

CURSO DE JAVASERVER FACES

EJERCICIO

MANEJO DE VALUE CHANGE LISTENER EN JSF



Experiencia y Conocimiento para tu vida

CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

OBJETIVO DEL EJERCICIO

El objetivo de este ejercicio es poner en práctica el concepto Value Change Listener.

Configuraremos varios campos extra, y asociaremos un evento cuando cambie el valor del campo código postal, esto permitirá modificar los valores de los campos Ciudad y Colonia en automático. El resultado se muestra a continuación:

Ubaldo

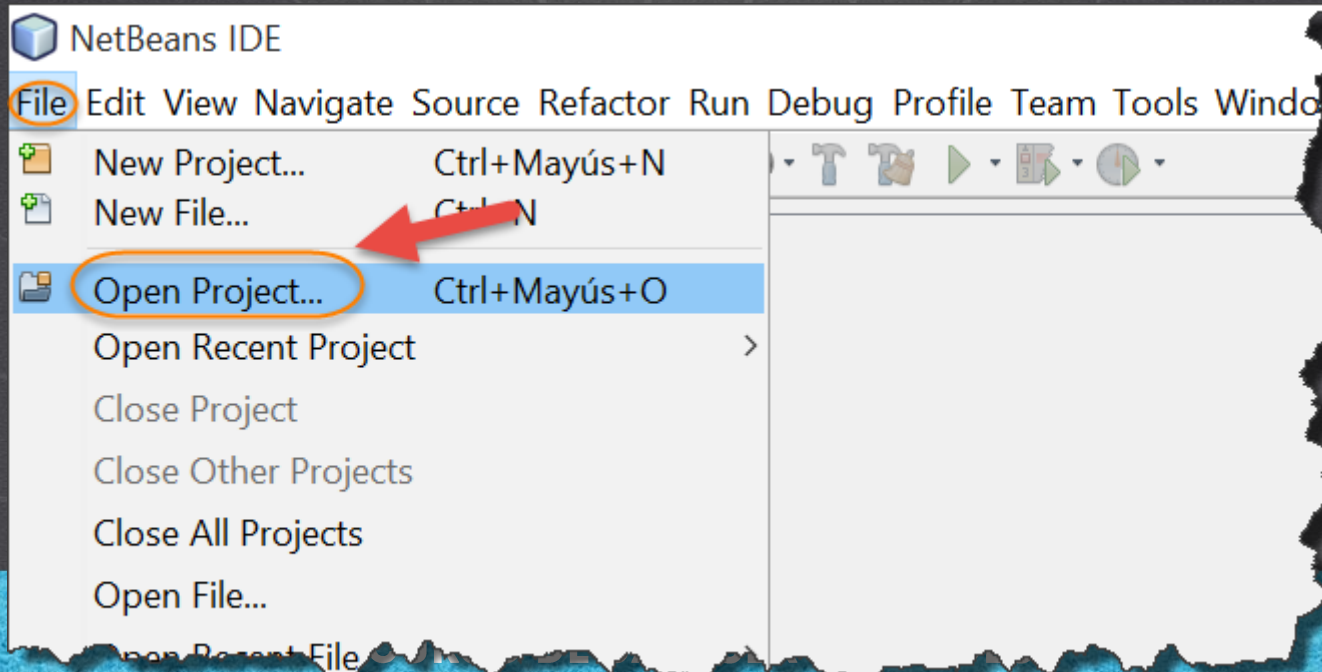
BolsaTrabajoV5

localhost:8080/BolsaTrabajoV5/faces/index.xhtml

Nombre	<input type="text" value="Introduce tu nombre"/>
Apellido	<input type="text" value="Introduce tu Apellido"/>
Sueldo deseado	<input type="text" value="0"/>
Fecha de Nacimiento	<input type="text"/>
Código Postal	<input type="text" value="03810"/>
Colonia	<input type="text" value="Nápoles"/>
Ciudad	<input type="text" value="Ciudad de Mexico"/>

