

# **CURSO DE JAVASERVER FACES**

## **EJERCICIO**

### **CICLO DE VIDA EN JSF**



Experiencia y Conocimiento para tu vida

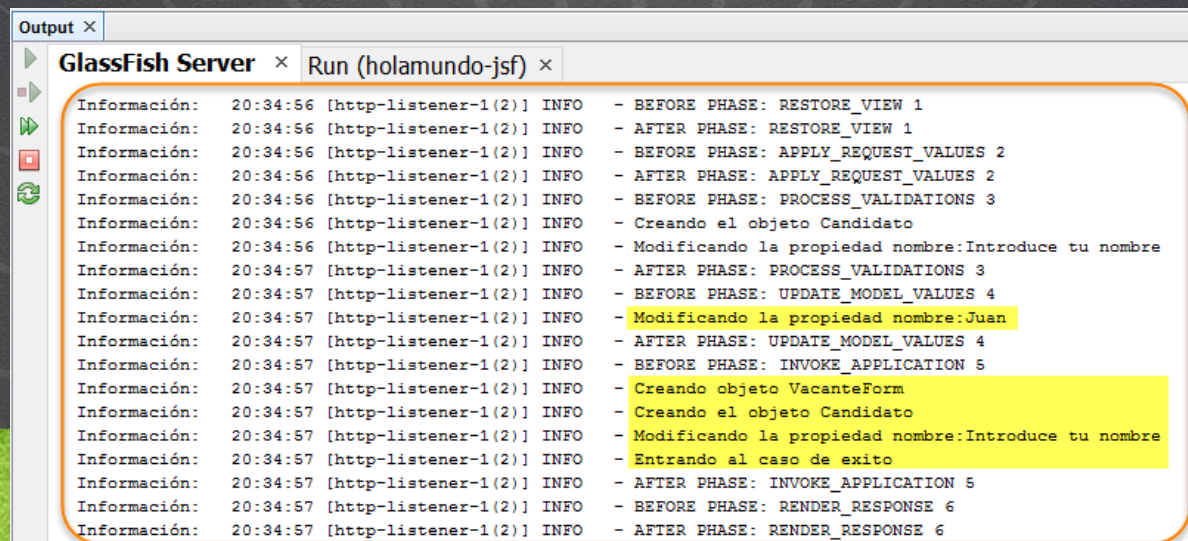
## **CURSO DE JAVASERVER FACES**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# OBJETIVO DEL EJERCICIO

En este ejercicio crearemos las clases necesarias para mostrar el ciclo de vida en la consola de nuestro servidor Java EE. Utilizaremos log4j para mostrar en la consola el resultado del ciclo de vida y cada una de sus etapas.

El resultado deberá ser similar al mostrado a continuación:

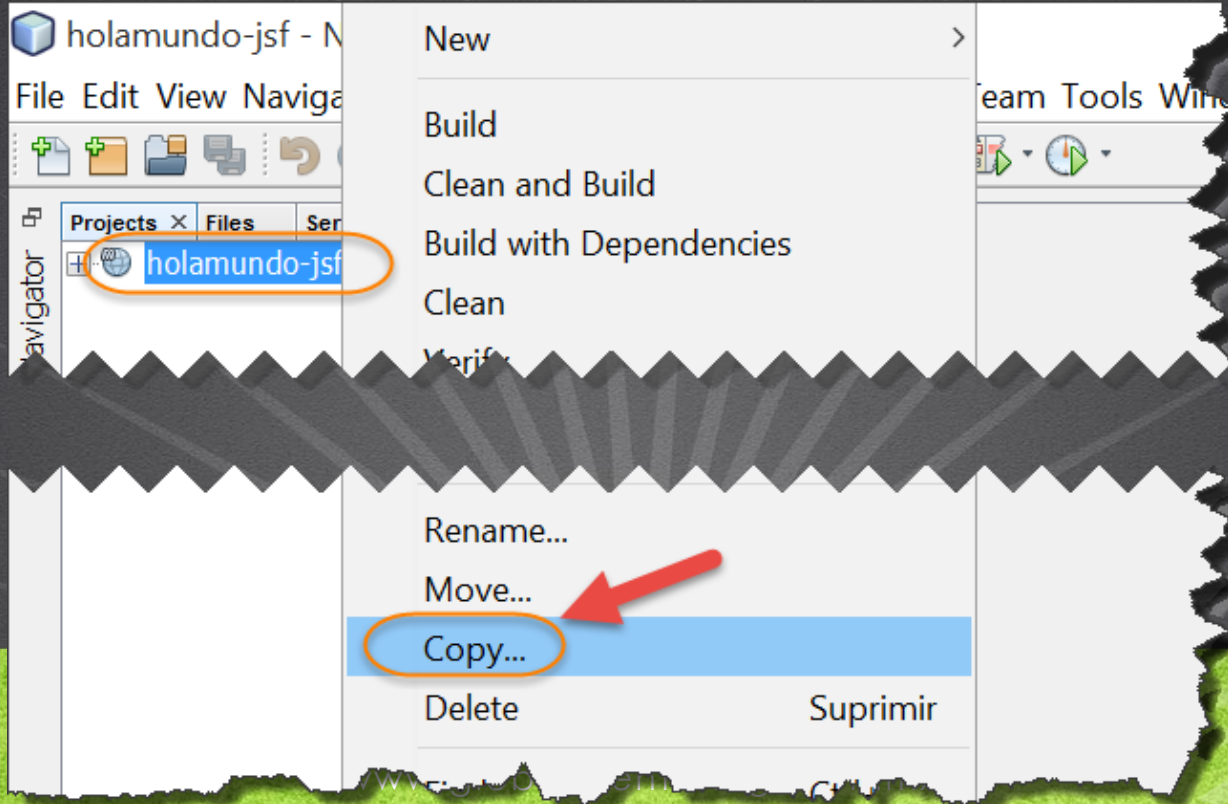


```
Output x
GlassFish Server x Run (holamundo-jsf) x
Información: 20:34:56 [http-listener-1(2)] INFO - BEFORE PHASE: RESTORE_VIEW 1
Información: 20:34:56 [http-listener-1(2)] INFO - AFTER PHASE: RESTORE_VIEW 1
Información: 20:34:56 [http-listener-1(2)] INFO - BEFORE PHASE: APPLY_REQUEST_VALUES 2
Información: 20:34:56 [http-listener-1(2)] INFO - AFTER PHASE: APPLY_REQUEST_VALUES 2
Información: 20:34:56 [http-listener-1(2)] INFO - BEFORE PHASE: PROCESS_VALIDATIONS 3
Información: 20:34:56 [http-listener-1(2)] INFO - Creando el objeto Candidato
Información: 20:34:56 [http-listener-1(2)] INFO - Modificando la propiedad nombre:Introduce tu nombre
Información: 20:34:57 [http-listener-1(2)] INFO - AFTER PHASE: PROCESS_VALIDATIONS 3
Información: 20:34:57 [http-listener-1(2)] INFO - BEFORE PHASE: UPDATE_MODEL_VALUES 4
Información: 20:34:57 [http-listener-1(2)] INFO - Modificando la propiedad nombre:Juan
Información: 20:34:57 [http-listener-1(2)] INFO - AFTER PHASE: UPDATE_MODEL_VALUES 4
Información: 20:34:57 [http-listener-1(2)] INFO - BEFORE PHASE: INVOKE_APPLICATION 5
Información: 20:34:57 [http-listener-1(2)] INFO - Creando objeto VacanteForm
Información: 20:34:57 [http-listener-1(2)] INFO - Creando el objeto Candidato
Información: 20:34:57 [http-listener-1(2)] INFO - Modificando la propiedad nombre:Introduce tu nombre
Información: 20:34:57 [http-listener-1(2)] INFO - Entrando al caso de exito
Información: 20:34:57 [http-listener-1(2)] INFO - AFTER PHASE: INVOKE_APPLICATION 5
Información: 20:34:57 [http-listener-1(2)] INFO - BEFORE PHASE: RENDER_RESPONSE 6
Información: 20:34:57 [http-listener-1(2)] INFO - AFTER PHASE: RENDER_RESPONSE 6
```



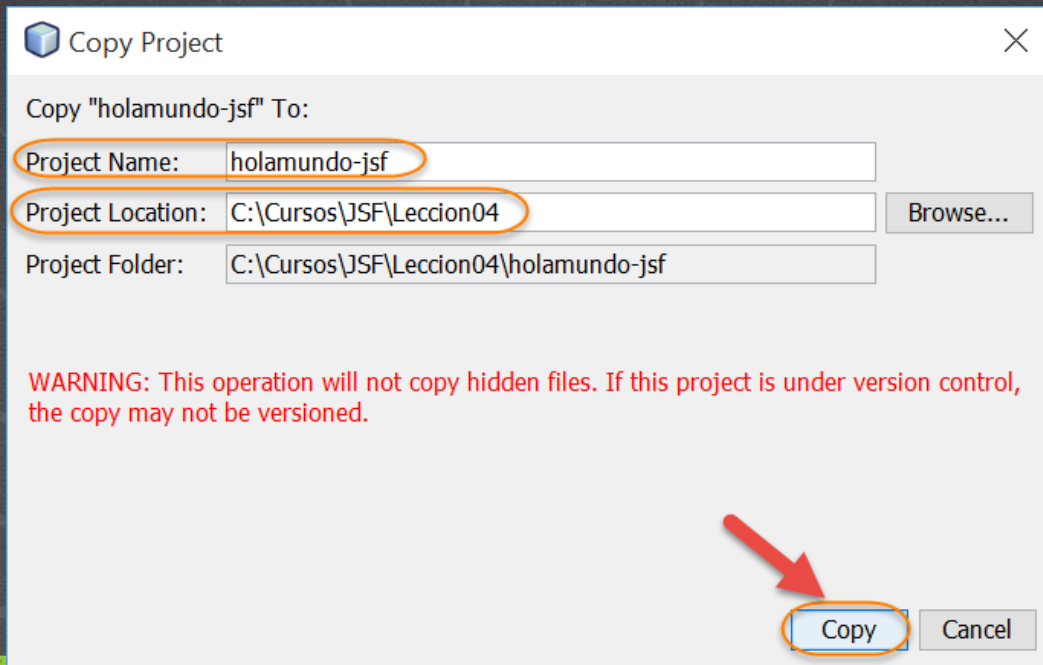
# PASO 1. COPIAR EL PROYECTO

Copiamos el proyecto holamundo-jsf (la última versión):



