Lab 08 – Empleo de ManageBean.

Este laboratorio parte de la forma de búsqueda de empleados.

Objetivo: Crear una menú de pestañas que simulen una agenda de contactos organizada alfabéticamente.

Al oprimir una de las pestañas, se ilustrará la lista de empleados cuyo LastName inicia con la respectiva letra.

Objetivo A: Crear un Backing Bean con la capacidad de capturar los eventos que suceden en la interfaz de usuario. Específicamente se desea crea un método que sirva como listener (escuchador) de los eventos producidos por las pestañas de letras.

Paso 1: Cree el método public void buscarPorLetra(String letra)
en el applicationModule. Su objetivo es crear una condición sobre el View Object de AdminEmpleados que aplique sobre el Query un filtro similar a “ LastName like ‘A%’ ”.

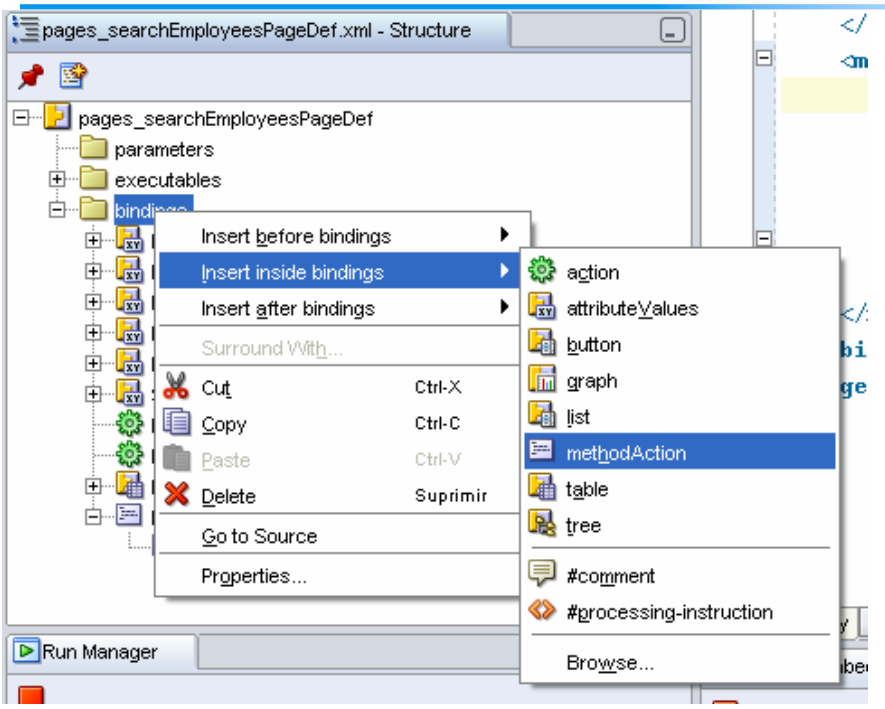
```
public void buscarPorLetra(String letra) {
  ViewObject vo = this.findViewObject("AdminEmpleados");
  ViewCriteria vc = vo.createViewCriteria();
  ViewCriteriaRow vcr = vc.createViewCriteriaRow();
  vcr.setAttribute("LastName", letra+"%");
  vc.add(vcr);
  vo.applyViewCriteria(vc);
  vo.executeQuery();
}
```

Paso 2: Exponga el método buscarPorLetra a los clientes, para que pueda ser invocado desde JSF.

a. Edite el aplicationModule y seleccione el método en la opción “Client Interface”.
b. Observe en el DataControl Palete esté disponible este método.

Paso 3: A través del Page Definition de la página JSP searchEmployees, cree un binding de tipo methodAction al método buscarPorLetra.

a. Inicialmente, asigne un valor constante al parámetro letra. Por ejemplo “A”.



