Introducción

En este laboratorio se muestra cómo realizar una aplicación JSF que implementa un caso de uso que permite hacer Login.

Objetivos

Los objetivos de este laboratorio son los siguientes:

- Desarrollar una aplicación JSF utilizando la herramienta Eclipse.
- Comprender cómo funciona una aplicación Web basada en el patrón arquitectural MVC (Modelo-Vista-Controlador).

Tareas a realizar

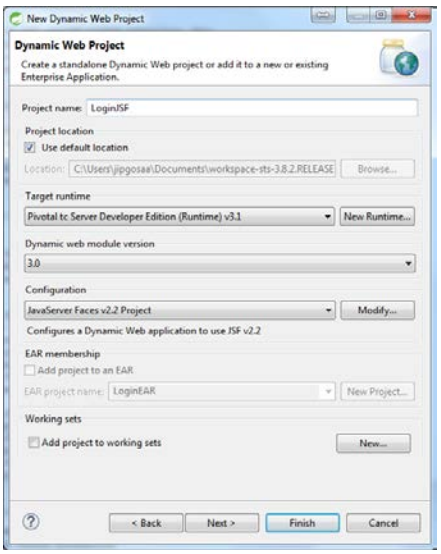
1. Crear y ejecutar un proyecto Java JSF con sus vistas (JSF) y modelos (beans)

Para ello hay que realizar los siguientes pasos:

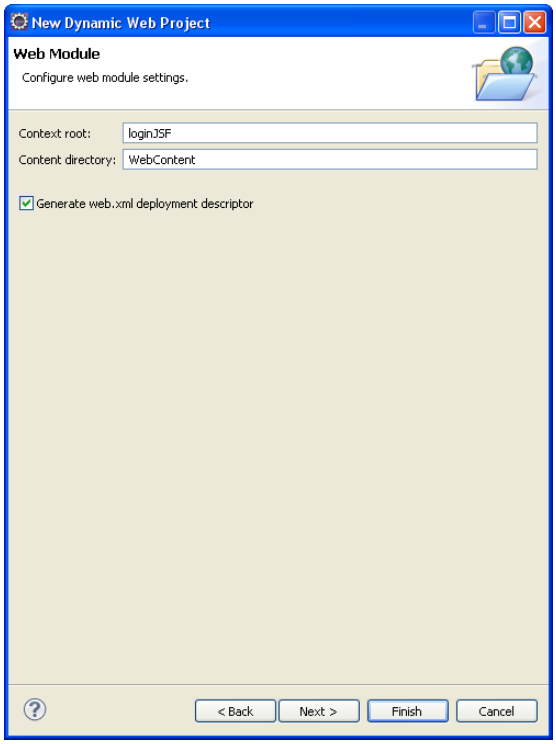
1.1.- Crear un proyecto JSF, junto con el fichero de configuración por defecto web.xml

Seleccionar: File => New => Project => Web => Dynamic Web Project

Seleccionar el contenedor web (en “Target runtime”) que se instala con el Spring Tool Suite (el servidor *Pivotal tc Server*, que es compatible con Apache Tomcat), la configuración JavaServer Faces v2.0, y dar un nombre al proyecto (LoginJSF, por ejemplo). La configuración JavaServer Faces v2.X, aparece si se selecciona la versión 3.0 en “Dynamic web module version”. Si no aparece el Runtime, entonces habrá que crearlo pulsando “New Runtime” y seleccionando el directorio donde se haya descargado el código del servidor Pivotal tc Server (junto al código del Spring Tool Suite).

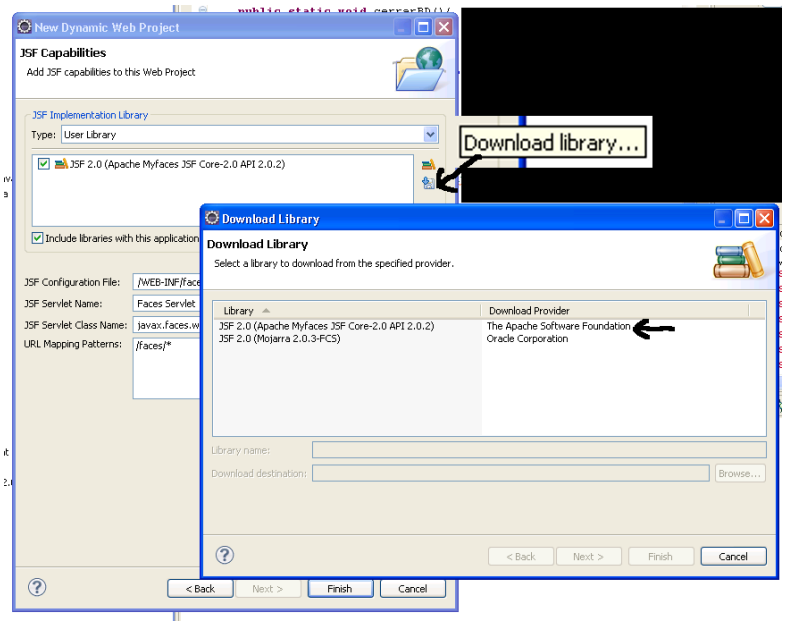


Tras pulsar Next 2 veces, activar la casilla “Generate web.xml”



Continuar con “Next” para ver las opciones por defecto, y aceptarlas finalmente con “Finish”.

Nota: Si en el paso donde se muestra la ventana de JSF Capabilities, donde se indican las librerías JSF, no aparece ninguna, entonces se pueden descargar (pulsando el icono anotado con “Download library” en la siguiente figura), para que el Eclipse busque las implementaciones disponibles de JSF y se pueda escoger una (por ejemplo, Myfaces, proporcionada por Apache Software Foundation).



1.2.- Crear las páginas JSF siguientes **(Capa de presentación / Vistas):**

Seleccionar: New => XHTML Page (o bien Other => Web => HTML File poniendo extensión .xhtml) => Dar los nombres: Login.xhtml, Hola.xhtml y Fulanito.xhtml. Es importante que los nombres acaben en .xhtml (ya que si no, no se identifican como páginas JSF)

Se puede generar una página JSF usando una plantilla que puede ser útil en algunos casos. En este caso no es necesario ya que se puede copiar y pegar el código que aparece debajo. Para usar una plantilla habría que seleccionar: Next => Escoger "Facelet Composition Page", por ejemplo.

