COMPONENTES PERSONALIZADOS PARA LA INTERFAZ DE USUARIO – Componentes RichFaces.

Autor:
Msc. Fredy Humberto Vera Rivera.
freve9@hotmail.com
Bucarmanga - Colombia 2.009

Los componentes de la interfaz de usuario (Componentes UI) son los elementos de un software que establecen la comunicación directa entre el cliente y el sistema. Se encargan principalmente del manejo de eventos producidos por el cliente, validación de datos y definen la navegación del cliente sobre las pantallas o paginas.

En la actualidad para el desarrollo de aplicaciones web en Java existen varios frameworks que contienen componentes reutilizables para la interfaz de usuario, entre estas librerías de componente encuentran Java Server Faces (JSF) que es la más utilizada y Richfaces que es una implementación de componentes JSF utilizando AJAX para su funcionamiento.

Las características principales del framework JSF son:

- El framework es una arquitectura basada en componentes.
- El componente JSF no es solo una serie de código HTML enviado e interpretado por un explorador web, el componente JSF es una combinación de elementos del lado del cliente con elementos del lado del servidor que representan al componente, incluye validación de datos, manipulación de eventos, conexión con la lógica del negocio de la aplicación, etc.
- JSF permite un paradigma orientado a componentes para construir interfaces de usuario bien diseñadas, personalizables basadas en componentes reutilizables.

Uno de los principales problemas para desarrollar un componente JSF es que el tiempo de implementación es muy largo y el desarrollo es complejo. Para la creación de un componente JSF se requiere implementar:

- La clase que hereda de UIComponent, que es la clase base para todos los componentes.
- La clase Tag, que define la etiqueta que se utiliza para utilizar el componente.
- La clase Converter si es necesaria. Que se utiliza para convertir los valores introducidos al componente en un objeto java determinado.
- La clase Renderer en caso que sea necesaria, la cual es la encargada de traducir la etiqueta JSF que representa el componente a código HTML.
- El archivo de configuración facesconfig.xml
- En el momento de utilizarlo en un ambiente facelet se debe suministrar el *.tablib.xml.

Para la creación de un componente de richfaces se requiere inclusive más tiempo. Adicionalemente se debe proveer:

- ListenerTagHandler
- Una clase para crear una interface *listener*.

- Un método de procesamiento de evento en la interfaz *listener.*
- Una clase Evento.
- Una clase Renderer específica para cada posible render kit utilizado con el componente.

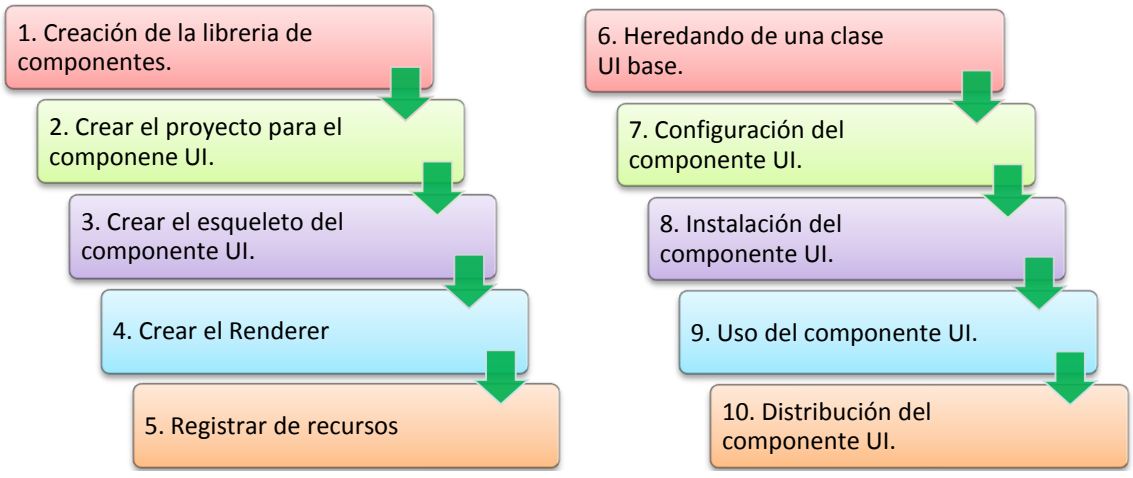
A pesar de la complejidad, el proceso de creación de componentes JSF es repetible según dicen Jonas Jacobi y John R. Fallows en el libro "Pro JSF and Ajax Bulling Rich Internet Components", en el cual definen el proceso en detalle. Con base en esta descripción la gente de Jboss realizaron el Kit de desarrollo de componentes de richfaces, el cual permite una creación más rápida y automatizada de componentes richfaces con soporte Ajax.