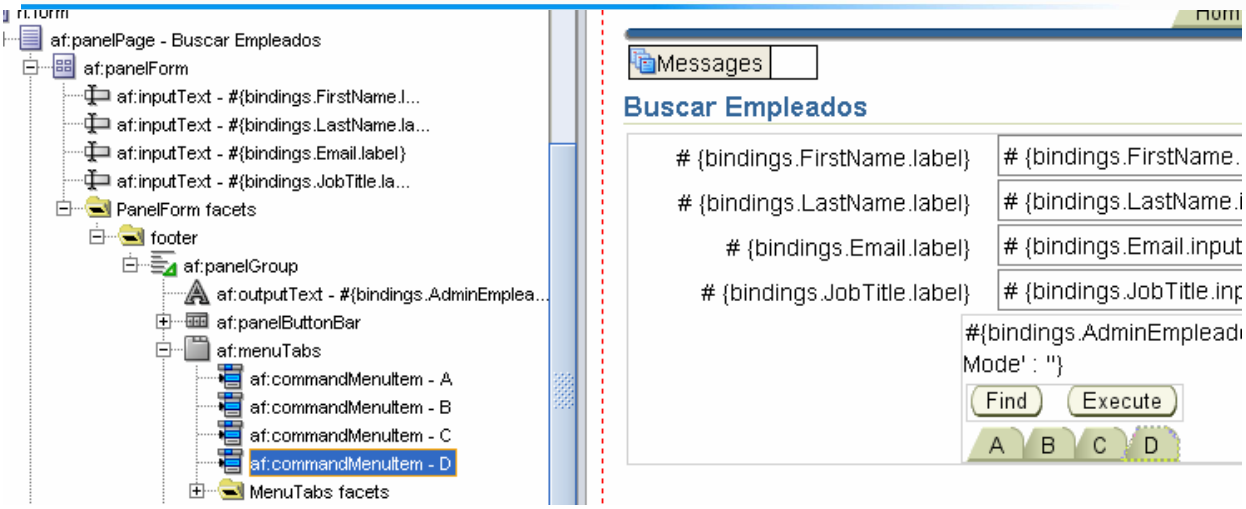
b. Verifique el archivo PageDefinition de la respectiva página:

```
52      <methodAction id="buscarPorLetra"
53                    InstanceName="DepartamentosServiceDataControl.dataProvider"
54                    DataControl="DepartamentosServiceDataControl"
55                    MethodName="buscarPorLetra" RequiresUpdateModel="true"
56                    Action="999" IsViewObjectMethod="false">
57        <NamedData NDName="letra" NDValue="A" NDType="java.lang.String"/>
58      </methodAction>
```

Paso 4: Crear la interfaz de usuario

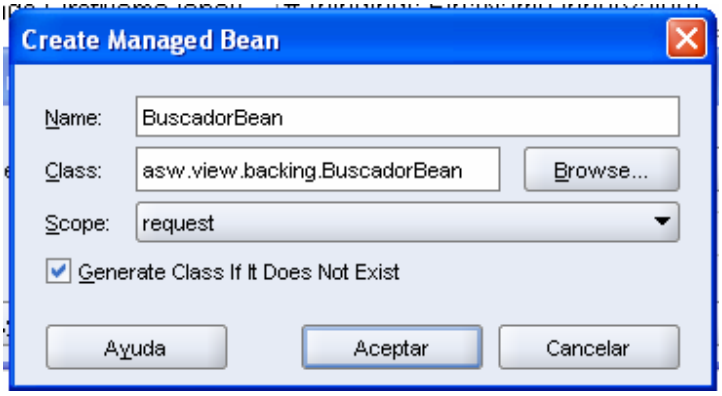
a. Adicione un menuTabs.
b. Adiciones tantos commandMenuItem como pestañas de letras desea tener.
c. Cambie el atributo Text de cada uno de estos commandMenuItem por una letra (emplee mayúsculas).

