



UNIVERSIDAD DEL QUINDÍO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Información general	
Actualizada por:	Einer Zapata
Duración estimada en minutos:	90
Docente:	Christian Andrés Candela
Guía no.	15
Nombre de la guía:	Capturando texto en JSF 2.3

Información de la Guía

OBJETIVO

Aprender a usar campos de captura de texto en JSF.

CONCEPTOS BÁSICOS

Manejo de Eclipse, Java, archivos de propiedades, xml y Glassfish.

CONTEXTUALIZACIÓN TEÓRICA

JSF (Java Server Faces) es un framework de presentación, usado para la construcción de interfaces web (páginas) que permitan la interacción con el usuario. JSF es un framework basado en componentes, hace uso del patrón de diseño MVC (Modelo Vista Controlador). Uno de los principales objetivos de JSF es separar la presentación (diseño de la página) de la lógica asociada a dicha presentación. Se puede pensar en JSF como en la librería swing (o JavaFX), con la diferencia que swing lo usamos para crear las ventanas de nuestras aplicaciones de escritorio, mientras que JSF nos permite la creación de páginas web dentro de nuestras aplicaciones empresariales. Teniendo en cuenta esto, es importante decir que al igual que en spring en JSF se pueden crear componentes propios con el fin de reutilizarlos posteriormente, o se puede hacer uso de componentes de terceros.

JSF ha sido incluido dentro de la especificación JEE como su capa de presentación. Siendo JSF un estándar, posee múltiples implementaciones de las cuales se puede escoger. De igual forma gracias a su inclusión como estándar, JSF está en continua evaluación y actualización.

JSF provee un componente de entrada de texto llamado **inputText**, este componente provee una interfaz a través de la cual el usuario ingresa un texto y el mismo es asignado a un atributo de un ManageBean, permitiendo así la manipulación de la información introducida por el usuario.

Los principales atributos del **inputText** son el **id** y el **value**. El **id** permite asignar un nombre al componente de texto que se está usando. Es importante tener en cuenta que no debe asignarse un mismo id a más de un componente JSF. El atributo **value** nos permite determinar el ManageBean y el atributo al cual se le asignará el valor ingresado por el usuario. Ejemplo:



Guía de laboratorio Área de Programación y Algoritmia

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

En este caso podemos ver un campo de entrada de texto cuyo nombre es **login**, y cuyo valor se asignará al atributo **login** del ManageBean **UsuarioBean**. También es posible expresar el `inputText` así:

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

NOTA: Siempre que se desee realizar una captura de datos, es importante tener en cuenta que los componentes de captura de información deben estar contenidos por un formulario así:

```
<h:form id="formAutenticacion">
  <h:inputText id="login" value="#{usuarioBean.login}" />
  ....
</h:form>
```

Como puede ver el formulario al igual que el `inputText` también cuenta con una propiedad **id**.

Por último, para lograr el envío de la información se requiere de un botón que registre la orden del usuario así:

```
<h:commandButton value="Enviar" action="mostrar" />
```

En el caso del botón podemos ver inicialmente dos atributos importantes, el **value** y el **action**. El **value** permite especificar el texto que contendrá el botón. El **action** nos permite determinar la acción a ser ejecutada por el botón. Si en la acción se da el nombre de una página JSF asumirá que debe redirigir al usuario hacia dicha página, si por el contrario se indica el nombre de un método en un ManageBean, JSF asume que debe invocar dicho método.

Ver <http://docs.oracle.com/javaee/6/tutorial/doc/bnatx.html>

Puede encontrar información en LEONARD, Anghel. JSF 2 CookBook . Birmingham – Mumbai: Packt Publishing junio 2010.

PRECAUCIONES Y RECOMENDACIONES

Recuerde que tanto los componentes de entrada como los botones deben estar dentro de un formulario para que la información pueda ser enviada de la página a los managed bean. Al nombrar las etiquetas hágalo en minúscula sostenida, si bien los navegadores no hacen distinción entre el uso de las mayúsculas y las minúsculas, el servidor de aplicaciones sí lo hace.

ARTEFACTOS

Se requiere tener instalado el JDK y un IDE para el desarrollo de aplicaciones (Eclipse JEE en su última versión), un servidor de aplicaciones que cumpla con las especificaciones de JEE, para esta práctica Glassfish.

EVALUACIÓN O RESULTADO

Se espera que el alumno pueda realizar emplear satisfactoriamente diferentes herramientas de captura de datos proporcionadas por JSF.



Procedimiento

1. Para el desarrollo de esta guía necesitara una base de datos en mysql, un proyecto de tipo maven – pom, el cual contenga un proyecto maven con soporte para el uso de JPA, uno con soporte para EJB, uno para web y otro proyecto maven configurado para la realización de pruebas. Además de una conexión a la base de datos para ser usada en la generación de las tablas.
2. Cree una clase con dos atributos de tipo String con sus respectivos métodos **get** y **set**. Marque la clase con las anotaciones Named y ApplicationScoped.

```
public class EjemploBean{  
    private String atributo1;  
    private String atributo2;  
    ...  
}
```

3. Adicionalmente cree un método que intercambie los valores de los dos atributos: Ejemplo

```
public void cambiar() {  
    String temp = atributo1;  
    atributo1 = atributo2;  
    atributo2 = temp;  
}
```

