# 0110510-jsf-logo.tiff

**Ejercicios Básicos**

## Ejercicio #1 Inicio de un proyecto con JSF

**Objetivo**: Crear un proyecto Maven que pueda ser desplegado en un servidor de aplicaciones (Jboss preferentemente) que implemente la tecnología JSF.

El resultado será similar al siguiente:

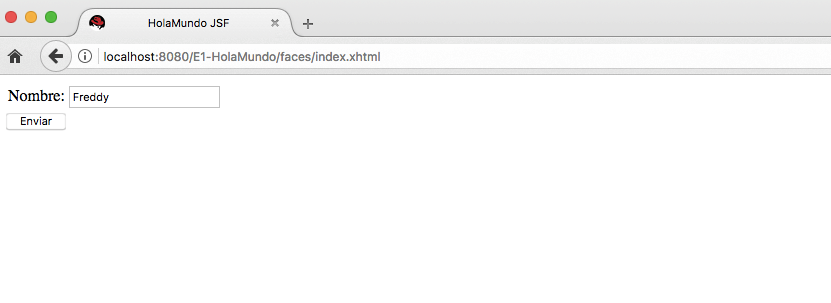


Ilustración 1 - Resultado esperado del ejercicio 1

Pasos a seguir:

1. Creación de un proyecto Maven con ayuda del IDE NetBeans como se muestra a continuación:

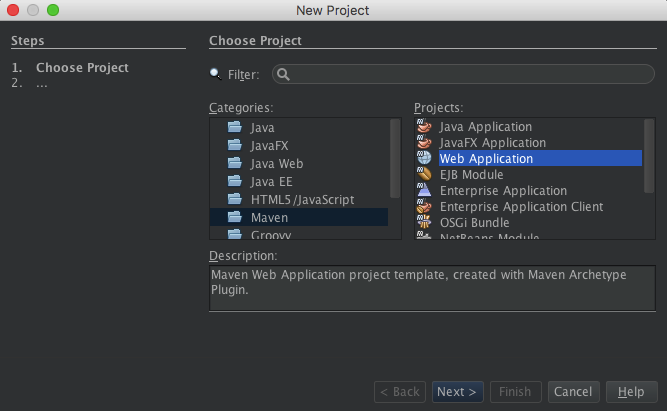


Ilustración 2 - Creación del proyecto con el arquetipo Maven Archetype Plugin

*Nota: este arquetipo ha sido utilizado anteriormente y también puede implementarse por medio de otros IDE´s como Eclipse o por medio de la terminal utilizando los comandos de Maven vistos al inicio del curso.*

1. Configurar el proyecto Maven proporcionando Nombre, ArtifactId, GroupId y la versión

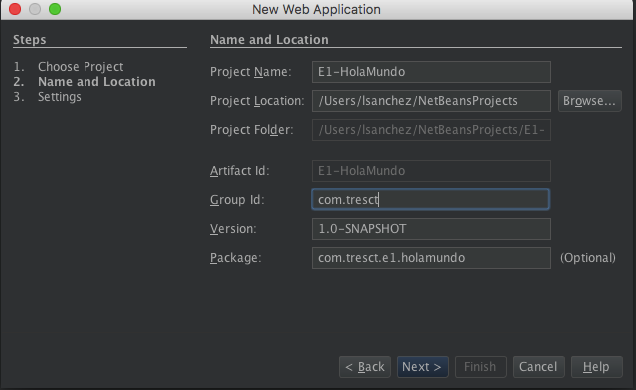


Ilustración 3 - Configuración de proyecto Maven

1. Elija el servidor de aplicaciones web de su preferencia:

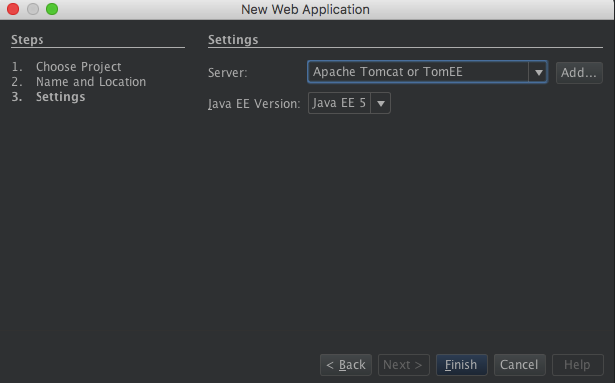


Ilustración 4 - Elección de servidor web

Para los ejercicios que se presentan en el curso utilizaré el servidor Jboss manejado desde la terminal del sistema. Los ejercicios pueden desplegarse en Jboss, GlasFish, Apache Tomcat, etc.

1. Revisar la arquitectura creada por la herramienta Maven:

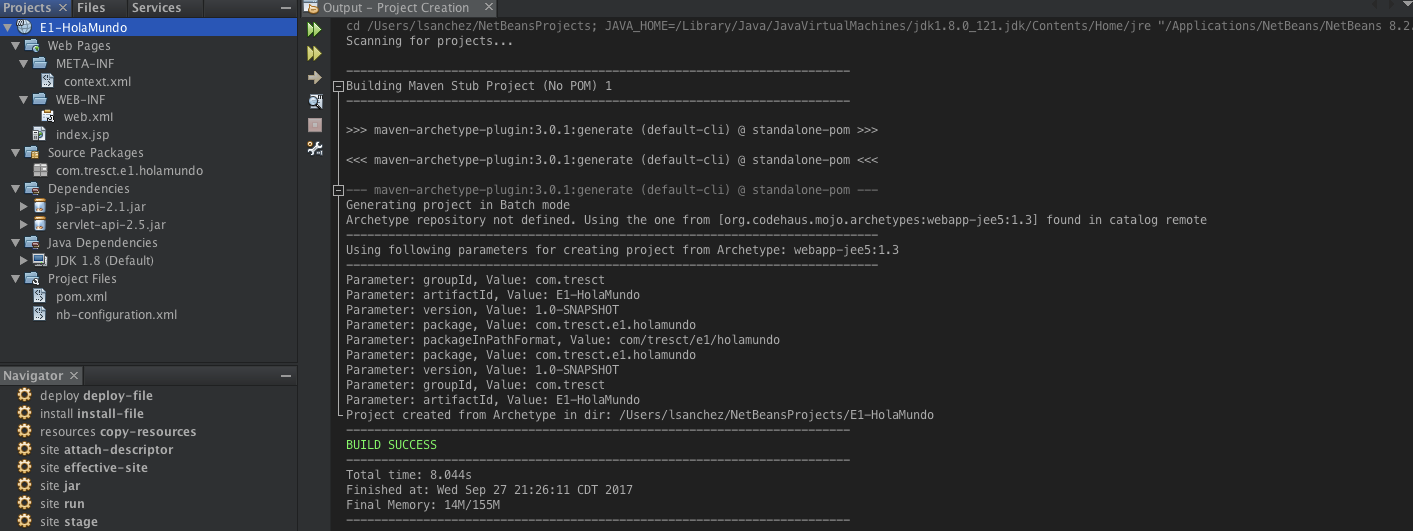


Ilustración 5 - Arquitectura Web creada por Maven

1. Abrir el archivo ***pom.xml*** y agregar las siguientes dependencias:

* ***jsf-impl.jar***
* ***jsf-api.jar***
* ***javax.servlet.jar***
* ***javax.servlet.jsp***

Obteniendo el siguiente resultado:



Ilustración 6 - Dependencias en pom.xml

Nota: Podemos agregar las dependencias de dos formas:

* Escribiendo el código en el pom.xml realizando búsquedas en MvnRepository:

<https://mvnrepository.com/artifact/com.sun.faces/jsf-api/2.2.14>

* Clic derecho sobre el código del pom.xml -> Insert code -> Dependency

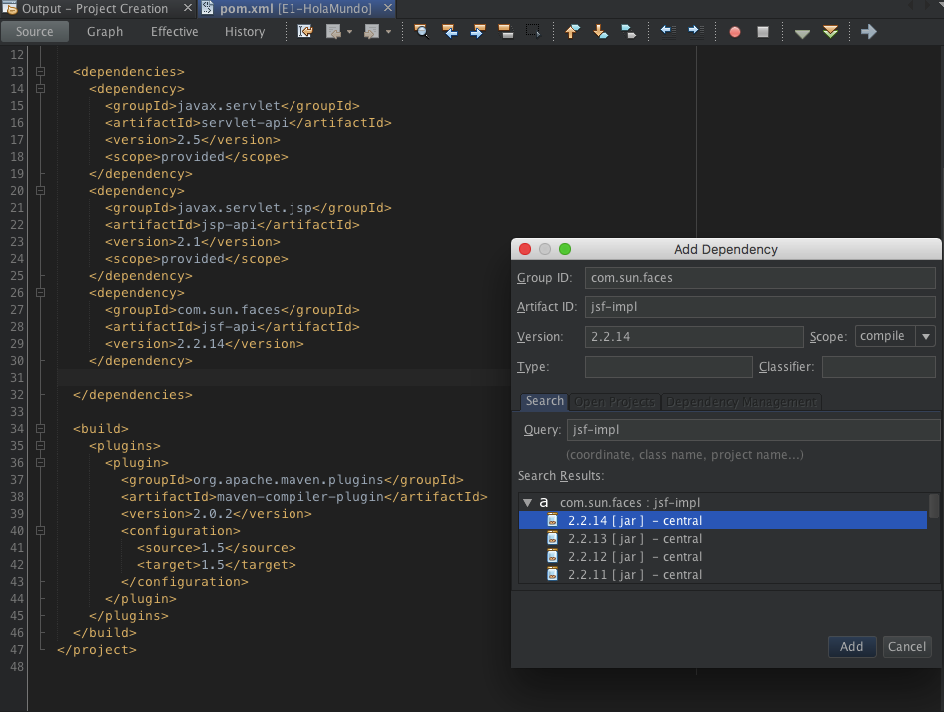


Ilustración 7 - Agregando dependencias desde el asistente

Dar clic en el botón **Clean and Build**  para descargar las bibliotecas necesarias especificadas en el pom.xml y observar los resultados como se muestra a continuación:

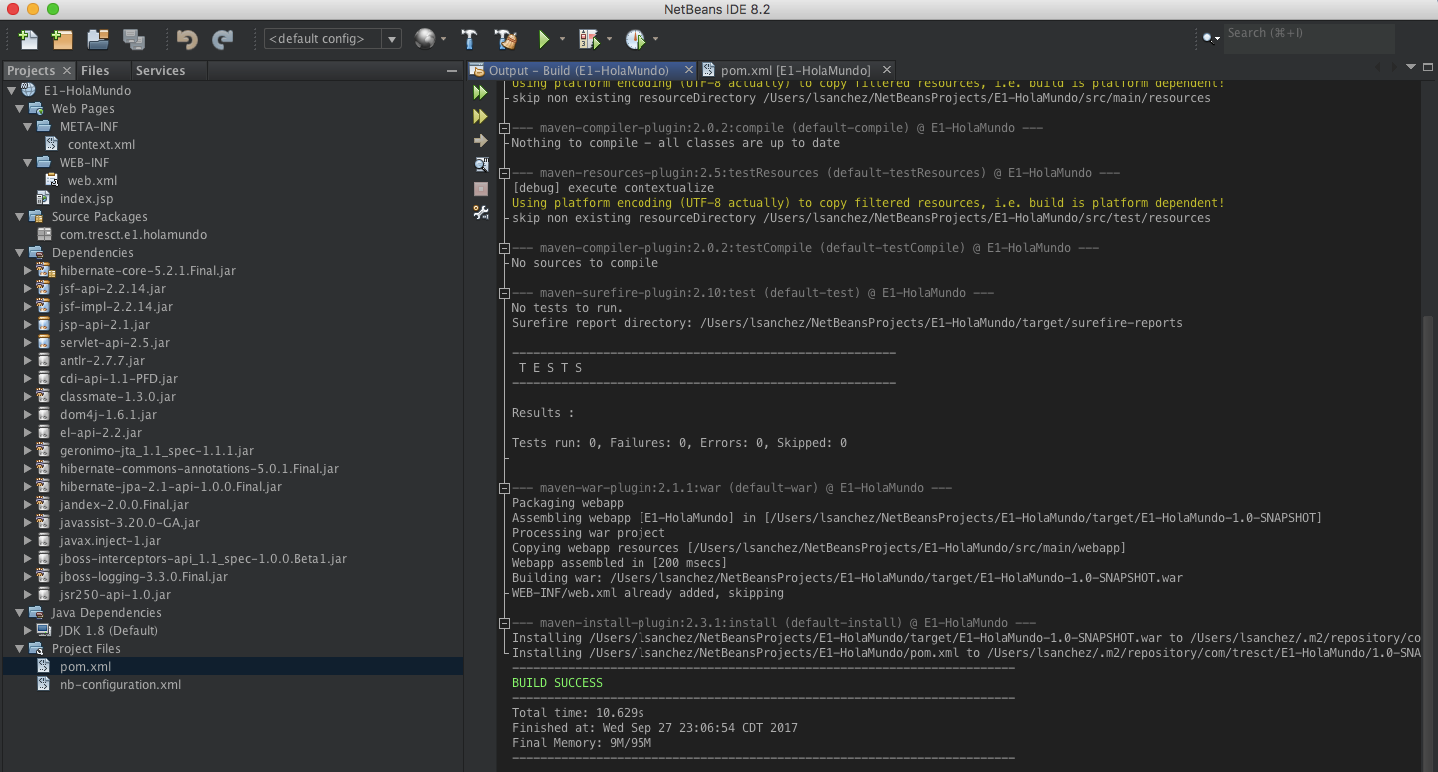


Ilustración 8 - Bibliotecas agregadas al proyecto

1. Remplazar el index.jsp por un index.xhtml

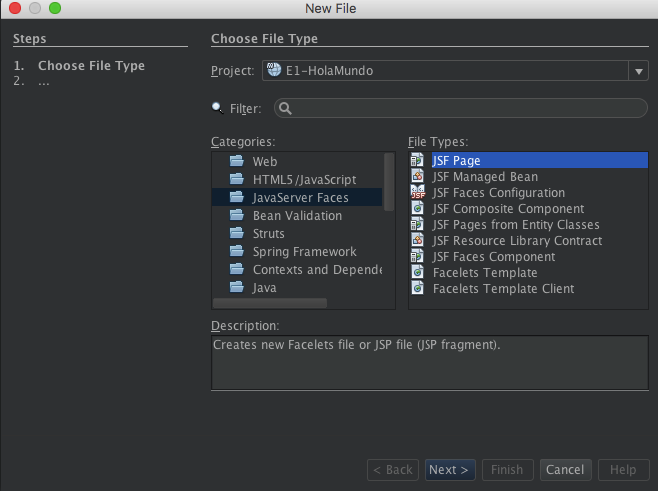


Ilustración 9 - Nuestro primer Facelet

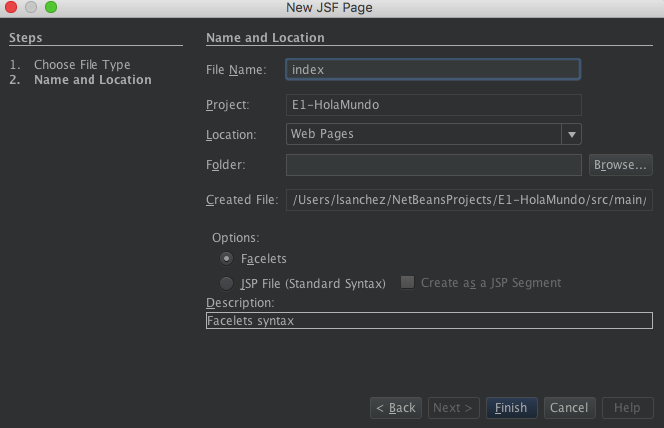
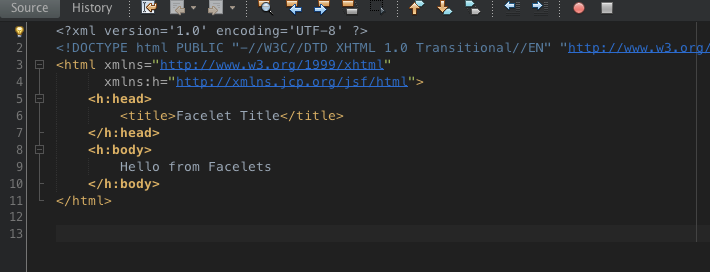


