

Índice

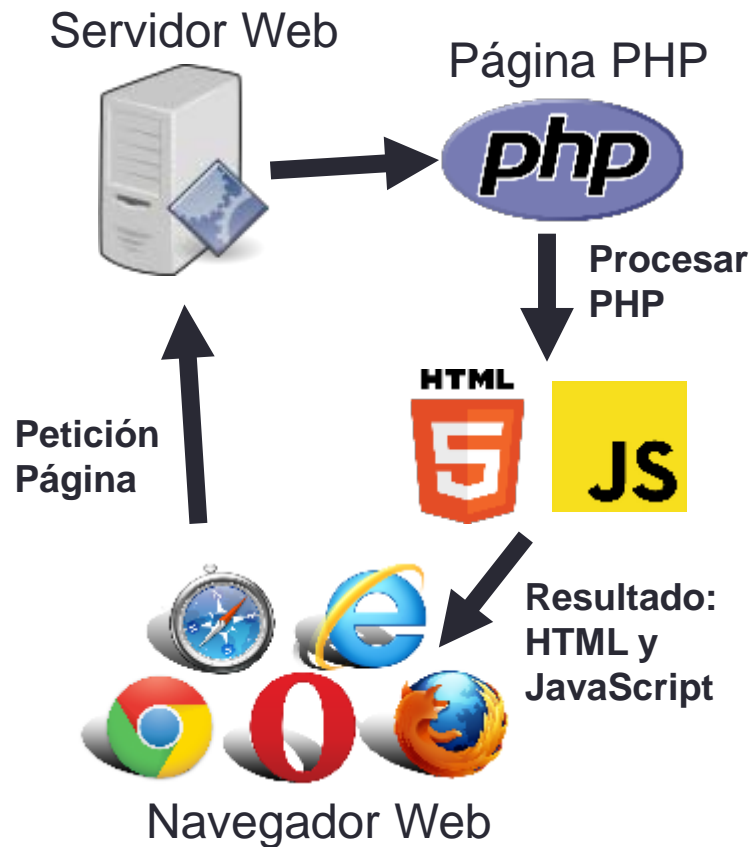
- 1) Introducción
- 2) Arquitectura Modelo Vista Controlador (MVC)
- 3) Java Server Faces
 - 3.1) Modelos
 - 3.2) Vistas
 - 3.3) Controlador y el fichero de configuración XML web.xml
 - 3.4) El fichero XML de configuración faces-config.xml
 - 3.5) Conversiones y validación de entrada en JSF
 - 3.6) Scope de los beans gestionados
 - 3.6) El ciclo de vida de una página JSF
 - 3.7) Otras características concretas en JSF

1) Introducción

- JSF es: (JSF 1.0 de 2004, JSF 2.2 de 2013)
 - un framework para construir interfaces gráficas de usuario (en aplicaciones Web)
 - la especificación estándar para la plataforma Java EE
- Como es sólo una especificación, se necesitan implementaciones/extensiones
 - Mojarra (implementación de Sun Microsystems / Oracle)
 - MyFaces (implementación de Apache)
 - RichFaces, PrimeFaces...(extensiones que ofrecen más componentes. Por ejemplo: soporte para Ajax,...)
- La solución Java anterior para aplicaciones Web eran los JSPs (1999) y los Servlets (1997)

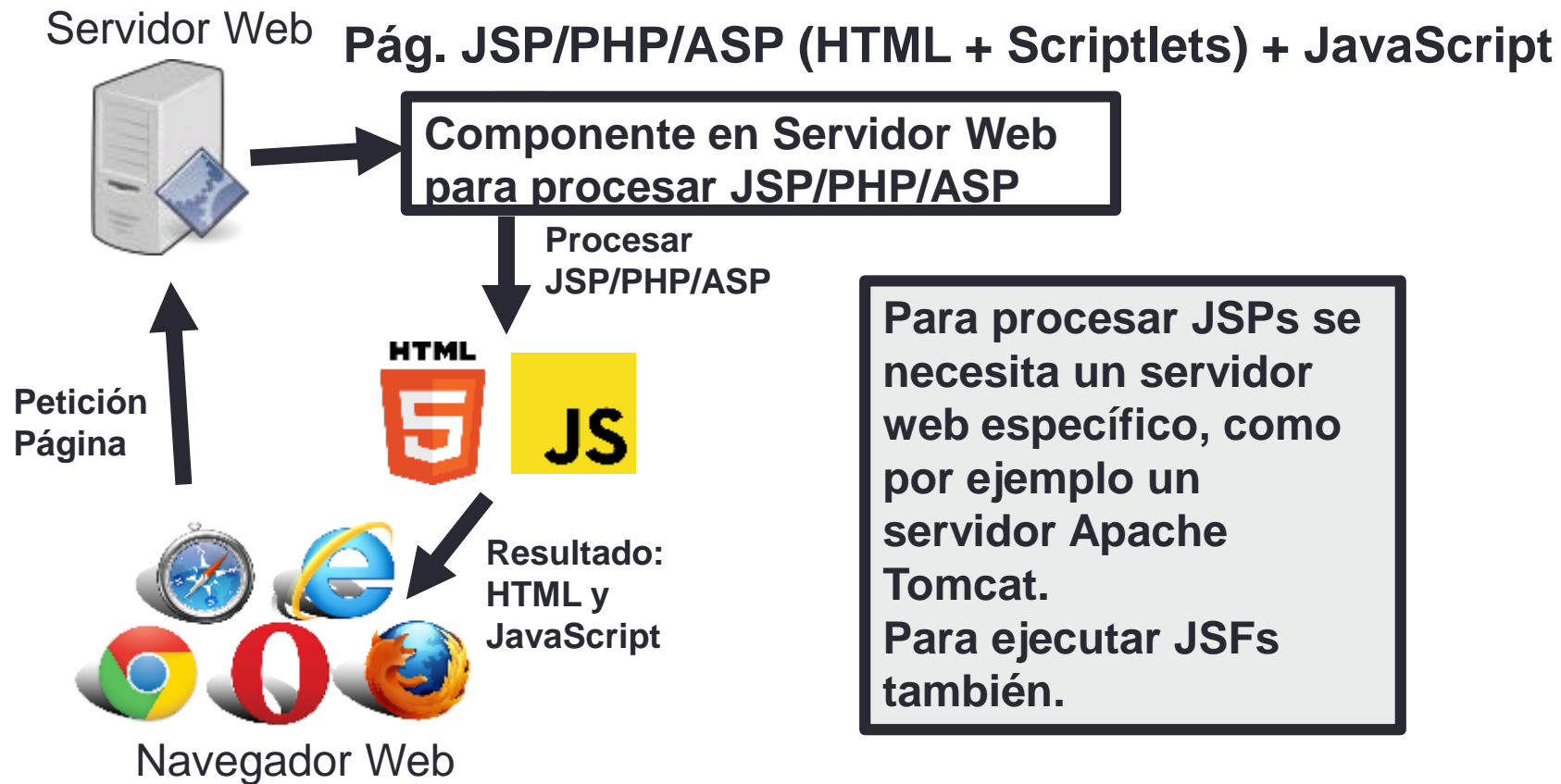
1) Introducción

- JSP (Java Server Pages) es la solución Java equivalente a PHP o a ASP (Microsoft)



1) Introducción

- JSP (Java Server Pages) es la solución Java equivalente a PHP o a ASP (Microsoft)



1) Introducción

- Problema: las páginas JSP deben indicar explícitamente:
 - las páginas JSP a las que se navega
 - la lógica del negocio que se invoca

http://localhost:8080/biblioWeb/TomarPrestamoCopia.jsp

TOMAR PRESTAMO COPIA LIBRO

Introduce el número de socio

Y la signatura:

Al pulsar el botón se invocará a la lógica del negocio correspondiente

```
<form action="ResultadoTomarPrestamoCopia.jsp" ...>
```