Login.xhtml

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

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
<f:view>
<h:head><title>Una simple aplicacion JavaServer Faces</title></h:head>
<h:body>
<h:form>
<h3>Por favor, introduzca su nombre de cuenta y password </h3>
<table>
<tr>
<td>Nombre:</td>
<td><h:inputText id="nom" value="#{login.nombre}"></h:inputText></td>
</tr>
<tr>
<td>Password:</td>
<td><h:inputSecret id="pass" value="#{login.password}"></h:inputSecret></td>
</tr>
</table>
<p>
<h:commandButton value="Aceptar" action="#{login.comprobar}"/>
</p>
</h:form>
</h:body>
</f:view>
</html>
```

Hola.xhtml

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

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">

<f:view>
<h:head>
      <title>Una simple aplicacion JavaServer Faces</title>
</h:head>
<h:body>
<h:form>
<h3> Bienvenido a JavaServer Faces,
<h:outputText value="#{login.nombre}"/>!
</h3>
</h:form>
</h:body>
</f:view>
</html>
```

Y Fulanito.xhtml

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

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">

<f:view>
<h:head>
      <title>Una simple aplicacion JavaServer Faces</title>
</h:head>
<h:body>
<h:form>
<h3> Fulanito, no seas pelma que tú no puedes entrar. </h3>
</h:form>
</h:body>
</f:view>
</html>
```

1.3.- Crear el Bean gestionado siguiente **(Capa de lógica del negocio / Modelo)**:

Seleccionar: New => Other => Java => Bean (o bien Class) => Dar el nombre LoginBean.java en el paquete modelo.bean

Nota: tras escribir en el programa fuente los atributos nombre y password (`String nombre; y String password;`), se pueden generar automáticamente los métodos accesores y modificadores. Para ello: Hacer click derecho en la clase LoginBean.java => Source => Generate getters and setters.

LoginBean.java

```
package modelo.bean;

public class LoginBean {
String nombre;
String password;

public LoginBean(){
}
public String getNombre() {
	return nombre;
}
public void setNombre(String nombre) {
	this.nombre = nombre;
}
public String getPassword() {
	return password;
}
public void setPassword(String password) {
	this.password = password;
}
public String comprobar() {
	if (nombre.equals("fulanito")) return "error";
	else return "ok";
}}
```

1.4.- Realizar la configuración del proyecto JSF.

Es necesario configurar el bean gestionado y la navegación en el fichero faces-config.xml (que se encuentra en WEB-INF):

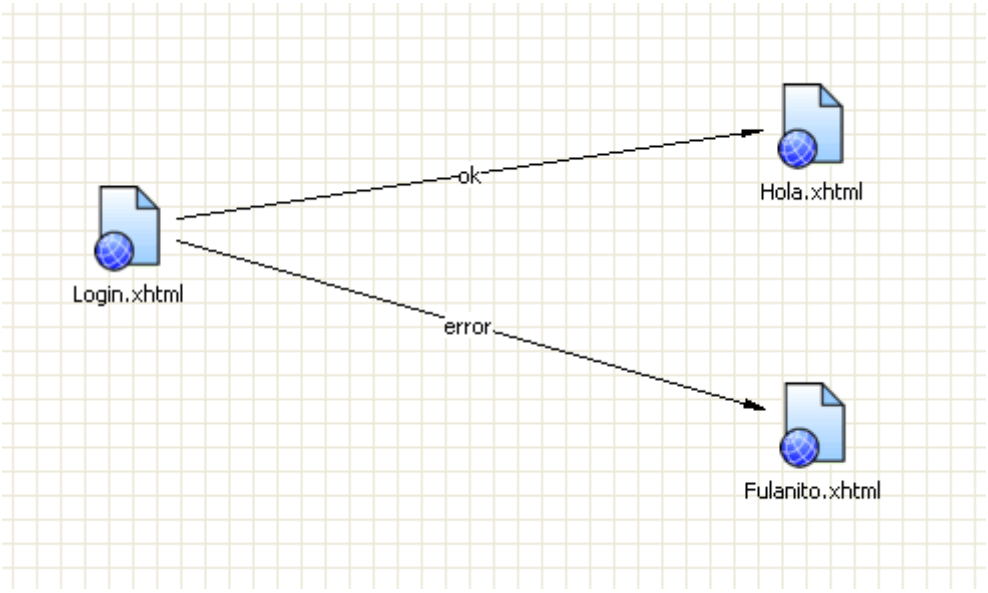
Hacer click derecho en el fichero faces-config.xml => Open with => Faces config editor

En las pestañas “Managed Bean” y “Navigation Rule” se configurarán los bean gestionados y la navegación respectivamente.

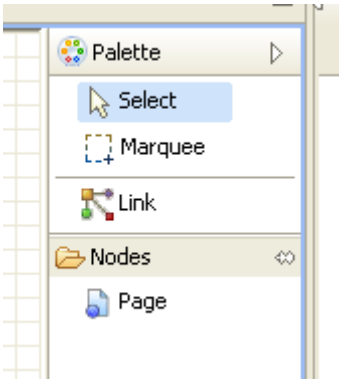
En la pestaña “Managed Bean”, pulsar Add => Browse => Seleccionar LoginBean en “Matching items:” En “Select entries:” se van escribiendo los primeros caracteres del nombre de la clase para que aparezca en “Matching items¨” y se pueda seleccionar.

Pulsar Next => Dar el nombre “login” al bean y seleccionar “session” como “scope” para el bean, y terminar con “Finish”.

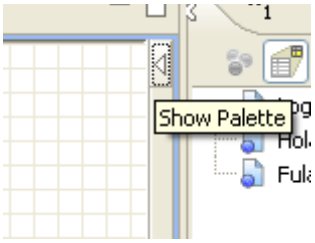
En la pestaña “Navigation Rule” => Arrastrar las páginas JSF desde el “Project Explorer” y añadir los siguientes enlaces.



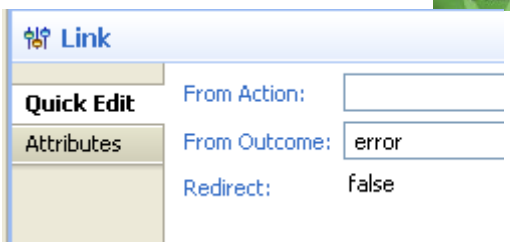
Para añadir un enlace, hay que seleccionar la “Palette” (arriba a la derecha en el editor de faces-config.xml) y, tras hacer click en “Link” se crea el enlace pinchando en los iconos de los dos JSF a enlazar (primero en el origen y luego en el destino).



Nota: si la “Palette” no está visible, se puede hacer visible en el triángulo arriba a la derecha siguiente:



Para etiquetar la flecha con la acción (***ok*** o ***error*** en nuestro caso) hay que seleccionar “Select” en la “Palette” => Hacer click derecho sobre la flecha => Show view => Properties => Escribir la acción en la propiedad “from-outcome”



Nota: si después de hacer lo anterior no son visibles las propiedades, hacer click derecho en la flecha sobre la que queremos ver las propiedades.

En la pestaña “Source” podremos ver el código generado de manera automática para el fichero faces-config.xml:

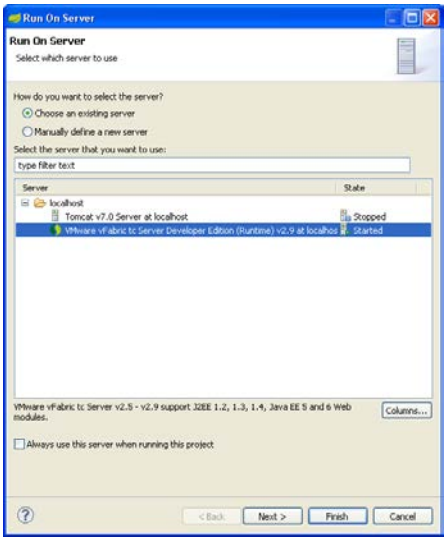
```
<?xml version="1.0" encoding="UTF-8"?>