Ilustración 10 - Configuración del Facelet

1. Descripción de nuestro index.xhtml



1. Remplazar el código del index.html por el siguiente:

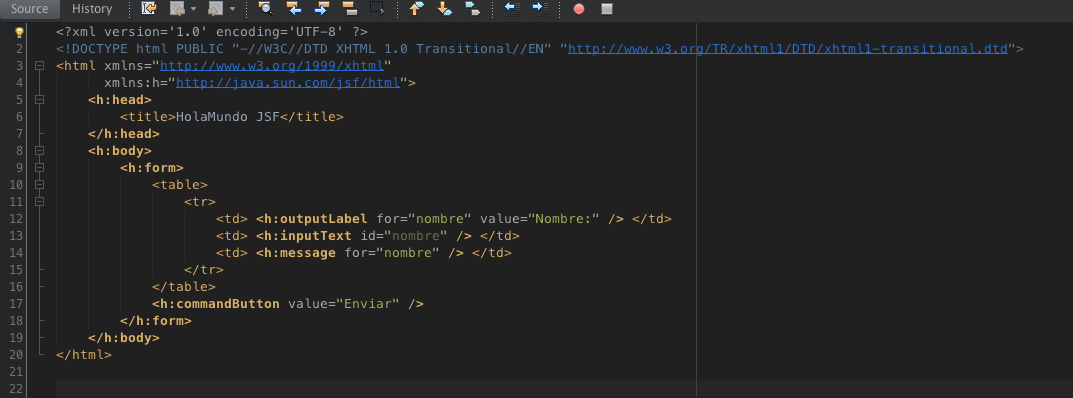


Ilustración 11 - Nuevo index.xhtml con un formulario JSF

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. <? xml version = '1.0'
2. encoding = 'UTF-8' ?> < !DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" > < html xmlns = "http://www.w3.org/1999/xhtml"
4. xmlns: h = "http://java.sun.com/jsf/html" > < h: head > < title > HolaMundo JSF < /title>  < /h:head> < h: body > < h: form > < table > < tr > < td > < h: outputLabel
5. for = "nombre"
6. value = "Nombre:" / > < /td> < td > < h: inputText id = "nombre" / > < /td> < td > < h: message
7. for = "nombre" / > < /td> < /tr>  < /table> < h: commandButton value = "Enviar" / > < /h:form> < /h:body> < /html>
8. Limpiar y construir el proyecto:

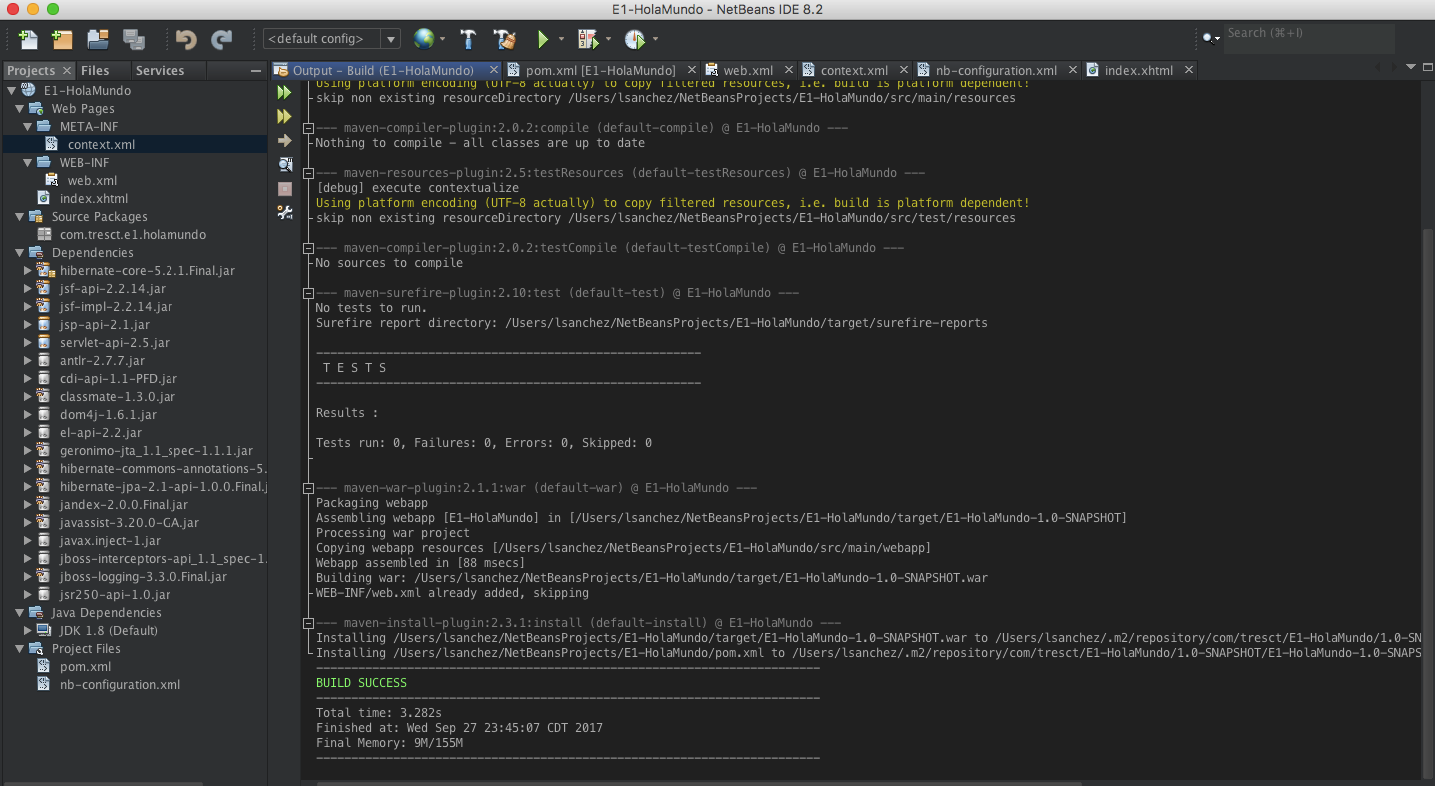


Ilustración 12 -Construcción del proyecto

1. Hacer el despliegue de la aplicación WEB.

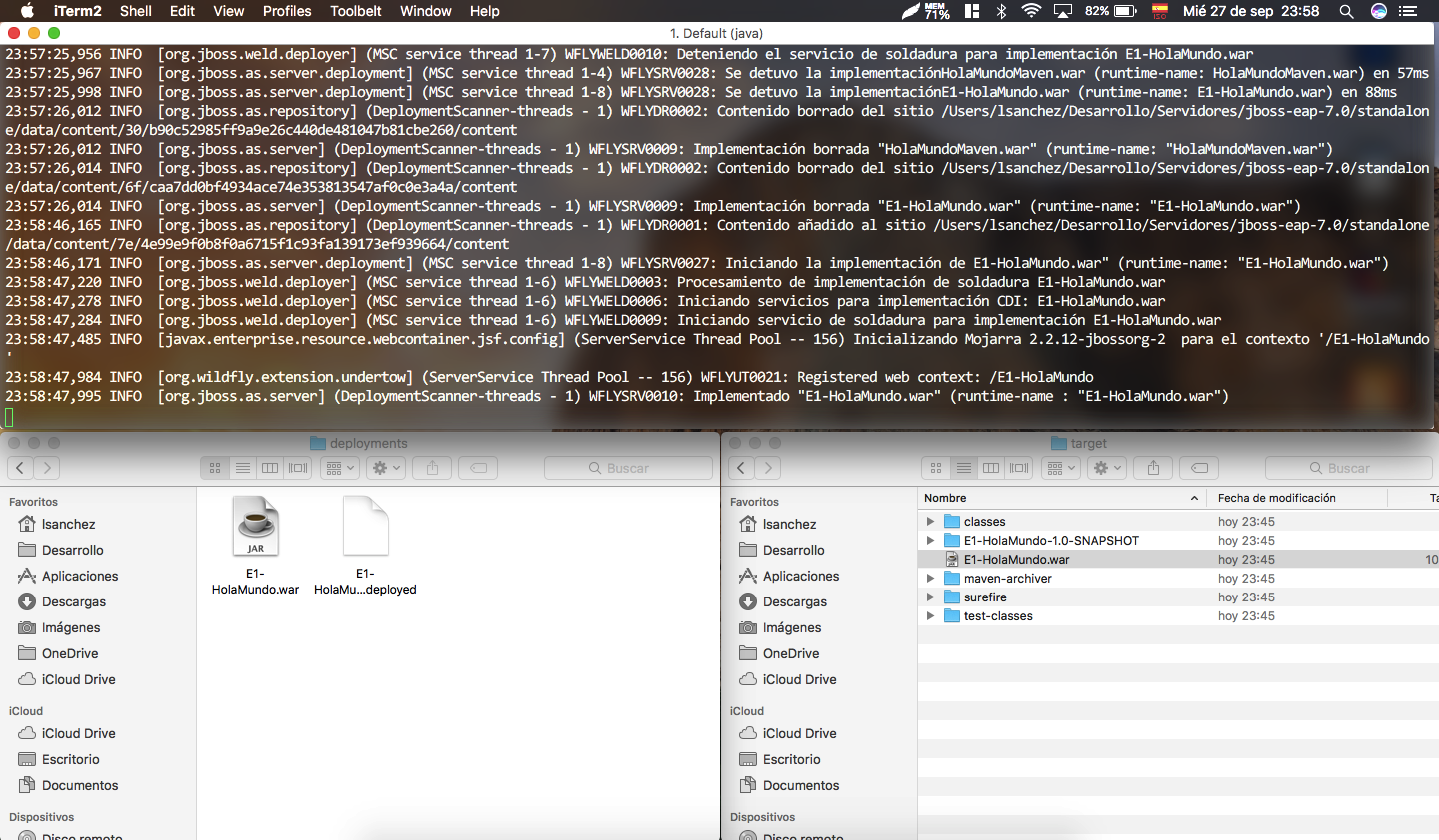


Ilustración 13 - Aplicación desplegada

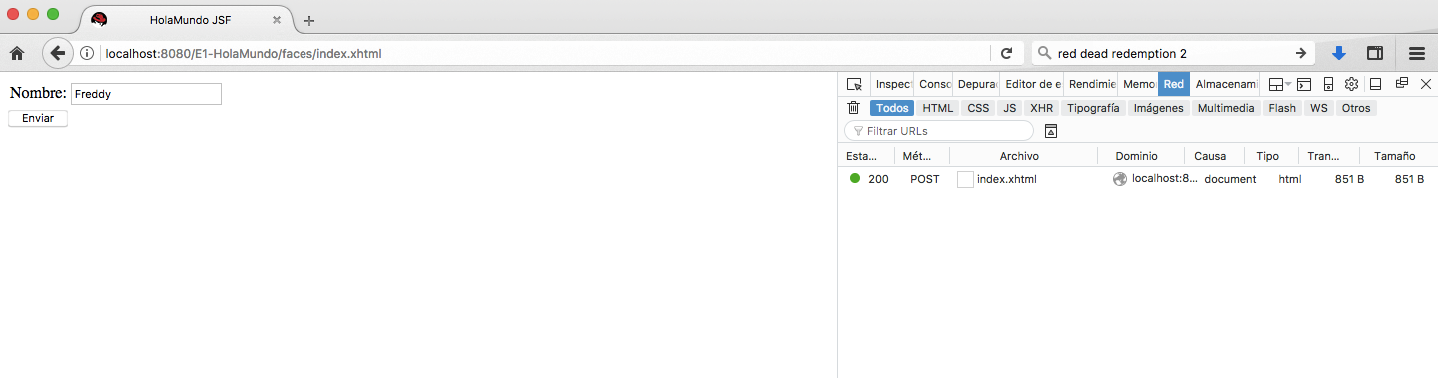


Ilustración 14 - Verificación del funcionamiento de nuestra aplicación en Mozilla Firefox

## Ejercicio #2 Model Managed Bean

* Objetivo: Introducir el concepto de este componente en nuestra aplicación y ubicar en que parte del MVC trabaja.
* Utilizar el concepto de anotaciones basado en las características de JSF 2.0 para declarar el Bean Administrado.
* Comprender la relación del Managed Bean con la Pagina JSF para desplegar valores a las vistas

Pasos a seguir

1. Agregar un Bean administrado al proyecto:

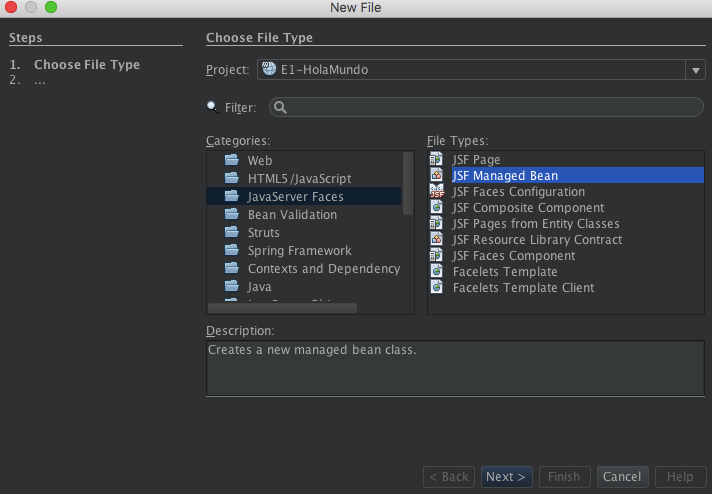


Ilustración 15 - Agregando un Bean Administrado por JSF

1. Configurarlo de la siguiente manera:

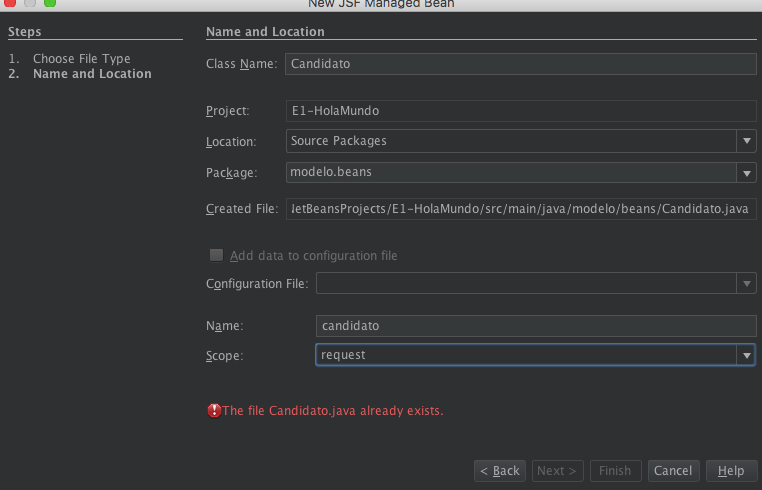
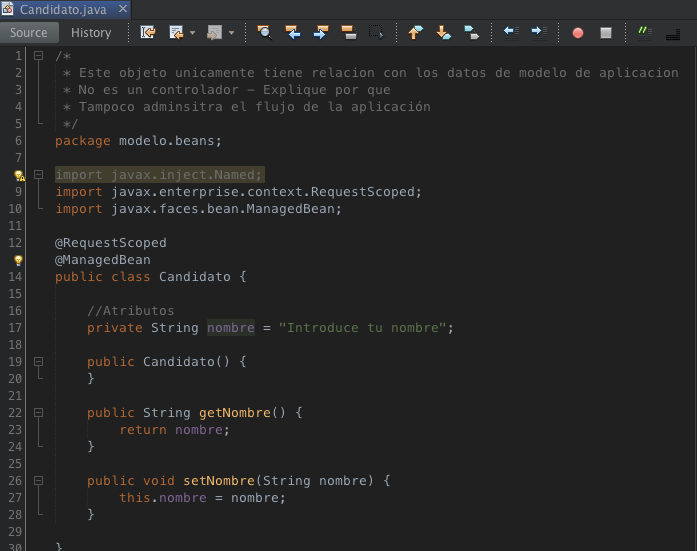


Ilustración 16 - Configuración del Bean Administrado

1. Escribir el siguiente código en el cuerpo del Bean Administrado



1. Enlazamos el Bean Administrado con la pagina JSF.

Ubicar el contenido de la página ***index.xhtml*** :

***<h:inputText id=”nombre” />***

***Sustituir con lo siguiente:***

***<h:inputText id=”nombre” value=”#{candidato.nombre}” />***

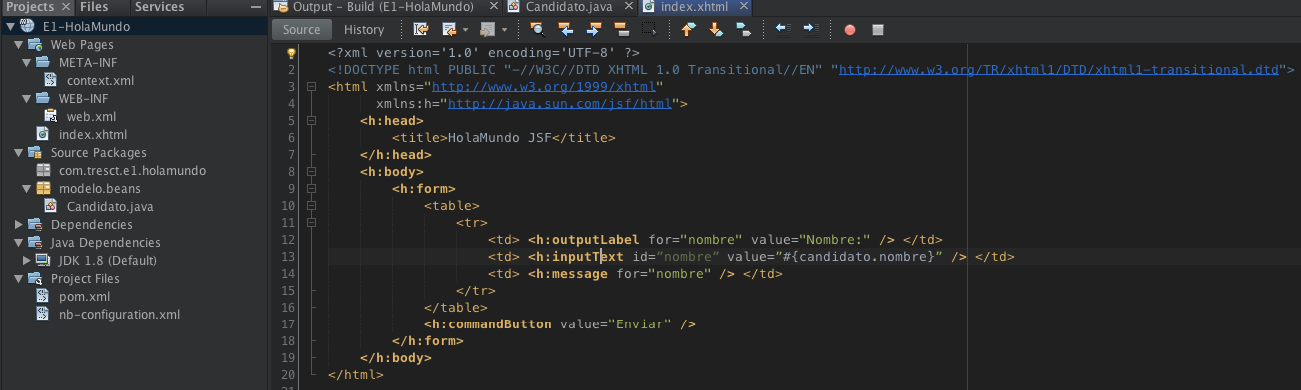
******

Ilustración 17 - Código del **index.xhtml**

1. Ejecutar la aplicación y observar el resultado:



Ilustración 18 - Enlazando valores del Bean a la vista por medio de EL...