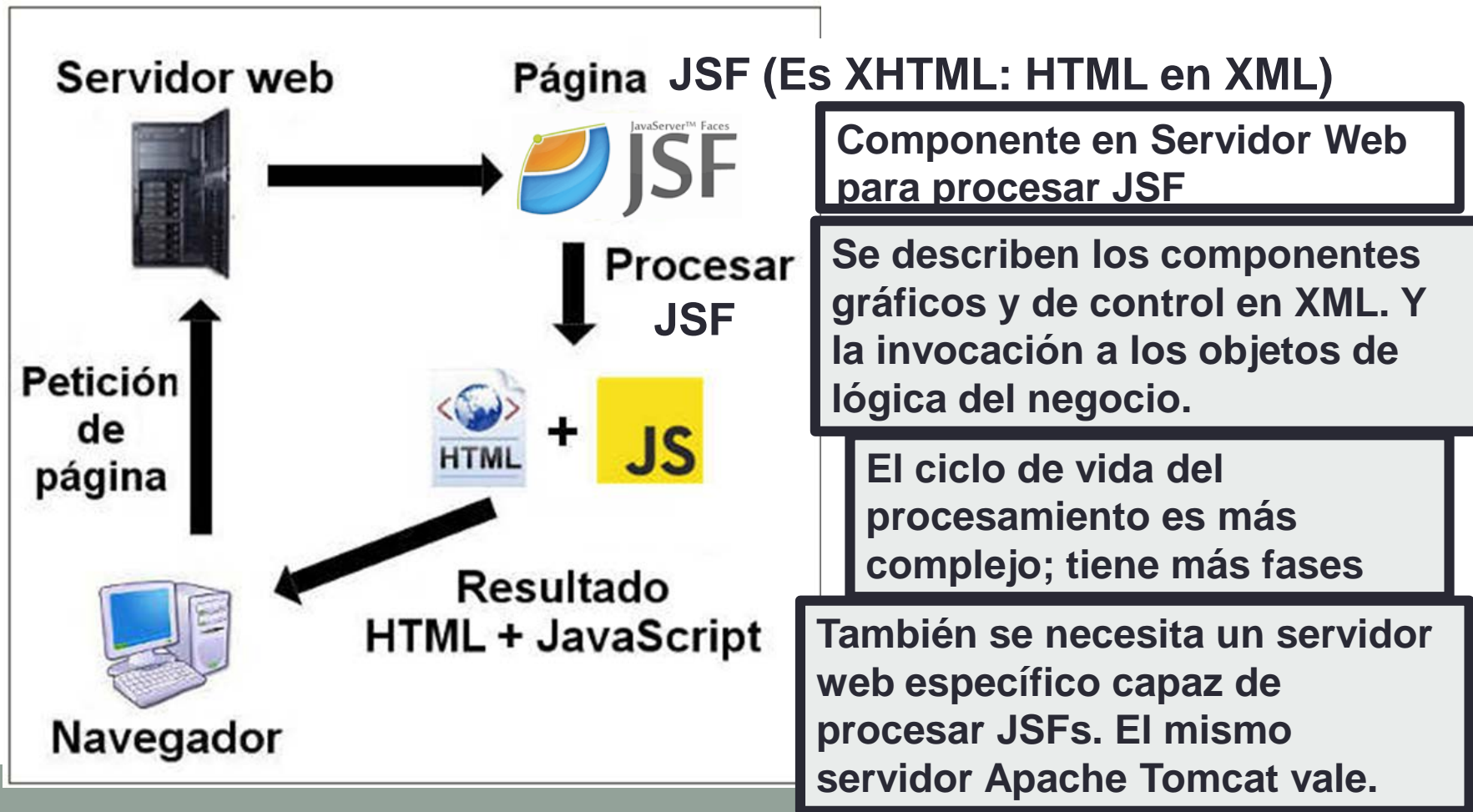
```
...
```

```
<input type="submit" value="Tomar Préstamo">
```

```
<jsp:useBean id="gestorPrestamo" class="beans.TomarPrestamoCopia" />  
<jsp:getProperty name="gestorPrestamo" property="copia" />
```

1) Introducción

- Con JSF, no se escriben páginas en HTML + código sino en XML (con etiquetas JSF)

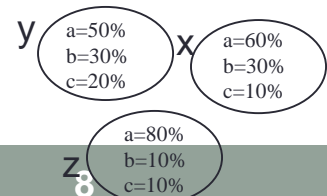
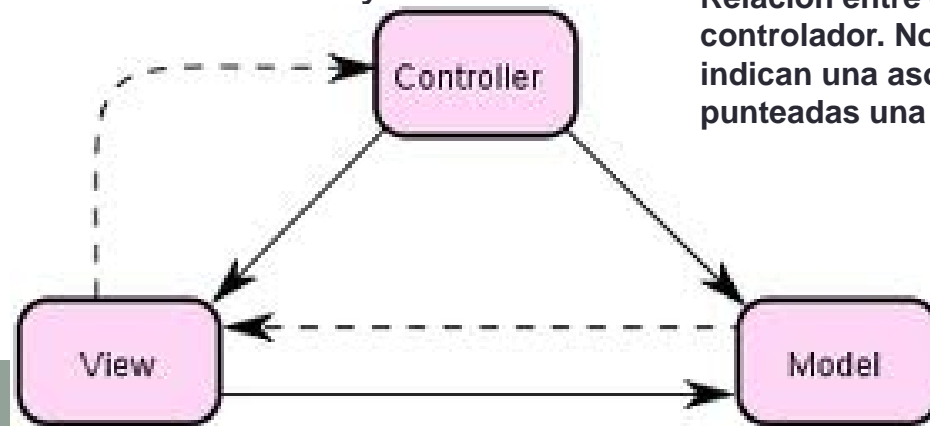
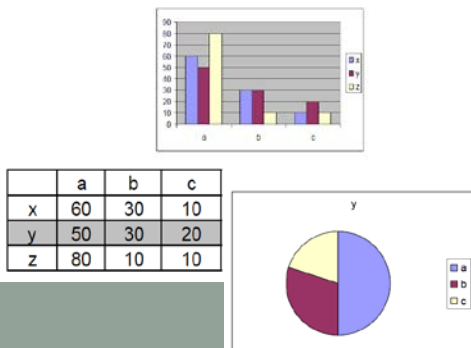


2) Arquitectura Modelo Vista Controlador (MVC)

- Modelo Vista Controlador (MVC) es un **patrón de arquitectura de software** que separa en una aplicación la interfaz de usuario de los datos y la lógica de negocio. Utiliza 3 componentes para ello:

- Modelo:** representación específica de la información con la cual el sistema opera.
- Vista:** la interfaz de usuario; presenta el modelo en un formato adecuado para interactuar.
- Controlador:** responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y a la vista

Relación entre el modelo, la vista y el controlador. Nota: las líneas sólidas indican una asociación directa, y las punteadas una indirecta

