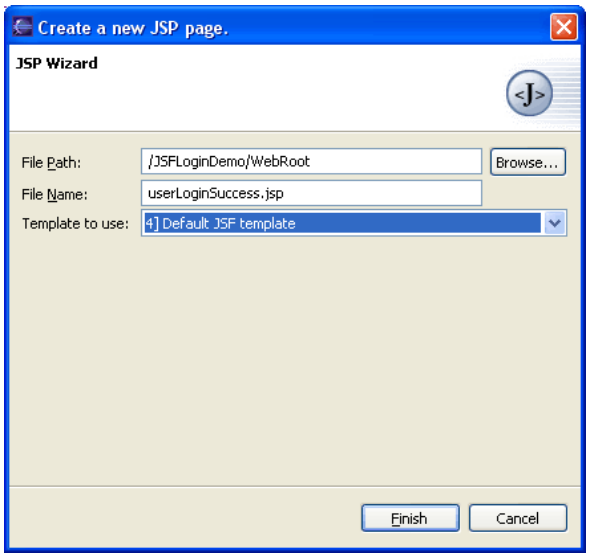


Creamos dos Páginas JSP

userLogin.jsp



userLoginSuccess.jsp



userLogin.jsp

con **<f:loadBundle...>** cargamos el archivo de etiquetas (labels). En este archivo definimos todas las etiquetas que vamos a mostrar en la página. Esto nos permite NO harcodear y de esta manera es muy simple la internacionalización de nuestra aplicación.

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

<html>
<body>

<f:view>
<f:loadBundle basename="com.jsfdemo.MessageBundle" var="bundle"/>

<h:form id="loginForm">
<table>
<tbody>
<tr>
<td><h:outputText value="#{bundle.user_name_label}" /></td>
<td>
```

Cada campo (o control) del formulario se corresponde con un atributo en un bean Java (en este caso la clase será UserBean)

```
<h:inputText id="userName" value="#{UserBean.userName}" required="true">
            <f:validateLength minimum="1" />
</h:inputText>


</td></tr><tr>
<td><h:outputText value="#{bundle.user password label}" /></td>
<td>


<h:inputSecret id="password" value="#{UserBean.password}" required="true">
            <f:validateLength minimum="3" />
</h:inputSecret>
```

Los mensajes se muestran con este tag

```
</td></tr><tr>
<td colspan="2">
            <h:messages style="font-weight: bold; color: #FF0000;" />
</td>
</tr><tr>



<td align="right" colspan="2">

<h:commandButton id="submit" action="#{UserBean.loginUser}"
       value="#{bundle.login_button_label}" />

</td> </tr>

</tbody>
</table>
</h:form>
</f:view>
</body>
</html>
```

El botón submit (action) esta relacionado con un método (en este caso loginUser() en el bean. Cuando se presione el botón se invocará el método correspondiente

El la página anterior podemos ver los siguientes puntos:

1. Los tags de JSF se dividen en **core** y **html**. Estos últimos son los que utilizamos para armar los formularios de datos para el usuario.

2. Levantamos el archivo de etiquetas **com.jsfdemo.MessageBundle** (asociado a la variable **bundle**). Luego para utilizar una etiqueta la sintaxis es: **`#{bundle.nombre_de_la_etiqueta}`**

3. Cada control del formulario esta asociado a un atributo del bean (en este caso UserBean).

4. Con el tag <h:messages...> mostramos todos los mensajes de error. Los mensajes de error se acumularán cuando se invoquen los métodos de procesamiento del bean (userBean)

5. Por último, se define un **commandButton** con un atributo **`action="#{UserBean.loginUser}"`**. Este atributo relaciona el botón con el método loginUser() del bean UserBean.

Programamos el Bean

UserBean.java

```
package com.jsfdemo.bean;

import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;

public class UserBean
{
    private String userName = null;
    private String password = null;

    // setters y getters
    public String getPassword(){ return password; }
    public void setPassword(String password){ this.password = password; }
    public String getUserName() { return userName; }
    public void setUserName(String userName) { this.userName = userName; }

    public String loginUser()
    {
        if(getUserName().equals("myeclipse") && getPassword().equals("myeclipse"))
        {
            return "success";
        }

        FacesContext facesContext = FacesContext.getCurrentInstance();

        String mssg="Usuario y/o Password incorrectos";

        FacesMessage facesMessage = new FacesMessage(mssg);

        facesContext.addMessage("loginForm", facesMessage);