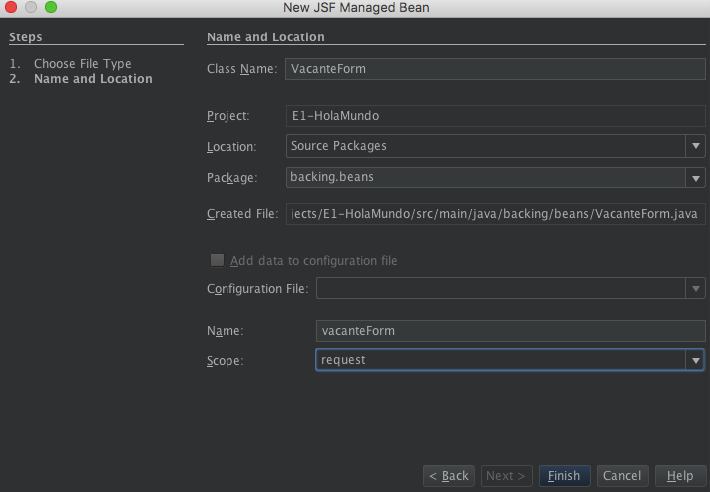
## Ejercicio # 3 Backing Beans

Objetivo:

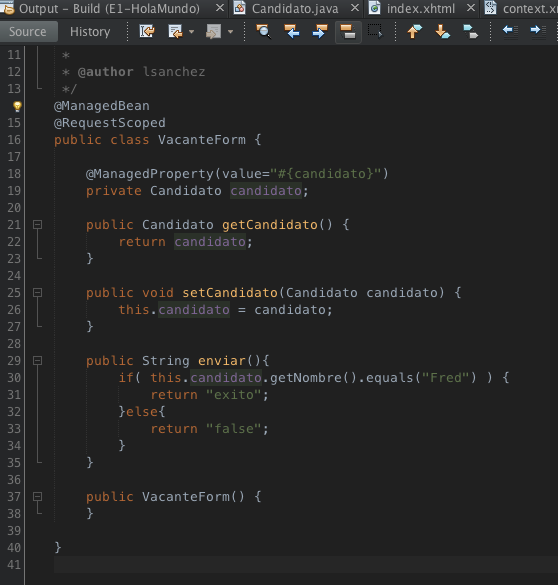
* Introducir el concepto de Backing Beans a nuestra aplicación.
* Utilizar el concepto de anotaciones para inyectar dependencias entre Managed Beans
* Adicionalmente agregar paginas JSF con las respectivas reglas de navegación con notación JSF 2.0
* Asociar funcionalidad al botón Enviar de nuestra página index.xhtml para ejecutar el flujo de la aplicación.

Pasos a seguir:

1. Agregar un JSF Managed Bean:

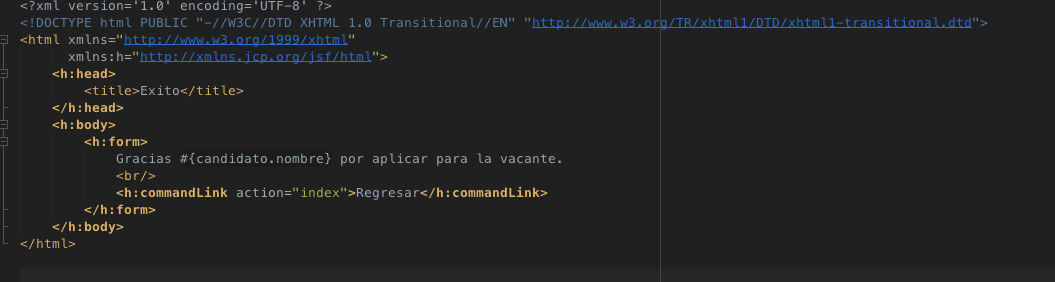


1. Escribir el siguiente código:

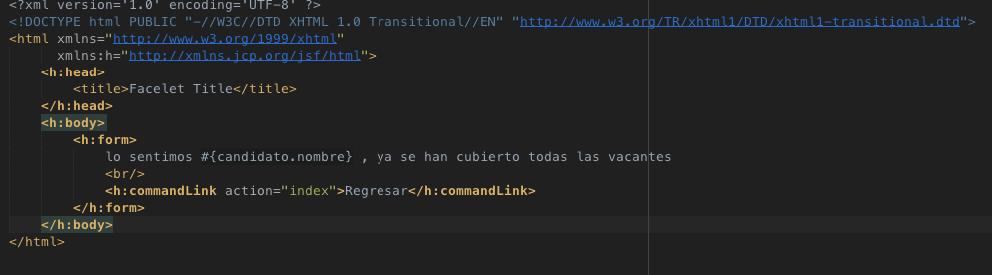


1. Agregar mensajes para el caso de éxito y fallo del flujo de la aplicación:

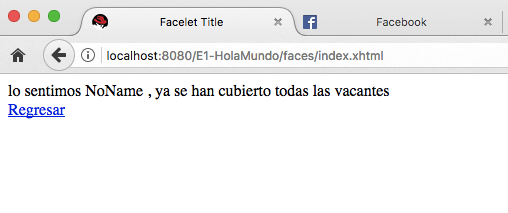
* Mensaje de éxito: Agregar un nuevo xhtml con el siguiente código:



1. Agregamos el xhtml para el fallo:



1. Finalmente se modifica el index de la siguiente manera:
2. Comente sus resultados:

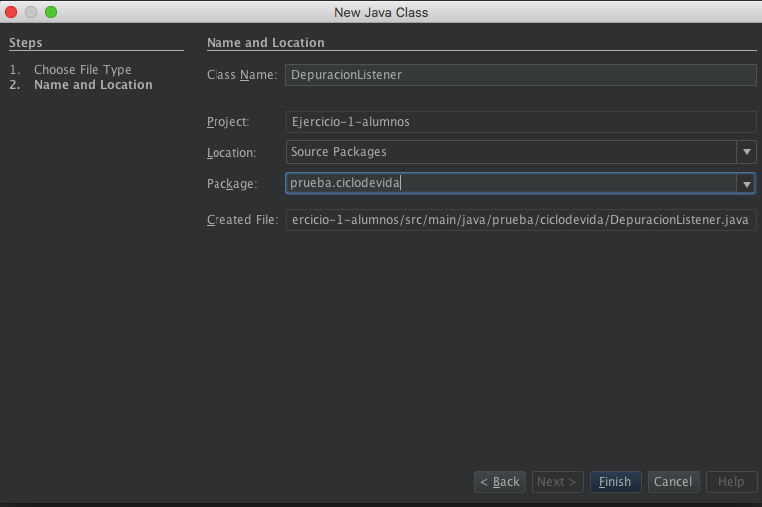


## Ejercicio # 4 Ciclo de vida de JSF

Objetivo: Conocer el funcionamiento del ciclo de vida de una aplicación JSF por medio de la visualización de los procesos que ocurren con los Managed Beans al ser instanciados y mantenidos por el Framework JSF.

Pasos a seguir:

1. Crear una copia del proyecto anterior o utilizar el mismo.
2. Crear una nueva clase Java de nombre **DepuracionListener** en el paquete: **prueba.ciclodevida** como se muestra a continuación:



1. Agregar los siguientes import a la clase:

***import javax.faces.event.PhaseEvent;***

***import javax.faces.event.PhaseId;***

***import org.apache.commons.logging.Log;***

***import org.apache.commons.logging.LogFactory;***

## Ejercicio # 5 Validaciones

Objetivo: conocer el funcionamiento de los validadores que nos proporciona jsf.