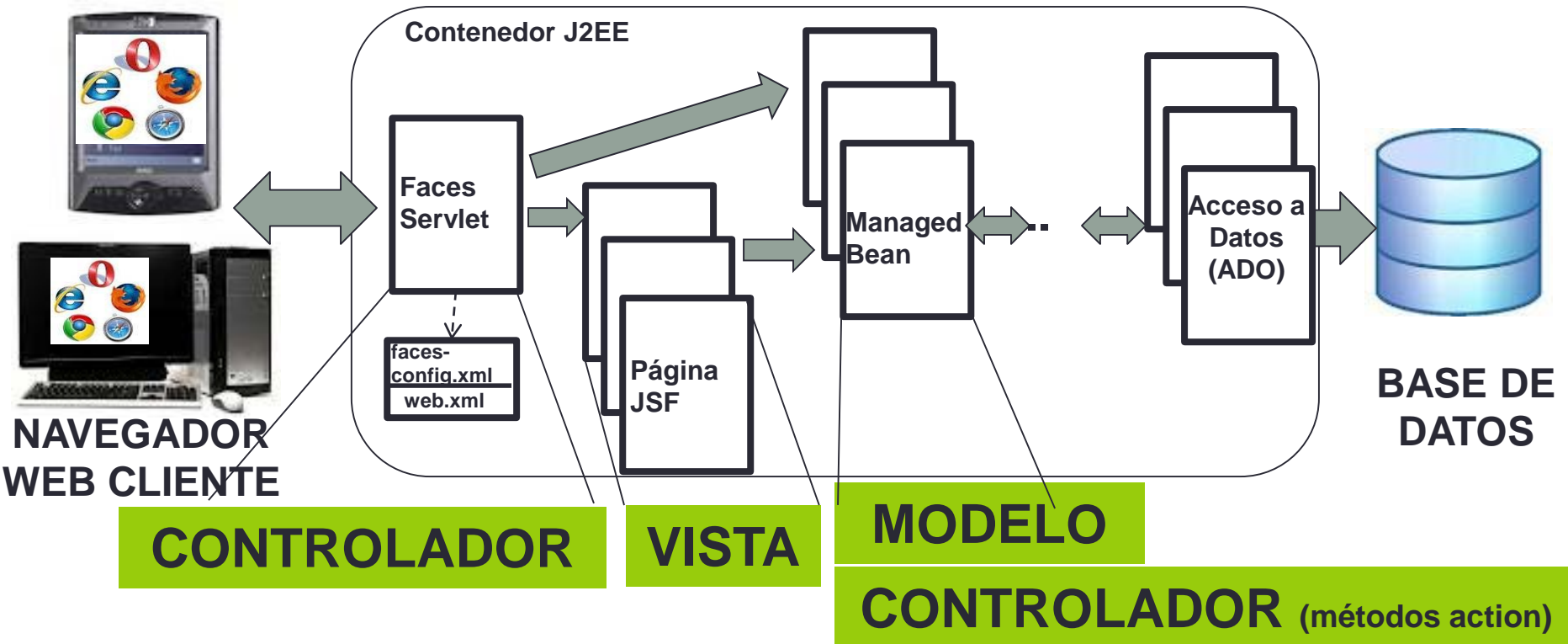


3) JSF es un framework que se basa en la arquitectura MVC

NIVEL/CAPA DE PRESENTACIÓN

NIVEL/CAPA DE LÓGICA DE NEGOCIO

NIVEL/CAPA DE DATOS



3.1) El Modelo en JSF son los Managed Bean (bean gestionados)

```
public class LoginBean {  
private String nombre;  
private String password;
```

```
public LoginBean(){}
```

```
// ATRIBUTO: nombre
```

```
public String getNombre() { return nombre; }
```

```
public void setNombre(String nuevoValor) {  
    nombre = nuevoValor; }
```

```
// ATRIBUTO: password
```

```
public String getPassword() { return password; }
```

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

```
}
```

Por favor, introduzca su nombre de cuenta y password

Nombre: nombre

Password: password

Aceptar

El bean permite obtener (get) y establecer (set) los valores de nombre y password

Falta el método "action" comprobar()

3.2) Las Vistas JSF son las páginas JSF (que son páginas dinámicas con componentes JSF)

```
<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></head>
<h:body>
<h:form>
<h3>Por favor, introduzca su nombre y password.</h3>
<table>
<tr><td>Nombre:</td>
<td><h:inputText value="#{login.nombre}"/></td></tr>
<tr><td>Password:</td>
<td><h:inputSecret value="#{login.password}"/></td></tr>
</table>
<p>
<h:commandButton value="Aceptar" action="ok"/>
</p>
</h:form>
</h:body>
</f:view> </html>
```

Librerías de ETIQUETAS JSF

Interacción con el BEAN "login"

Interacción con el CONTROLADOR. No llamada a una VISTA

Los componentes que comienzan por "h" crean formularios HTML(elementos gráficos como botones, campos de texto...). Los que comienzan por "f" sirven para manejo de eventos, conversiones de datos, validaciones,...

3.3) El Controlador en JSF es un Servlet (el Faces Servlet)

- El controlador es responsable de la navegación entre las vistas
- JSF nos proporciona uno que podemos utilizar
- **Se declara en el fichero de configuración web.xml**

```
<web-app>
<servlet>
<servlet-name>Faces Servlet</servlet-name>
<servlet-class>javax.faces.webapp.FacesServlet</servletclass>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>Faces Servlet</servlet-name>
<url-pattern>/faces/*</url-pattern>
</servlet-mapping>
<welcome-file-list>
<welcome-file>faces/Login.xhtml</welcome-file>
</welcome-file-list>
</web-app>
```

Declaración del CONTROLADOR utilizado

Patrón URL que permite distinguir si un fichero es JSF o no, y por tanto, procesarse como tal o no.

Nombre de la vista de inicio

3.4) faces-config.xml: Relación entre la referencia al Bean en la vista y la clase que lo implementa

```
<td> <h:inputText value="#{login.nombre}"/></td></tr>
```

Expresión escrita en EL (Expression Language) de JSF. Permite más posibilidades de acceso a propiedades.

Se declara en el fichero de configuración faces-config.xml

```
<managed-bean>  
<managed-bean-name>login</managed-bean-name>  
<managed-bean-class>LoginBean</managed-bean-class>  
<managed-bean-scope>session</managed-bean-scope>  
</managed-bean>
```

NOTA: Se puede cambiar la clase del BEAN sin necesidad de modificar la vista JSF, sólo cambiando la configuración.

3.4) faces-config.xml: Relación “estática” entre la acción y la vista a la que se navega o redirecciona

La navegación entre vistas se declara en el fichero de configuración faces-config.xml

```
<navigation-rule>  
<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>
```

Esa regla dice que: cuando se está visualizando la vista Login.xhtml, si se lanza la acción “ok” el controlador cargará la vista “Hola.xhtml”

La acción “ok” se lanzará cuando ocurra el evento pulsar el botón “Aceptar” que se encuentra definido en Login.xhtml:

```
<h:commandButton value="Aceptar" action="ok"/>
```

NOTA: se puede cambiar la vista a la que se redirecciona (Hola.xhtml) sin necesidad de modificar la vista JSF (Login.xhtml), sólo cambiando la configuración (el fichero faces-config.xml)

3.4) faces-config.xml: Relación “dinámica” entre la acción y la vista a la que se navega o redirecciona, dependiendo de lo que devuelva el modelo*

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

En este caso, la acción se lanza desde una clase bean (*) de manera “dinámica”

```
<navigation-rule>
<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>
<navigation-rule>
<from-view-id>/Login.xhtml</from-view-id>
<navigation-case>
<from-outcome>error</from-outcome>
<to-view-id>/Error.xhtml</to-view-id>
</navigation-case>
</navigation-rule>
```

Contenido de faces-config.xml que define la navegación, dependiendo de la acción

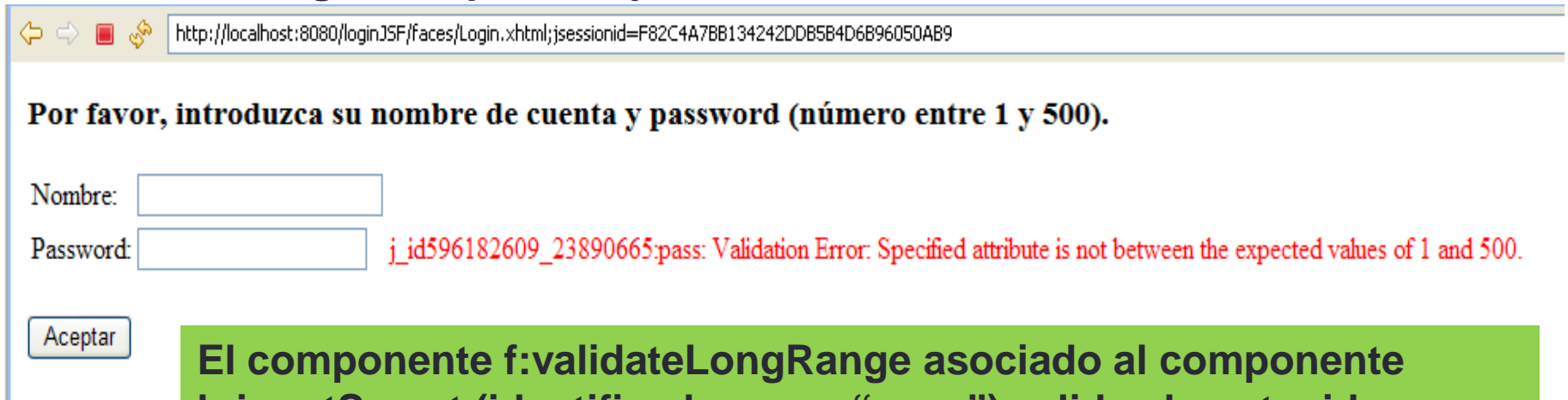
```
public class LoginBean {
...
public String comprobar() {
if (...) return “login”;
else return “error”;}
}
```