Las características principales del CDK de richfaces son:

- Inicio rápido del desarrollo. El desarrollo de un nuevo componente inicia con una estructura pre generada, la cual contiene todos los archivos necesarios para el desarrollo.
- Solo se necesita especificar las propiedades del componente en un meta-data y el código específico del componente. Todos los otros artefactos descritos atrás son generados automáticamente.
- Ciclo de vida del desarrollo independiente, cada componente se desarrolla separadamente de los otros.
- Facilidad de automatizar las pruebas.
- Soporte para Ajax y para adicionar Ajax a componentes existentes.
- Optimización para diferentes implementaciones de JSF.

Los pasos necesarios para la creación un componente reutilizable se puede apreciar en la siguiente figura.

Figura 1. Fases para la creación de componentes UI.



A continuación se detallan las actividades principales que se tienen que hacer en cada fase hasta tener implementado el componente UI. Para una mayor referencia se puede estudiar el documento CDK developer guide [18].

1. Creación y configuración de la librería de componentes UI.

Antes de crear la librería de componentes es necesario tener instaladas las siguientes herramientas, las cuales permiten crear la librería de componentes y los propios componentes. Las herramientas son:

- JDK1.5 o superior
- Apache Maven 2.0.9 o superior
- Apache Tomcat 6.0
- Browser

En primer lugar se debe tener configurado el repositorio de Maven, luego se debe crear un directorio donde quedaran almacenados los componentes, dentro del directorio se debe crear el archivo de configuración pom.xml. El contenido de los archivos de configuración y del pom.xml se puede consultar en el documento CDK developer guide [18] a partir de la página 6.

Los elementos más importantes del archivo de configuración pom.xml de la librería de componente creada se pueden apreciar en la siguiente tabla.

ELEMENTOS DEL pom.xml DE LA LIBRERÍA DE COMPONENTES UI.	
Elemento	**Descripción**
groupId	Prefijo para el paquete Java de la librería de componentes.
url	Namespace para la librería de componentes. Es el nombre o alias que se utiliza en las cabecera de las páginas para utilizar los componentes. Se usa en el TDL (Tag Definition Languaje).
version	Versión de la librería de componentes.
scope	Es usado para limitar la transitividad de una dependencia, afecta al classpath usado para varias tareas de construcción. El scope "Provided" indica que se espera que el JDK o el contenedor provean las dependencias en tiempo de ejecución.

2. Creación del proyecto para el desarrollo del componente UI.

Para crear el proyecto para su componente se debe ingresar al directorio donde se encuentra su librería y ejecutar el siguiente comando.

```
mvn archetype:create –DarchetypeGroupId = org.richfaces.cdk –darchetypeArtifactId = maven-archetype-jsf-component -DarchetypeVersion=3.3.0.GA -DartifactId=nombreComponente
```

El commando anterior crea una estructura de directorios como se muestra en la siguiente figura.