<faces-config
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
    version="2.0">
      <managed-bean>
            <managed-bean-name>login</managed-bean-name>
            <managed-bean-class>modelo.bean.LoginBean</managed-bean-class>
            <managed-bean-scope>session</managed-bean-scope>
      </managed-bean>
      <navigation-rule>
            <display-name>Login.xhtml</display-name>
            <from-view-id>/Login.xhtml</from-view-id>
            <navigation-case>
                  <from-outcome>error</from-outcome>
                  <to-view-id>/Fulanito.xhtml</to-view-id>
            </navigation-case>
      </navigation-rule>
      <navigation-rule>
            <display-name>Login.xhtml</display-name>
            <from-view-id>/Login.xhtml</from-view-id>
            <navigation-case>
                  <from-outcome>ok</from-outcome>
                  <to-view-id>/Hola.xhtml</to-view-id>
            </navigation-case>
      </navigation-rule>
</faces-config>
```

1.5.- Ejecutar la aplicación JSF en el navegador del Eclipse

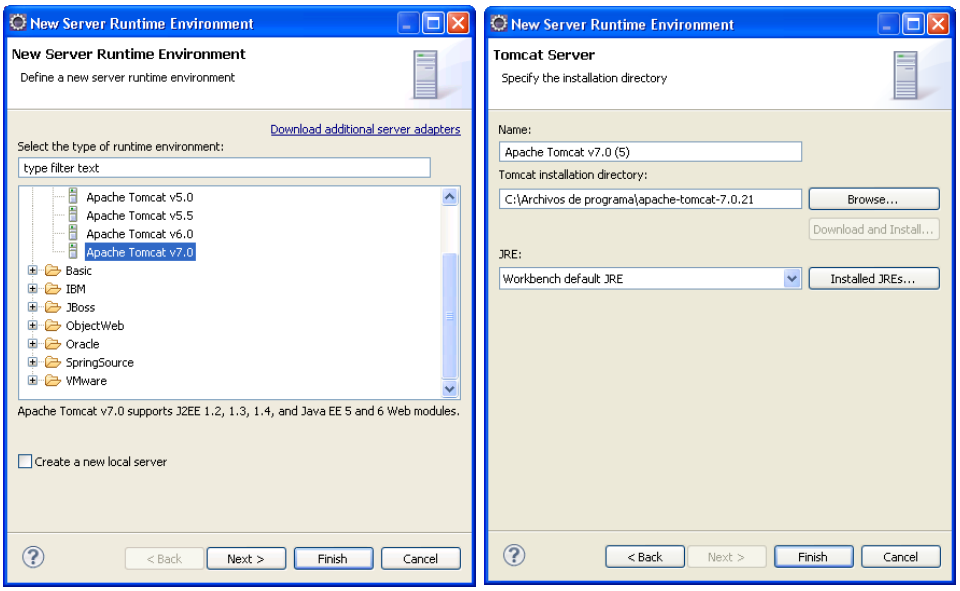
Click derecho en “Login.xhtml” => Run As => Run On Server

Habrá que escoger la instancia del servidor web, en nuestro caso el servidor Apache (o el servidor tc Server), que tal vez ya haya sido lanzado anteriormente. Por ejemplo, se puede seleccionar uno ya lanzado (Started): “Sel. VMware tcServer” => Finish



Si ya estaba lanzado y hemos hecho algún cambio en las páginas JSF o en los bean gestionados, hay que pararlo previamente (en la ventana “Servers”, hacer click derecho sobre el servidor Web => Stop), o al menos quitar la aplicación web lanzada ya en el servidor (en la misma ventana, hacer click derecho sobre la aplicación lanzada y escoger “Remove”).

Nota: Si no hay “Target runtime” se puede crear uno pulsando “New Runtime”, seleccionando por ejemplo un servidor “Apache Tomcat v7.0” => Next => Browse y seleccionar el directorio raíz donde se encuentre una instalación de Apache (en la figura C:\Archivos de programa\apache-tomcat-7.0.21).



Se puede comprobar que, con esta implementación de la lógica del negocio, se permite entrar a todos los usuarios excepto a “fulanito”, y no hace falta password.

1.6.- Ejecutar la aplicación JSF definiendo un JSF de inicio

Para ejecutar la aplicación, se ha seleccionado exactamente el JSF de inicio de esta manera: Click derecho en "Login.xhtml" => Run As => Run On Server

Se puede definir en el fichero de configuración web.xml cuál es el JSF de inicio, de tal manera que se pueda ejecutar la aplicación directamente: Click derecho en el proyecto LoginJSF => Run As => Run On Server

Se hace añadiendo `faces/Login.xhtml` en la lista de ficheros `<welcome-file-list>` del fichero de configuración web.xml de WEB-INF

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-ins
  <display-name>LoginJSF</display-name>
  <welcome-file-list>
    <welcome-file>faces/Login.xhtml</welcome-file>
    <welcome-file>index.html</welcome-file>
```