# PASO 1. COPIAR EL PROYECTO

Copiamos el proyecto holamundo-jsf (la última versión):

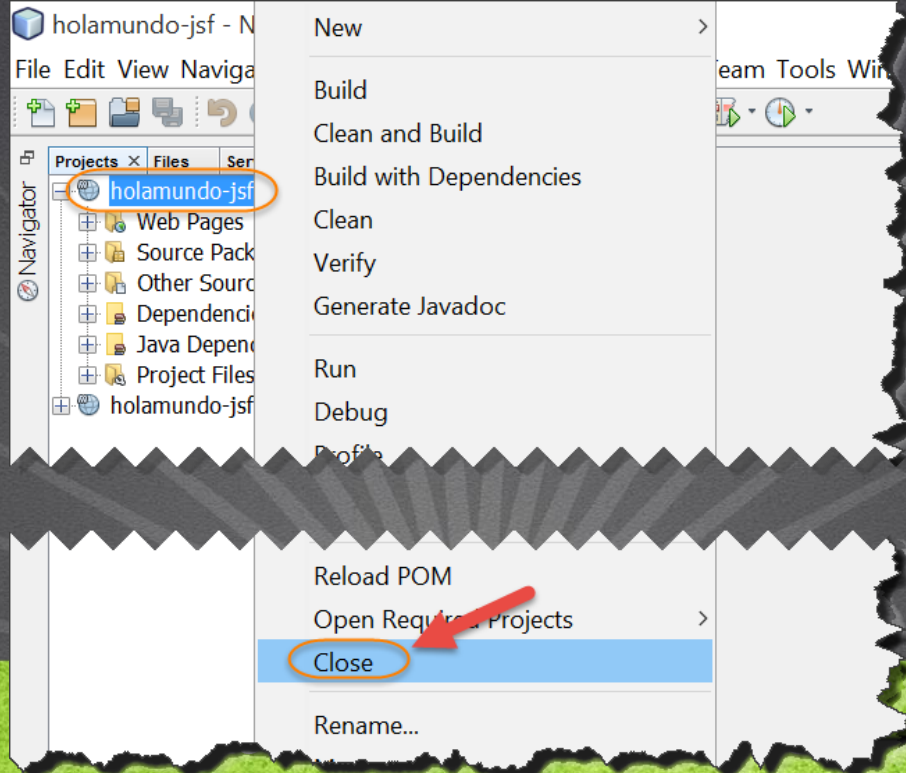


**CURSO DE JAVASERVER FACES**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# PASO 2. CERRAR PROYECTO

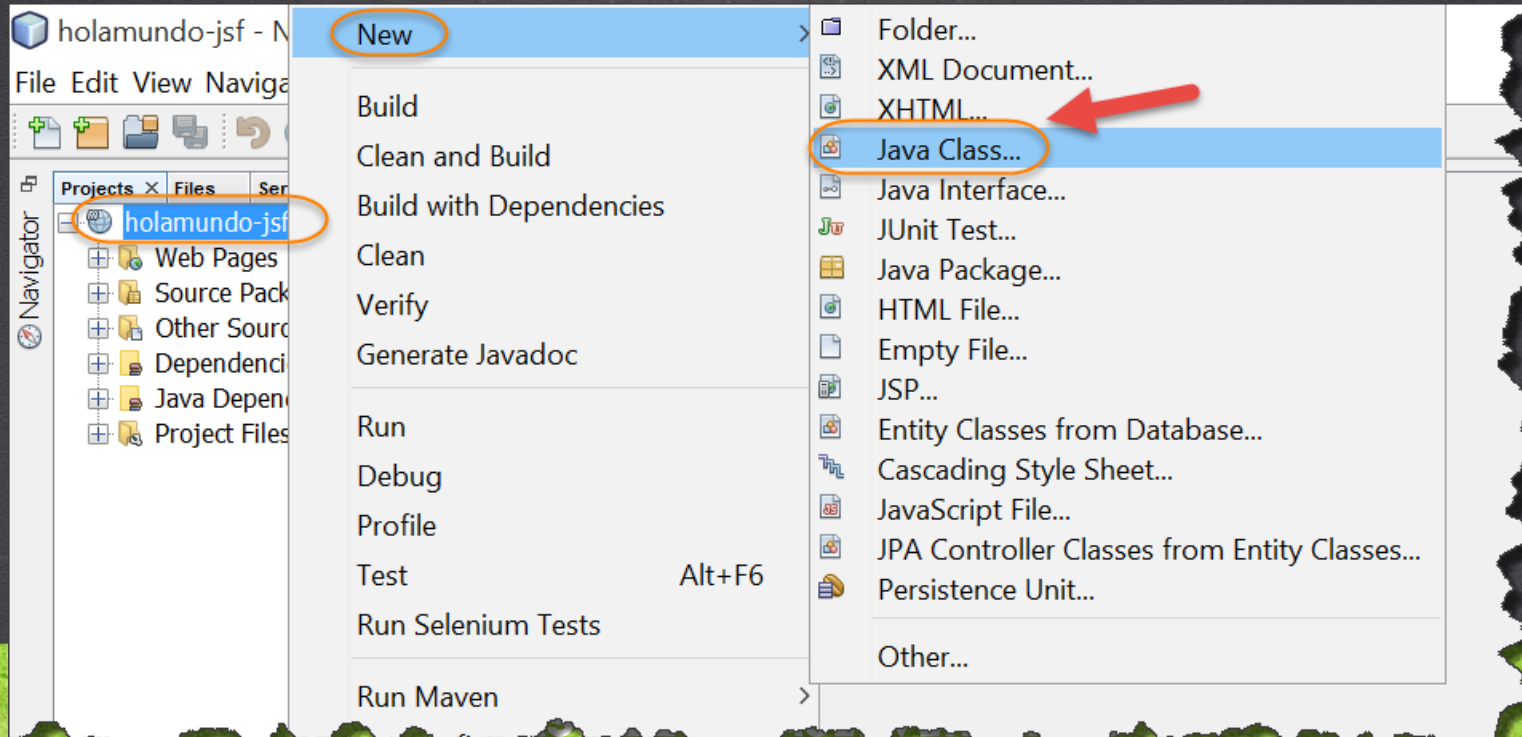
Cerramos el proyecto anterior, y dejamos solo el nuevo abierto:





# PASO 3. CREAR UNA CLASE

Creamos la clase DepuracionListener:



# PASO 3. CREAR UNA CLASE

Creamos la clase DepuracionListener:

New Java Class

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name: DepuracionListener

Project: holamundo-jsf

Location: Source Packages

Package: beans.ciclovida

Created File: C:\Leccion04\holamundo-jsf\src\main\java\beans\ciclovida\DepuracionListener.java

< Back Next > **Finish** Cancel Help

**CURSO DE JAVASERVER FACES**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# PASO 4. MODIFICAMOS EL CÓDIGO

Archivo DepuracionListener.java:

Dar click para ir al código

```
package beans.ciclovida;

import javax.faces.event.PhaseEvent;
import javax.faces.event.PhaseId;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class DepuracionListener implements javax.faces.event.PhaseListener {
    Logger log = LogManager.getRootLogger();

    @Override
    public void afterPhase(PhaseEvent phaseEvent) {

        if (log.isInfoEnabled()) {
            log.info("AFTER PHASE: " + phaseEvent.getPhaseId().toString());
        }
    }

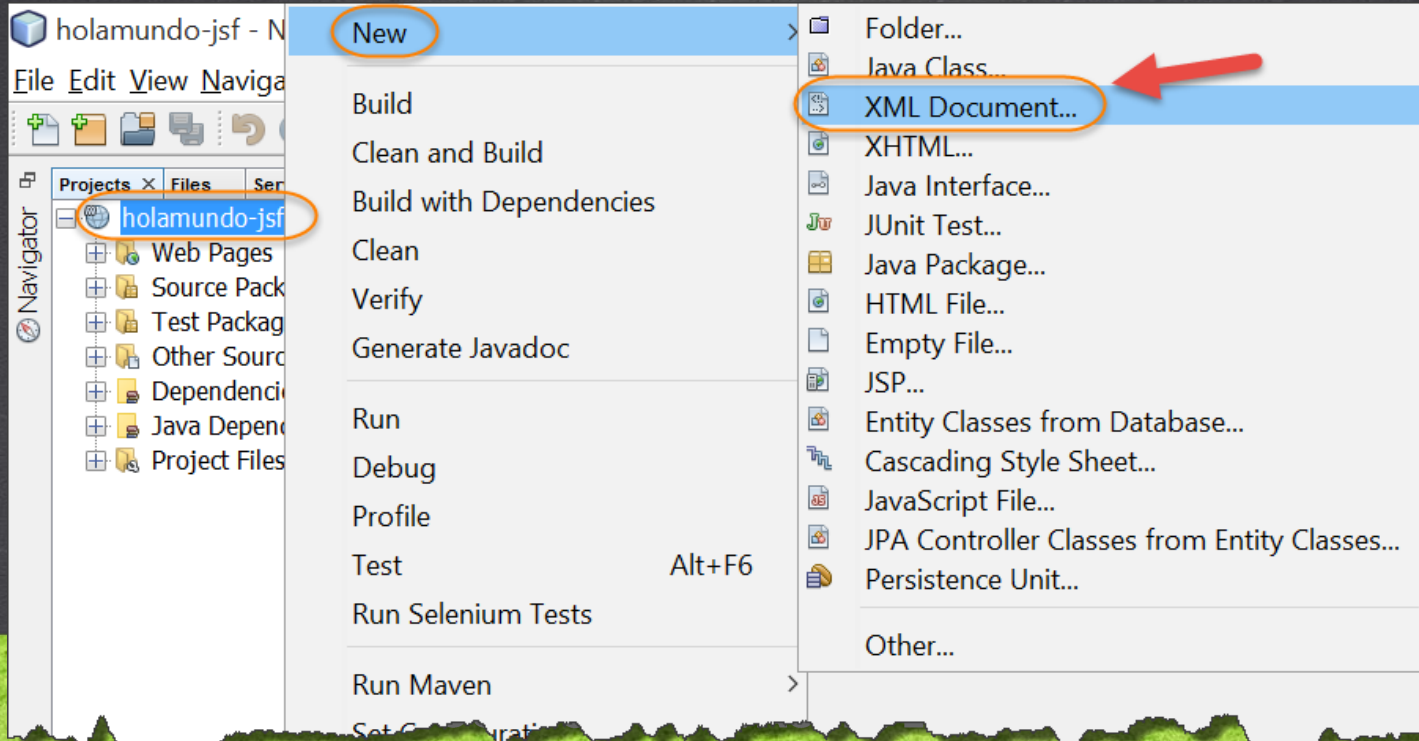
    @Override
    public void beforePhase(PhaseEvent phaseEvent) {
        if (log.isInfoEnabled()) {
            log.info("BEFORE PHASE: " + phaseEvent.getPhaseId().toString());
        }
    }

    @Override
    public PhaseId getPhaseId() {
        return PhaseId.ANY_PHASE;
    }
}
```