(*) los métodos action de los bean son parte del CONTROLADOR y no del MODELO. En el JSF aparece exactamente el nombre del método (comprobar)

3.5) JSF ofrece además componentes para validación de datos de entrada

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

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



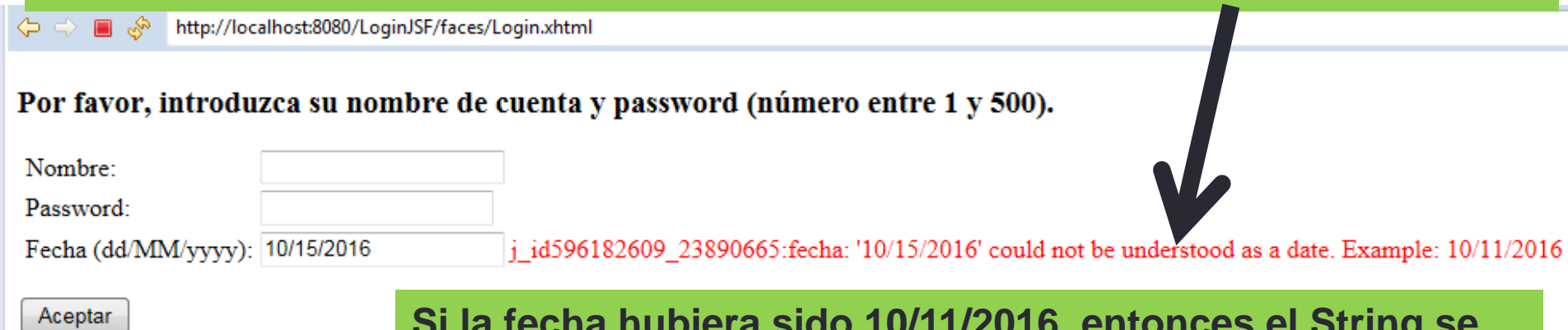
The screenshot shows a web browser window with the URL `http://localhost:8080/loginJSF/faces/Login.xhtml;jsessionid=F82C4A7BB134242DDDB5B4D6B96050AB9`. The page content includes the text "Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).". Below this text are two input fields: "Nombre:" followed by an empty text box, and "Password:" followed by an empty text box. To the right of the password field, a red error message is displayed: "j_id596182609_23890665:pass: Validation Error: Specified attribute is not between the expected values of 1 and 500.". At the bottom left of the form area, there is a button labeled "Aceptar".

El componente `f:validateLongRange` asociado al componente `h:inputSecret` (identificado como "pass") valida el contenido que se escribe en la caja de texto. Debe ser un número entre 1 y 500, y si no lo es, se escribe un mensaje de error (en color rojo) en el componente `h:message` con identificador "pass"

3.5) JSF ofrece además componentes para conversión de datos de entrada

```
<h:inputText id="fecha" value="#{login.fecha}">
  <f:convertDateTime pattern="dd/MM/yyyy"/></h:inputText>
<h:message for="fecha" style="color:red" />
```

El componente `f:convertDateTime` asociado al `h:inputText` (identificado como "fecha") convierte el String de entrada en un objeto Java Date. Y si no puede, muestra también el error.



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

Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:

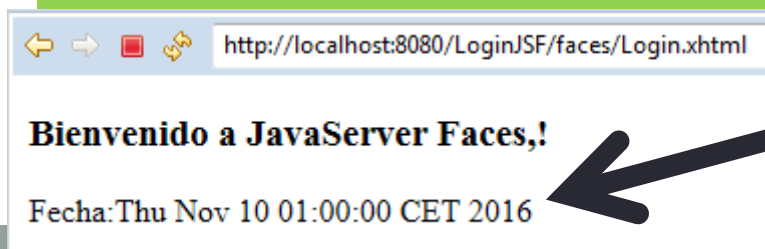
Password:

Fecha (dd/MM/yyyy): j_id596182609_23890665:fecha: '10/15/2016' could not be understood as a date. Example: 10/11/2016

Aceptar

A large black arrow points from the text above to the error message in the screenshot.

Si la fecha hubiera sido 10/11/2016, entonces el String se convertiría a un tipo Date de Java, como puede verse aquí:



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

Bienvenido a JavaServer Faces,!

Fecha: Thu Nov 10 01:00:00 CET 2016

A large black arrow points from the text above to the date in the screenshot.

3.6) Scope de los beans gestionados

- El controlador de JSF es quien se encarga de crear los objetos de las clases Bean.
- Cuándo los crea y cuándo están activos depende del alcance (scope) definido para cada bean gestionado (session, request, application,...)

```
<managed-bean>  
<managed-bean-name>login</managed-bean-name>  
<managed-bean-class>LoginBean</managed-bean-class>  
<managed-bean-scope>  </managed-bean-scope>  
</managed-bean>
```

  **session/request/application/...**

3.6) Scope de un bean gestionado: request

Con scope “request” el controlador crea un nuevo bean por cada petición HTTP desde el navegador

The screenshot shows the IDE's configuration for a Managed Bean. On the left, a tree view shows the project structure with 'login' selected under the 'request' scope. On the right, the configuration panel shows: Managed Bean name*: login, Managed Bean class*: modelo.bean.LoginBean, and Managed Bean scope*: request (circled in black). Below this, the 'Initialization' section is visible, and the Managed Bean class type is set to 'General class'.

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

Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:
Password:

Por favor, es og

Estudiante

(1) Primera petición del JSF Login.xhtml

(2) Tras la acción del botón, se carga una nueva vista Hola.xhtml

(3) Se realiza una nueva petición de JSF Hola.xhtml

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

Bienvenido a JavaServer Faces,!

Identificado como:

Bienvenido a JavaServer Faces, Kepa!

Identificado como: estudiante

(4) Se crea un nuevo bean que no “recuerda” los valores

Nota: con scope “view” crearía un nuevo bean tras pulsar el botón

3.6) Scope de un bean gestionado: request

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

Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:
Password:
Fecha (dd/MM/yyyy):

Aceptar

¿Qué mostraría al hacer click en el enlace, si scope es "request"?

Se crearía un nuevo bean que no recordaría los valores

Por favor, escoge el tipo de usuario.

Estudiante ▾

Link que accede a: <http://localhost:8080/LoginJSF/faces/Hola.xhtml>

← → http://localhost:8080/LoginJSF/faces/Hola.xhtml

Bienvenido a JavaServer Faces,!

Identificado como:
en la fecha:

El link se consigue con el siguiente tag de JSF: (h:link)

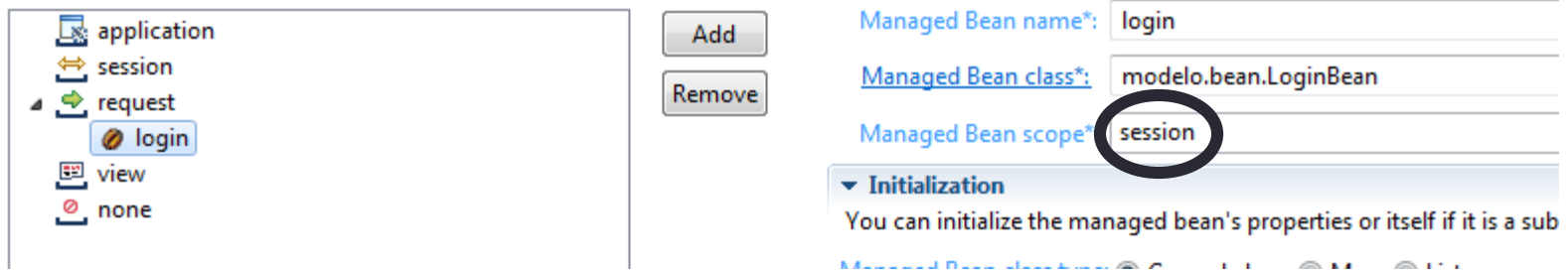
```
<h:link value="Link que accede a: http://localhost:8080/LoginJSF/faces/Hola.xhtml" outcome="Hola.xhtml"/>
```

Que se "renderiza" en el siguiente código HTML

```
<a href="/LoginJSF/faces/Hola.xhtml">Link que accede a: http://localhost:8080/LoginJSF/faces/Hola.xhtml</a>
```

3.6) Scope de un bean gestionado: session

Con scope “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



The screenshot shows the IDE's configuration for a managed bean. On the left, a tree view shows the bean is associated with the 'login' request. On the right, the configuration fields are: Managed Bean name*: login, Managed Bean class*: modelo.bean.LoginBean, and Managed Bean scope*: session (circled in black). Below these fields is an 'Initialization' section.