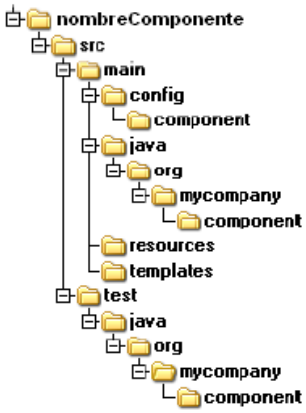
Los directorios generados más importantes son:

src/main/config: Contiene los archivos meta-data, que son archivos xml necesarios para la configuración del componente.

src/main/java: Contiene el código java el pregenerado y el escrito por el desarrollador.

src/main/resources: Se usa para almacenar los recursos del componente, tales como, imágenes, código javascript, archivos css, etc. También contiene el archivo resource-config.xml para el registro de recursos.

Figura 2. Estructura del proyecto para la creación de componentes UI.



src/main/templates: Contiene las plantillas jsp utilizadas para la generación del componente.

Se debe adicionar el plugin maven-compiler-plugin a las sección plugins en el archivo pom.xml generado en el directiorio del componente UI. Se debe adicionar el siguiente código:

```
<plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <inherited>true</inherited>
        <configuration>
                <source>1.5</source>
                <target>1.5</target>
        </configuration>
</plugin>
```

3. **Crear el esqueleto del componente UI.**

Para crear el esqueleto del componente se debe ejecutar el siguiente comando en el directorio generado para el componente a desarrollar.

```
mvn cdk:create -Dname=nombreComponente
```

El resultado de este comando es la generación de tres archivos principales:

- Un archivo de configuración XML (nombreComponente.xml), para los metadatos del componente. Se genera en el directorio src/main/config**.**
- Una clase UI. Generada en el directorio src/main/java/org/mycompany/component. Su nombre es UINombreComponente.java.
- Una plantilla jsp, generada en el directorio src/main/templates.

4. **Crear el renderer.**

El funcionamiento del componente se centra principalmente en dos acciones: codificar y decodificar datos. La decodificación (decode) es el proceso de conversión los parámetros request a valores del componente; la codificación (encode) es el proceso de convertir los valores actuales del componente en las correspondientes etiquetas para ser mostradas en la página. En la siguiente figura se puede apreciar en el ciclo de vida de los componentes JSF y el momento cuando ocurren los procesos de codificación y decodificación.

Figura 3. Fases del ciclo de vida de un componente JSF.

Un componente JSF consta principalmente de dos partes: las clase UIClass del componente y el renderer. La clase UIClass es responsable del estado y ambiente del componente UI. La clase renderer se encarga de la representación del componente, genera las etiquetas apropiadas en el cliente y convierte la información suministrada por el cliente en valores del componente. Para un componente UI es necesario crear las siguientes clases:

- **NombreComponenteRenderer** en la cual se sobreescribe el método encode() para codificar etiquetas y recursos. Esta clase se genera al instalar el componente, se crea a partir del mecanismo de plantilas jspx creadas al generar el componente.
- **NombreComponenteRendererBase** se sobreescribe el método decode() y se define el convertidor para la clase. En esta clase se definen los método utilizados en la pagina jsp para la generación del componente.

Para la creación de la plantilla que genera la clase renderer se procede a modificar la página /src/main/templates/org/mycompany/ htmlNombreComponente.jspx. Se crea con etiquetas de HTML 4.0 y a partir de tags predefinidos, la referencia para los tag que se puedan utilizar en la plantilla se puede apreciar en el capítulo 11 del CDK developer guide. En la plantilla también se puede utilizar etiquetas html normales y código jsp tradicional. En el capítulo de desarrollo del componente reutilizable para la División de Servicios de Información se entrará más en detalle en la creación de esta plantilla, de la clase RendererBase y de los convertidores.