# PASO 5. CREAR ARCHIVO XML

Creamos el archivo faces-config.xml:



# PASO 5. CREAR ARCHIVO XML

Creamos el archivo faces-config.xml:

New XML Document

**Steps**

1. Choose File Type
2. **Name and Location**
3. Select Document Type
4. ...

**Name and Location**

File Name: faces-config

Project: holamundo-jsf

Folder: src/main/webapp/WEB-INF [Browse...](#)

Created File: amundo-jsf\src/main/webapp/WEB-INF\faces-config.xml

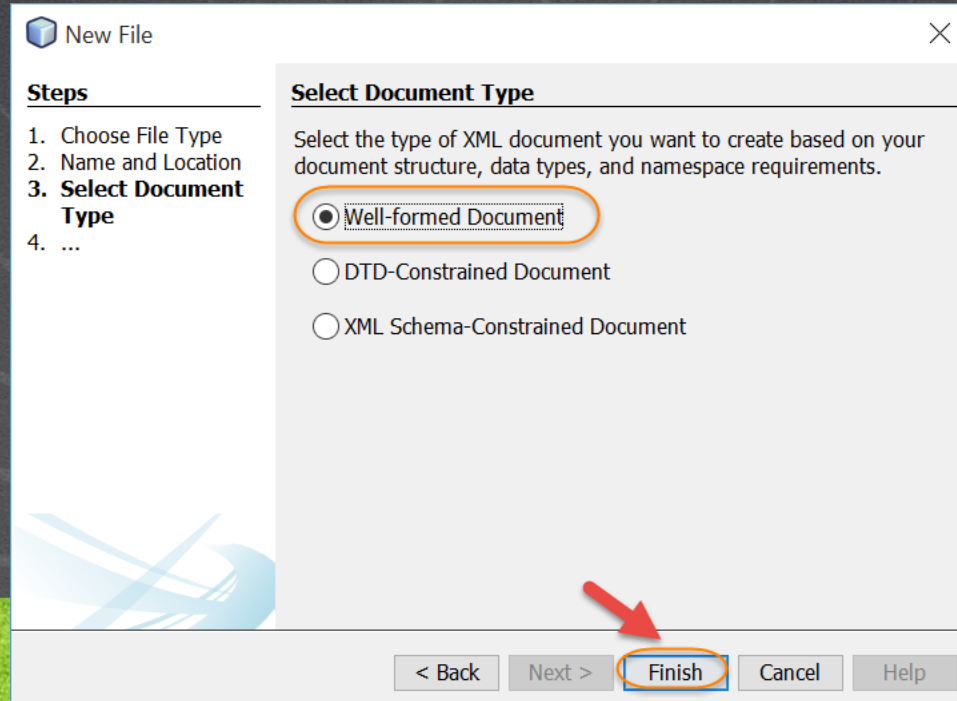
< Back **Next >** Finish Cancel Help

**CURSO DE JAVASERVER FACES**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# PASO 5. CREAR ARCHIVO XML

Creamos el archivo faces-config.xml. Seleccionamos cualquier opción, no es importante ya que lo vamos a sobrescribir:





# PASO 6. MODIFICAMOS EL CÓDIGO

## Archivo faces-config.xml:

Dar click para ir al código

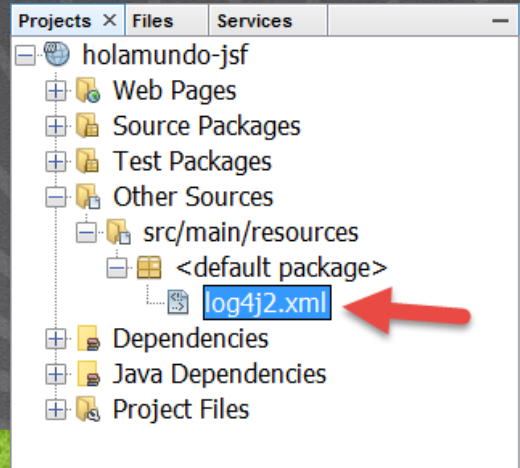
```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd" version="2.2">

    <lifecycle>
        <phase-listener>
            beans.ciclovida.DepuracionListener
        </phase-listener>
    </lifecycle>

</faces-config>
```