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

Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:

Password:

Por favor, escoge el tipo de usuario.

Estudiante

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

Bienvenido a JavaServer Faces,Kepa!

Identificado como:estudiante

¿Y si hubiera una nueva petición de página?

Una simple aplicaci... Una simple aplicaci... Login.xhtml

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

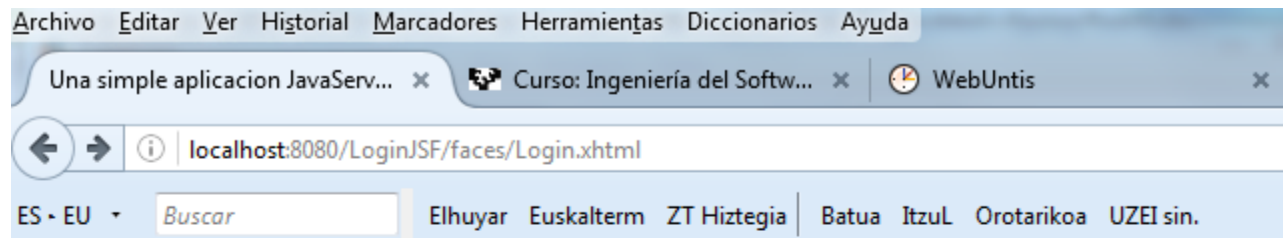
Bienvenido a JavaServer Faces,Kepa!

Identificado como:estudiante

Usa el mismo bean que ya ha creado para la sesión, por lo que sí recuerda los valores

3.6) Scope de un bean gestionado: session

¿Y si se invocara la aplicación desde otro navegador (en la misma máquina o en otra)?



Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:

Password:

Por favor, escoge el tipo de usuario.

Estudiante ▾

Entonces se crearían nuevos beans para cada una de las sesiones, y no recordaría los valores

3.6) Scope de un bean gestionado: application

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

Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:

Password:

Por favor, escoge el tipo de usuario.

Estudiante

El scope "application" permite compartir el bean entre sesiones. El bean se crea al lanzarse el servidor web y se mantiene mientras esté en marcha

Archivo Editar Ver Historial Marcadores Herramientas Diccionarios Ayuda

Una simpl... x Curso: In... WebUntis Sorting al... How can ...

localhost:8080/LoginJSF/f

egela

ES · EU Buscar Elhuyar Euskalterm ZT Hiztegia Batua ItzuL Orotarikoa U

Por favor, introduzca su nombre de cuenta y password (número

Nombre:

Password:

Por favor, escoge el tipo de usuario.

Estudiante

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

Jubilación ... Donate Spl... Una si...

Archivo Edición Ver Favoritos Herramientas Ayuda

Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:

Password:

Por favor, escoge el tipo de usuario.

Estudiante

175%

Nota: con `redisplay="true"` en `h:inputSecret` s

3.7) El ciclo de vida (CV) de una página JSF

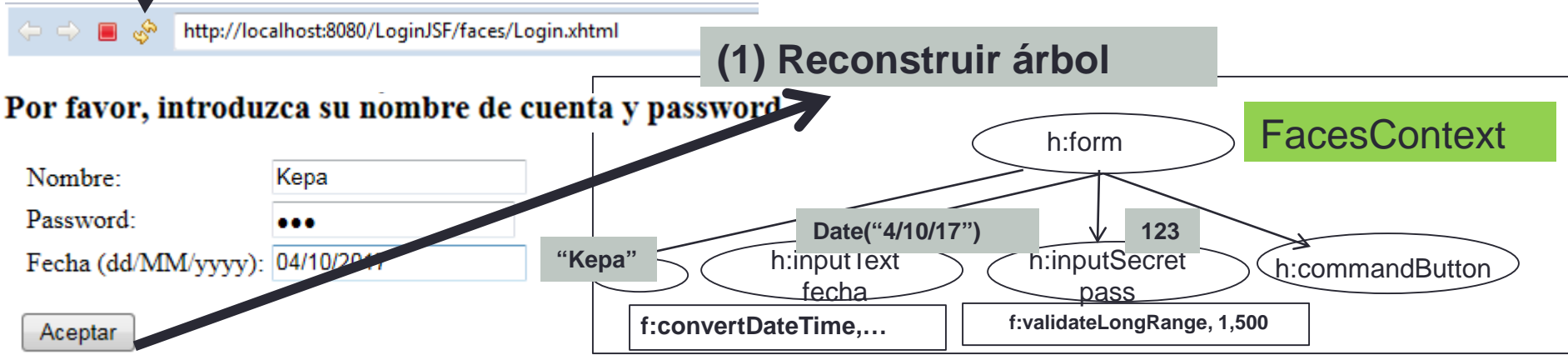
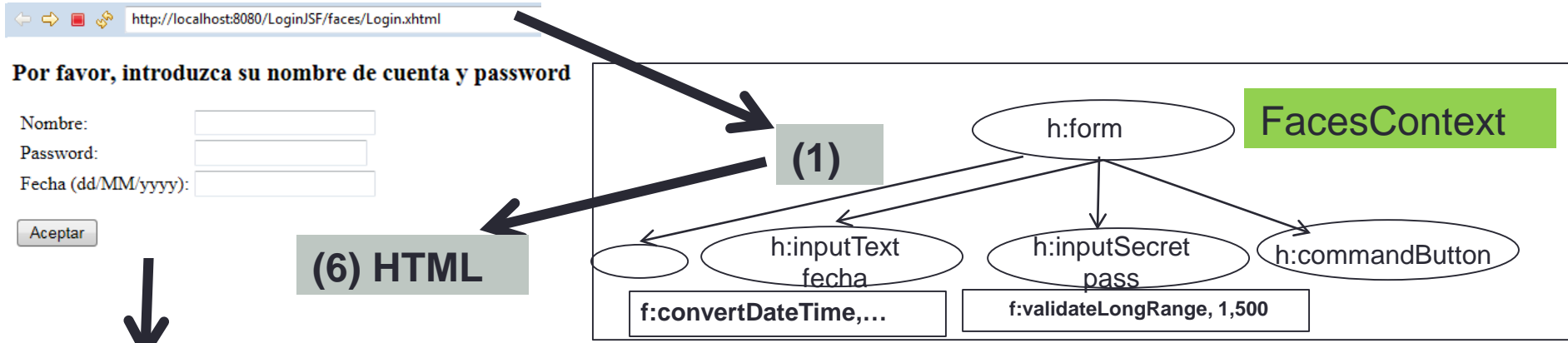
- **1) Reconstituir árbol de componentes:** el controlador (re)construye en memoria (FacesContext) la estructura de componentes de la página junto con manejadores de eventos y validadores.

La clase FacesContext permite acceder a la estructura, valores ...

- **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 pág HTML de respuesta, mostrando los componentes del árbol, valores y errores de la vista a mostrar.

Durante el CV, si sucede algún evento concreto que lo requiera o si se detectan errores, se pasa directamente a la última fase (6) sin completar todas las fases anteriores.