4. Cree una página xhtml con el nombre de su preferencia.
5. Para enriquecer la página con elementos que interactúen con los ManagedBean se requiere adicionar a la etiqueta html una declaración del conjunto de elementos a ser usado. Modifique la etiqueta html de su página así:

```
<html xmlns="http://www.w3.org/1999/xhtml"  
    xmlns:jsf="http://xmlns.jcp.org/jsf"  
    xmlns:pt="http://xmlns.jcp.org/jsf/passthrough"  
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"  
    xmlns:h="http://xmlns.jcp.org/jsf/html"  
    xmlns:f="http://xmlns.jcp.org/jsf/core">
```

6. En el body de su página cree un formulario con dos campos de texto (los cuales representarán en la página las dos variables de la clase) y un boton así:

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

7. Los elementos del formulario deben ser modificados para que interactúen con JSF.

```
<form jsf:id="miFormulario" >  
    <input type="text" jsf:value="#{ejemploBean.atributo1}" />  
    <input type="text" jsf:value="#{ejemploBean.atributo2}" />  
    <input type="submit" jsf:action="#{ejemploBean.cambiar}" value="Aceptar" />  
</form>
```



Guía de laboratorio Área de Programación y Algoritmia

Al form se adiciono un identificador para convertirlo en un formulario JSF, de igual forma los campos de texto se han modificado adicionando el atributo `jsf:value` para vincularlos a las variables de la clase, de esta forma si se modifica la variable el texto del campo de texto es modificado, y de igual forma si el campo de texto es modificado la variable es modificada en consecuencia. Por su parte, al botón se ha adicionado el atributo `jsf:action`, que permite ejecutar el método `cambiar` cuando el botón es presionado.

8. Cree una segunda página, y en el body adicione el siguiente formulario.

```
<h:form>
  <h:inputText />
  <h:inputText />
  <h:commandButton />
</h:form>
```

Como puede ver la sintaxis usada para la creación de este formulario es diferente. Mientras que en el ejemplo anterior se usaron componentes HTML5 para enlazarlos con JSF, este formulario usa componentes JSF.

9. Enlace los componentes con los elementos de la clase `ManagedBean`.

```
<h:form>
  <h:inputText value="#{ejemploBean.atributo1}" />
  <h:inputText value="#{ejemploBean.atributo2}" />
  <h:commandButton action="#{ejemploBean.cambiar}" />
</h:form>
```

Puede usar cualquiera de las dos sintaxis, de hecho, puede llegar a combinar el uso de componentes HTML5 con componentes JSF.

10. Corra el EAR en el servidor y compruebe el funcionamiento de su página.
11. Cree una página y una clase `ManagedBean` que le permitan registrar Empleados.
12. Cree una clase con el nombre de `EmpleadoBean` dicha clase debe tener como atributos los necesarios para crear un empleado o los que considere pertinente para que concuerden con los de su entidad. No olvide generar los métodos `get` y `set`.
13. Adicione a la clase las anotaciones `Named` y `ApplicationScoped`.
14. Para poder registrar el empleado se deberá usar un EJB que cumpla con dicha tarea. Declare una instancia del EJB en su clase `ManagedBean` y adiciónale la anotación `@EJB`

```
@EJB
private AdministradorEJB administradorEJB;
```

15. Cree un método `registrar` que cree un `Empleado` y lo registre usando su EJB.
16. Cree una página con el nombre de `registrarEmpleado.xhtml`. Modifique la etiqueta `html` para que pueda hacer uso de los elementos de JSF.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:jsf="http://xmlns.jcp.org/jsf">
```



Guía de laboratorio Área de Programación y Algoritmia

```
xmlns:pt="http://xmlns.jcp.org/jsf/passthrough"
xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
xmlns:h="http://xmlns.jcp.org/jsf/html"
xmlns:f="http://xmlns.jcp.org/jsf/core">
```

17. Al interior de la página cree un formulario que solicite al usuario los campos necesarios para la creación del empleado.

```
<form jsf:id="formEmpleado" >
  <label for="nombreEmpleado" >Nombre:</label>
  <input id="nombreEmpleado" type="text" jsf:value="#{empleadoBean.nombre}" />
  <input type="submit" jsf:action="#{empleadoBean.registrar}" value="Aceptar" />
</form>
```

18. Corra el EAR en el servidor y verifique el funcionamiento de su página.

19. Para indicar que el campo nombreEmpleado es requerido adicione al campo de texto el atributo required="required"

```
<form jsf:id="formEmpleado" >
  <label for="nombreEmpleado" >Nombre:</label>
  <input id="nombreEmpleado" type="text" required="required" jsf:value="#{empleadoBean.nombre}" />
  <input type="submit" jsf:action="#{empleadoBean.registrar}" value="Aceptar" />
</form>
```

20. Corra el EAR en el servidor y verifique el funcionamiento de su página.

21. Finalmente, es importante poder mostrar al usuario mensajes de éxito o error cuando se está realizando la operación de registro. Adicione a su método registrar en el ManagedBean un try catch que permita determinar si se presentó o no errores al realizar el registro. En caso de que se tenga éxito en el registro adicione las siguientes líneas.

```
FacesMessage facesMsg = new FacesMessage(FacesMessage.SEVERITY_INFO, "Registro exitoso", "Registro exitoso");
FacesContext.getCurrentInstance().addMessage(null, facesMsg);
```

22. Si se presentó algún error al tratar de registrar el empleado en el catch adicione las siguientes líneas.

```
FacesMessage facesMsg = new FacesMessage(FacesMessage.SEVERITY_ERROR, e.getMessage(), e.getMessage());
FacesContext.getCurrentInstance().addMessage(null, facesMsg);
```

23. En la página registrar adicione la etiqueta `<h:messages />
`.

24. Despliegue y verifique los resultados.