5. **Registrar recursos.**

Los recursos deben ser registrados en el archivo resource-config.xml. Se puede registrar imgenes, JavaScript , CSS, XCSS, SWF, (X)HTML, XML, Logs, JAR, etc), a continuación se presenta un ejemplo del registro de una imagen.

```
<resource>
        <name>org/mycompany/renderkit/html/images/icono.png</name>
        <path>org/mycompany/renderkit/html/images/icono.png</path>
</resource>
```

6. Heredando de una clase UI base.

La clase base para todos los componentes JSF es UIComponent, Al abrir el archivo generado en la creación del esqueleto del componente se puede apreciar que la clase UINombreComponente.java hereda de UIComponentBase, la cual es una sublcase de UIComponent y provee una implementación por defecto de todos los métodos abstractos de UIComponent.

7. Configuración del componente UI.

El archivo de configuración fue generado en el momento de crear el esqueleto del component UI, su nombre es nombreComponente.xml, se utiliza para la generación de la clase UINombreComponente completa, el Tag Handler para JSP, el archivo faces-config.xml y los descriptores para JSP y Facelets.

Consta principalmente de las siguientes partes:

- Mapeo a la clase UINombreComponente.

  ```
  <name>org.mycompany.NombreComponente</name>
  <family>org.mycompany.NombreComponente</family>
  <classname>org.mycompany.component.html.HtmlNombreCompon</classname>
  <superclass>org.mycompany.component.UINombreComponente</superclass>
  ```

- Mapeo de la clase Renderer y de la plantilla jspx.

  ```
  <renderer generate="true" override="true">
  <name>org.mycompany.NombreComponenteRenderer</name>
  <template>org/mycompany/htmlNombreComponente.jspx</template>
  </renderer>
  ```

- El tag handler para Jsp

  ```
  <tag>
  <name>NombreComponente</name>
  <classname>org.mycompany.taglib.NombreComponenteTag</classname>
  <superclass>
          org.ajax4jsf.webapp.taglib.HtmlComponentTagBase
  </superclass>
  </tag>
  ```

- Propiedades del componente.

  ```
  <property>
          <name>value</name>
  ```

```
        <classname>java.lang.Object</classname>
        <description>The value of the component</description>
    </property>
    <property>
        <name>title</name>
        <classname>java.lang.String</classname>
        <description>Defines a title of the component</description>
        <defaultvalue>&amp;quot;NombreComponente&amp;quot;</defaultvalue>
    </property>
```

Si quiere incluir en el componente definición para eventos html, se debe adicionar las siguientes entidades en el archivo de configuración.

```
&ui_component_attributes;
&html_events;
&ui_input_attributes;
```

8. Instalación del componente UI.

Para realizar la instalación del componente se procede a ejecutar el siguiente comando:

```
mvn clean install
```

El cual genera en el directorio /target/classes/META-INF se encuentran los archivos *.tdl, *.taglib.xml, resoruces-config.xml y faces-config.xml.

La clase NombreComponenteTag se genera en directorio /target /clases /org /mycompany / taglib.

Con la generación de estos archivos y del jar nombreComponente-1.0-SNAPSHOT.jar se culmina la creación del componente, el jar se puede encontrar en el directorio target.

9. Uso del componente UI

Para utilizar el componente en una página jsp se tiene que poner la siguiente cabecera:

```
<%@ taglib uri="http://mycompany.org/NombreComponente" prefix="my"%>
```

Luego se utiliza la etiqueta del componente:

```
<my:NombreComponente value="#{bean.text}">
```

Se puede resaltar que la dirección http://mycompany.org es la url suministrada en el archivo de configuración pom.xml de la biblioteca de componentes creada al principio de este capítulo.

10. Distribución del componente UI.

Para la distribución del componente se debe suministrar a los desarrolladores el archivo nombreComponente-1.0-SNAPSHOT.jar para que lo copien en la librería del proyecto

donde se piensa utilizar y también se debe copiar en la librería del servidor de aplicaciones para ser reconocido en tiempo de ejecución.

Referencias:

CDK Developer Guide. – Jboss Comunity.