Nota: es necesario añadir "`faces/`" antes del nombre del JSF para indicar que se trata de un JSF, y para que lo procese como tal y no como un simple fichero HTML. En el mismo fichero web.xml se define el patrón para identificar a los JSFs:

```
<url-pattern>/faces/*</url-pattern>
```

1.7.- Ejecutar la aplicación JSF desde un navegador

Aunque la aplicación se está visualizando en un navegador dentro del entorno de desarrollo de Eclipse, se puede ejecutar desde cualquier navegador web, accediendo a la dirección URL siguiente:

http://localhost:8080/LoginJSF/faces/Login.xhtml

(si es que 8080 es el puerto del Servidor Web que se ha lanzado desde Eclipse y los nombres de la aplicación y página JSF son LoginJSF y Login.xhtml, respectivamente)

Como ya está definido el JSF de inicio, entonces se puede acceder directamente a la URL de la aplicación:

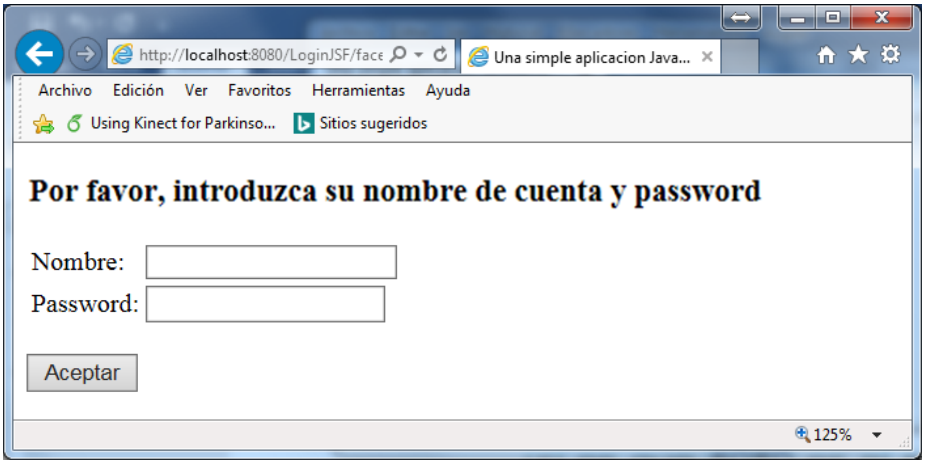
http://localhost:8080/LoginJSF/

Por supuesto, la aplicación Web se puede ejecutar desde un navegador Web de cualquier ordenador que se encuentre accesible por red, poniendo el número de IP de vuestra máquina en vez de "localhost" en la URL:

http://NUM_IP:8080/LoginJSF/faces/Login.xhtml o bien http://NUM_IP:8080/LoginJSF/

En la siguiente imagen se puede ver cómo se ejecuta desde el navegador Mozilla Firefox:

O en Internet Explorer:



2. Añadir componentes de validación, de conversión y visualización de errores.

2.1.- Para validar la entrada de datos, se pueden utilizar componentes JSF. Podría, por ejemplo, asegurarse que el password sea un número entre 1 y 500.

Para realizarlo, hay que incluir el siguiente componente de validación de datos de entrada asociado a la caja de texto donde se introduce el password (componente JSF `inputSecret` al que se le ha dado el nombre *"pass"*):

```
<f:validateLongRange minimum="1" maximum="500"/>
```

El validador se define dentro del componente (entre `h:inputSecret` y `/h:inputSecret`):

```
<h:inputSecret id="pass" value="#{login.password}">
        <f:validateLongRange minimum="1" maximum="500"/>
</h:inputSecret>
```

En el caso en que la entrada para *"pass"* no sea válida, entonces se mostrará en color rojo el mensaje de error correspondiente en otro componente JSF de tipo `h:message`.

```
<h:message for="pass" style="color:red" />
```

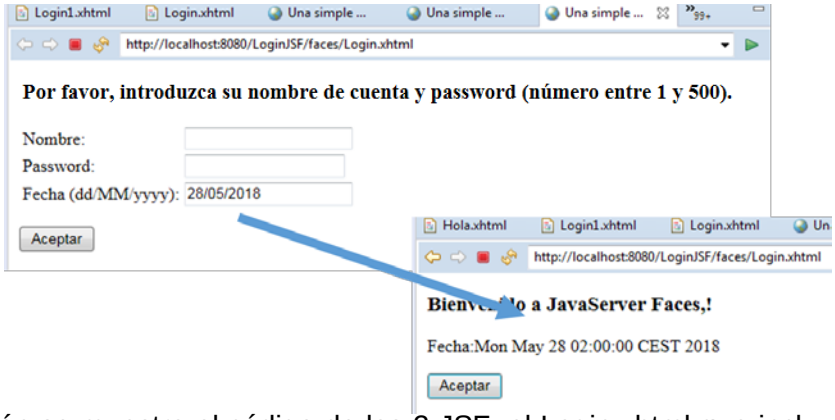
A continuación se añade el código completo del JSF Login.xhtml:

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

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
<f:view>
<h:head><title>Una simple aplicacion JavaServer Faces</title></h:head>
<h:body>
<h:form>
<h3>Por favor, introduzca su nombre de cuenta y password (número entre 1 y
500). </h3>