# PASO 7. MODIFICAR ARCHIVO XML

Modificamos el archivo log4j2.xml. Agregamos la clase de DepuracionListener y la ponemos en modo DEBUG para que podamos observar las fases por las que pasa el ciclo de vida de JSF al momento de hacer una petición, y recibir la respuesta de la aplicación web. El archivo se encuentra en la ruta siguiente:



**CURSO DE JAVASERVER FACES**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# PASO 7. MODIFICAMOS EL CÓDIGO

Archivo log4j2.xml:

Dar click para ir al código

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="INFO">
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss} [%t] %-5level %logger{36} - %msg%n" />
    </Console>
  </Appenders>
  <Loggers>
    <logger name="beans.ciclovida.DepuracionListener" level="DEBUG" additivity="false">
      <AppenderRef ref="Console"/>
    </logger>
    <Root level="info">
      <AppenderRef ref="Console" />
    </Root>
  </Loggers>
</Configuration>
```



## PASO 8. MODIFICAR ARCHIVO XML

Modificar la clase Candidato.java para mandar al log del servidor si se ha creado un objeto desde el constructor de la clase, y otro mensaje para saber si se ha modificado la propiedad de nombre.



Experiencia y Conocimiento para tu vida

**CURSO DE JAVASERVER FACES**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# PASO 8. MODIFICAMOS EL CÓDIGO

## Archivo Candidato.java:

Dar click para ir al código

```
package beans.model;

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;

    public Candidato() {
        log.info("Creando el objeto Candidato");
        this.setNombre("Introduce tu nombre");
    }

    public String getNombre() {
        return nombre;
    }

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

## PASO 9. MODIFICAR ARCHIVO XML

Modificar la clase VacanteForm.java para mandar al log del servidor si se ha creado un objeto y también si se ha entrado en el caso de éxito o fallo, según el valor introducido en la caja de texto.



Experiencia y Conocimiento para tu vida

**CURSO DE JAVASERVER FACES**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



# PASO 9. 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.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");
    }

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

# PASO 9. MODIFICAMOS EL CÓDIGO

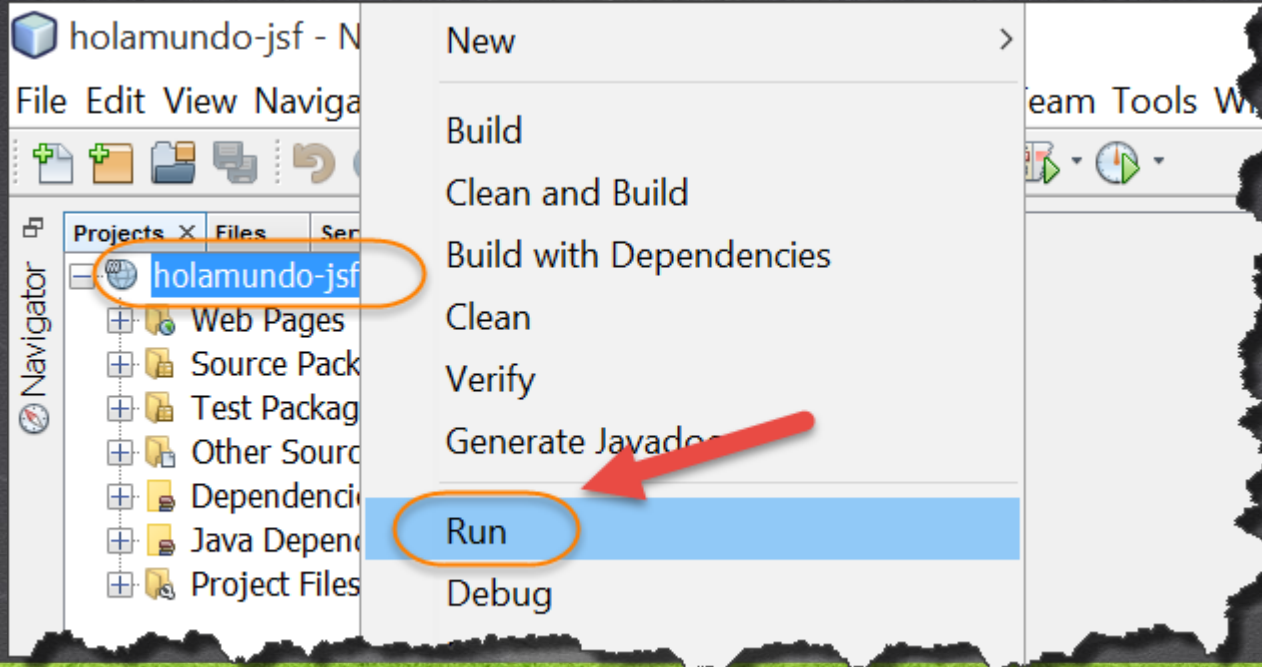
[Archivo VacanteForm.java:](#)

Dar click para ir al código

```
public String enviar() {  
    if (this.candidato.getNombre().equals("Juan")) {  
        log.info("Entrando al caso de exito");  
        return "exito";  
    } else {  
        log.info("Entrando al caso de fallo");  
        return "fallo";  
    }  
}
```