Pasos a seguir:

1. Creación de un proyecto Maven en NetBeans IDE (BolsaTrabajoV1) y agregar las siguientes dependencias:



Ilustración 19 – Dependencias

1. Crear una arquitectura similar a nuestro proyecto anterior

El proyecto deberá quedar de la siguiente forma:

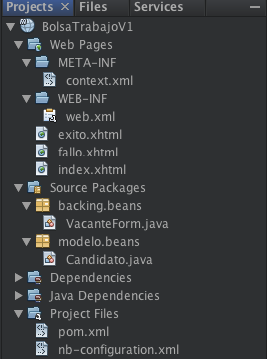


Ilustración 20 - Arquitectura del proyecto

1. Validar el campo “nombre” que se encuentra en el index.xhtml

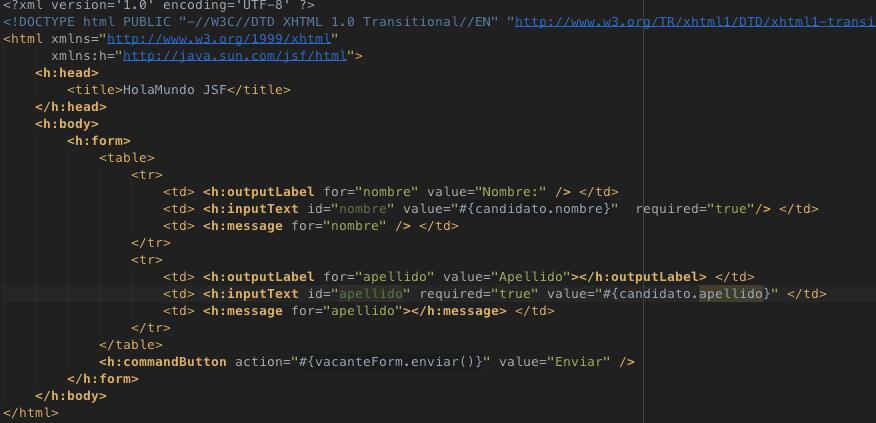


Ilustración 21 - Nuevo index.xhtml

1. Modificar el ManagedBean para Candidato. Agregando el atributo faltante (Apellido)

Se deberá agregar la propiedad y los metodos set y get.



Ilustración 22 - Nuevo Bean

1. Agregaremos un nuevo atributo: **sueldoDeseado** . El objetivo de este nuevo atributo es validar un valor numérico entre un rango de valores proporcionado por el desarrollador.

Para ello, modificaremos el index.xhtml como se muestra a continuación:

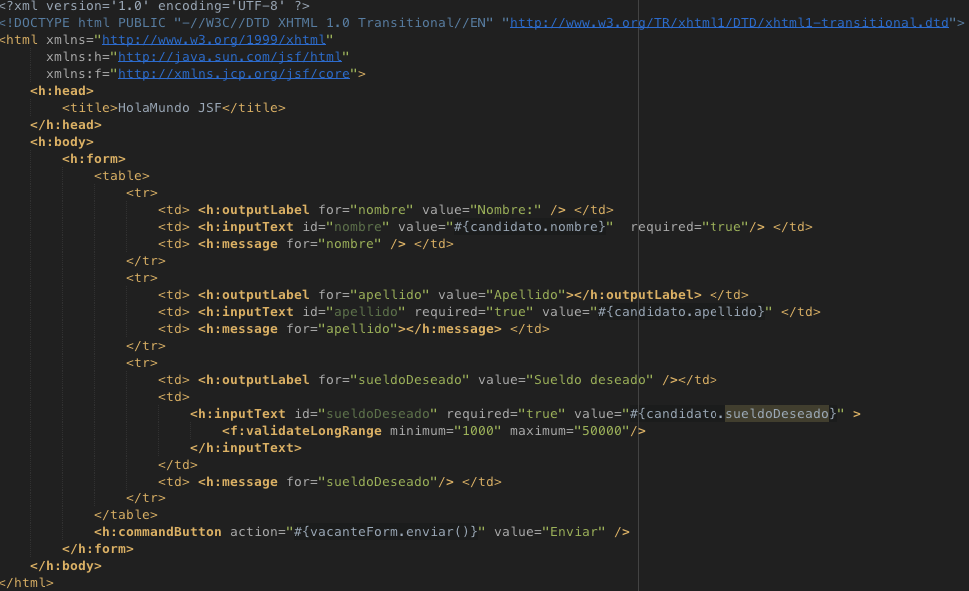


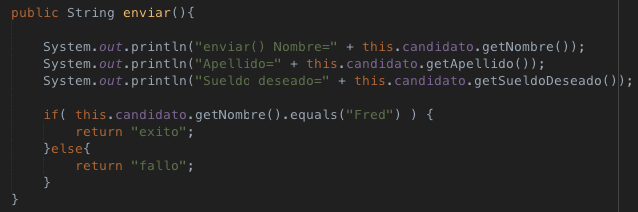
Ilustración 23 - Se agrega atributo sueldoDeseado

1. Modificamos el ManagedBean con el atributo nuevo.





1. Modificar el ManagedBean: VacanteForm con el siguiente Log



8- Ejecutar la aplicación y observar los resultados:

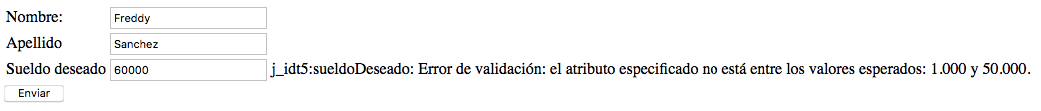
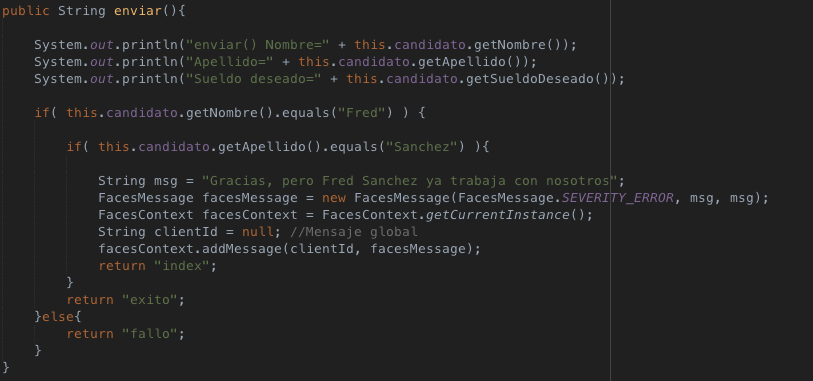
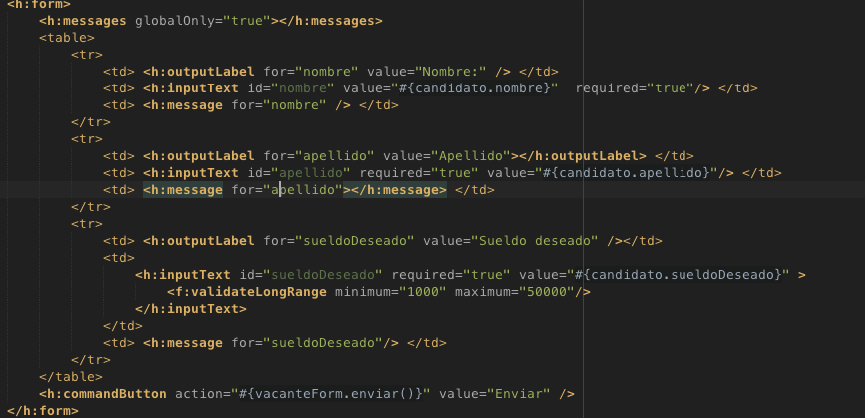


Ilustración 24 - Se realizan pruebas de validaciones

1. Realizar validaciones interdependientes: Para ello se modificara el Managed Bean VacanteForm.java y dentro del método enviar(), inmediatamente después de la llave del if que valida a un usuario especifico.



También se modificará el index.xhtml. agregando la siguiente línea inmediatamente después de la etiqueta <h:form> ….



1. Realizar las validaciones correspondientes:

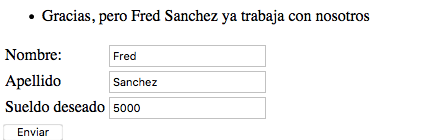


Ilustración 25 - Caso de prueba 1





Ilustración 26 - Caso de prueba 2

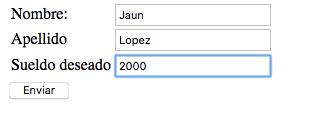




Ilustración 27 - Caso de validación 3

No olvide revisar el log en la consola del servidor.

