Especialista en Aplicaciones y Servicios Web con Java Enterprise

JSF Sesión 1

Introducción a JavaServer Faces

Índice

- ¿Qué es JavaServer Faces?
- Configuración
- Ciclo de proceso de una página JSF
- Componentes para la GUI
- Validación, conversión y tratamiento de errores
- Modelo de datos (beans de respaldo: *backing* beans)
- Acciones y navegación entre páginas
- Resumen

Struts y JSF

- Struts nace como framework para implementar MVC en aplicaciones web
- JSF es para el desarrollo de componentes web, pero…
- Solapamientos Struts-JSF
  - Componentes web (punto fuerte en JSF)
  - Validación-conversión de datos
  - MVC (punto fuerte en Struts): beans + acciones + navegación
- El desarrollador original de Struts y el líder de la especificación de JSF son la misma persona (un tal Craig McClanahan)
  - JSF y Struts son compatibles

JavaServer Faces ofrece

- Un conjunto estándar de componentes gráficos (widgets) para la web
- Vinculación entre componentes y *beans* de Java
- Soporte para validación y conversión automática de datos
- Modelo de navegación entre páginas
- Modelo de eventos similar al de Swing
- Soporte para la internacionalización

Configurar una aplicación JSF

- Una aplicación JSF es una aplicación web estándar
- Instalar JSF (implementación de referencia de Sun, versión 1.1.01) en WEB-INF/lib
  - Librerías JSF (JAR): `jsf-api.jar`, `jsf-impl.jar`
  - Librerías commons (JAR)
  - JSTL (hay que utilizar la versión apropiada para nuestro Tomcat): `jstl.jar`, `standard.jar`
- Configurar JSF
  - `web.xml`: redirigir las peticiones al servet `FacesServlet`
  - `faces-config.xml`: fichero de configuración propio

Probar ejemplos

- jsf-hola-mundo
- jsf-adios-mundo
- jsf-guess-number

web.xml para JSF

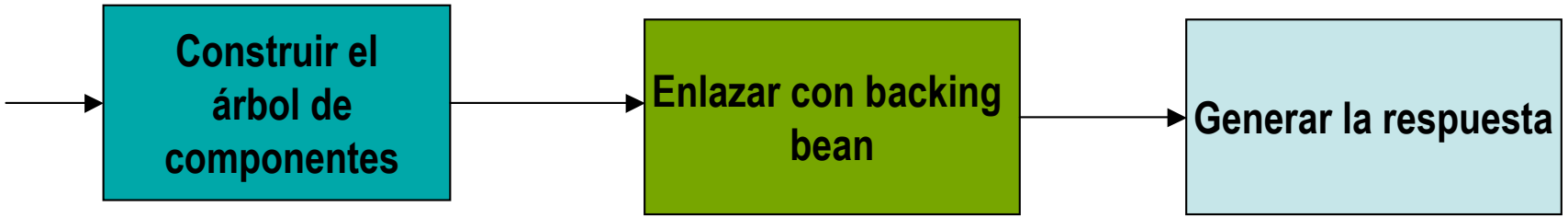
```
<web-app>
   ...
   <!-- Definición del Servlet que procesa las páginas JSF -->
   <servlet>
      <servlet-name>Faces Servlet</servlet-name>
      <servlet-class>javax.faces.webapp.FacesServlet</servlet-
   class>
      <load-on-startup> 1 </load-on-startup>
   </servlet>
   <!-- Mapeado del servlet con la URL correspondiente -->
   <servlet-mapping>
      <servlet-name>Faces Servlet</servlet-name>
      <url-pattern>*.jsf</url-pattern>
   </servlet-mapping>
   ...

</web-app>
```

