Navegación de páginas Web utilizando JSF

Desarrollo y despliegue de componentes software–II

1 Diferentes formas de navegación de páginas con JSF

Ahora vamos a trabajar con las reglas de navegación que proporciona JSF para poder indicar qué vista del modelo queremos mostrar cuando pulsamos un "botón" o picamos en un enlace incluido en la página inicial.

Las opciones que tenemos son las siguientes:

1. Navegación implícita (o la navegación entre páginas sin necesidad de definir reglas)
2. Reglas de navegación definidas en un MB ("*managed bean*")
3. Reglas de navegación con condiciones que se resuelven mediante paso de parámetros
4. Reglas de navegación que se definen en el archivo de configuración denominado `faces-config.xml` (que está situado dentro del subdirectorio `src/main/webapp/WEB_INF` de nuestro proyecto)

1.1 Navegación implícita

Se trata de realizar una *auto–navegación* dentro de una página, ya que JSF 2.0 incluye una herramienta denominada *resolutor de vista de páginas*, de tal manera que sólo tenemos que incluir un nombre de vista en el atributo `action` de un *componenteIU* de `home.xhtml` y el marco de trabajo JSF se encargará de buscar de forma automática la vista más adecuada de la página. Por ejemplo, si incluimos en la página el siguiente código:

```
<h3>Utilizando la salida JSF</h3>
<h:commandButton action="pagina2" value="Pagina 2" />
```

Cuando se pulse el botón con nombre *Pagina 2*, el marco de trabajo JSF determinará el nombre de la página que contiene la vista que se quiere mostrar: `pagina2.xhtml`. Buscará en el directorio actual, es decir, donde esté ubicado el archivo `home.xhtml` del componente previamente desplegado en Tomcat.

1.2 Auto–navegación utilizando un MB

Si hubiéramos programado un método (p.e.:`moverHaciaPagina1()`) para devolver una vista del modelo como una página xhtml dentro de un MB, tal como muestra el siguiente código:

```
@ManagedBean(name = "controladorNavegacion", eager = true)
@RequestScoped
public class controladorNavegacion implements Serializable{ //Este es el MB
  private static final long serialVersionUID = 1L;
  @ManagedProperty(value="#{param.pageId}")
```

```
  private String pageId;
  ...
  public String moverHaciaPagina1(){
   return "pagina1";
  }
...
}
```

En este caso, para obtener la página que necesitamos en la vista, incluiremos el nombre del método cualificado dentro del atributo `action` de un *componenteIU*:

```
<h3>Utilizando un "Managed Bean"</h3>
<h:commandButton action="#{controladorNavegacion.moverHaciaPagina1}"
 value="Pagina 1" />
```

De manera similar al caso anterior, cuando se pulse el botón denominado *"Pagina 1"*, el marco de trabajo JSF determinará que el nombre de la página que contiene la vista es: `pagina1.xhtml` y lo buscará en el sitio adecuado del proyecto desplegado en Tomcat.

1.3 Navegación condicional

Un MB puede, mediante el uso del atributo `param` y su parámetro `PageId`, que pertenece a una anotación `ManagedProperty`, seleccionar la vista que va a mostrar entre varias posibles alternativas:

```
@ManagedBean(name = "controladorNavegacion", eager = true)
@RequestScoped
public class ControladorNavegacion implements Serializable{ //Este es el MB
  private static final long serialVersionUID = 1L;
  @ManagedProperty(value="#{param.pageId}") //atributo y su parametro
  private String pageId;
  ...
  //navegacion condicional basada en el valor de pageId
   //si pageId es 1 mostrar page1.xhtml,
   //si pageId es 2 mostrar page2.xhtml
   //si no, mostrar home.xhtml
  public String mostrarPagina(){
    if(pageId == null){
       return "home";
    }
    if(pageId.equals("1")){
      return "pagina1";
      }else if(pageId.equals("2")){
        return "pagina2";
        }else{
          return "home";
          }
    }
...
  }
}
```

El valor actual del parámetro `pageId` se obtiene desde dentro de un *componenteIU* incluído en una página JSF como un parámetro solicitado, por ejmplo, picando en un enlace de la página:

```
<h:form>
<h:commandLink action="#{controladorNavegacion.mostrarPagina}"
value=" Pagina 1 ">
<f:param name="pageId" value="1" />
</h:commandLink>
<h:commandLink action="#{controladorNavegacion.mostrarPagina}"
value=" Pagina 2 ">
<f:param name="pageId" value="2" />
</h:commandLink>
<h:commandLink action="#{controladorNavegacion.mostrarPagina}"
value=" Volver a Inicio ">
<f:param name="pageId" value="3" />
</h:commandLink>
</h:form>
```

1.4 Navegación entre páginas utilizando reglas de navegación

Otra posibilidad que tiene el desarrollador de aplicaciones Web con el marco de trabajo JSF es la definir la navegación por las páginas de la aplicación a través del archivo de configuración `faces-config.xhtml`, que se ubica dentro del subdirectorio `WEB_INF` de nuestro proyecto.