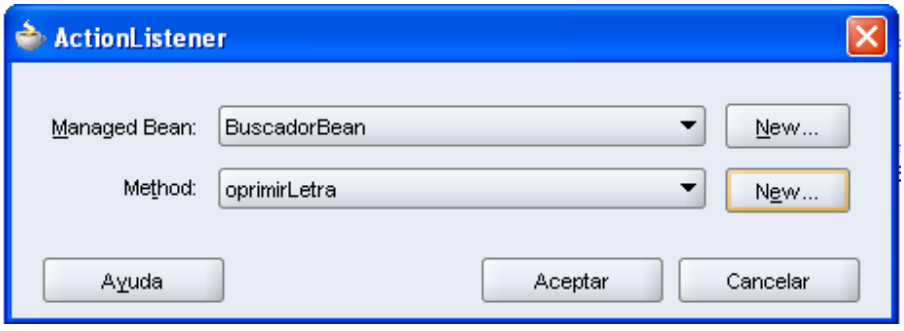
Lab 08 – Empleo de ManageBean.



Paso 5: Para cada uno de los menuItems (pestañas de letras) el valor #{BuscadorBean.oprimirLetra} para el atributo ActionListener.

Nota: Para la primera pestaña será necesario crear Managed Bean y el respectivo método oprimirLetra.





Paso 6: Complete el código del bean BuscadorBean creado en el paso anterior:

Lab 08 – Empleo de ManageBean.

```
1  package asw.view.backing;
2  import javax.faces.event.ActionEvent;
3  import oracle.binding.BindingContainer;
4  import oracle.binding.OperationBinding;
5
6  public class BuscadorBean {
7      private BindingContainer bindings;
8      public BindingContainer getBindings() {
9          return this.bindings;
10     }
11     public void setBindings(BindingContainer bindings) {
12         this.bindings = bindings;
13     }
14     public BuscadorBean() {
15     }
16
17
18     public void oprimirLetra(ActionEvent actionEvent) {
19         BindingContainer bindings = getBindings();
20         OperationBinding operationBinding =
21             bindings.getOperationBinding("buscarPorLetra");
22         Object result = operationBinding.execute();
23     }
24 }
```

Los métodos getBindings y setBindings son empleados para recibir el objeto **bindings** del contexto del contenedor. Este objeto será asignado al BuscadorBean en el archivo faces-config.xml (ver paso 7).

En la línea de código 21 se está invocando el methodBinding que se encuentra declarado en la PageDef de está página (ver paso 3).

Paso 7: Edite el archivo faces-config.xml para declarar la propiedad **bindings** en el bean BuscadorBean:

```
52  <managed-bean>
53    <managed-bean-name>BuscadorBean</managed-bean-name>
54    <managed-bean-class>asw.view.backing.BuscadorBean</managed-bean-class>
55    <managed-bean-scope>request</managed-bean-scope>
56    <managed-property>
57      <property-name>bindings</property-name>
58      <value>#{bindings}</value>
59    </managed-property>
60  </managed-bean>
61  <managed-bean>
```

Paso 8: Prueba la aplicación.

a. Qué sucede con la tabla resultado de Employees?
b. Las pestañas cambian? Se observa que oprimió el usuario?