        return "failure";
    }
}
```

Un atributo por cada campo del formulario de la página JSP

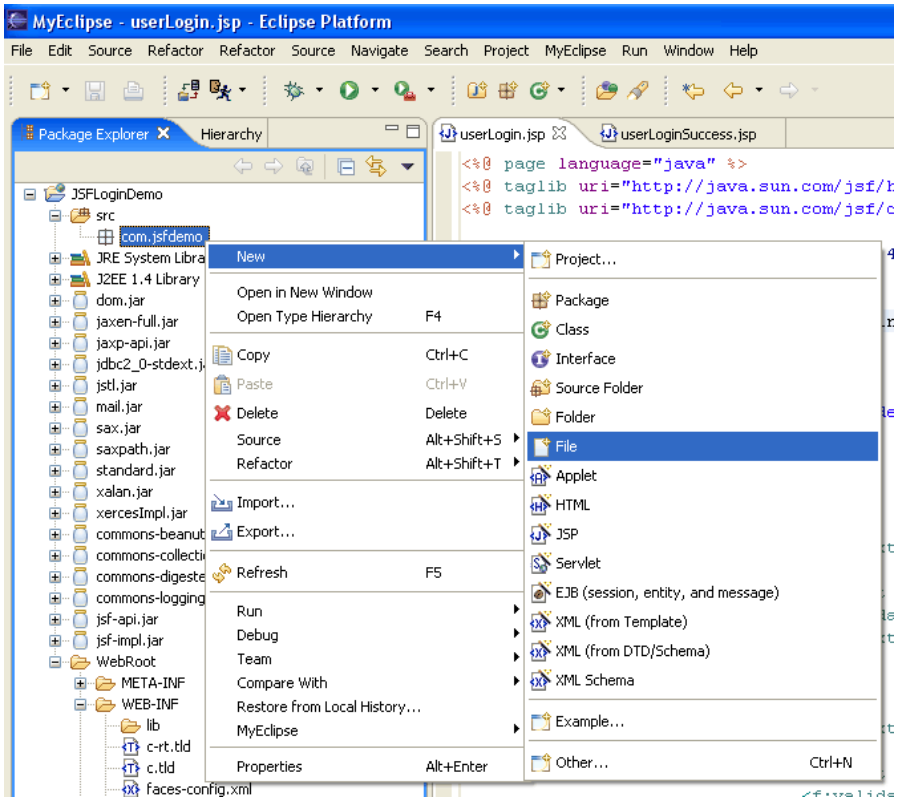
Método relacionado con el submit (action command) del formulario

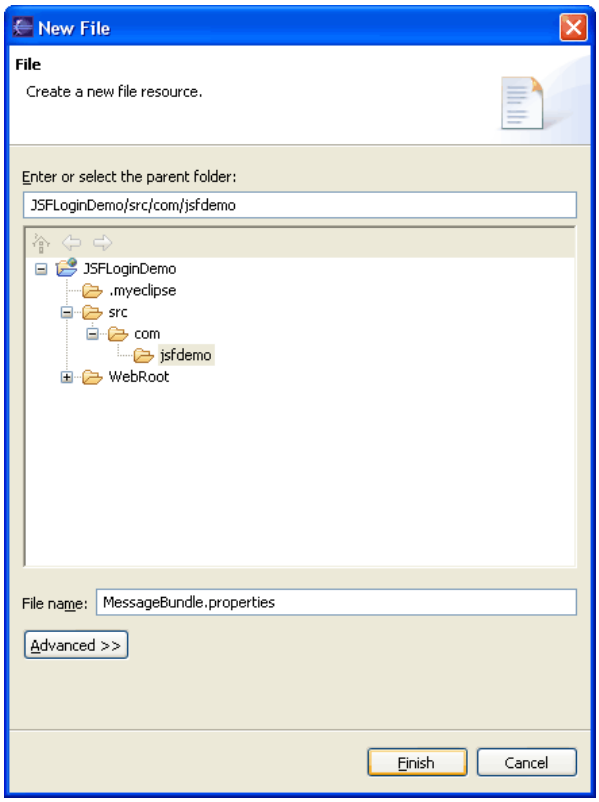
Accediendo al **FacesContext** podemos pasar mensajes para que se muestren en la página JSP

El método retorna un String. Este string es análogo al forward de Struts. Luego definiremos para cada uno de estos strings cual es la siguiente pagina JSP que se debe mostrar

Como podemos ver, cada página tiene asociado un Bean. El bean contiene y procesa los datos del formulario de la página JSP. Comparándolo con Struts podríamos decir que el bean cumple las funciones del Form y del Action. El retorno String de los métodos serían el forward de Struts.

Ahora vamos a crear el archivo de mensajes





El archivo debe estar en un package. En este caso lo ubicaremos en:

com.jsfdemo

MessageBundle.properties

```
user_name_label=User Name:
user_password_label=Password:
login_button_label=Login
```

El archivo faces-config.xml

Ahora tenemos que configurar el WEB-INF/**faces-config.xml**. En este archivo definimos las reglas de navegación. Recordemos que los métodos del bean retornan un String. Aquí definiremos a que página ir según el string que retorne el bean. también en este archivo vamos a definir los beans que utilizaremos en las páginas JSP.

faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE faces-config PUBLIC "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN"
"http://java.sun.com/dtd/web-facesconfig_1_1.dtd">

<faces-config>

      <navigation-rule>
            <from-view-id>/userLogin.jsp</from-view-id>
            <navigation-case>
                  <from-outcome>success</from-outcome>
                  <to-view-id>/userLoginSuccess.jsp</to-view-id>
            </navigation-case>
            <navigation-case>
                  <from-outcome>failure</from-outcome>
                  <to-view-id>/userLogin.jsp</to-view-id>
            </navigation-case>
      </navigation-rule>

      <managed-bean>
            <managed-bean-name>UserBean</managed-bean-name>
            <managed-bean-class>com.jsfdemo.bean.UserBean</managed-bean-class>
            <managed-bean-scope>session</managed-bean-scope>
      </managed-bean>

</faces-config>
```

Este documento es una adaptación del tutorial publicado en:
http://www.myeclipseide.com/images/tutorials/JSF/SFLoginDemoTutorial.html