3.7) El ciclo de vida (CV) de una página JSF



(1) Reconstruir árbol

(2) Aplicar valores petición y convertir "4/10/17" a Date

(3) Realizar validación de 123 (1<=123<=500)

(4) Actualizar bean

(6) Renderizar HTML

**(5) Invocar aplicación (action, link,...)
¿Otra pág/vista?=>(1)**

árbol para
Hola.xhtml

@LoginBean

"Kepa"
123
Date(4,10,2017)

Con errores de validación se pasaría a (6) sin pasar por (4)(5)

3.8) Otras características concretas en JSF

- Evitar validaciones/conversiones
- Uso de Ajax para actualizar parcialmente código HTML de la vista
- Uso de CSS y JavaScript
- Errores generados en los beans (modelo)
- Convertidores String-Objetos (omnifaces)

3.8) Evitar validaciones/conversiones

← → 🚫 🏠 http://localhost:8080/LoginJSF/faces/Login.xhtml

Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:
Password:
Fecha (dd/MM/yyyy):

Aceptar Ir a fulanito

← → 🚫 🏠 http://localhost:8080/LoginJSF/faces/Login.xhtml

Fulanito, no seas pelma que tú no puedes entrar.

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

¿Qué pasa si se pulsa “Ir a fulanito” siendo el password/fecha inválidos?

Login.xhtml Fulanito.xhtml Una simple ... Una simple ... Una simple ... Una simple ... Una simple ... 42

← → 🚫 🏠 http://localhost:8080/LoginJSF/faces/Login.xhtml

Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:

Password:

j_id596182609_23890665:pass: Validation Error: Specified attribute is not between the expected values of 1 and 500.

Fecha (dd/MM/yyyy):

j_id596182609_23890665:fecha: '50/3/2016' could not be understood as a date. Example: 10/11/2016

Aceptar Ir a fulanito

Tras validar (paso 3) no se ejecutan los pasos 4 y 5 de CV sino el 6 => No procesa “action” de commandButton

3.8) Evitar validaciones/conversiones

Con atributo `immediate="true"` se indica al controlador que no realice conversiones ni validaciones, ni actualice el bean

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

Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:

Password:

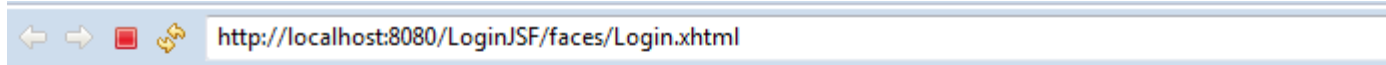
Fecha (dd/MM/yyyy):

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

Fulanito, no seas pelma que tú no puedes entrar.

Se salta los pasos 3 y 4 del CV. Va directo al 5

3.8) Uso de Ajax para actualizar parcialmente código HTML de la vista



Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500). .

Nombre:

Password:

Fecha (dd/MM/yyyy):

(1) Escribo “Alf” como nombre

(2) Escribo “748” como password

(3) Escribo “10/11/2016” como fecha

¿Cuándo se mostrará el error de que el password no está entre 1 y 500?

Respuesta: tras pulsar el botón de Aceptar (commandButton con action), que “renderiza” el HTML (paso 6 del CV) después de aplicar las validaciones (paso 4 del CV) y muestra el mensaje de error

3.8) Uso de Ajax para actualizar parcialmente código HTML de la vista

Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:

Password: j_id596182609_23890665:pass: Validation Error: Specified attribute is not between the expected values of 1 and 500.

Fecha (dd/MM/yyyy):

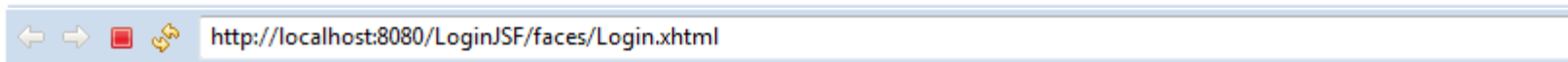
• j_id596182609_23890665:pass: Validation Error: Specified attribute is not between the expected values of 1 and 500.

```
<h:messages id="mensajes" style="color: blue"/>
```

Componente donde se muestran, en este caso, todos los mensajes y al que se le ha dado el nombre “mensajes” (opcional)

Respuesta: tras pulsar el botón de Aceptar (commandButton con action), que “renderiza” el HTML (paso 6 del CV) después de aplicar las validaciones (paso 4 del CV) y muestra el mensaje de error

3.8) Uso de Ajax para actualizar parcialmente código HTML de la vista



Por favor, introduzca su nombre de cuenta y password (número entre 1 y 500).

Nombre:

Alf

(1) Escribo "Alf" como nombre

Password:

...

(2) Escribo "748" como password

Fecha (dd/MM/yyyy):

- `j_id596182609_23890665:pass: Validation Error: Specified attribute is not between the expected values of 1 and 500.`

Aceptar

Ir a fulanito

(3) Al posicionarme en la caja de la fecha, tras haber modificado "pass" se ejecuta el código Ajax que parcialmente actualiza el código HTML de "mensajes"

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

Código Ajax en JSF

3.8) Uso de CSS

```
<h:head>  
  <h:outputStylesheet library="css" name="style.css" />  
</h:head>
```

En el h:head de un JSF se define un fichero llamado "style.css" con código CSS



El fichero debe colocarse en el subdirectorio "resources" de WebContent, en un subdirectorio con el mismo nombre que la librería ("css")

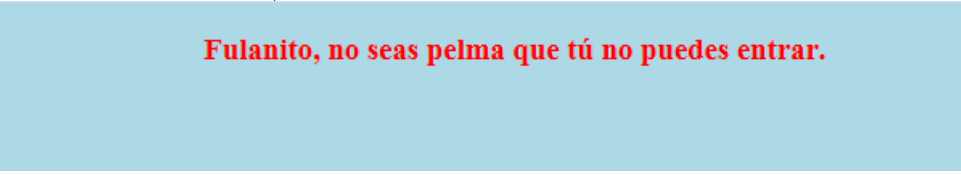
Contenido del CSS:



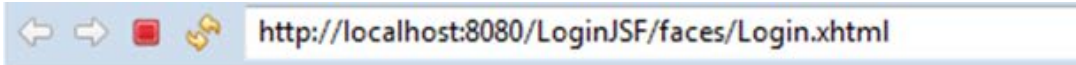
```
body {  
  background-color: lightblue;  
}  
  
h3 {  
  color: red;  
  text-align: center;  
}  
  
p {  
  font-family: verdana;  
  font-size: 20px;  
}
```

```
<h3>  
Fulanito, no seas pelma que tú no puedes entrar.  
</h3>
```

Resultado de aplicar el CSS al HTML:



3.8) Uso de JavaScript



Por favor, introduzca su nombre de cuenta y password

Nombre:
Password:
Fecha (dd/MM/yyyy):

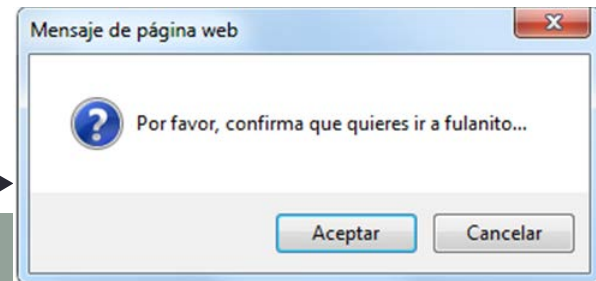
Por favor, escoge el tipo de usuario.

Estudiante

Se puede añadir código JavaScript para que se ejecute en eventos de componentes JSF

```
<h:commandButton value="Ir a fulanito" onclick="if (!confirm('Por favor, confirma que quieres ir a fulanito...')) return false"/>
```

Al pulsar el botón:



3.8) Uso de JavaScript

En el JSF Hola.xhtml

```
<h:head>  
  <h:outputScript library="js" name="infoJavaScript.js" />  
</h:head>
```

Definición del fichero JavaScript

Que debe colocarse también en "resources"

Contenido del JS:

```
function showMessage(m){  
  alert(m);  
}  
showMessage("Estos son tus datos:");
```

Y llamada a JavaScript en el botón "Aceptar" de Hola.xhtml

```
<h:commandButton value="Aceptar"  
  onclick="showMessage('No definido dónde ir')"/>
```

Ejecución:

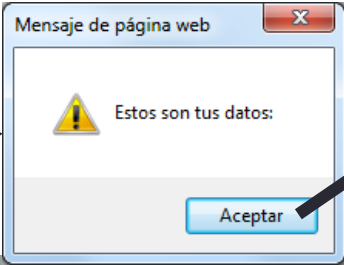
Por favor, introduzca su nombre de cuenta y password

Nombre:
Password:
Fecha (dd/MM/yyyy):

Por favor, escoge el tipo de usuario.