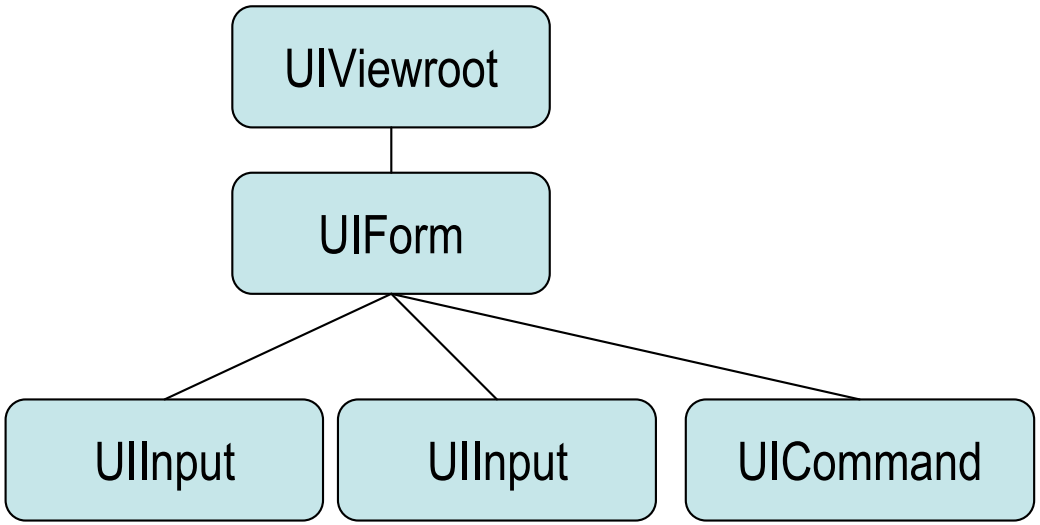
Ejemplo de página JSF

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<html>
<head> <title>Ejemplo de página JSF</title> </head>
   <body>
      <f:view>
          <h:form>
            <h:inputText value="#{userBean.name}"/>
            <h:inputSecret value="#{userBean.password}"/>
            <h:commandButton value="Enter"
                  action="#{userBean.doLogin}"/>
          </h:form>
      </f:view>
   </body>
</html>
```

- Acción
- Backing bean
- GUI

El ciclo de proceso de una página JSF: 1er acceso






```
<form>
  <input type="text" value=""/>
  <input type="secret" value=""/>
 <input type="submit" value="Entrar"/>
<form>
```

Componentes para la interfaz

- Basados en *taglibs*
  - Core
  - HTML

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

- Todas las etiquetas JSF deben ir dentro de un `f:view`

```
<f:view>
        <!-- definición de componentes JSF -->
</f:view>
```

Componentes para la interfaz

- Los nombres de los componentes siguen la sintaxis `tipoRenderer`
  - `inputText, outputText, commandLink,…`
- Formularios (excepción a la regla anterior)

  <h:form> ... </h:form>
- Comandos
  - `commandButton:` botón, `commandLink:` enlace (tanto botones como enlaces deben estar dentro de formularios)
  - `action`: método a ejecutar

```
<h:commandButton value= "Entrar" action="#{gestorUsuarios.doLogin}"/>
<h:commandLink action="#{gestorUsuarios.doPersonal}">
    <h:outputText value="Mi página"/>
</h:commandLink>
```

Componentes de entrada

- Algunos ejemplos
  - Entrada de texto

```
<h:inputText value="#{userBean.name}"/>
<h:inputSecret value="#{userBean.password}"/>
```

  - Conjunto de botones de radio

```
<h:selectOneRadio value="#{userBean.estado}">
                <h:selectitem itemLabel="soltero"
                  itemValue="soltero"/>
                <h:selectitem itemLabel="casado"
                  itemValue="soltero"/>
 </h:selectOneRadio >
```

Beans de respaldo

- Similares a los *beans* de JSP, con propiedades y métodos get y set
- Pasos:
  - Definir la clase Java con el *bean*
  - Instanciarlo

    En la propia página

    ```
    <jsp:useBean id="usuario" class="beans.Usuario"
                                 scope="request"/>
    ```

    En `faces-config.xml`

    ```
    <managed-bean>
       <managed-bean-name>usuario</managed-bean-name>
       <managed-bean-class>
          beans.Usuario
       </managed-bean-class>
       <managed-bean-scope>request</managed-bean-scope>
    </managed-bean>
    ```

  - Vincular los datos del *bean* con los componentes web

    ```
    <h:inputText value="#{usuario.login}"/>
    ```

Acciones

- Se definen en los beans de respaldo
- Indicar el método que implementa la acción