1.4.1 Salida esperada desde un *componenteIU*

Cada regla de navegación define cómo ir de una página que se especifica en la etiqueta `<from--tree--id>` a otras páginas de la aplicación Web:

```
<?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">
<navigation-rule>
<from-tree-id>/bienvenida.jsp</from-tree-id>
<navigation-case>
  <from-outcome>success</from-outcome>
  <to-tree-id>/respuesta.jsp</to-tree-id>
</navigation-case>
</navigation-rule>
<navigation-rule>
<from-tree-id>/respuesta.jsp</from-tree-id>
<navigation-case>
  <from-outcome>success</from-outcome>
  <to-tree-id>/bienvenida.jsp</to-tree-id>
```

```
</navigation-case>
</navigation-rule>
</faces-config>
```

El elemento `navigation-rule` puede contener cualquier número de elementos `navigation-case`, cada uno de los cuales abrirá posteriormente una página distinta que se define dentro de la etiqueta `to-tree-id`, basándose en el valor actual del atributo `action` dentro de la etiqueta `command_button` que está incluida en la página:

```
<h:command_button id="submit" action="success" label="Enviar"/>
```

El valor de salida esperado "`success`" del ejemplo anterior se identifica con el atributo `<from-outcome>`, es decir, la salida que se espera ("`success`") se produce desde el *componenteIU* `<h:command_button ...>` incluido en el formulario (por ejemplo, en la página `home.xhtml`).

1.4.2 Salida esperada devuelta por un método

El valor de salida que se indica en `<from-outcome>` podría salir también del valor que se devuelva en la ejecución de un método (`procesarPaginaX()`) incluido en un MB. En este caso el archivo `faces-config.xhtml` contendrá la siguiente información:

```
<?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">
<navigation-rule>
<from-view-id>home.xhtml</from-view-id>
<navigation-case>
<from-action>#{controladorNavegacion.procesarPagina1}</from-action>
<from-outcome>pagina</from-outcome>
<to-view-id>pagina1.jsf</to-view-id>
</navigation-case>
<navigation-case>
<from-action>#{controladorNavegacion.procesarPagina2}</from-action>
<from-outcome>pagina</from-outcome>
<to-view-id>pagina2.jsf</to-view-id>
</navigation-case>
</navigation-rule>
</faces-config>
```

Incluso se admite que los métodos del MB devuelvan el mismo nombre de página, ya que el marco de trabajo JSF será capaz de encontrar la página adecuada después de procesar las reglas de navegación que están definidas en el archivo anterior:

```
@ManagedBean(name = "controladorNavegacion", eager = true)
@RequestScoped
public class ControladorNavegacion implements Serializable{ //Este es el MB
```

```
  private static final long serialVersionUID = 1L;
  @ManagedProperty(value="#{param.pageId}")
  private String pageId;
  ...
  public String procesarPagina1(){
   return "pagina";
  }

  public String procesarPagina2(){
   return "pagina";
  }
 }
```

El valor de salida esperado `pagina`, declarado en `<from-outcome>`, en este caso se envía seleccionando un enlace representado por el *componenteIU* `<h:command_link ...>` incluido en el formulario Web:

```
<h:form>
<h:commandLink action="#{controladorNavegacion.procesarPagina1}"
value=" Pagina 1 " />
<h:commandLink action="#{controladorNavegacion.procesarPagina2}"
value=" Pagina 2 " />
</h:form>
```

1.4.3 Navegación hacia adelante con/sin redireccionamiento de URL a la página de destino

JSF por defecto realiza un re-envío de la página del servidor, es decir, siempre se mantiene la misma URL en el visor del navegado cuando pasa a visitar otra página Web. Para permitir que cambie la URL que se muestra en el buscador del navegador cuando navegamos a otras páginas hay que añadir la orden `faces.redirect=true` al final del nombre de la vista en el atributo `action` del *componenteIU* que utilicemos en el formulario:

```
<h:form>
<h3>Hacia Adelante</h3>
<h:commandButton action="pagina1" value="Pagina 1 " />
<h3>Redireccionamiento</h3>
<h:commandButton action="pagina1?faces-redirect=true"
value="Pagina 1 " />
</h:form>
```

2 Ejercicio

Ahora nos proponemos compilar, ejecutar y desplegar una aplicación desarrollada utilizando el marco de trabajo JSF para probar las facilidades de navegación que hemos introducido anteriormente. La aplicación Web ha de mostrar la siguiente pantalla de inicio:

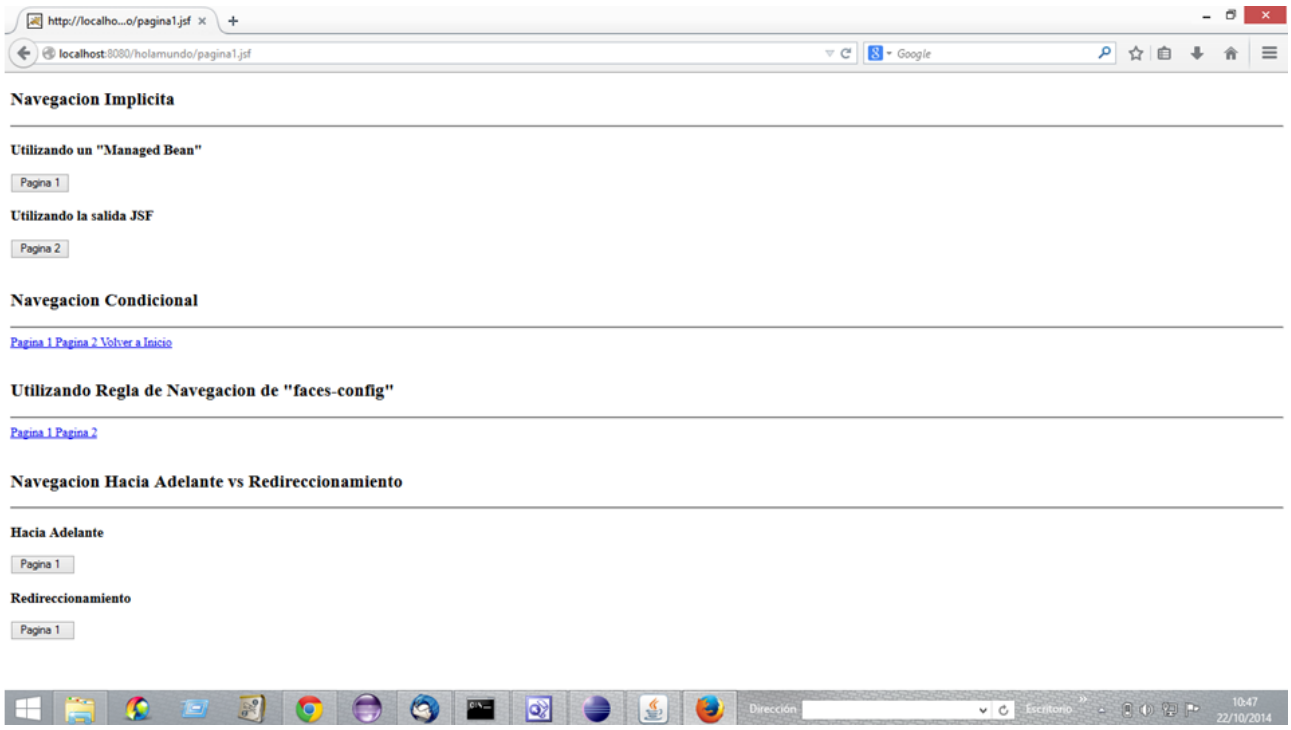

Figura 1: Aplicación que utiliza la tecnología JSF para navegar por varias páginas Web

Los pasos a seguir para realizarlo correctamente son los siguientes:

1. Crear un proyecto (o reutilizar el proyecto `holamundo` de la primera parte de esta práctica) dentro del paquete Java de prueba (src/main/java/prueba)
2. Crear una clase `ControladorNavegacion.java` en el paquete `prueba` anterior
3. Crear un archivo `faces-config.xhtml` en el subdirectorio `WEB_INF` del proyecto
4. Actualizar el archivo web.xml del proyecto
5. Crear las clases `pagina1.xhtml`, `pagina2.xhtml` y `home.xhtml` en el subdirectorio `WEB_INF` del proyecto
6. Compilar y ejecutar el proyecto (*"Run As"*) y asegurarse que la lógica de la aplicación funciona como se esperaba, es decir, la aplicación comprueba todas las facilidades de navegación de páginas Web que ofrece JSF que hemos introducido al comienzo de este documento
7. Finalmente, construir (*"Maven Install"*) el archivo `.WAR` de la aplicación y desplegarlo en el servidor Web Tomcat de Apache
8. Arrancar la aplicación Web desplegada utilizando la URL adecuada según la configuración de tu ordenador

Ayuda

La parte inicial de las páginas, tal como `home.xhtml` ha de contener la siguiente cabecera:

```
<?xml version="1.0" encoding="UTF-8"?>
<!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:f="http://java.sun.com/jsf/core"
xmlns:h="http://java.sun.com/jsf/html">
```

Créditos

- `http://www.tutorialspoint.com/jsf/`
- `http://programacion.net/articulo/introduccion_a_la_tecnologia_javaserver_faces_233`
- J. Murach, M. Urban. “Java Servlets and JSP”. Murach, 2014