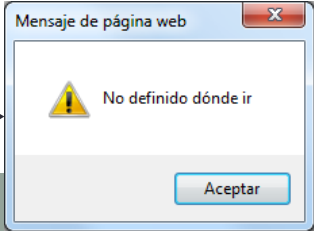
Profesor

Al cargarse Hola.xhtml



Bienvenido a JavaServer Faces, Alf!

Identificado como: profesor
con password:123
Fecha: Tue Nov 07 01:00:00 CET 2017



3.8) Errores generados en los beans (modelo)

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

Por favor, introduzca su nombre de cuenta y password

Nombre:
Password:
Fecha (dd/MM/yyyy):

Aceptar Ir a fulanito

Por favor, escoge el tipo de usuario.

Estudiante

Errores en tiempo de ejecución detectados en el modelo (ej: red o BD desconectada) o porque afectan a varios componentes de entrada (ej: nombre y password de distinta longitud)

El error se le pasa a FacesContext (ver Ciclo Vida JSF)

```
public String comprobar() {  
    if (nombre.length()!=password.length()){  
        FacesContext.getCurrentInstance().addMessage(null, new FacesMessage("Error: tamaño de nombre y password no coinciden."));  
        return null;  
    }  
    if (nombre.equals("fulanito")) return "error";  
    else return "ok";  
}
```

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

Por favor, introduzca su nombre de cuenta y password

Nombre:
Password:
Fecha (dd/MM/yyyy):

Aceptar Ir a fulanito

Para mostrar esos mensajes globales viene bien el componente h:messages

```
<h:messages id="mensajes" style="color: blue" />
```

• Error: tamaño de nombre y password no coinciden.

3.8) Uso de PrimeFaces

PrimeFaces es implementación open source con varias extensiones

https://www.primefaces.org/docs/guide/primefaces_user_guide_6_1.pdf

QUERY AVAILABILITY

Rural house:

1: Ezkio

Nov 2017

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

First day:

Number of nights: 0

Mostrar Ofertas

Volver

calendar de Prime Faces

```
<h:outputText value="First day:" />
<p:calendar id="arrDay" value="#{queryAvailabilityBean.arrivalDay}" navigator="true" mode="inline">
</p:calendar>
```


3.8) Uso de PrimeFaces

No olvidar:

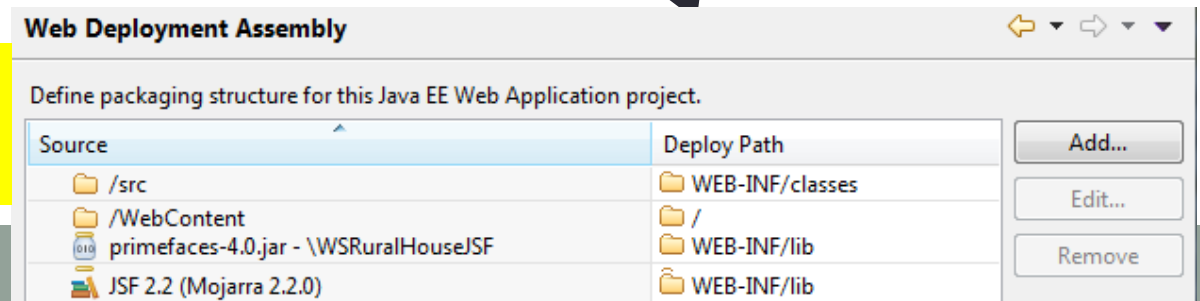
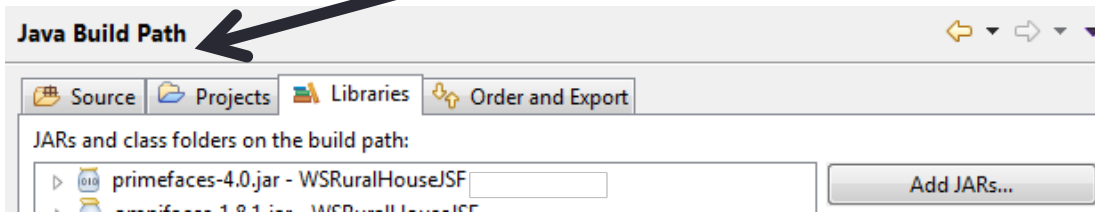
Añadir al JSF la librería de etiquetas p y usar `<h:head>` en vez de `<head>`

```
<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"
      xmlns:p="http://primefaces.org/ui">
```

```
<h:head>
  <title>Query Availability</title>
</h:head>
```



Añadir la librería al proyecto y al despliegue de la aplicación Web



La librería de etiquetas <http://primefaces.prime.com.tr/ui> es específica para Prime Faces 2.X Usad: <http://primefaces.org/ui>

3.8) Uso de PrimeFaces

Si queréis que la fecha seleccionada se vea antes de ejecutar un “action” de commandButton

QUERY AVAILABILITY

Rural house: 1: Ezkio

First day:

Number of nights:

Mostrar Ofertas

Ofertas						
Num oferta	Prim					

Calendar: Nov 2017

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

QUERY AVAILABILITY

Rural house: 1: Ezkio

First day: 14/11/17

Number of nights: 0

Mostrar Ofertas Volver

Ofertas disponibles en esas fechas				
Num oferta	Primer día	Último día	Precio	Casa rural

```
<p:calendar id="arrDay" value="#{queryAvailabilityBean.arrivalDay}" navigator="true" mode="popup">  
</p:calendar>
```

Se puede usar el mode “popup” de p:calendar

3.8) Uso de PrimeFaces

Si queréis que la fecha seleccionada se vea antes de ejecutar un “action” de commandButton

QUERY AVAILABILITY

Rural house:

First day:

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Number of nights:

14/11/2017

O bien se puede hacer mediante los eventos Ajax proporcionados por PrimeFaces para el componente p:calendar

Se define una acción Ajax para que cuando se seleccione una fecha en el calendario se actualice la etiqueta FirstDayLabel

```
<p:calendar id="arrDay" value="#{queryAvailabilityBean.arrivalDay}" navigator="true" mode="inline">
  <p:ajax event="dateSelect" update="FirstDayLabel"/>
</p:calendar>

<h:outputLabel id="FirstDayLabel" value="#{queryAvailabilityBean.arrivalDay}">
<f:convertDateTime pattern="dd/MM/yyyy" timeZone="CET"/> </h:outputLabel>
```

Acción Ajax

3.8) Uso de PrimeFaces

Pero la acción Ajax permite más cosas como...

ejecutar un método “oyente” o “listener” cuando suceda el evento (seleccionar fecha).

```
<p:calendar id="arrDay" value="#{queryAvailabilityBean.arrivalDay}" navigator="true" mode="inline">  
  <p:ajax event="dateSelect" listener="#{queryAvailabilityBean.onDateSelect}" update="FirstDayLabel"/>  
</p:calendar>
```

Método “listener”

```
public void onDateSelect(SelectEvent event) {  
  // FacesContext facesContext = FacesContext.getCurrentInstance();  
  // SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");  
  // facesContext.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO, "Date Selected", format.format(event.getObject())));  
  System.out.println("Evento: " + event.getObject());  
}
```

Se puede obtener info del contexto del evento

QUERY AVAILABILITY

Rural house:

1: Ezkio

First day:

Nov		2017				
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Problems Javadoc Declaration Console Git Staging History Ser
Pivotal tc Server Developer Edition v3.1 [Pivotal tc Server] C:\Program Files\Java\jre1.8.0_111\bin\java
Evento: Wed Nov 15 00:00:00 CET 2017

Number of nights: 0

Mostrar Ofertas

Volver

15/11/2017

3.8) Convertidores String-Objetos (omnifaces)

```
<h:selectOneMenu value="#{setAvailabilityBean.casa}">
    <f:selectItems value="#{setAvailabilityBean.casas}" />
</h:selectOneMenu>
```

```
public List<RuralHouse> getCasas() {
    casas=facadeBL.getAllRuralHouses();
    return casas;
}
```