<table>
<tr>
<td>Nombre:</td>
<td><h:inputText id="nom" value="#{login.nombre}"></h:inputText></td>
</tr>
<tr>
<td>Password:</td>
<td><h:inputSecret id="pass" value="#{login.password}">
        <f:validateLongRange minimum="1" maximum="500"/></h:inputSecret></td>
    <td> <h:message for="pass" style="color:red" /></td>
</tr>
</table>
<p>
<h:commandButton value="Aceptar" action="#{login.comprobar}"/>
</p>
</h:form>
</h:body>
</f:view>
</html>
```

2.2.- El uso de conversores de datos es interesante para transformar los datos de entrada introducidos en los formularios (que son cadenas de caracteres, de tipo String) en objetos Java. En este caso se puede utilizar un conversor de datos `f:convertDateTime` para convertir un String en un objeto Java de la clase Date. En la siguiente imagen se puede comprobar que el String de entrada “28/05/2018” se ha convertido en un objeto de Date, que al mostrarlo en la vista Hola.xhtml aparece como “Mon May 28 02 00:00 CEST 2018”



A continuación se muestra el código de los 2 JSF: el Login.xhtml que incluye el nuevo componente `h:inputText` para introducir la fecha y el conversor `f:convertDateTime`; y el JSF Hola.xhtml que muestra la información de la fecha introducida.

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

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
<f:view>
<head><title>Una simple aplicacion JavaServer Faces</title></head>
<body>
<h:form>
<h3>Por favor, introduzca su nombre de cuenta y password (número entre 1 y
500). </h3>
<table>
<tr>
<td>Nombre:</td>
<td><h:inputText id="nom" value="#{login.nombre}"></h:inputText></td>
</tr>
<tr>
<td>Password:</td>
<td><h:inputSecret id="pass" redisplay="true" value="#{login.password}">
        <f:validateLongRange minimum="1" maximum="500"/></h:inputSecret></td>
    <td> <h:message for="pass" style="color:red" /></td>
</tr>
    <td>Fecha (dd/MM/yyyy):</td>
    <td><h:inputText id="fecha" value="#{login.fecha}">
       <f:convertDateTime pattern="dd/MM/yyyy"/>
       </h:inputText></td>
    <td>  <h:message for="fecha" style="color:red" /> </td>
</table>
<p>
<h:commandButton value="Aceptar" action="#{login.comprobar}"/>
</p>
<p>
<h:messages  id="mensajes" style="color: blue" />
</p></h:form>
</body>
</f:view>
</html>
```

Hola.xhtml

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

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">

<f:view>
<h:head>
      <title>Una simple aplicacion JavaServer Faces</title>
</h:head>
<h:body>
<h:form>
<h3> Bienvenido a JavaServer Faces,
<h:outputText value="#{login.nombre}"/>!
</h3>
<p> Fecha: <h:outputText value="#{login.fecha}"/></p>
</h:form>
</h:body>
</f:view>
</html>
```

Además, para gestionar la entrada y salida de la fecha por parte del bean, se necesita modificar el bean `LoginBean`. Hay que añadir un atributo `fecha` y sus métodos accesor y modificador.

```
Date fecha;

public Date getFecha() {
      return fecha;
}
public void setFecha(Date fecha) {
      this.fecha = fecha;
}
```

2.3.- Mostrar los mensajes de error.

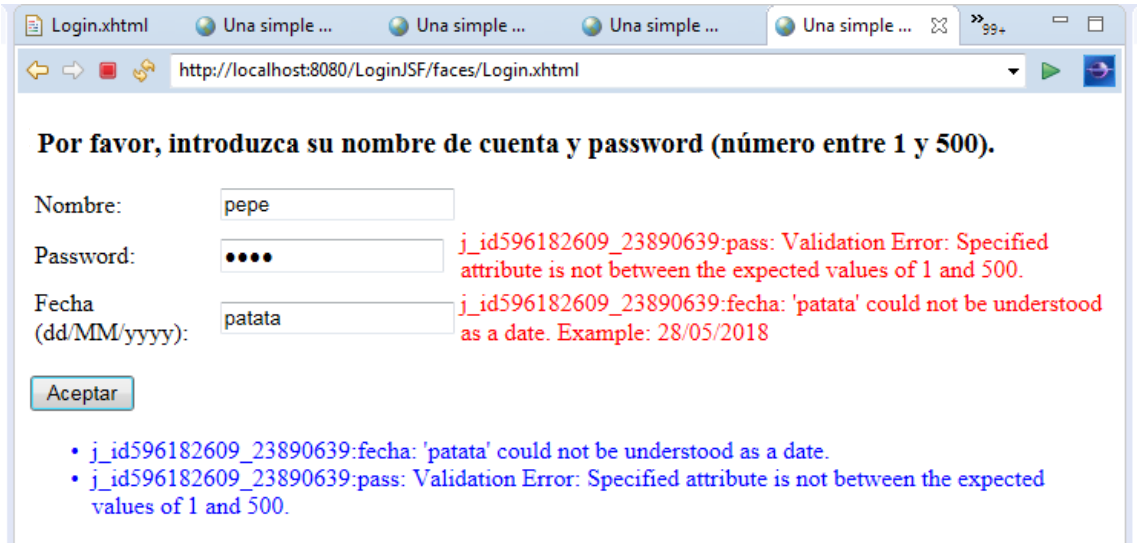
Ya se ha señalado anteriormente que para mostrar los errores de validación se pueden utilizar componentes JSF del tipo `h:message`. De hecho se ha utilizado el siguiente para mostrar los errores generados en el componente llamado *"pass"*

```
<h:message for="pass" style="color:red" />
```

En el JSF Hola.xhtml hay otro componente similar donde se muestran los errores asociados al componente llamado *"fecha"*. Dichos errores se generarán cuando el string introducido para la fecha no siga el patrón *"dd/MM/yyyy"* que se ha asignado a dicho componente. Nota: no confundir en el patrón *mm* con *MM*, ya que en minúsculas son minutos y no meses.

