Curso Spring – RichFaces – Lo esencial de JSF

Marzo 2011 – JoeDayz

Susan Inga

Contenido

- ✓ Introducción
- ✓ ¿Qué aprenderemos?
- ✓ Ganaremos experiencia en…
- ✓ Introducción a JSF
- ✓ Características principales
- ✓ Comparativas…
- ✓ Implementación
- ✓ Ciclo de vida de una petición JSF
- ✓ Configurando la infraestructura de JSF
- ✓ Paso 1: Evaluando el estado inicial de la aplicación web
- ✓ Paso 2: Verificando la configuración del backend
- ✓ Paso 3: Usando Facelets
- ✓ Paso 4: Configurando el FacesServlet
- ✓ Paso 5: Revisando el faces-config.xml
- ✓ Paso 6: Implementando los beans de gestión
- ✓ Paso 7: Administrando los beans
- ✓ Paso 8: Revisando las plantillas de la aplicación
- ✓ Paso 9: Listando las cuentas
- ✓ Paso 10: Mostrando detalles de las cuentas
- ✓ Paso 11: Registrando vistas

Reward Dining

Lo esencial de JSF

Introducción

- En este laboratorio implementarás controladores básicos para invocar funcionalidades de la aplicación y visualizar resultados para el usuario con tecnología JSF.

¿Qué aprenderemos?

- ✓ ¿Cómo construir la infraestructura que requiere una aplicación con JSF?
- ✓ ¿Cómo exponer beans como endpoints mapeados hacia URLs de la aplicación web?

Ganaremos experiencia en…

- ✓ FacesServlet
- ✓ <navigation-rule/>
- ✓ Facelets

Introducción a JSF

- El objetivo de la tecnología JavaServer Faces es desarrollar aplicaciones web de forma parecida a como se construyen aplicaciones locales con Java Swing, AWT o cualquier otra API similar.

- Para el desarrollo Web tradicionalmente se usan páginas JSP que atienden peticiones y en base a ellas devuelven respuestas hacia el cliente en formato HTML para mostrar los resultados solicitados.

Introducción a JSF

- JavaServer Faces pretende simplificar la construcción de las aplicaciones Web brindando un entorno de trabajo que gestiona las acciones generadas por el usuario en una página HTML. Para luego traducirlas a eventos que son enviados al servidor con el objetivo de regenerar la página original y pero mostrando los cambios pertinentes provocados por dichas acciones.

- Al final, la intención es desarrollar aplicaciones Java en las que el cliente no es una ventana de la clase JFrame o similar, sino una página HTML.

Características principales

- Constituye un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones web basadas en plataforma Java y en el patrón MVC.
- Los principales componentes una API y una implementación de referencia para:
  - Representar componentes de interfaz de usuario y administrar su estado.
  - Manejar eventos, validar en el lado del servidor y convertir datos.
  - Definir la navegación entre páginas
  - Soportar internacionalización y accesibilidad.
  - Proporcionar extensibilidad para todas las características anteriores.
  - Una librería de etiquetas JavaServer Pages personalizadas para dibujar componentes UI dentro de una página JSP.