SET AVAILABILITY

List of houses:

First day:

Price:

Last day:

Las casas se obtienen invocando a la lógica del negocio

```
facadeBL=new FacadeImplementationWS();
```

Mejor: que haya una única instancia de FacadeImplementationWS

```
public static ApplicationFacadeInterfaceWS facadeBL;

public QueryAvailabilityBean(){
    facadeBL=FacadeBean.getBusinessLogic();
}
```

3.8) Convertidores String-Objetos (omnifaces)

```
public class FacadeBean {  
  
    private static FacadeBean singleton = new FacadeBean( );  
  
    private static ApplicationFacadeInterfaceWS facadeInterface;  
  
    private FacadeBean(){  
        try {  
  
            facadeInterface=new FacadeImplementationWS();  
  
        } catch (Exception e) {  
            System.out.println("FacadeBean: error creando la lógica del negocio: "+e.getMessage());  
        }  
    }  
    public static ApplicationFacadeInterfaceWS getBusinessLogic( ) {  
  
        return facadeInterface;  
    }  
}
```

Se trata de una clase singleton FacadeBean, que sólo puede tener una instancia (constructor private), que es la que crea la instancia de FacadeImplementationWS que devuelve con getBusinessLogic. FacadeImplementationWS no es singleton, pero así sólo creamos una instancia de ella

3.8) Convertidores String-Objetos (omnifaces)

```
<h:selectOneMenu value="#{setAvailabilityBean.casa}">
    <f:selectItems value="#{setAvailabilityBean.casas}" />
</h:selectOneMenu>
```

```
public List<RuralHouse> getCasas() {
    casas=facadeBL.getAllRuralHouses();
    return casas;
}

public void setCasa(RuralHouse casa) {
    this.casa = casa;
}
```

SET AVAILABILITY

List of houses:

First day:

Price:

Last day:

¿Qué sucedería al seleccionar una casa de la lista desplegable?

Se produce un error porque lo que se envía al BEAN es un String (ej: 1:Ezkio) y no un objeto de RuralHouse, que es lo que espera el parámetro “casa” del método “setCasa”

HAY QUE INDICAR CÓMO CONVERTIR OBJETOS EN STRINGS Y VICEVERSA

3.8) Convertidores String-Objetos (omnifaces)

La clase siguiente de omnifaces lo hace:

```
converter="omnifaces.SelectItemsConverter"
```

```
<h:selectOneMenu value="#{setAvailabilityBean.casa}" converter="omnifaces.SelectItemsConverter">  
  <f:selectItems value="#{setAvailabilityBean.casas}"/>  
</h:selectOneMenu>
```

Para que funcione hay que:

1) Añadir la librería al proyecto y al despliegue de la aplicación Web

The screenshot shows two Eclipse IDE configuration windows. The 'Java Build Path' window on the left has the 'Libraries' tab selected, showing 'omnifaces-1.8.1.jar - WSRuralHouseJSF' in the list. The 'Web Deployment Assembly' window on the right shows a table with 'Source' and 'Deploy Path' columns. The 'omnifaces-1.8.1.jar' is listed in the 'Source' column, and its corresponding 'WEB-INF/lib' directory is listed in the 'Deploy Path' column. Two black arrows point from the 'omnifaces-1.8.1.jar' entry in the 'Libraries' list to the 'omnifaces-1.8.1.jar' entry in the 'Web Deployment Assembly' table.

Source	Deploy Path
/src	WEB-INF/classes
/WebContent	/
db4o-8.0.249.16098-all-java5.jar - \	WEB-INF/lib
JSF 2.2 (Mojarra 2.2.0)	WEB-INF/lib
omnifaces-1.8.1.jar - \WSRuralHou:	WEB-INF/lib

Nota: Usar omnifaces 1.8 para JSF 2.0

3.8) Convertidores String-Objetos (omnifaces)

2) Además, se necesita que la clase sobre la que actúa el conversor (RuralHouse) tenga una buena implementación de toString, hashCode y equals, porque son utilizados por omnifaces.SelectItemsConverter

```
public String toString() {  
    return this.houseNumber + ": " + this.city;  
}
```

Lo que mostrará en el
selectOneMenu



```
@Override  
public boolean equals(Object obj) {  
    RuralHouse other = (RuralHouse) obj;  
    if (this == obj)  
        return true;  
    if (obj == null)  
        return false;  
    if (getClass() != obj.getClass())  
        return false;  
    if (houseNumber != other.houseNumber) // ASÍ PU  
    if (!houseNumber.equals(other.houseNumber))  
        return false;  
    return true;  
}
```

Cuando se selecciona un elemento en la lista, entonces comprueba si ese elemento es igual (usando el método equals) alguno de los elementos que están en la lista (que vuelve a obtener con “getCasas” para comprobar que todavía es una opción válida)

3.8) Convertidores String-Objetos (omnifaces)

Por lo tanto: lo siguiente haría que se volvieran a traer todas las casas de la BD tras seleccionar una opción en la lista desplegable, y podría dar un error si el método “equals” no está bien definido

```
public List<RuralHouse> getCasas() {  
    casas=facadeBL.getAllRuralHouses();  
    return casas;  
}
```

Si se desea que el contenido de las casas disponibles no cambie durante la “vida del bean” entonces se podría hacer así:

```
public QueryAvailabilityBean(){  
    facadeBL=FacadeBean.getBusinessLogic();  
    casas=facadeBL.getAllRuralHouses();  
}  
  
public List<RuralHouse> getCasas() {  
    return casas;  
}
```

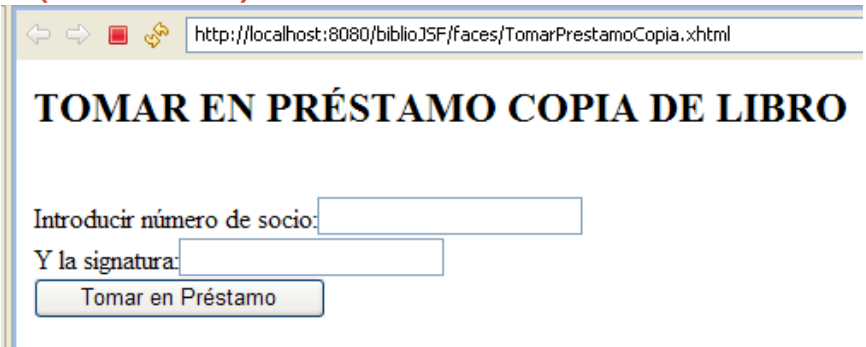
Problema solucionado: en el JSF no aparecen explícitamente llamadas explícitas a las vistas (JSFs) y modelo (el bean)

```

...
<f:view>
<h:form>
Introducir número de socio:
<h:inputText
value="#{tomarPrestamoCopia.numSocio}" />
<br/> Y la signatura:
<h:inputText value="#{tomarPrestamoCopia.signatura}" />
<br/>

```

La vista JSF



El bean (mod+contr.)

```

<h:commandButton action="#{tomarPrestamoCopia.getCopia}"
value="Tomar en Préstamo"></h:commandButton>
</h:form>
</f:view>

```

```

package beans;
public class TomarPrestamoCopia {
public String getCopia() {
...return "No hay copias libres";
...return "ok";}
}

```

```

</h:body>
</html>
<faces-config ...
<managed-bean>
<managed-bean-name>tomarPrestamoCopia</managed-bean-name>
<managed-bean-class>beans.TomarPrestamoCopia</managed-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean> ...
<navigation-rule>
<display-name>TomarPrestamoCopia.xhtml</display-name>
<from-view-id>/TomarPrestamoCopia.xhtml</from-view-id>
<navigation-case>
<from-outcome>ok</from-outcome>
<to-view-id>/TomadaCopiaEnPrestamo.xhtml</to-view-id>...

```

Configuración de beans y navegación entre vistas, según acciones

Referencias

- <http://www.tutorialspoint.com/jsf/index.htm>
- <http://www.coreservlets.com/JSF-Tutorial/>
- <https://javaee.github.io/javaxserverfaces-spec/>
- Las siguientes en castellano:
- <http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/sesion01-apuntes.html>
- <https://www.apuntesdejava.com/p/tutorial-jsf-22.html>