```
<f:convertDateTime pattern="dd/MM/yyyy"/>
```

Por último, en el JSF Hola.xhtml se ha añadido otro componente `h:messages` no asociado con un componente en concreto. En el mismo se mostrarán los mensajes de error generados durante la ejecución del JSF (aquellos que se generan durante el ciclo de vida del JSF y que se envían al FacesContext). Conviene incluir un componente `h:messages` para identificar esos errores, porque si no, no se visualizarían y sería difícil depurar la aplicación. En la siguiente imagen se puede ver una ejecución donde se han introducido valores incorrectos para el password y para la fecha. Ambos errores se muestran en los componentes `h:message` y `h:messages`.



Nota: En el componente *"pass"* se visualizan los puntos después de haber pulsado el botón “Aceptar” porque se ha añadido el atributo `redisplay="true"`

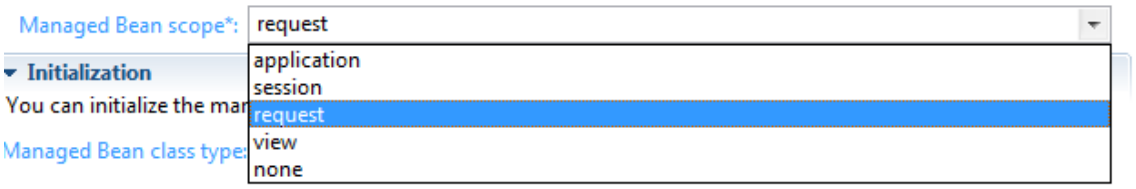
```
<h:inputSecret id="pass" redisplay="true" value="#{login.password}">
```

3. Scope de los beans gestionados: request, session y application.

En este apartado vamos a comprobar la diferencia entre distintos tipos de scope que se pueden asignar a los beans gestionados, y que determinan cuándo se crean las instancias o beans, y si son compartidas o no.

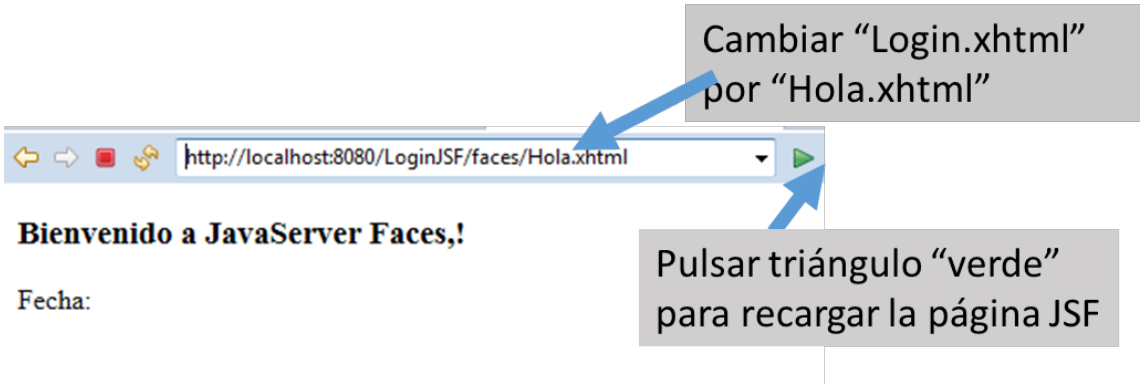
- Cuando el scope del bean gestionado es “request” el controlador de Faces (el FacesServlet) crea un nuevo bean por cada petición HTTP realizada desde el navegador.
- Cuando el scope del bean gestionado es “session” el controlador crea un bean al comenzar una sesión HTTP y lo usa mientras el navegador no se cierre y la sesión no caduque.
- El scope “application” permite compartir el bean entre sesiones. El bean se crea al lanzarse el servidor web y se mantiene mientras dicho servidor se encuentre en ejecución.

Para cambiar el scope: faces-config.xml => Open With Faces Config Editor => Pestaña “Managed Bean” y seleccionar el scope



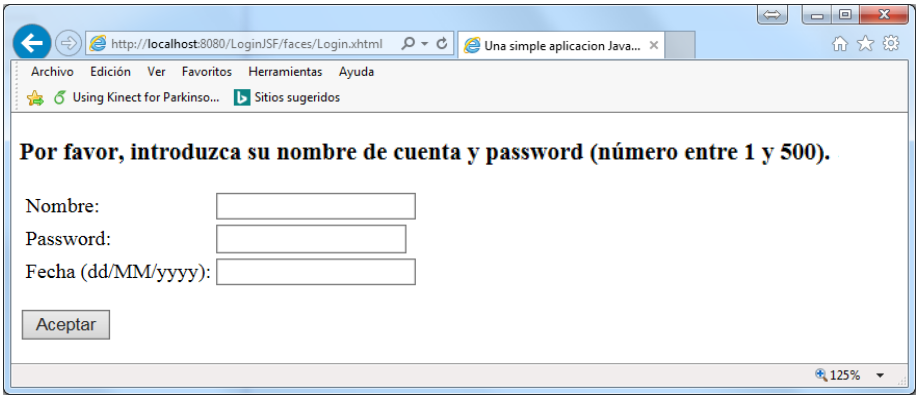
Poner el scope “request” => Run JSF Login.xhtml => Escribir los datos de entrada “Alf”, “123” y “29/05/2018” y pulsar botón “Aceptar” => Se visualizará la vista Hola.xhtml con los valores introducidos en la vista “Login.xhtml”. Esto es así porque no se ha producido una nueva petición HTTP desde el navegador y por tanto el scope “request” es adecuado en este caso.

Si se cambia Login.xhtml por Hola.xhtml y se recarga la página, esto es, se realiza una nueva petición HTTP al JSF Hola.xhtml entonces se comprobará que no se recuerdan los valores anteriores del bean, debido a que se habrá creado una nueva instancia del bean (scope “request”)



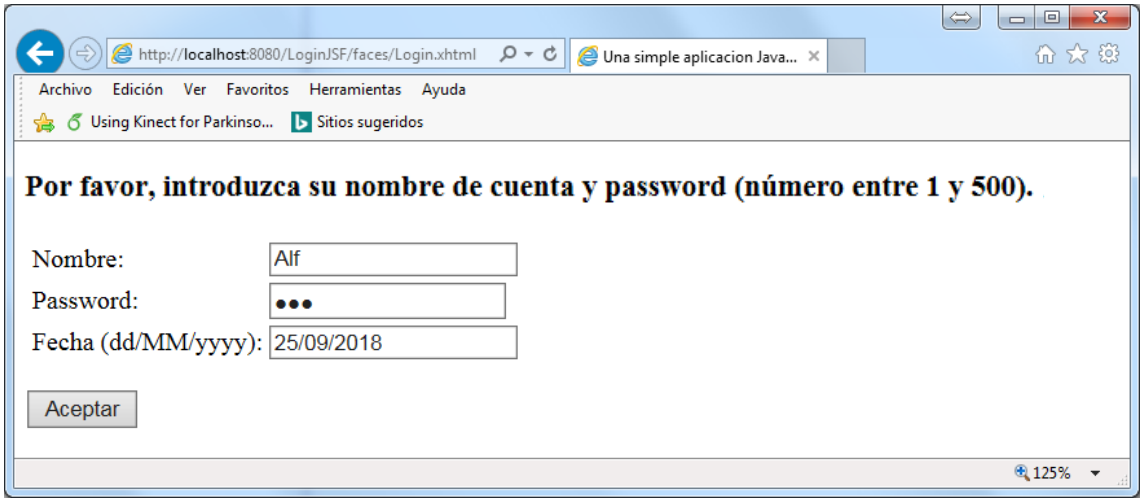
Poner el scope “session” => Run JSF Login.xhtml => Escribir los datos de entrada “Alf”, “123” y “29/05/2018” y pulsar botón “Aceptar” => Se visualizará la vista Hola.xhtml con los valores introducidos en la vista “Login.xhtml”. Pero también se visualizan dichos valores si se vuelve a recargar el JSF Hola.xhtml. Y también se visualizan cuando se vuelve a recargar el JSF Login.xhtml.