Características principales

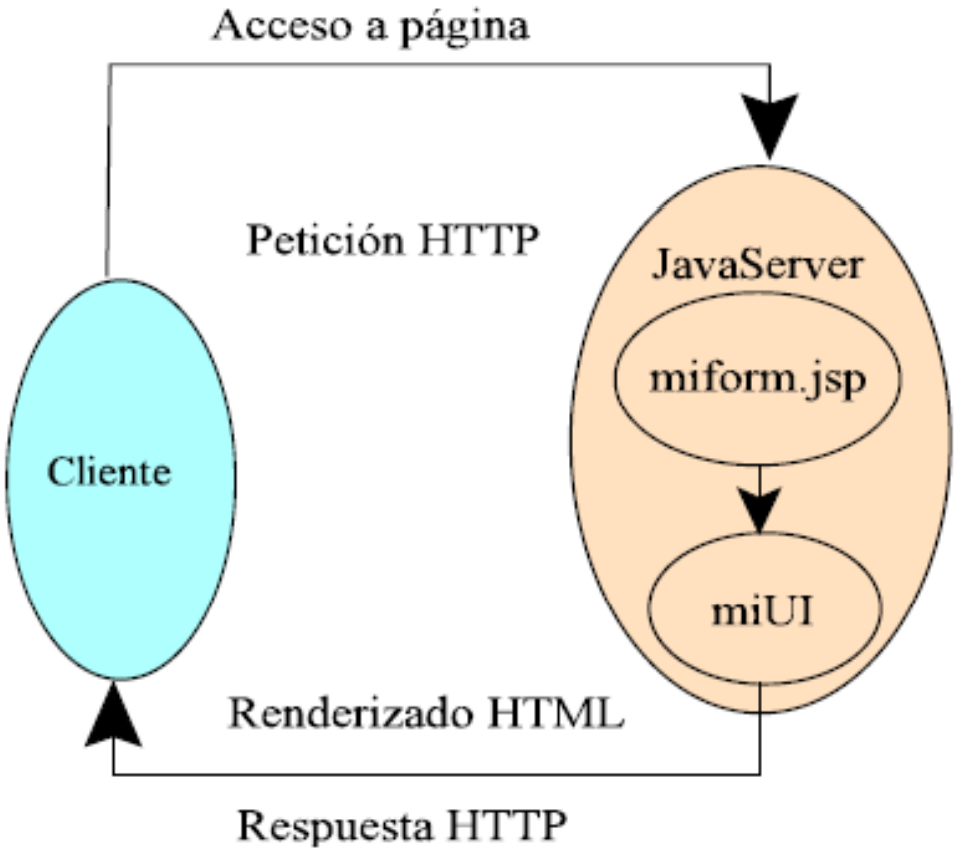


Diagrama de una aplicación JSF

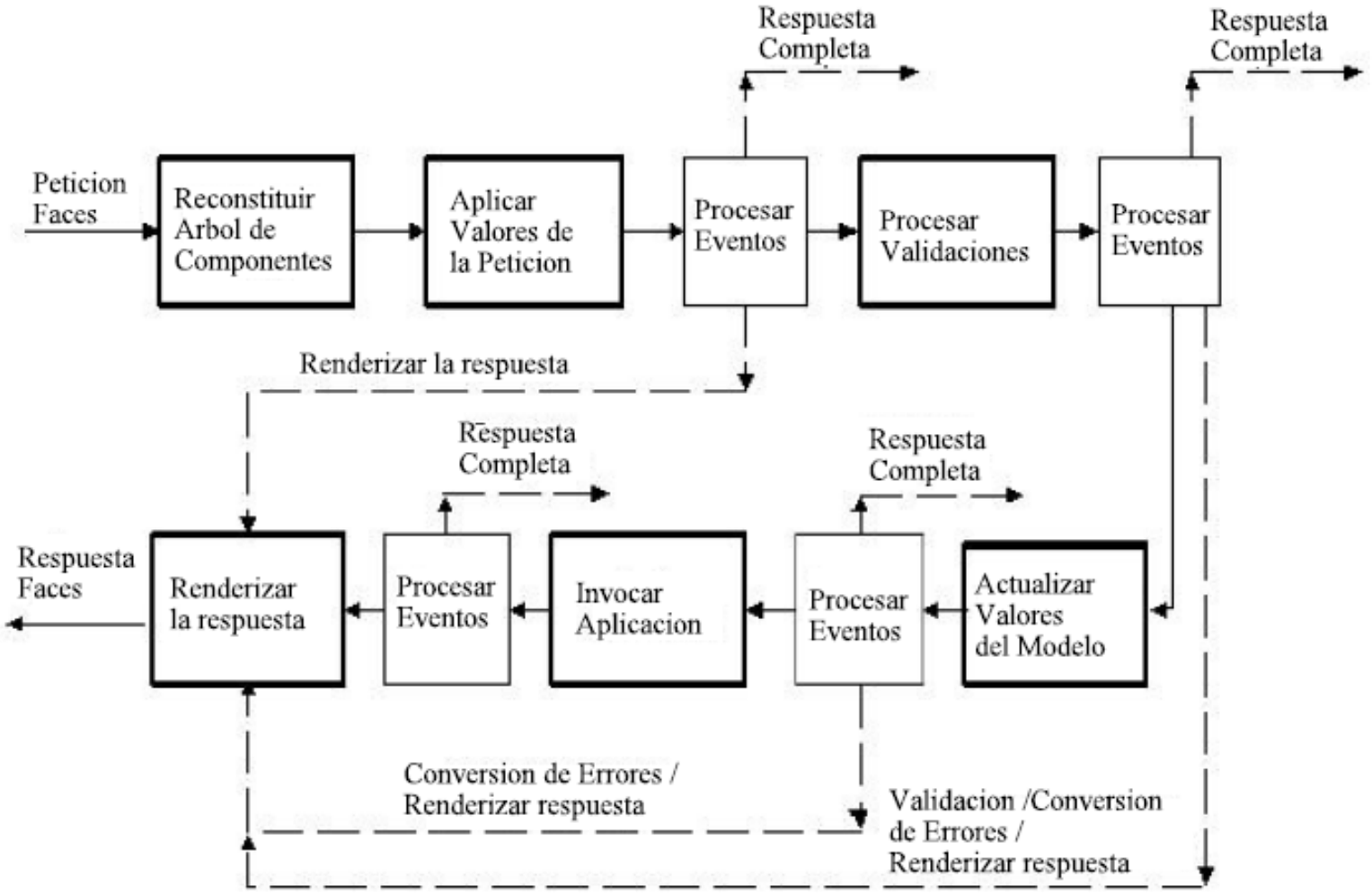
Comparativas…

- ✓ Al igual que muchos frameworks de vista, JSF pretende normalizar y sobretodo estandarizar el desarrollo Web.
- ✓ Es posterior a Struts, por ello ha usado toda la experiencia de ese framework y tiene como líder de especificación al mismo creador de Struts.
- ✓ Una de sus principales ventajas es el poder hacer uso de herramientas de diseño visual.
- ✓ Otra de las ventajas de que sea una especificación estándar es que hay distintos fabricantes. Lo cual permite tener una cantidad razonable de opciones para su uso.

Implementación

- En el desarrollo del laboratorio podremos observar que la mayoría de funcionalidades consta y/o requiere de:
  - Una página donde mostrar lo solicitado.
  - Dicha página consta de componentes para mostrar o capturar información.
  - Un bean que maneje los datos a renderizar ó
  - Un bean (clase Java) en la que sus campos son accedidos siguiendo métodos getter y setter para la captura de información.
  - Un archivo de configuración para la aplicación que contiene recursos del bean y las reglas de navegación. Por defecto, este archivo se denomina face-config.xml.
- La aplicación requerirá de:
  - Archivos necesarios para el servlet el web.xml e index.html que redireccionan el usuario a la dirección correcta para la página de entrada en el sistema.

Ciclo de vida de una petición JSF



Ciclo de vida petición-respuesta de una página JSF

Configurando la infraestructura de JSF

- ✓ Ahora si podemos iniciar con la configuración de la aplicación.

Paso 1: Evaluando el estado inicial de la aplicación web

- La aplicación web que estas desarrollando debe permitir a los usuarios ver un resumen de todas las cuentas del sistema, luego poder ver los detalles acerca de una cada una de las cuentas, en particular de una elegida previamente. La funcionalidad debería quedar como se muestra a continuación:

Figura 1: GET pages/accountSummary.jsf: Ver la lista de todas las cuentas y con link en el nombre para ver los detalles