```
<h:commandButton value="Entrar"
                 action="#{usuario.doLogin}"/>
```

- Dicho método debe enlazar con la capa de negocio y devolver un String, indicando el resultado de la acción

- En faces-config.xml se especifica a qué vista ir según el resultado de la acción

Navegación entre páginas

- Se decide a qué página ir en función de hasta 3 factores
  - Página actual <from-view-id>
  - Qué acción se ha ejecutado (opcional) <from-action>
  - Cuál ha sido el resultado (String) de la acción (opcional) <from-outcome>
- Reglas de navegación definidas en `faces-config.xml`

```
<navigation-rule>
    <from-view-id>/login.jsp</from-view-id>
      <navigation-case>
                <from-action>#{gestorUsuarios.doLogin}</from-action>
                <from-outcome>OK</from-outcome>
                <to-view-id>/personal.jsp</to-view-id>
      </navigation-case>
      <navigation-case>
                <from-action>#{gestorUsuarios.doLogin}</from-action>
                <from-outcome>error</from-outcome>
                <to-view-id>/error.jsp</to-view-id>
      </navigation-case>
</navigation-rule>
```

Probar ejemplos

- jsf-login

Validación de datos y gestión de errores

- Conversores (validación sintáctica)

```
<h:inputText value="#{usuario.fechaNacimiento}">
   <f:convertDateTime pattern="dd/MM/yyyy"/>  <!-- idem a java.text.SimpleDateFormat --
>
</h:inputText>
```

- Validadores (validación semántica)

```
 <h:form id="formu">
    <h:inputText value="#{usuario.login}" required="true" id="login"/> <br>
    <h:inputSecret value="#{usuario.password}" required="true">
        <f:validateLength minimum="6" />
    </h:inputSecret>
    ...
```

- Mostrar errores

```
 <h:message for="formu:login"/>
 <h:messages/>  <!-- para todos los campos -->
 <h:messages globalOnly="true"> <!-- solo los globales -->
```

Mensajes de error propios

- La implementación de JSF ya tendrá mensajes de error predefinidos (en inglés)
- Definir un fichero .properties con los mensajes

```
<faces-config>
        <application>
                <message-bundle>mensajes</message-bundle>
        </application>
        ...
</faces-config>
```

- Los mensajes tienen un nombre clave especificado en el estándar JSF

```
javax.faces.component.UIInput.CONVERSION=Error en el formato
```

Resumiendo: ¿qué es JSF?

- Un conjunto de controles GUI basados en Web y manejadores asociados
  - JSF proporciona una gran cantidad de controles GUI orientados a HTML junto con código para manejar los eventos
- Una versión mejorada de Struts
  - Al igual que Struts, JSF puede ser visto como un framework MVC para construir formularios HTML, validar sus valores, invocar la lógica de negocio y mostrar los resultados.
- La parte VC del MVC
  - El *modelo* de JSF son los *backing beans*, que no representan el modelo de la aplicación. Por eso, JSF se puede ver sobre todo como un framework para construir la *Vista* y el *Controlador*.

Resumiendo: ventajas de JSF

- Controles GUI HTML
- Manejo de eventos
- Managed beans
- Lenguaje de expresiones
- Conversión y validación de las entradas en los campos
- Navegación centralizada en un fichero de configuración

Resumiendo: inconvenientes de JSF

- Curva de aprendizaje bastante pronunciada
- Mala documentación
- Aplicaciones difíciles de depurar
- Escaso soporte de herramientas
- Difícil de integrar con otros frameworks
- Poco introducido en la industria (en comparación con Struts)

¿Preguntas?