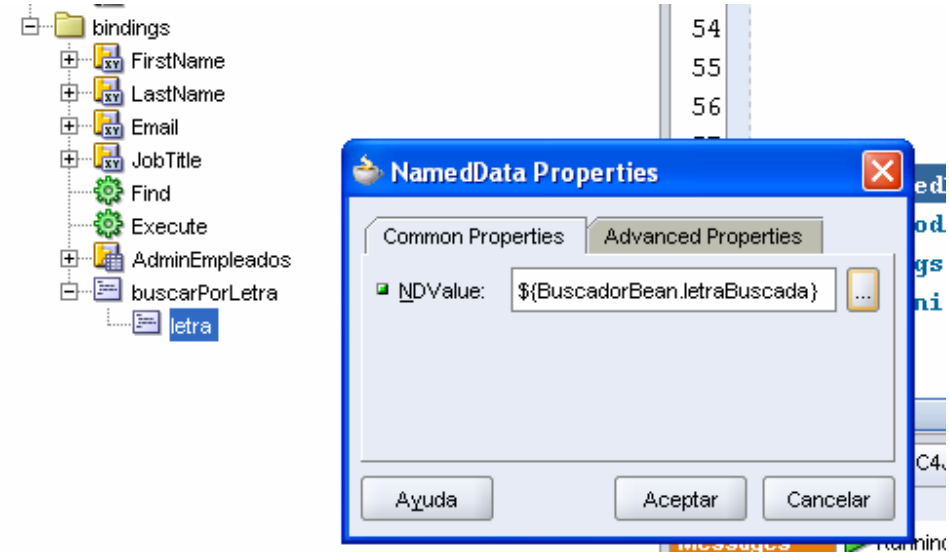
Objetivo B: A continuación se manipula la interfaz de usuario para reflejar las acciones del usuario. Adicionalmente se creará una propiedad (a nivel del bean BuscadorBean) que almacene la letra que el usuario oprimió. Está propiedad podrá ser referenciada desde una EL.

Paso 9: Modificar el MethodAction en la PageDefinition para hacer variable el parámetro “letra” y de esta forma asegurar que se filtre correctamente.

a. Adicionar el atributo letraBuscada en la clase BuscadorBean. Crear los respectivos métodos set y get.
b. Asignar al atributo letraBuscada el valor de la etiqueta de la pestaña que fue seleccionada por el usuario.

```
29    public void oprimirLetra(ActionEvent actionEvent) {
30        /*
31         * Declarar variables locales del método
32         */
33        CoreCommandMenuItem pestanaOprimida;
34        BindingContainer bindings;
35        OperationBinding operationBinding;
36        Object resultado;
37        /*
38         * Almacenar en la propiedad LetraBuscada el valor de la etiquete
39         * de la pestaña que lanzo el evento
40         */
41        pestanaOprimida = (CoreCommandMenuItem) actionEvent.getComponent();
42        this.setLetraBuscada(pestanaOprimida.getText());
43        /*
44         * Invocar la ejecución del procedimiento que se está implementado
45         * en el Business Services.
46         */
47        bindings = getBindings();
48        operationBinding = bindings.getOperationBinding("buscarPorLetra");
49        resultado = operationBinding.execute();
50    }
```

c. En el PageDefinition, asignar al parámetro “letra” el valor variable “BuscadorBean.letraBuscada.“



Lab 08 – Empleo de ManageBean.

Paso 10: Probar la aplicación

Paso 11: En este instante la aplicación funciona correctamente, sin embargo el usuario no recibe una adecuada retroalimentación de la acción, ya que la pestaña oprimida no se observa seleccionada. Modifique, desde el BuscadorBean, la apariencia de las pestañas para indicar al usuario cual fue la letra seleccionada:

a. Adicionar el siguiente segmento de código al método oprimirLetra:

```
34          CoreMenuTabs uiTabs;
35          Iterator recorrer;
36          BindingContainer bindings;
37          OperationBinding operationBinding;
38          Object resultado;
39          /*
40           * Almacenar en la propiedad LetraBuscada el valor de la etiquete
41           * de la pestaña que lanzo el evento
42           */
43          pestanaOprimida = (CoreCommandMenuItem) actionEvent.getComponent();
44          this.setLetraBuscada(pestanaOprimida.getText());
45          /*
46           * Apagar todas la pestañas, para luego encender la seleccionada por el
47           * usuario.
48           */
49          uiTabs = (CoreMenuTabs) pestanaOprimida.getParent();
50          recorrer = uiTabs.getChildren().iterator();
51          while (recorrer.hasNext()){
52            CoreCommandMenuItem pestana;
53            pestana = (CoreCommandMenuItem) recorrer.next();
54            pestana.setSelected(false);       // apagar
55          }
56          pestanaOprimida.setSelected(true); // encender
57          /*
```