Paso 1: Evaluando el estado inicial de la aplicación web

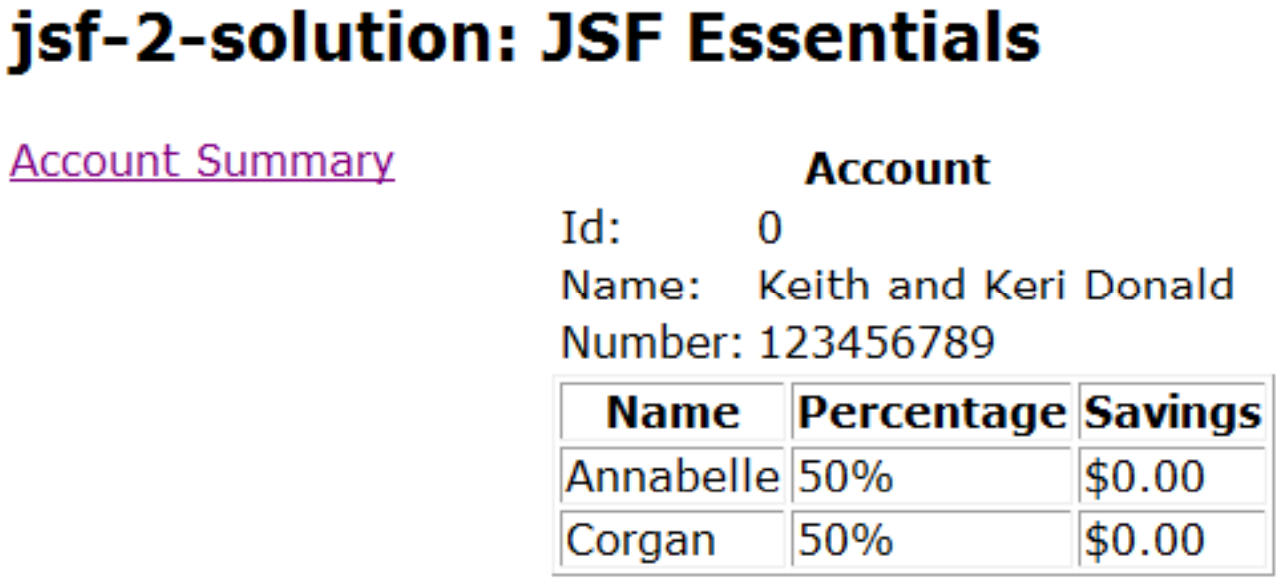


Figura 2: GET /pages/accountDetails.htm?entityId=0: Ver los detalles de la cuenta con id '0'

Paso 1: Evaluando el estado inicial de la aplicación web

- ✓ Actualmente, esta funcionalidad está implementada a medias. En este primer paso evaluaremos el estado inicial de la aplicación.
- ✓ Empieza por desplegar la aplicación para este proyecto como está. Una vez desplegado, verás que la aplicación muestra un error de tipo 404, lo cual nos indica que el request no ha podido ser procesado. Hay muchas razones por las cuales ha sucedido esto.

Paso 2: Verificando la configuración del backend

- ✓ Rápidamente evalúa la configuración inicial del "backend" de esta aplicación. Para hacer esto, abre el archivo web.xml del directorio src/main/webapp/WEB-INF.
- ✓ Notarás que un ContextLoaderListener ya ha sido definido. Este listener esta configurado para cargar todos los recursos en esta capa de la aplicación que se encuentran declarados en application-config.xml. Abre este archivo para ver los beans que levantará esta capa. Puedes visualizar este archivo usando el Spring IDE.
- ✓ Como puedes observar, aún no se le ha indicado al ContextLoaderListener que application-config.xml es el archivo que debe usar para cargar los recursos. Hay que indicárselo completando el TODO 01.
- ✓ Con ello ya hemos configurado que al iniciar nuestra aplicación nuestro Web Application Context de Spring cargue al desplegar nuestro sistema.

Paso 3: Usando Facelets

- Facelets es un poderoso sistema de plantillas que permite definir vistas JSF usando plantillas tipo HTML, reduciendo la cantidad de código necesario para integrar componentes dentro de una vista y no depende de un contenedor web.
- Si se ha trabajando con JSPs, trabajar con facelets es sumamente parecido.
- Para configurar el uso de facelets, observa el parámetro javax.faces.DEFAULT_SUFFIX. Este debe indicarle a nuestra aplicación cual va a ser el sufijo a usar de nuestras páginas. Completa el TODO 02 con .xhtm.

Paso 4: Configurando el FacesServlet

- El elemento principal de la infraestructura central de una aplicación construida con JSF es el FacesServlet.
- Este es el front controller para la aplicación JSF. El recibe todos los request para la aplicación e inicializa los componentes JSF antes que el JSP o el XHTML sea renderizado.
- Verifica en el web.xml y navega hacia la definición Faces Servlet. Nota que este es un FacesServlet y que todos los requests *jsf le son mapeados a él.