# PASO 10. EJECUTAMOS EL PROYECTO

Ejecutamos el proyecto:



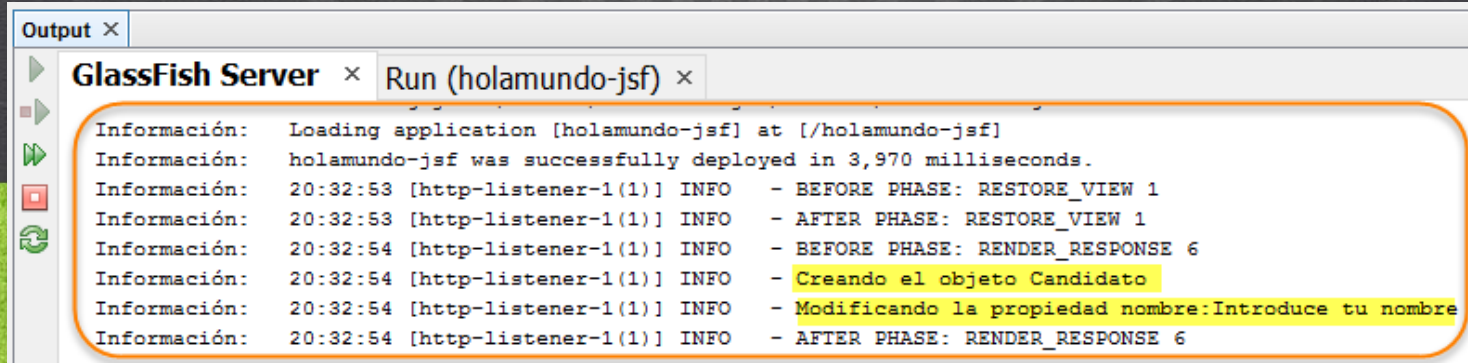
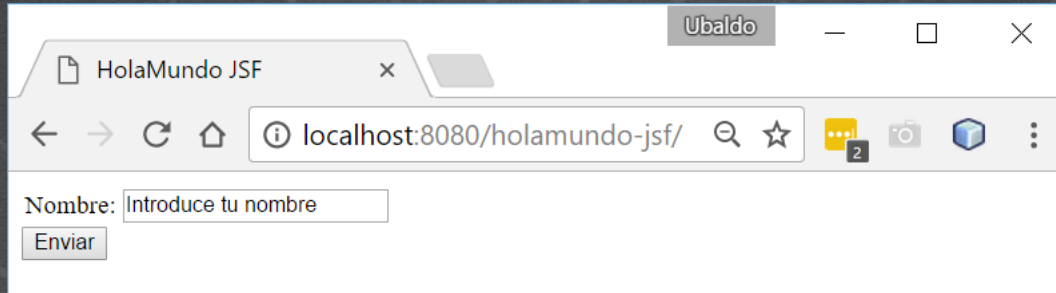
**CURSO DE JAVASERVER FACES**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



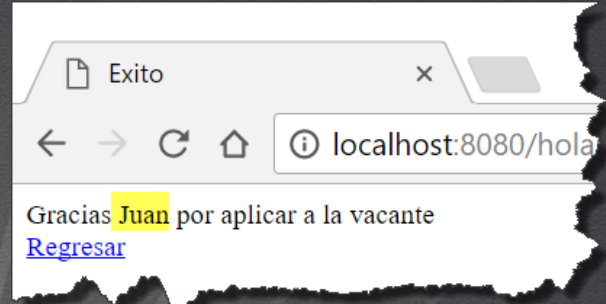
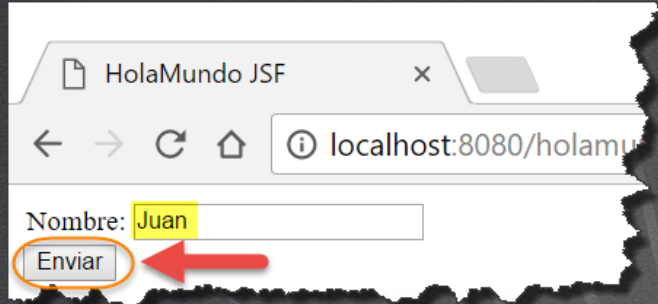
# PASO 10. EJECUTAMOS EL PROYECTO

Ejecutamos nuestra aplicación y obtendremos los valores de cada una de las fases del ciclo de vida de JSF. Esta clase nos permitirá observar en qué fase nos encontramos al procesar las acciones. Cuando se muestra la página inicial obtenemos la siguiente salida:



# PASO 10. EJECUTAMOS EL PROYECTO

Al proporcionar el valor de Juan en el campo de texto vemos la siguiente salida:



```
Output x
GlassFish Server x Run (holamundo-jsf) x
Información: 20:34:56 [http-listener-1(2)] INFO - BEFORE PHASE: RESTORE_VIEW 1
Información: 20:34:56 [http-listener-1(2)] INFO - AFTER PHASE: RESTORE_VIEW 1
Información: 20:34:56 [http-listener-1(2)] INFO - BEFORE PHASE: APPLY_REQUEST_VALUES 2
Información: 20:34:56 [http-listener-1(2)] INFO - AFTER PHASE: APPLY_REQUEST_VALUES 2
Información: 20:34:56 [http-listener-1(2)] INFO - BEFORE PHASE: PROCESS_VALIDATIONS 3
Información: 20:34:56 [http-listener-1(2)] INFO - Creando el objeto Candidato
Información: 20:34:56 [http-listener-1(2)] INFO - Modificando la propiedad nombre:Introduce tu nombre
Información: 20:34:57 [http-listener-1(2)] INFO - AFTER PHASE: PROCESS_VALIDATIONS 3
Información: 20:34:57 [http-listener-1(2)] INFO - BEFORE PHASE: UPDATE_MODEL_VALUES 4
Información: 20:34:57 [http-listener-1(2)] INFO - Modificando la propiedad nombre:Juan
Información: 20:34:57 [http-listener-1(2)] INFO - AFTER PHASE: UPDATE_MODEL_VALUES 4
Información: 20:34:57 [http-listener-1(2)] INFO - BEFORE PHASE: INVOKE_APPLICATION 5
Información: 20:34:57 [http-listener-1(2)] INFO - Creando objeto VacanteForm
Información: 20:34:57 [http-listener-1(2)] INFO - Creando el objeto Candidato
Información: 20:34:57 [http-listener-1(2)] INFO - Modificando la propiedad nombre:Introduce tu nombre
Información: 20:34:57 [http-listener-1(2)] INFO - Entrando al caso de exito
Información: 20:34:57 [http-listener-1(2)] INFO - AFTER PHASE: INVOKE_APPLICATION 5
Información: 20:34:57 [http-listener-1(2)] INFO - BEFORE PHASE: RENDER_RESPONSE 6
Información: 20:34:57 [http-listener-1(2)] INFO - AFTER PHASE: RENDER_RESPONSE 6
```



# PASO 10. EJECUTAMOS EL PROYECTO

Al presionar el botón de regresar observamos la siguiente salida, la cual es distinta a cuando se solicitó por primera vez la página al iniciar la aplicación:

The diagram illustrates the sequence of events when navigating back to a page in a JSF application. It starts with a browser window titled 'Exito' displaying 'Gracias Juan por aplicar a la vacante' and a 'Regresar' button. An arrow points to another browser window titled 'HolaMundo JSF' showing a form with the text 'Nombre: Introduce tu nombre' and an 'Enviar' button. A curved arrow then points to the 'Output' console of the 'GlassFish Server', which shows the server-side processing of the request, including phase names and the creation/modification of the 'Candidato' object.

**Browser Window 1 (Exito):**

Gracias Juan por aplicar a la vacante  
[Regresar](#)

**Browser Window 2 (HolaMundo JSF):**

Nombre: Introduce tu nombre  
Enviar

**GlassFish Server Output (Run (holamundo-jsf)):**

```
Información: 20:42:25 [http-listener-1(4)] INFO - BEFORE PHASE: RESTORE_VIEW 1
Información: 20:42:25 [http-listener-1(4)] INFO - AFTER PHASE: RESTORE_VIEW 1
Información: 20:42:25 [http-listener-1(4)] INFO - BEFORE PHASE: APPLY_REQUEST_VALUES 2
Información: 20:42:25 [http-listener-1(4)] INFO - AFTER PHASE: APPLY_REQUEST_VALUES 2
Información: 20:42:25 [http-listener-1(4)] INFO - BEFORE PHASE: PROCESS_VALIDATIONS 3
Información: 20:42:25 [http-listener-1(4)] INFO - AFTER PHASE: PROCESS_VALIDATIONS 3
Información: 20:42:25 [http-listener-1(4)] INFO - BEFORE PHASE: UPDATE_MODEL_VALUES 4
Información: 20:42:25 [http-listener-1(4)] INFO - AFTER PHASE: UPDATE_MODEL_VALUES 4
Información: 20:42:25 [http-listener-1(4)] INFO - BEFORE PHASE: INVOKE_APPLICATION 5
Información: 20:42:25 [http-listener-1(4)] INFO - AFTER PHASE: INVOKE_APPLICATION 5
Información: 20:42:25 [http-listener-1(4)] INFO - BEFORE PHASE: RENDER_RESPONSE 6
Información: 20:42:25 [http-listener-1(4)] INFO - Creando el objeto Candidato
Información: 20:42:25 [http-listener-1(4)] INFO - Modificando la propiedad nombre:Introduce tu nombre
Información: 20:42:25 [http-listener-1(4)] INFO - AFTER PHASE: RENDER_RESPONSE 6
```



# CONCLUSIÓN DEL EJERCICIO

Con este ejercicio hemos puesto en práctica el ciclo de vida en JSF.

Hicimos varias pruebas y observamos como en cada una de ellas obtenemos ciertos valores en distintas etapas del ciclo de vida de JSF. Esta clase de `DepuracionListener` nos va a servir para depurar nuestra aplicación y poder detectar errores más fácilmente y en qué fase o etapa se está provocando el error.



Experiencia y Conocimiento para tu vida

**CURSO DE JAVASERVER FACES**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

**CURSO ONLINE**

# **JAVASERVER FACES (JSF)**

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

**CURSO DE JAVASERVER FACES**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)