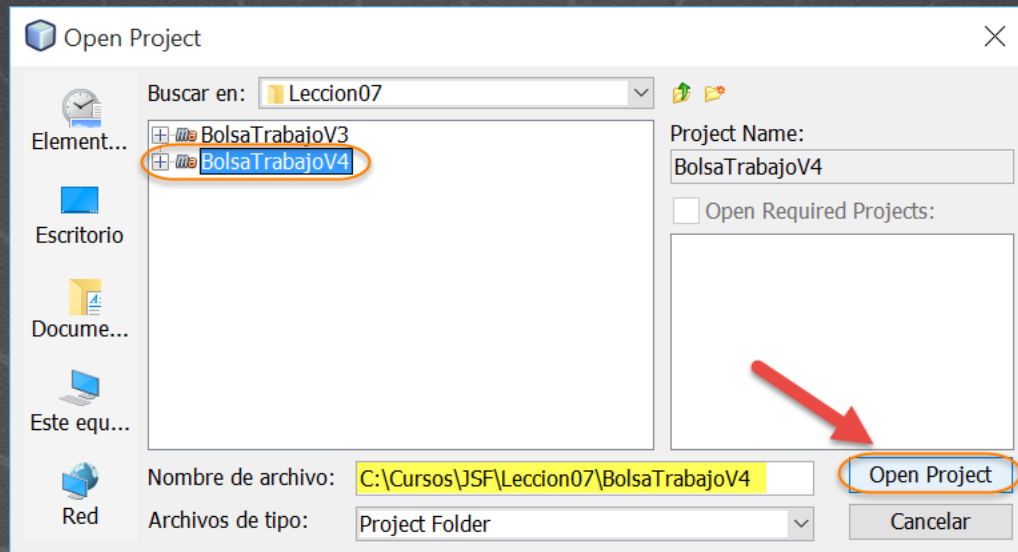
PASO 1. ABRIR EL PROYECTO

Abrimos el proyecto BolsaTrabajoV4, sólo en caso de que no lo tengamos ya abierto:



PASO 1. ABRIR EL PROYECTO

Abrimos el proyecto BolsaTrabajoV4, sólo en caso de que no lo tengamos ya abierto:

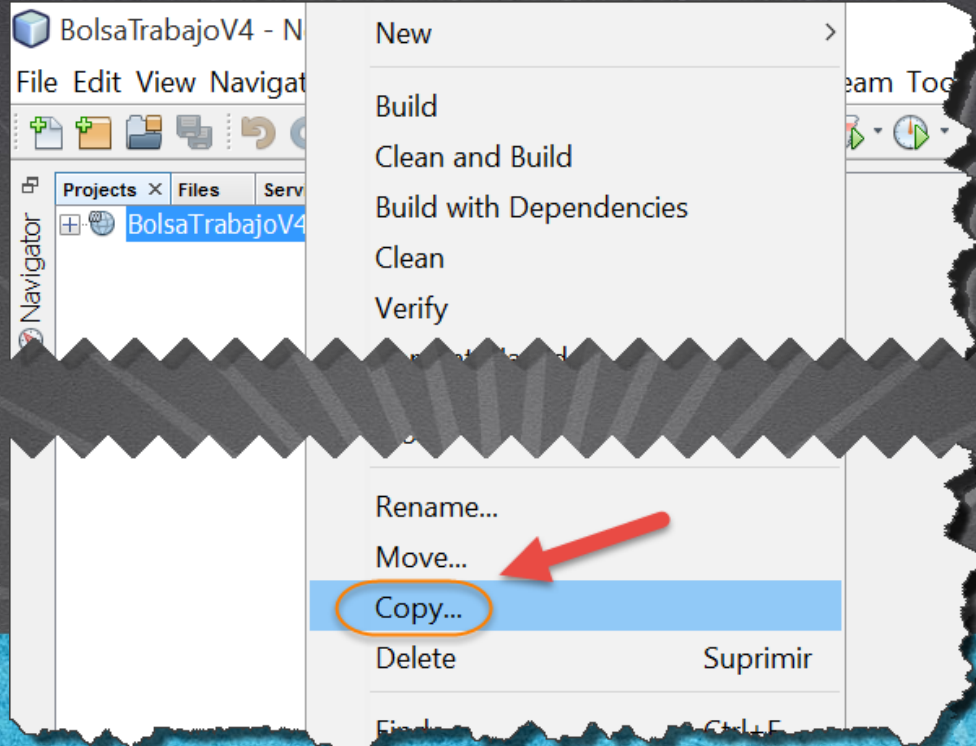


CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

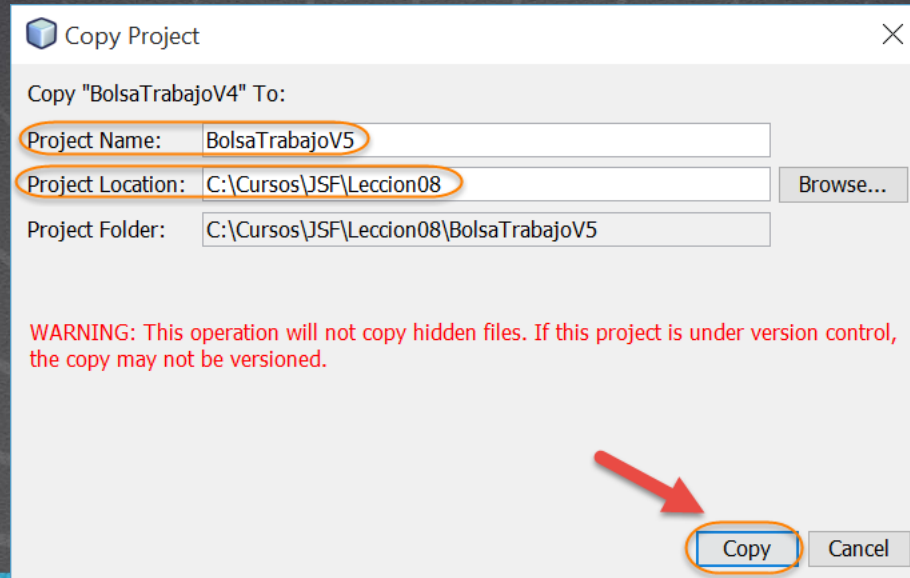
PASO 2. COPIAR EL PROYECTO

Copiamos el proyecto BolsaTrabajoV4:



PASO 2. COPIAR EL PROYECTO

Copiamos el proyecto BolsaTrabajoV4 y lo renombramos a BolsaTrabajoV5 y lo depositamos en la nueva ruta mostrada:

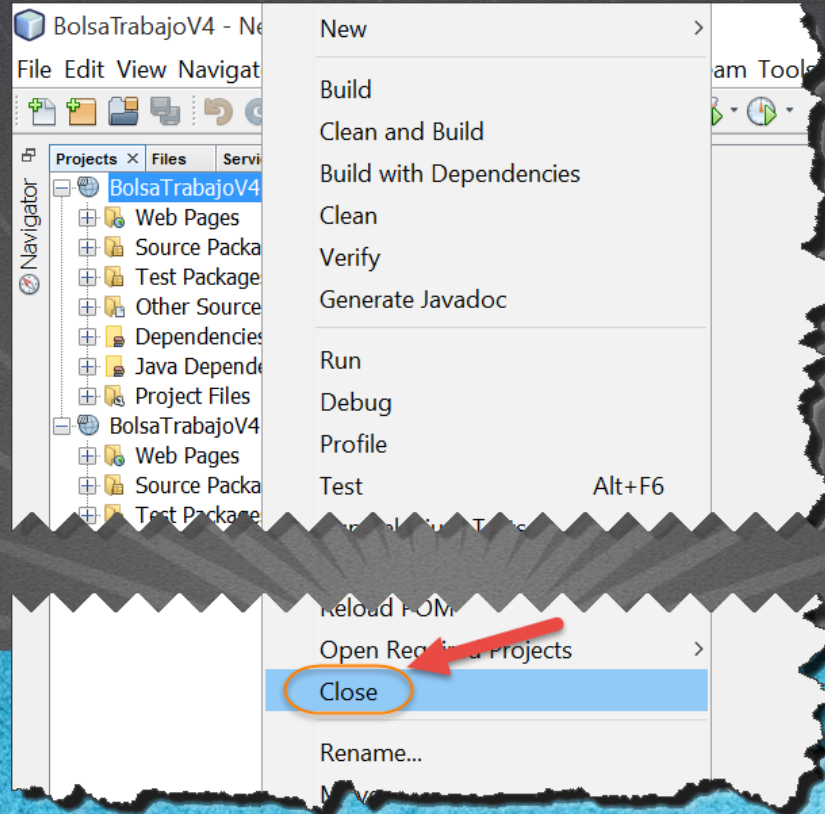


CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

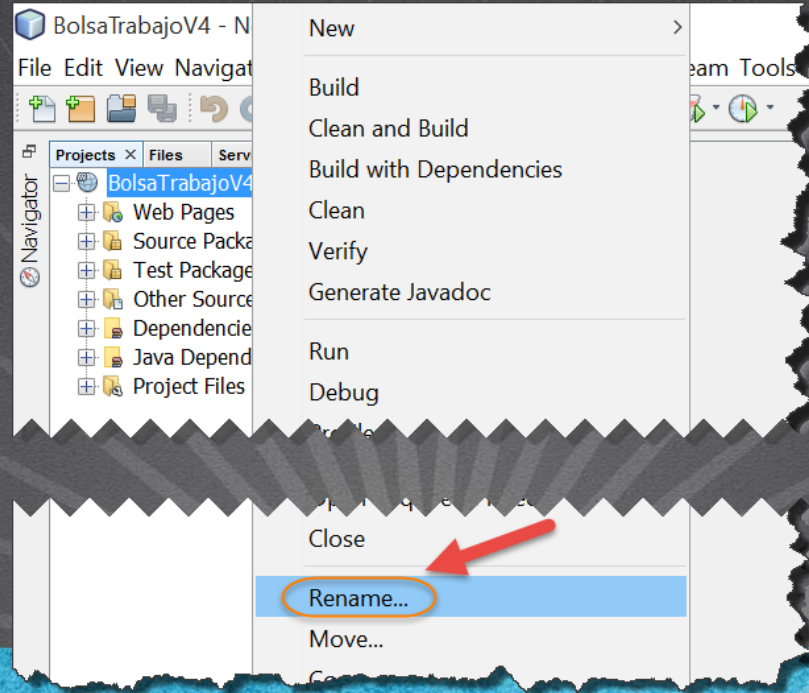
PASO 3. CERRAR PROYECTO

Cerramos el proyecto anterior, y dejamos solo el nuevo abierto:



PASO 4. RENOMBRAR PROYECTO

Cambiamos el nombre del proyecto a BolsaTrabajoV5:

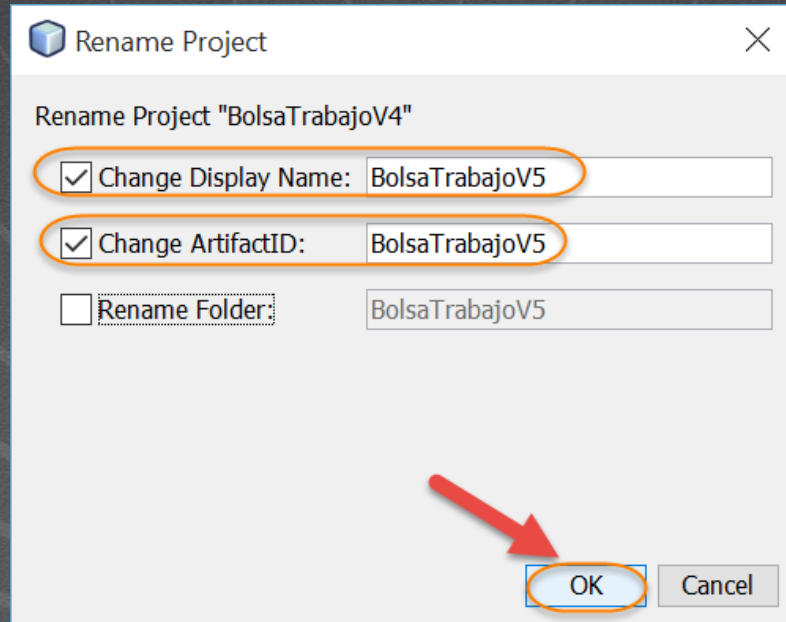


CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

PASO 4. RENOMBRAR PROYECTO

Cambiamos el nombre del proyecto a BolsaTrabajoV5:



CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

PASO 5. CAMBIAR UN ARCHIVO XHTML

Cambiamos el archivo index.xhtml como sigue:

Agregar al tag h:form el atributo `id = "vacanteForm"`, el resultado debe ser como el siguiente:

```
<h:form id="vacanteForm">
```


PASO 5. CAMBIAR UN ARCHIVO XHTML

Agregar el atributo codigoPostal al archivo index.xhtml.
Agregar el siguiente código, creando un nuevo renglón:

```
<tr>
  <td><h:outputLabel for="codigoPostal"
    value="#{msgs['vacanteForm.codigoPostal']}" /></td>
  <td><h:inputText id="codigoPostal" immediate="true"
    onchange="this.form.submit()" required="true"
    value="#{candidato.codigoPostal}"
    valueChangeListener="#{vacanteForm.codigoPostalListener}" /></td>
  <td><h:message for="codigoPostal" /></td>
</tr>
```

PASO 5. CAMBIAR UN ARCHIVO XHTML

Agregar el atributo colonia al archivo index.xhtml. Agregar el siguiente código, creando un nuevo renglón:

```
<tr>
  <td><h:outputLabel for="colonia" value="#{msgs['vacanteForm.colonia']}"
  /></td>
  <td><h:inputText id="colonia" required="true"
    value="#{candidato.colonia}" /></td>
  <td><h:message for="colonia" /></td>
</tr>
```


PASO 5. CAMBIAR UN ARCHIVO XHTML

Agregar el atributo ciudad al archivo index.xhtml. Agregar el siguiente código, creando un nuevo renglón:

```
<tr>
  <td><h:outputLabel for="ciudad" value="#{msgs['vacanteForm.ciudad']}"
  /></td>
  <td><h:inputText id="ciudad" required="true"
  value="#{candidato.ciudad}" /></td>
  <td><h:message for="ciudad" /></td>
</tr>
```

Agregar el atributo

PASO 5. MODIFICAMOS EL CÓDIGO

Archivo index.xhtml:

Dar click para ir al código

```
<?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:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
  <h:head>
    <title>BolsaTrabajoV5</title>
  </h:head>
  <h:body>
    <h:form id="vacanteForm">
      <h:messages globalOnly="true" ></h:messages>
      <table>
        <tr>
          <td><h:outputLabel for="nombre" value="#{msgs['vacanteForm.nombre']}" /></td>
          <td><h:inputText id="nombre" required="true" value="#{candidato.nombre}" /> </td>
          <td><h:message for="nombre" /></td>
        </tr>
        <tr>
          <td><h:outputLabel for="apellido" value="#{msgs['vacanteForm.apellido']}" /></td>
          <td><h:inputText id="apellido" required="true" value="#{candidato.apellido}" /></td>
          <td><h:message for="apellido" /></td>
        </tr>
      </table>
    </h:form>
  </h:body>
</html>
```


PASO 5. MODIFICAMOS EL CÓDIGO

[Archivo index.xhtml:](#)

Dar click para ir al código

```
<tr>
  <td><h:outputLabel for="sueldoDeseado" value="#{msgs['vacanteForm.sueldoDeseado']}" /></td>
  <td><h:inputText id="sueldoDeseado" required="true"
    value="#{candidato.sueldoDeseado}"
    <f:validateLongRange minimum="10000" maximum="50000" />
  </h:inputText></td>
  <td><h:message for="sueldoDeseado" /></td>
</tr>
<tr>
  <td><h:outputLabel for="fechaNacimiento" value="#{msgs['vacanteForm.nacimiento']}" /></td>
  <td><h:inputText id="fechaNacimiento" required="true"
    value="#{candidato.fechaNacimiento}"
    <f:convertDateTime pattern="MM/dd/yyyy" />
  </h:inputText></td>
  <td><h:message for="fechaNacimiento" /></td>
</tr>
<tr>
  <td><h:outputLabel for="codigoPostal" value="#{msgs['vacanteForm.codigoPostal']}" /></td>
  <td><h:inputText id="codigoPostal" immediate="true"
    onchange="this.form.submit()" required="true"
    value="#{candidato.codigoPostal}"
    valueChangeListener="#{vacanteForm.codigoPostalListener}" /></td>
  <td><h:message for="codigoPostal" /></td>
</tr>
```

PASO 5. MODIFICAMOS EL CÓDIGO

Archivo index.xhtml:

Dar click para ir al código

```
<tr>
  <td><h:outputLabel for="colonia" value="#{msgs['vacanteForm.colonia']}" /></td>
  <td><h:inputText id="colonia" required="true"
    value="#{candidato.colonia}" /></td>
  <td><h:message for="colonia" /></td>
</tr>
<tr>
  <td><h:outputLabel for="ciudad" value="#{msgs['vacanteForm.ciudad']}" /></td>
  <td><h:inputText id="ciudad" required="true"
    value="#{candidato.ciudad}" /></td>
  <td><h:message for="ciudad" /></td>
</tr>
</table>
<h:commandButton action="#{vacanteForm.enviar}"
  value="#{msgs['vacanteForm.enviar']}" />
</h:form>
</h:body>
</html>
```


PASO 6. CAMBIAR UNA CLASE JAVA

Cambiamos el código de la clase Candidato.java.

Agregamos los siguientes campos:

```
private String codigoPostal;  
private String colonia;  
private String ciudad;
```

Además los métodos getters/setters de cada atributo. Al final la clase Candidato.java debe quedar como sigue:

PASO 6. MODIFICAMOS EL CÓDIGO

Archivo Candidato.java:

Dar click para ir al código

```
package beans.model;

import java.util.Date;
import javax.inject.Named;
import javax.enterprise.context.RequestScoped;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

@RequestScoped
@Named
public class Candidato {

    Logger log = LogManager.getRootLogger();
    private String nombre = "Introduce tu nombre";
    private String apellido = "Introduce tu Apellido";
    private int sueldoDeseado;
    private Date fechaNacimiento;
    private String codigoPostal;
    private String colonia;
    private String ciudad;

    public Candidato() {
        log.info("Creando el objeto Candidato");
    }

    public String getNombre() {
        return nombre;
    }
}
```


PASO 6. MODIFICAMOS EL CÓDIGO

Archivo Candidato.java:

Dar click para ir al código

```
public void setNombre(String nombre) {
    this.nombre = nombre;
    log.info("Modificando la propiedad nombre:" + this.nombre);
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
    log.info("Modificando la propiedad apellido:" + this.apellido);
}

public int getSueldoDeseado() {
    return sueldoDeseado;
}

public void setSueldoDeseado(int sueldoDeseado) {
    this.sueldoDeseado = sueldoDeseado;
    log.info("Modificando la propiedad sueldoDeseado:" + this.sueldoDeseado);
}

public Date getFechaNacimiento() {
    return fechaNacimiento;
}
```