Sin embargo, si se accediera en otro navegador al JSF Login.xhtml, entonces no se visualizarían los valores del bean, ya que al tratarse de otra sesión, se crea un nuevo bean. En el navegador se escribe: http://localhost:8080/LoginJSF/faces/Login.xhtml



Por último se puede comprobar que el bean se comparte entre sesiones con el scope "application".

Poner el scope "application" en el bean => Run JSF Login.xhtml => Escribir los datos de entrada "Alf", "123" y "29/05/2018" y pulsar botón "Aceptar" => Se visualizará la vista Hola.xhtml con los valores introducidos en la vista "Login.xhtml" => Acceder en otro navegador al JSF http://localhost:8080/LoginJSF/faces/Login.xhtml y comprobar que se muestran los valores introducidos en la otra sesión.

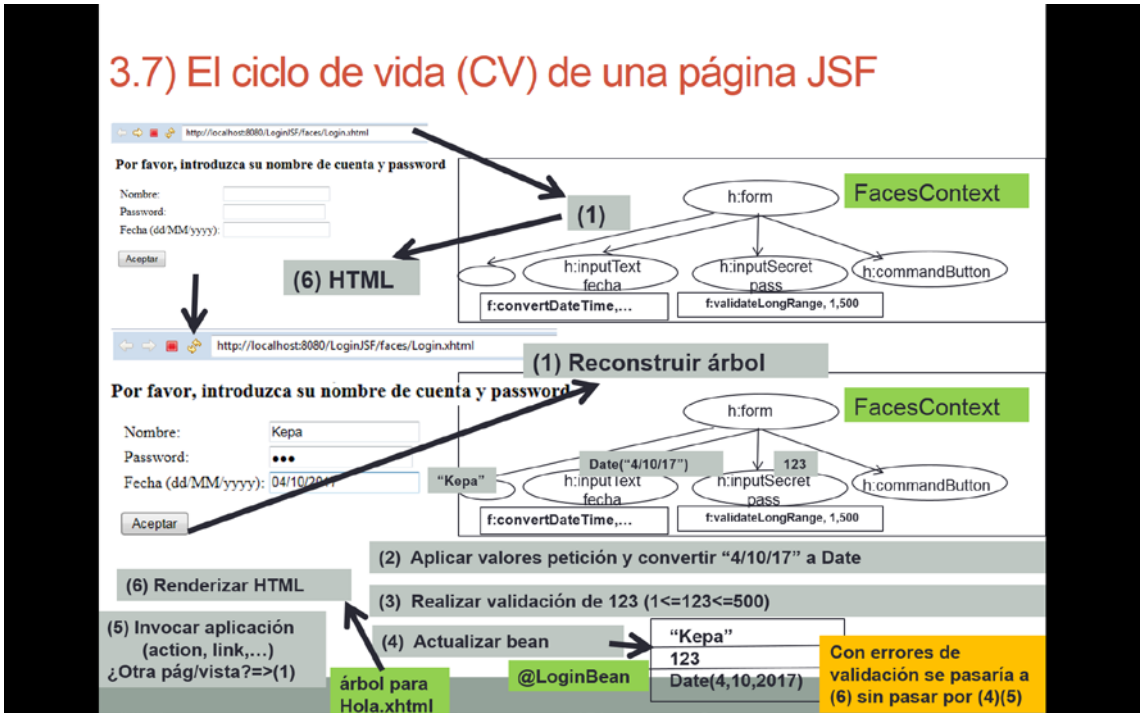


4. El Ciclo de Vida (CV) de una página JSF

Es necesario entender que durante el procesamiento de una página JSF se aplican algunas fases en un orden determinado, siendo la última de ellas la obtención del código HTML (renderización) que es el que se muestra en el navegador web. Durante el procesamiento del JSF, dependiendo de si suceden algunos eventos concretos o se detectan errores, se pasa directamente a la última fase sin completar todas las fases anteriores. Estas son las fases del ciclo de vida de una página JSF:

1) Reconstituir árbol de componentes: el controlador (re)construye en memoria (FacesContext) la estructura de componentes de la página junto con manejadores de eventos y validadores.
2) Aplicar valores de la petición: cada componente extrae sus valores de los parámetros "request", calcula nuevos valores según conversiones, y los almacena localmente.
3) Procesar validaciones: Se aplican todos los validadores registrados.
4) Actualizar los valores del modelo: Se actualizan propiedades de beans con los valores ya convertidos, validados y almacenados en los componentes
5) Invocar aplicación: Se tratan los eventos a nivel de aplicación como enviar un formulario (h:commandButton) o seguir un enlace a otra página (h:link).
6) Renderizar la respuesta: se genera la página HTML de respuesta, mostrando los componentes del árbol, valores y errores de la vista a mostrar.

Ejecutar el JSF Login.xhtml con los parámetros "Kepa", "123" y "04/10/2017" e intentar analizar por qué fases pasa tras pulsar el botón "Aceptar" antes de que se cargue la vista Hola.xhtml, la cual se visualiza con los valores del mismo bean.



Ejecutar a continuación el mismo JSF Login.xhtml pero con los parámetros "Kepa", "1234" y "04/10/2017". Intentar analizar por qué fases pasa tras pulsar el botón "Aceptar", y comprobar que no se pasa por las fases (4) y (5), sino que se visualiza la misma vista Login.xhtml con los errores asociados al validador.

5. Modificar el CV del JSF: Evitar validaciones y conversiones

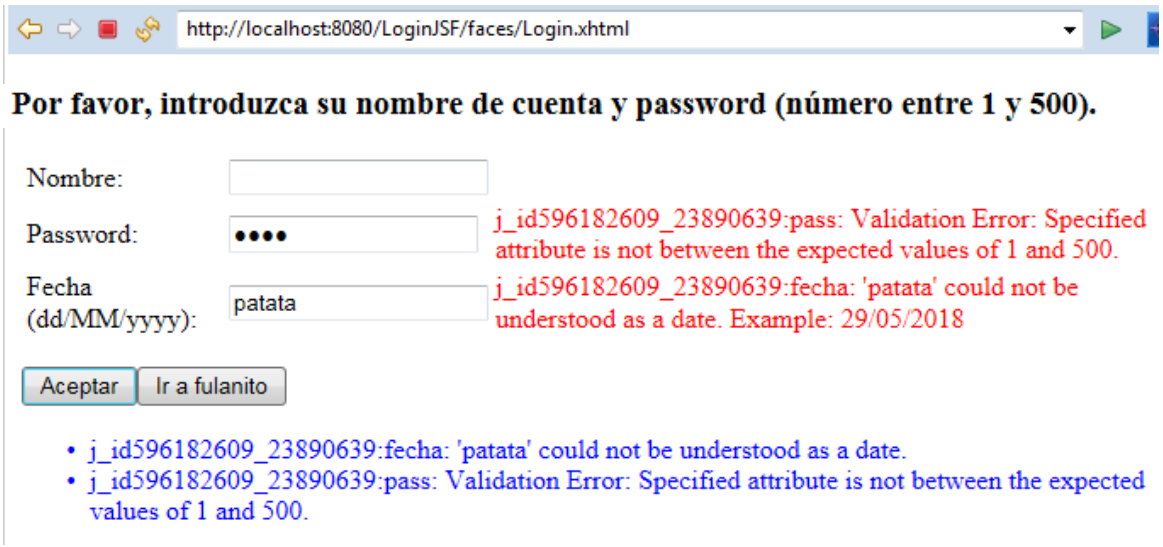
Añadir en el JSF Login.xhtml un botón “Ir a fulanito” que cuando se pulse, se cargue la vista Fulanito.xhtml, la cual simplemente indica que Fulanito no puede entrar al sistema. El código necesario para conseguirlo es el siguiente:

```
<h:commandButton value="Ir a fulanito" action="error"></h:commandButton>
```

Y debe colocarse justo a continuación del código correspondiente al botón “Aceptar”:

```
<h:commandButton value="Aceptar" action="#{login.comprobar}"/>
```

Ejecutar el JSF: Run Login.xhtml => Introducir los valores “1234” en password y “patata” como fecha => Pulsar el botón “Ir a fulanito” => Comprobar que no va al JSF Fulanito.xhtml porque existen errores en la validación y en el formato de fecha (de acuerdo con el ciclo de vida JSF). Sin embargo, ese ciclo de vida no es muy adecuado en este caso porque impide dirigirse a la vista a la que se desea ir, aunque haya errores en los datos del formulario, que no van a ser usados en la siguiente vista.



Ejecutar el JSF: Run Login.xhtml => Introducir los valores “1234” en password y “patata” como fecha => Pulsar el botón “Ir a fulanito” => Comprobar que esta vez sí carga la vista Fulanito.xhtml.

El código del JSF Login.xhtml es el siguiente:

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

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
<f:view>
<h:head><title>Una simple aplicacion JavaServer Faces</title></h:head>
<h:body>
<h:form>
<h3>Por favor, introduzca su nombre de cuenta y password (número entre 1 y
500). </h3>
<table>
<tr>
<td>Nombre:</td>
<td><h:inputText id="nom" value="#{login.nombre}"></h:inputText></td>
</tr>
<tr>
<td>Password:</td>
<td><h:inputSecret id="pass" redisplay="true" value="#{login.password}">
        <f:validateLongRange minimum="1" maximum="500"/></h:inputSecret></td>
    <td> <h:message for="pass" style="color:red" /></td>
</tr>
    <td>Fecha (dd/MM/yyyy):</td>
    <td><h:inputText id="fecha" value="#{login.fecha}">
       <f:convertDateTime pattern="dd/MM/yyyy"/>
       </h:inputText></td>
    <td>  <h:message for="fecha" style="color:red" /> </td>
</table>
<p>
<h:commandButton value="Aceptar" action="#{login.comprobar}"/>
<h:commandButton value="Ir a fulanito" immediate="true"
action="error"></h:commandButton>
</p>
<p>
<h:messages  id="mensajes" style="color: blue" />
</p>
</h:form>
</h:body>
</f:view>
</html>
```

6. Modificar el CV del JSF: usar Ajax para actualizar parcialmente el HTML

Añadir en el JSF Login.xhtml un componente Ajax que funciona de esta manera: cuando en tiempo de ejecución se modifica el `h:inputSecret` que se llama *"pass"*, en ese momento se genera (o renderiza) el HTML correspondiente al componente `h:messages` que se llama *"mensajes"*. Esto es, no hay que esperar a pulsar el botón "Aceptar", que es un botón `h:commandButton`. En este caso se generaría el HTML en la última fase del CV tras procesar los eventos a nivel de aplicación (como pulsar un botón `h:commandButton` o un enlace `h:link`). El componente Ajax es el siguiente:

```
<f:ajax execute="pass" render="mensajes"/>
```

Y se puede colocar asociado al mismo componente *"pass"*

```
<h:inputSecret id="pass" redisplay="true" value="#{login.password}" >
        <f:validateLongRange minimum="1" maximum="500"/>
        <f:ajax execute="pass" render="mensajes"/>
</h:inputSecret>
```

Ejecutar el JSF: Run Login.xhtml => Introducir los valores "1234" en password y hacer click en otro componente como el de nombre o fecha (no pulsar la tecla "enter" porque equivale a pulsar el primer `h:commandButton` del formulario, que es un evento a nivel de aplicación) => Ver el mensaje de error de validación mostrado en *"mensajes"*, como resultado de la ejecución del componente Ajax.