Paso 5: Revisando el faces-config.xml

- El archivo faces-config.xml permitirá configurar la aplicación, administrar los beans, conversores, validadores y sobretodo las reglas de navegación.
- Además, permite la comunicación entre el modelo y el sistema JSF.
- Navega hacia la carpeta WEB-INF y encontrarás la plantilla del archivo faces-config.xml. Ábrelo.

```
<application>
    <locale-config>
        <default-locale>en</default-locale>
    </locale-config>
    <message-bundle>validation</message-bundle>
    <el-resolver>org.springframework.web.jsf.el.SpringBeanFacesELResolver</el-resolver>
</application>
```

- En el vemos especificado como EL-resolver a SpringBeanFacesELResolver, esto nos ayudará a usar Expression Languaje para resolver los beans que dispone Spring en su contexto definido en el archivo application-context.xml.
- Asimismo, le indicamos el archivo de propiedades a usar en la aplicación: validation.properties.

Paso 6: Implementando los beans de gestión

- ✓ Las aplicaciones JSF usan un bean por cada página que contengan. Este bean debe definir las propiedades y los métodos asociados con los componentes de la interfaz.
- ✓ Navega hacia la carpeta src/main/java y en el paquete accounts.web encontrarás la clase BasePage.java. Revísala.
- ✓ Es una clase utilitaria para hacer uso de métodos comunes como obtener un parámetro del request, agregar mensajes, errores, etc.
- ✓ Heredando de esta clase, para obtener todos sus beneficios, debemos crear el controlador (o bean) que nos permita realizar el listado de las cuentas tal cual se ha solicitado.
- ✓ En el paquete accounts.web crear la clase AccountController e implementa un método que devuelva un listado de cuenta haciendo uso del AccountManager ya implementado y declarado en el application-config.xml.

Paso 7: Administrando los beans

- Para comunicar el controlador (bean) que acabamos de crear con la vista a la cual debe enviar información, debemos declararlo en el archivo faces-config.xml.

```
<managed-bean>
    <description>account</description>
    <managed-bean-name>account</managed-bean-name>
    <managed-bean-class>accounts.web.AccountController</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
    <managed-property>
        <property-name>accountManager</property-name>
        <value>#{accountManager}</value>
    </managed-property>
</managed-bean>
```

Paso 8: Revisando las plantillas de la aplicación

- ✓ Navega hacia la carpeta webapp/pages/template y revisa el contenido.
- ✓ Puedes observar como se construye la plantilla que se va a respetar en toda la aplicación.
- ✓ Ahora navega hacia la webapp/pages y revisa el archivo welcome.xhtml, es la página a visualizarse al iniciar la aplicación.
- ✓ Una vez terminados los cambios, procede a levantar la aplicación.
- ✓ Si todo salió correcto, continua con el siguiente paso!

Paso 9: Listando las cuentas

- ✓ En la misma carpeta encontrarás el archivo accountSummary.xhtml, aquí es donde debemos hacer los ajustes necesarios para renderizar la información que enviamos desde nuestro AccountController.
- ✓ Una vez terminados los cambios, procede a levantar la aplicación.
- ✓ Si encuentras errores, consulta con tu instructor que puede suceder.
- ✓ Si todo salió correcto, continua con el siguiente paso.

Paso 10: Mostrando detalles de las cuentas

- ✓ Ahora que nuestra app ya funciona y se encuentra totalmente configurada, procedemos a implementar la segunda funcionalidad requerida.
- ✓ En nuestra clase AccountController añadimos un método que nos permita capturar el id de una cuenta y con el realicemos una búsqueda de la cuenta solicitada.
- ✓ Una vez encontrada la cuenta buscada, la asignamos a un atributo de nuestra clase y devolvemos el nombre de la vista en la que deseamos se renderize la información de la cuenta.
- ✓ Esta vista en accountDetails.xhtml. Revisa dicha página.
- ✓ Ahora, dirígete hacia el archivo faces-config.xml

Paso 11: Registrando vistas

- En el archivo faces-config.xml debemos registrar la regla de navegación que nos permita identificar a la cadena “accountsDetails” que devolvemos en nuestro nuevo método de AccountController, para que en la aplicación se pueda renderizar dicha vista.

```
<navigation-rule>
    <from-view-id>/*</from-view-id>
    <navigation-case>
        <from-outcome>accountDetails</from-outcome>
        <to-view-id>/pages/accountDetails.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
```

- Ahora completa la información en accountDetails.xhtml.
- Levanta la aplicación si todo está correcto, has terminado el lab!

Enjoy it!