PASO 6. MODIFICAMOS EL CÓDIGO

Archivo Candidato.java:

Dar click para ir al código

```
public void setFechaNacimiento(Date fechaNacimiento) {
    this.fechaNacimiento = fechaNacimiento;
    log.info("Modificando la propiedad fechaNacimiento:" + this.fechaNacimiento);
}

public String getCodigoPostal() {
    return codigoPostal;
}

public void setCodigoPostal(String codigoPostal) {
    this.codigoPostal = codigoPostal;
}

public String getColonia() {
    return colonia;
}

public void setColonia(String colonia) {
    this.colonia = colonia;
}

public String getCiudad() {
    return ciudad;
}

public void setCiudad(String ciudad) {
    this.ciudad = ciudad;
}
}
```


PASO 7. CAMBIAR UNA CLASE JAVA

Cambiamos el código de la clase VacanteForm.java. Agregamos un método de tipo Value Change Listener, para que cuando modifiquemos el valor en el campo código postal, en automático llene los campos de Colonia y Ciudad.

La clase VacanteForm debe quedar como sigue:



Experiencia y Conocimiento para tu vida

CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

PASO 7. MODIFICAMOS EL CÓDIGO

Archivo VacanteForm.java:

Dar click para ir al código

```
package beans.backing;

import beans.model.Candidato;
import javax.inject.Named;
import javax.enterprise.context.RequestScoped;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIInput;
import javax.faces.component.UIViewRoot;
import javax.faces.context.FacesContext;
import javax.faces.event.ValueChangeEvent;
import javax.inject.Inject;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

@RequestScoped
@Named
public class VacanteForm {

    Logger log = LogManager.getRootLogger();

    @Inject
    private Candidato candidato;

    public VacanteForm() {
        log.info("Creando objeto VacanteForm");
    }
}
```


PASO 7. MODIFICAMOS EL CÓDIGO

Archivo VacanteForm.java:

Dar click para ir al código

```
public void setCandidato(Candidato candidato) {
    this.candidato = candidato;
}

//Metodo de flujo de control
public String enviar() {

    System.out.println("enviar() Nombre=" + this.candidato.getNombre());
    System.out.println("enviar() Apellido=" + this.candidato.getApellido());
    System.out.println("enviar() Sueldo deseado" + this.candidato.getSueldoDeseado());

    if (this.candidato.getNombre().equals("Juan")) {

        if (this.candidato.getApellido().equals("Perez")) {
            String msg = "Gracias, pero Juan Perez ya trabaja con nosotros.";
            FacesMessage facesMessage = new FacesMessage(FacesMessage.SEVERITY_ERROR, msg, msg);
            FacesContext facesContext = FacesContext.getCurrentInstance();
            String clientId = null; //Este es un mensaje global
            facesContext.addMessage(clientId, facesMessage);
            return "index";
        }
        return "exito";//exito.xhtml
    } else {
        return "fallo"; //fallo.xhtml
    }
}
```

PASO 7. MODIFICAMOS EL CÓDIGO

Archivo VacanteForm.java:

Dar click para ir al código

```
//Metodo de tipo Value Change Listener
public void codigoPostalListener(ValueChangeEvent valueChangeEvent) {
    FacesContext facesContext = FacesContext.getCurrentInstance();
    UIViewRoot uiViewRoot = facesContext.getViewRoot();
    String newCodigoPostal = (String) valueChangeEvent.getNewValue();
    if ("03810".equals(newCodigoPostal)) {

        log.info("Modificamos los valores de colonia y ciudad dinamicamente con ValueChangeListener");
        //Utilizamos el nombre del form de index.xhtml para encontrar el componente
        UIInput coloniaInputText = (UIInput) uiViewRoot.findComponent("vacanteForm:colonia");
        String colonia = "Nápoles";
        coloniaInputText.setValue(colonia);
        coloniaInputText.setSubmittedValue(colonia);

        UIInput ciudadInputText = (UIInput) uiViewRoot.findComponent("vacanteForm:ciudad");
        String ciudad = "Ciudad de Mexico";
        ciudadInputText.setValue(ciudad);
        ciudadInputText.setSubmittedValue(ciudad);

        facesContext.renderResponse();
    }
}
```


PASO 8. CAMBIAR UN ARCHIVO XML

Cambiamos el archivo faces-config.xml para colocar nuevamente el idioma en español.

```
<default-locale>es</default-locale>
```



Experiencia y Conocimiento para tu vida

CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

PASO 8. MODIFICAMOS EL CÓDIGO

Archivo faces-config.xml:

Dar click para ir al código

```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd" version="2.2">

  <lifecycle>
    <phase-listener>
      beans.ciclovida.DepuracionListener
    </phase-listener>
  </lifecycle>
  <application>
    <locale-config>
      <default-locale>es</default-locale>
    </locale-config>
    <!--etiquetas del formulario-->
    <resource-bundle>
      <!--No se necesita agregar la extension del archivo-->
      <base-name>mensajes</base-name>
      <var>msgs</var>
    </resource-bundle>
    <!--cambio de textos de validadores-->
    <message-bundle>jsf</message-bundle>
  </application>
</faces-config>
```

PASO 9. CAMBIAR UN ARCHIVO PROPERTIES

Cambiamos el archivo mensajes.properties y el archivo mensajes_en.properties para agregar los campos nuevos.

Los archivos deben quedar como siguen:



Experiencia y Conocimiento para tu vida

CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

PASO 9. MODIFICAMOS EL CÓDIGO

[Archivo mensajes.properties:](#)

Dar click para ir al código

```
vacanteForm.nombre=Nombre  
vacanteForm.apellido=Apellido  
vacanteForm.sueldoDeseado=Sueldo deseado  
vacanteForm.nacimiento=Fecha de Nacimiento  
vacanteForm.codigoPostal=Código Postal  
vacanteForm.colonia=Colonia  
vacanteForm.ciudad=Ciudad  
vacanteForm.enviar=Enviar
```

CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

PASO 10. MODIFICAMOS EL CÓDIGO

Archivo mensajes_en.properties:

Dar click para ir al código

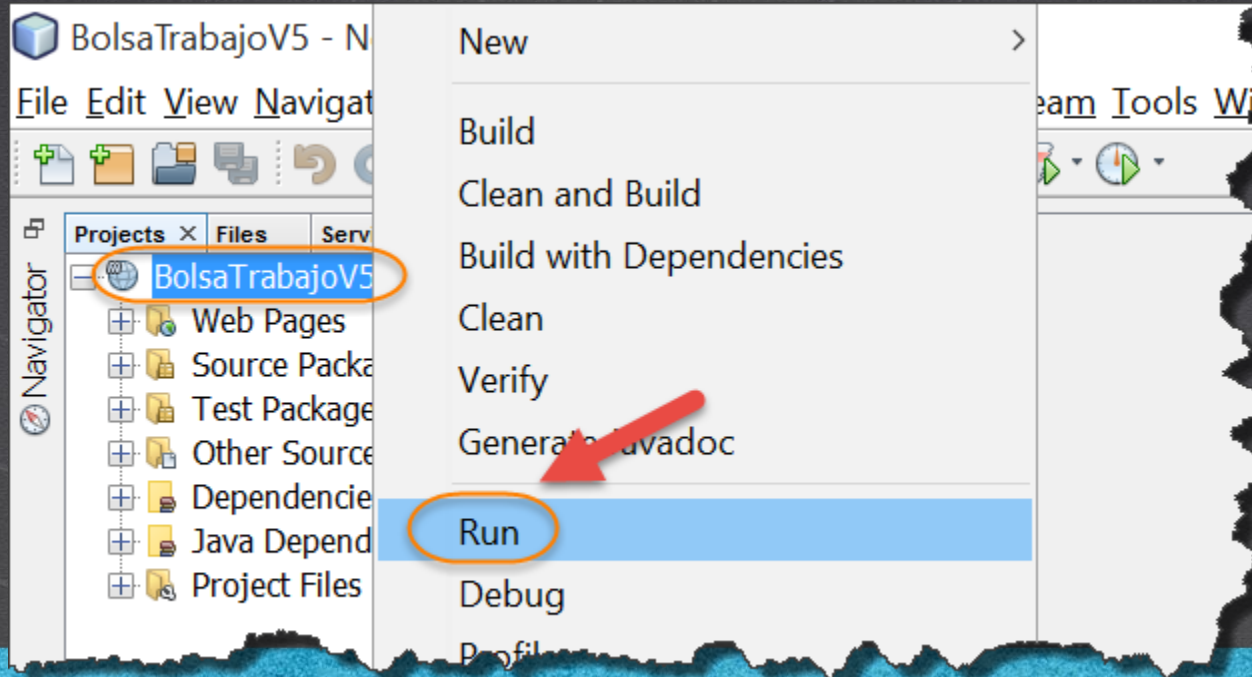
```
vacanteForm.nombre=Name  
vacanteForm.apellido=Last Name  
vacanteForm.sueldoDeseado=Salary desired  
vacanteForm.nacimiento=Date of birth  
vacanteForm.codigoPostal=Zip Code  
vacanteForm.colonia=Suburb  
vacanteForm.ciudad=City  
vacanteForm.enviar=Submit
```

CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

PASO 11. EJECUTAMOS EL PROYECTO

Ejecutamos el proyecto:



CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

PASO 11. EJECUTAMOS EL PROYECTO

Proporcionamos el valor de 03810 en el valor de código postal para observar que los campos de colonia y ciudad se llenan de manera automática. El resultado es como sigue:

Ubaldo

BolsaTrabajoV5

localhost:8080/BolsaTrabajoV5/faces/index.xhtml

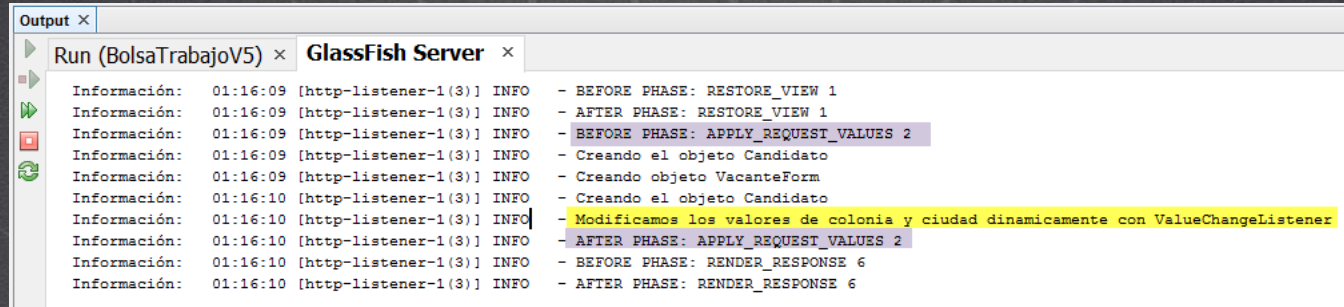
Nombre	<input type="text" value="Introduce tu nombre"/>
Apellido	<input type="text" value="Introduce tu Apellido"/>
Sueldo deseado	<input type="text" value="0"/>
Fecha de Nacimiento	<input type="text"/>
Código Postal	<input type="text" value="03810"/>
Colonia	<input type="text" value="Nápoles"/>
Ciudad	<input type="text" value="Ciudad de Mexico"/>

CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx

TAREAS EXTRAS

- Realizar la prueba con el código postal con el valor 03810 y revisar en el log del servidor el resultado de ejecutar el código value change listener y verificar el resultado según el ciclo de vida de JSF.



```
Output x
Run (BolsaTrabajoV5) x GlassFish Server x
Información: 01:16:09 [http-listener-1(3)] INFO - BEFORE PHASE: RESTORE_VIEW 1
Información: 01:16:09 [http-listener-1(3)] INFO - AFTER PHASE: RESTORE_VIEW 1
Información: 01:16:09 [http-listener-1(3)] INFO - BEFORE PHASE: APPLY_REQUEST_VALUES 2
Información: 01:16:09 [http-listener-1(3)] INFO - Creando el objeto Candidato
Información: 01:16:09 [http-listener-1(3)] INFO - Creando objeto VacanteForm
Información: 01:16:10 [http-listener-1(3)] INFO - Creando el objeto Candidato
Información: 01:16:10 [http-listener-1(3)] INFO - Modificamos los valores de colonia y ciudad dinamicamente con ValueChangeListener
Información: 01:16:10 [http-listener-1(3)] INFO - AFTER PHASE: APPLY_REQUEST_VALUES 2
Información: 01:16:10 [http-listener-1(3)] INFO - BEFORE PHASE: RENDER_RESPONSE 6
Información: 01:16:10 [http-listener-1(3)] INFO - AFTER PHASE: RENDER_RESPONSE 6
```

- Realizar la prueba con otros datos para observar que el comportamiento no es el mismo que con el código postal codificado para cambiar los valores de colonia y ciudad.

CONCLUSIÓN DEL EJERCICIO

- Con este ejercicio hemos agregado el concepto de Value Change Listener con JSF.
- Realizamos varios cambios en el formulario index.xhtml, sin embargo el cambio más importante es que al momento de cambiar el valor de código postal, en automático se manda a llamar un método en el Managed Bean de VacanteForm.java, y este método a su vez encuentra los elementos de colonia y ciudad del formulario y desde código Java modifica dinámicamente estos valores desde la clase VacanteForm.java.

CURSO ONLINE

JAVASERVER FACES (JSF)

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

CURSO DE JAVASERVER FACES

www.globalmentoring.com.mx