## Ejercicio # 6 Convertidores

Objetivo: Conocer el funcionamiento de los convertidores en JSF.

Se implementará un tipo de conversión implícita ente tipos de datos de los formularios y los controladores (Vista - Controlador).

Pasos a seguir:

1. Copiar el proyecto anterior (BolsaTrabajoV1) a BolsaTrabajoV2 de la siguiente manera:

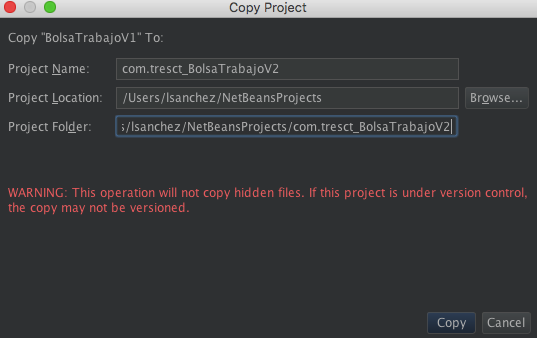


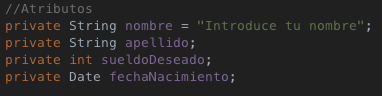
Ilustración 28 - Usar el asistente de copiado

1. En la clase “Candidato.java” cambiar el tipo de dato del atributo “sueldoDeseado” de String a int. Quedando de la siguiente forma:



Ilustración 29 - Nuevos tipos de datos para atributo y metodos

1. Agregamos el atributo de fecha de nacimiento:



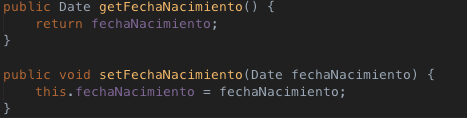


Ilustración 30 - Fecha (import java.util.Date)

1. Agregar un nuevo campo sobre el index.xhtml para el manejo de fecha.

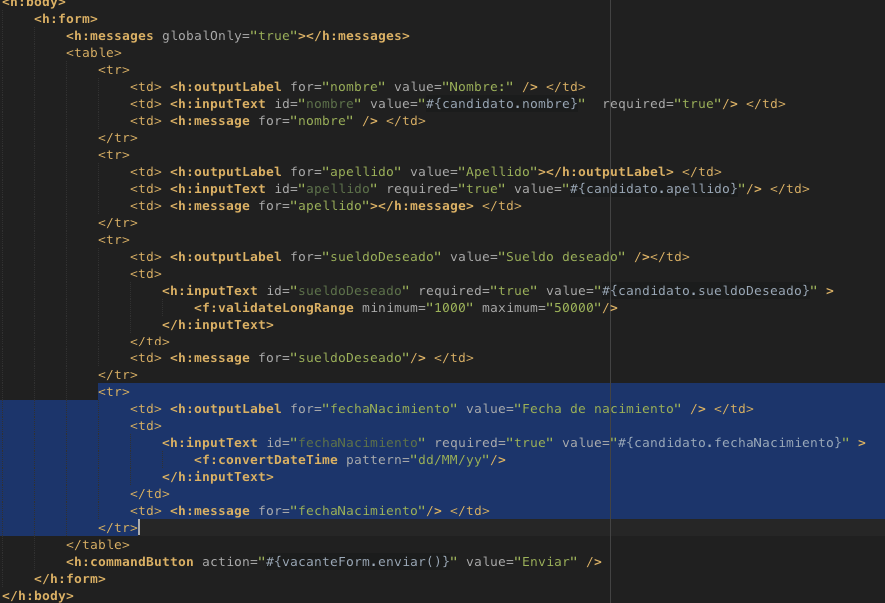


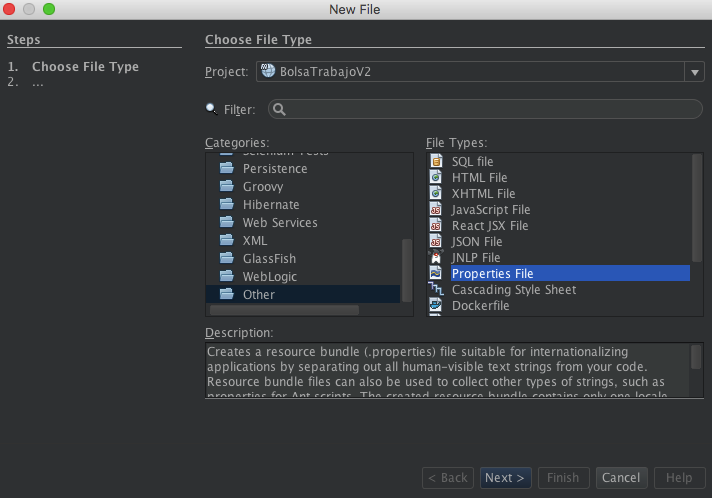
Ilustración 31 - Probar la aplicación y revisar los casos de exito y fallo

## Ejercicio # 6 Mensajes

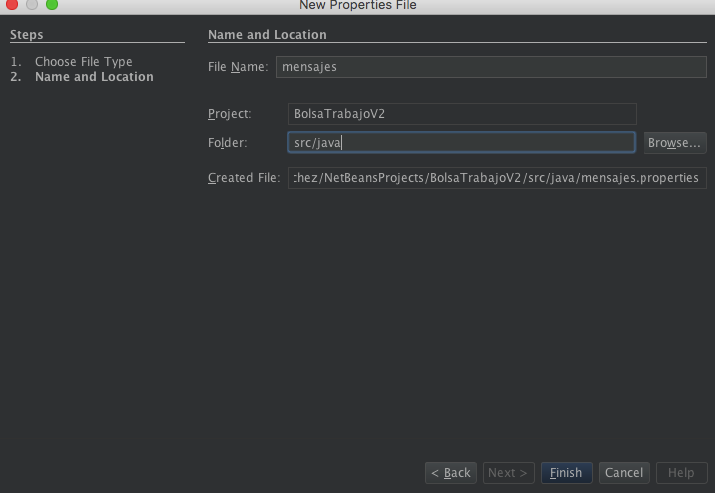
Objetivo: Poner en practica el concpeto de Mensajes en JSF. Por medio de estos, lograremos cambiar cada una de las etiquetas utilizadas del tipo : h:outputLabel por etiquetas que provienen de un archivo de propiedades.

Pasos a seguir:

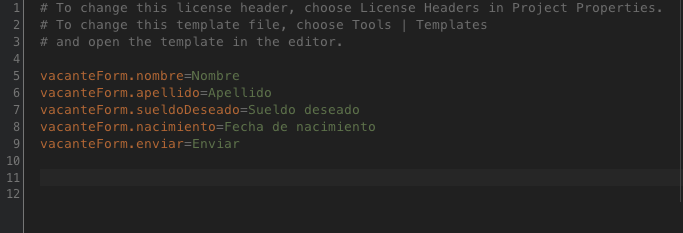
1. Realizaremos los cambios sobre el proyecto V2 de BolsaTrabajo, por lo tanto, crearemos un archivo de propiedades de la siguiente manera:



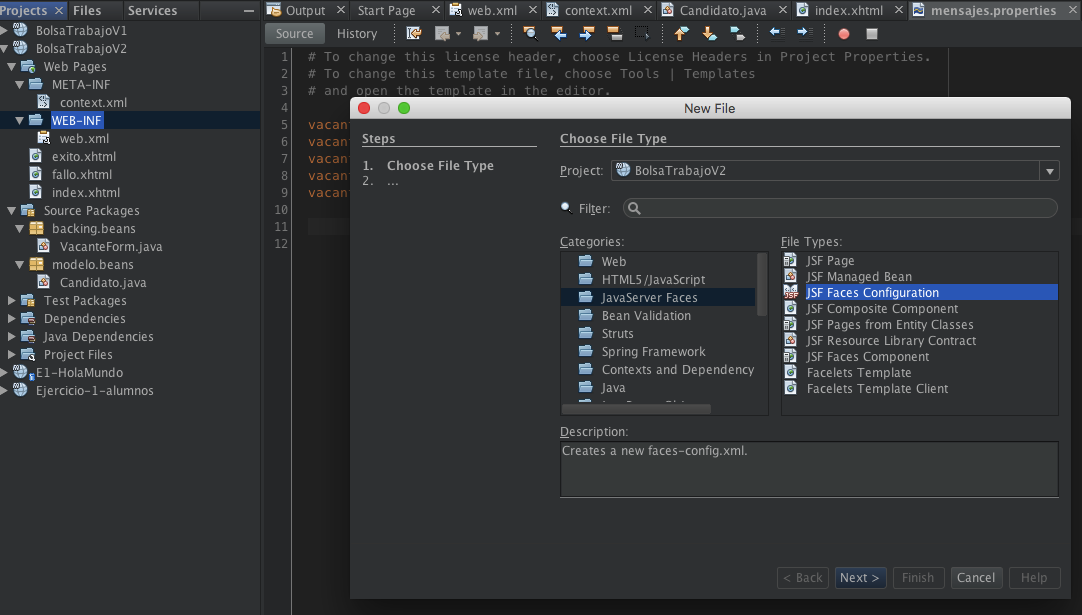
Se configura de la siguiente manera:



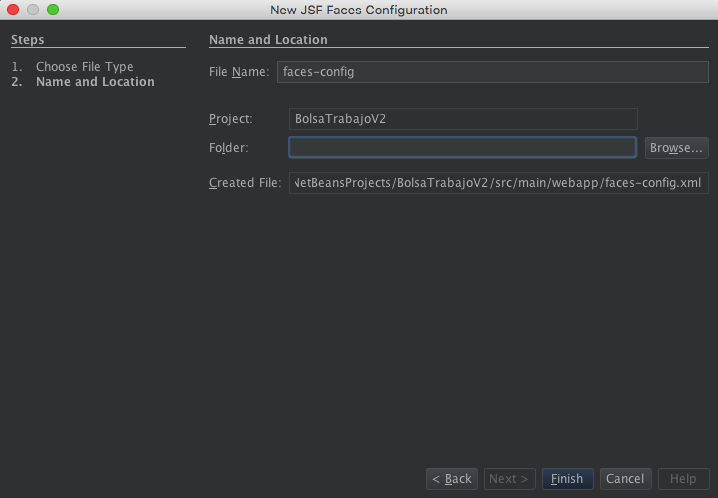
1. Escribir las siguientes propiedades:



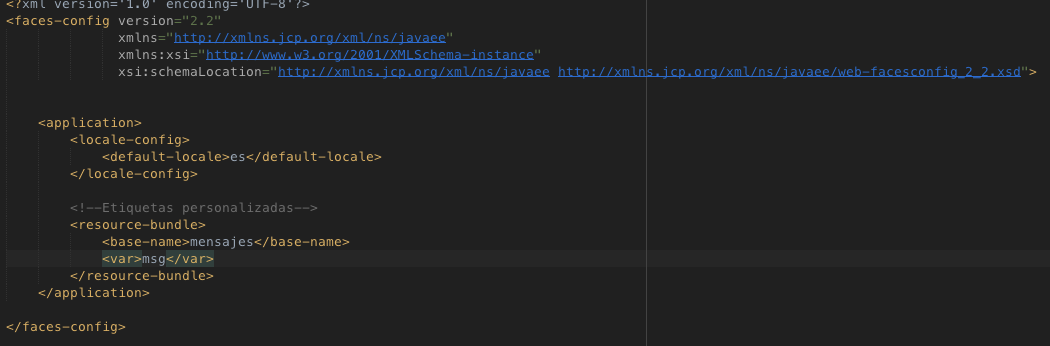
1. Creación del archivo ***faces-config.xml:***



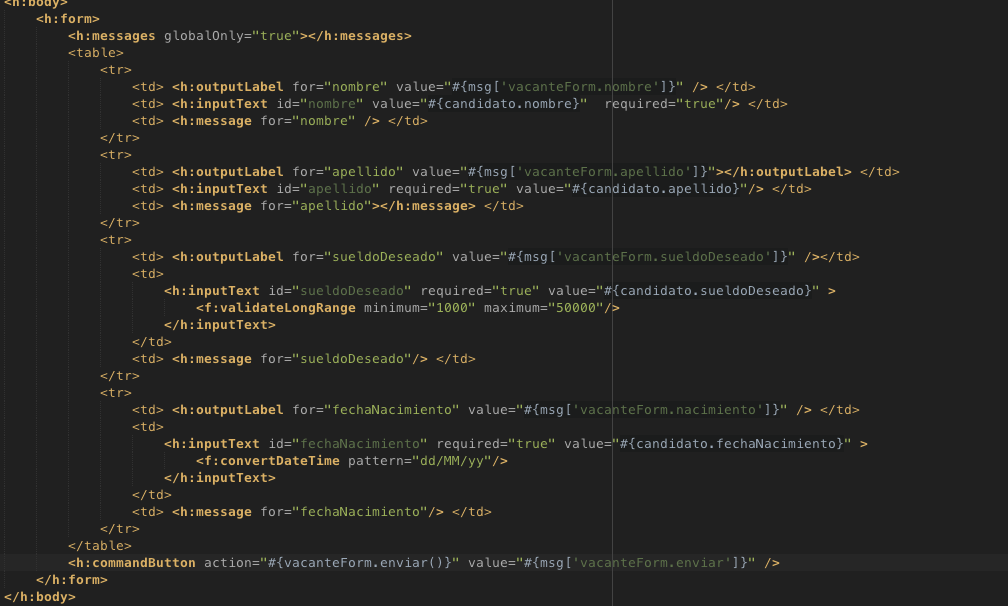
Se utilizará la siguiente configuración:



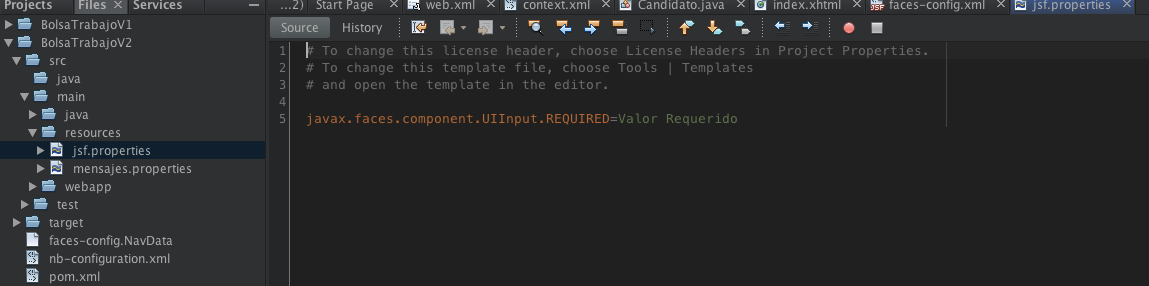
Dentro del archivo faces-config.xml escribir el siguiente código:



1. Remplazar las etiquetas en index.xhtml de la siguiente manera:



1. Cambiar los mensajes estándar de error de JSF por medio de otro archivo:



Atención: Debe revisar la arquitectura que se propone en el ejercicio.

Debido a que utilizamos una herramienta para la construcción de proyectos, la ruta de los properties debe incluirse así: src/main/resources.

## Ejercicio # 7 Internacionalización en JSF

Objetivo: El objetivo de este ejercicio es poner en práctica el concepto de Internacionalización en JSF. Esto lo lograremos al crear un archivo de propiedades con distinto idioma, basado en la especificación I18n de Java.

Pasos a seguir:

1. Realizaremos una copia de nuestro proyecto y cambiaremos el nombre a: BolsaTrabajoV3

Dando como resultado una arquitectura similar a la que se muestra a continuación:

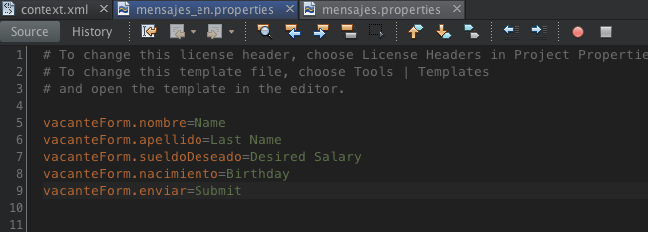


No olvide hacer un rename del artifact id y el contexto de la aplicación deberá llamarse igual al proyecto por comodidad.

1. Creación del archivo de propiedades: mensajes\_en.properties



1. Escribimos las siguientes propiedades en el archivo mensajes\_en.properties



1. Cambiar el idioma en el archivo fces-config.xml

Este paso es sumamente sencillo ya que únicamente se debe modificar la siguiente línea del archivo de configuración: (es) -> (en)



1. Ejecutar la aplicación y observar los cambios:

