



## Ejercicio 10

Aplicación Web con JSF/PrimeFaces,  
EJB y JPA

## Objetivo del Ejercicio

- El objetivo del ejercicio crear una aplicación Web que Listar las Personas utilizando la arquitectura de JSF/AJAX/PrimeFaces, EJB y JPA.



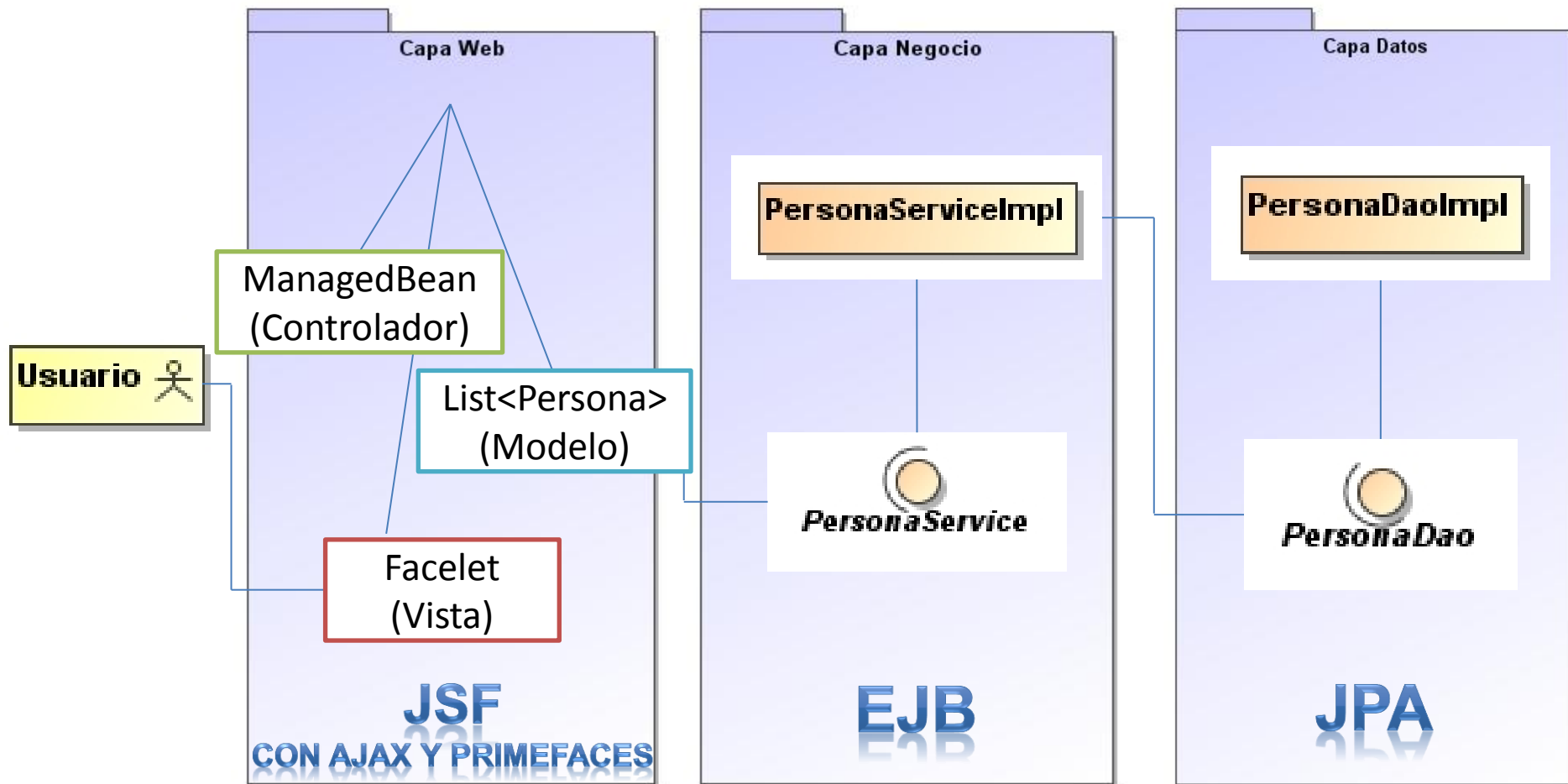
The screenshot shows a web browser window titled "Listar Personas". The address bar displays "http://localhost:8080/sga-jsf/faces/index.xhtml". The main content area features a table titled "Lista de Personas" with four columns: "Nombre", "Apellido Paterno", "eMail", and "Opciones". The table contains three rows of data. Below the table is a "Regresar" button.

Lista de Personas			
Nombre	Apellido Paterno	eMail	Opciones
Juan	Perez	juanperez@gmail.com4	
Oscar	Gomez	ogomez@mail.com	
Angelica	Lara	alara@mail.com	

[Regresar](#)

# Diagrama de Clases

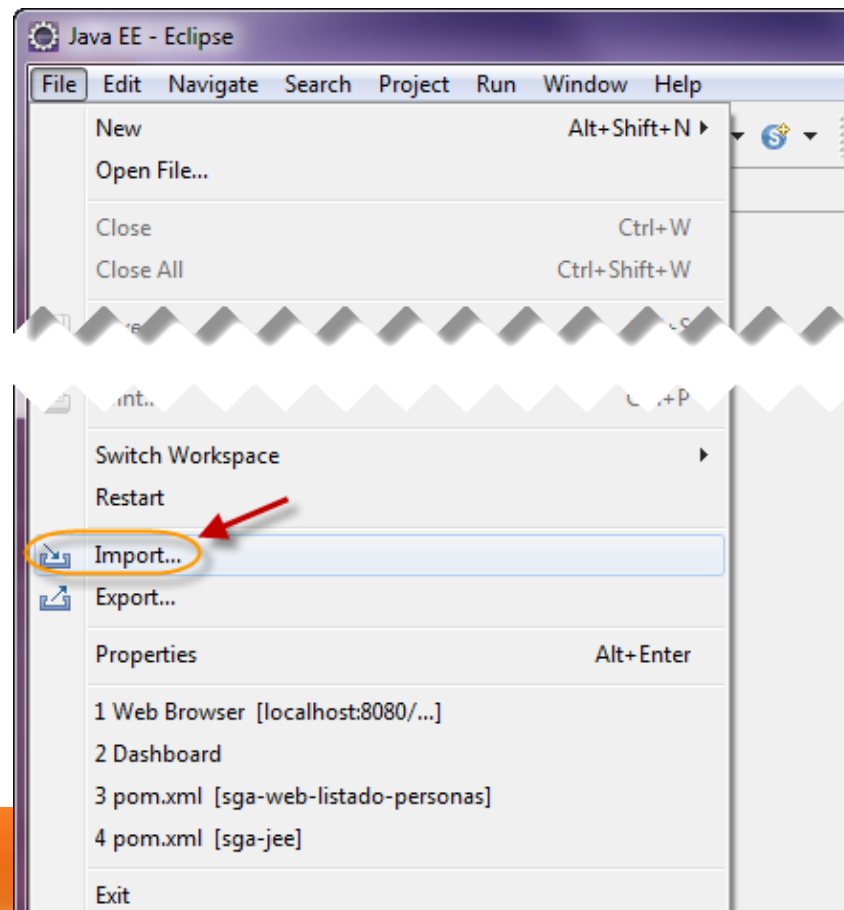
- Este es el Diagrama de Clases del Ejercicio, donde se pueden observar la Arquitectura de 3 capas de nuestro Sistema.



## Paso 1. Cargar el proyecto sga-jee

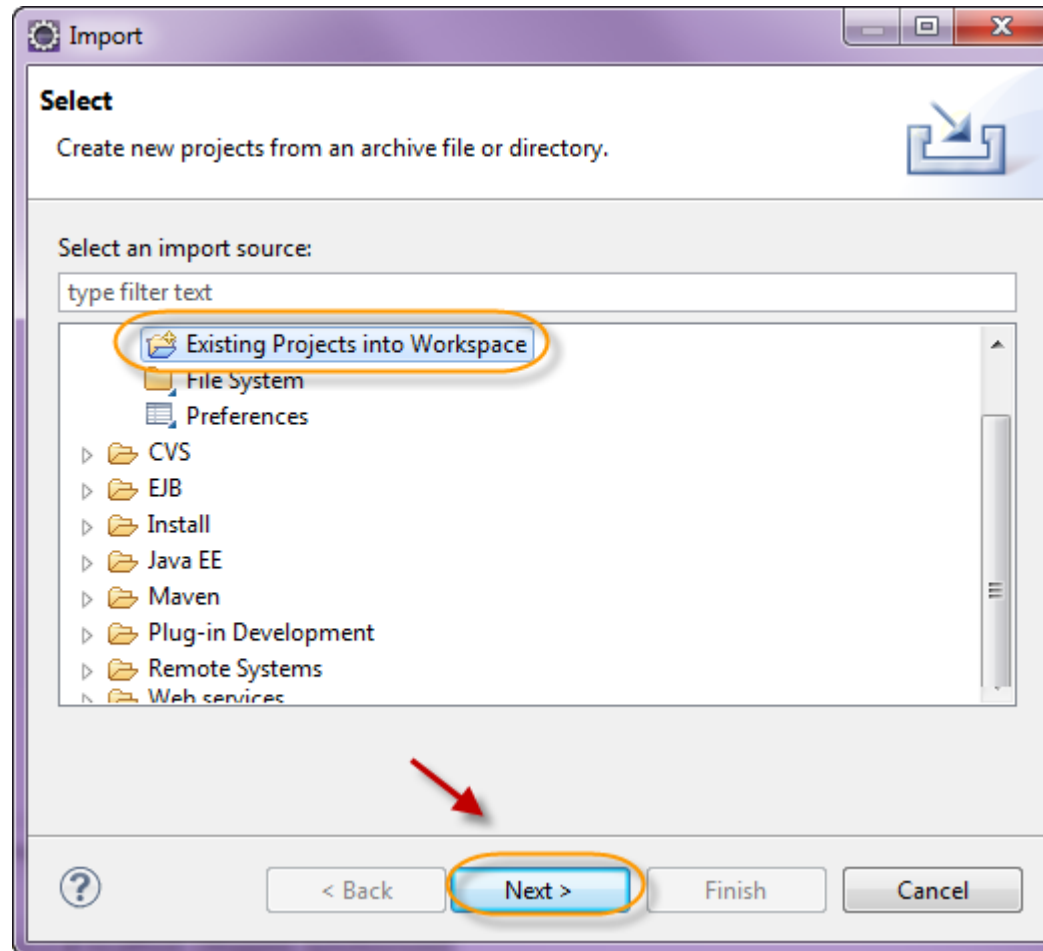
Como primer paso, cargamos en el workspace de Eclipse el proyecto sga-jee que se encuentra en la lección 4 de los ejercicios del curso.

**Nota: Si ya se tiene cargado este proyecto se puede omitir el paso 1 y el paso 2 siguientes:**



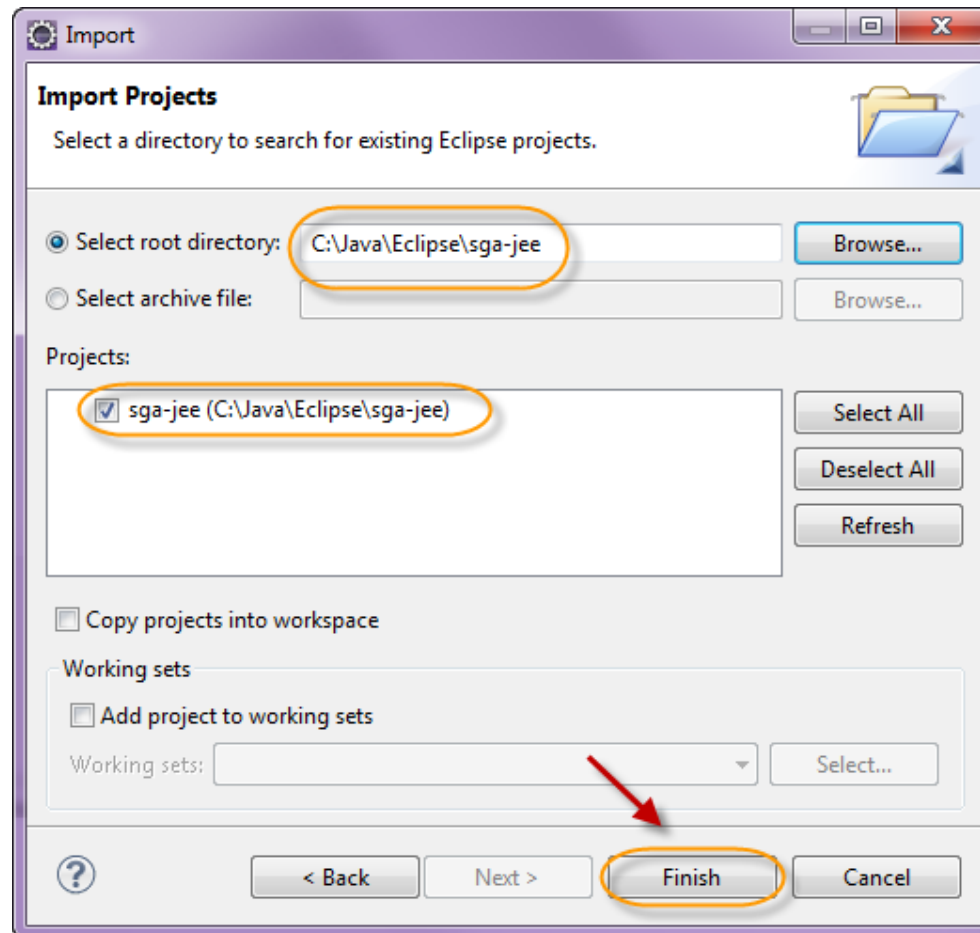
## Paso 1. Cargar el proyecto sga-jee (cont)

Cargamos el proyecto sga-jee de los ejercicios de la lección 4.



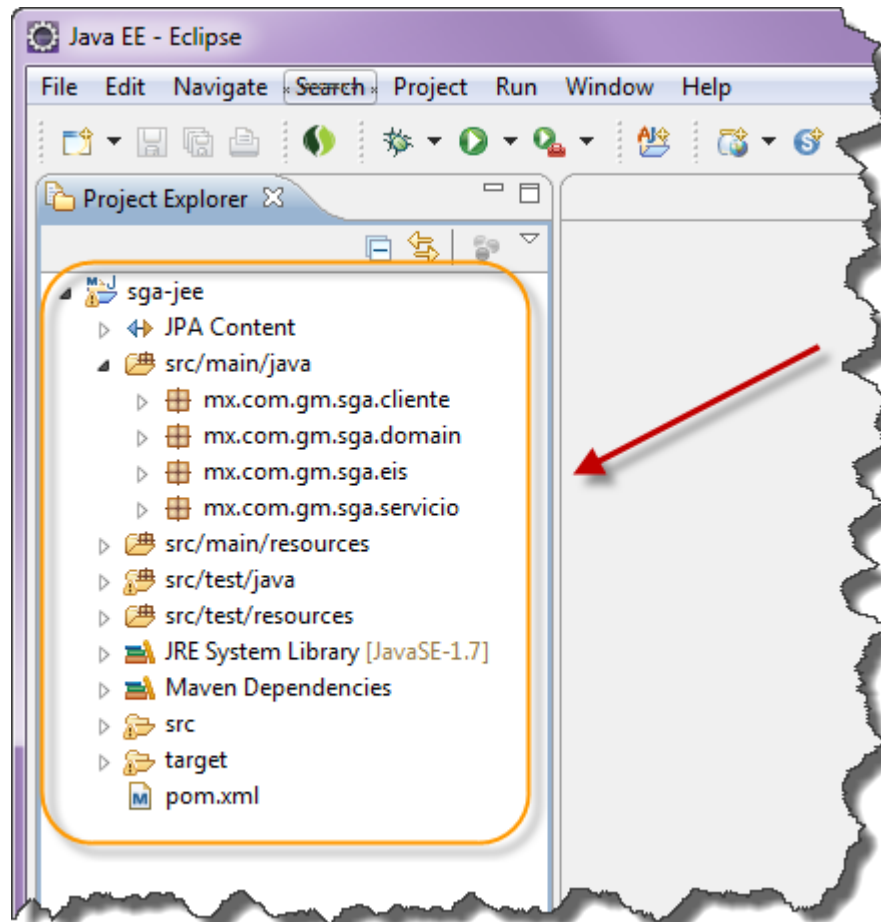
## Paso 1. Cargar el proyecto sga-jee (cont)

Proporcionamos la ruta donde hayamos descomprimido el archivo (si ya existía en nuestro workspace, lo eliminamos previamente y lo volvemos a cargar):



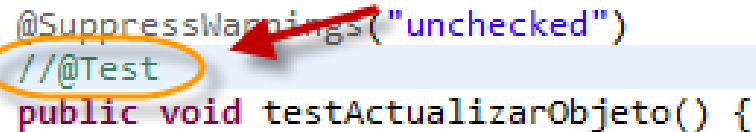
## Paso 1. Cargar el proyecto sga-jee (cont)

Observamos la estructura del proyecto y que no contenga errores:



## Paso 1. Cargar el proyecto sga-jee (cont)

En caso de estar trabajando con el proyecto de la lección 3, se deberán desactivar TODAS las pruebas unitarias:



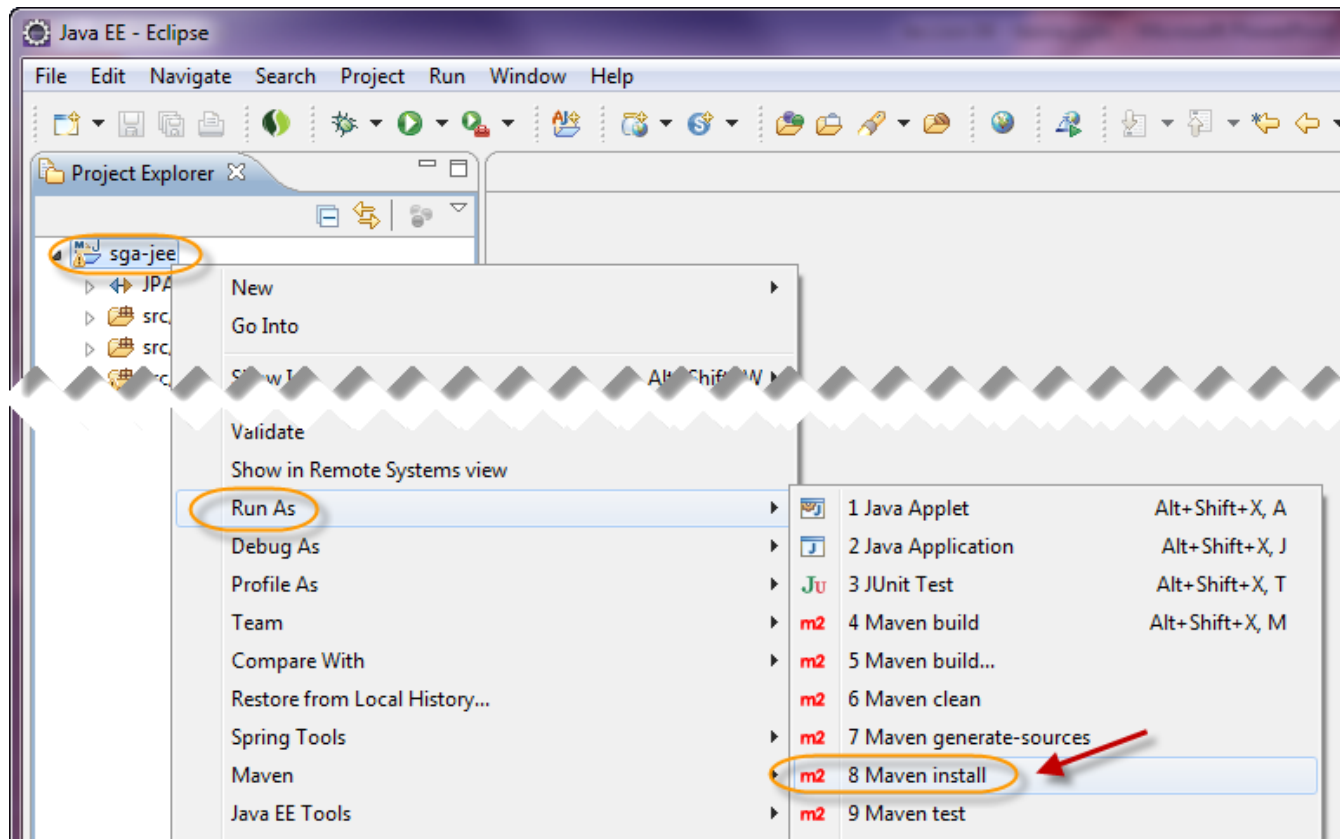
```
@SuppressWarnings("unchecked")  
//@Test  
public void testActualizarObjeto() {
```

The image shows a code snippet with three lines. The first line is `@SuppressWarnings("unchecked")`. The second line is `//@Test`, where the `@Test` part is circled in orange. A red arrow points from the `@Test` part of the second line to the `@SuppressWarnings` annotation on the first line. The third line is `public void testActualizarObjeto() {`.



## Paso 2. Instalar el proyecto sga-jee

Instalaremos el proyecto en el repositorio local de Maven. Esto permitirá posteriormente agregar el proyecto sga-jee como una dependencia de Maven a otros proyectos. **Nota: Si ya se tiene instalado el proyecto del ejercicio anterior, se puede omitir este paso por completo:**



## Paso 2. Instalar el proyecto sga-jee (cont)

Deberemos obtener un resultado similar al siguiente:

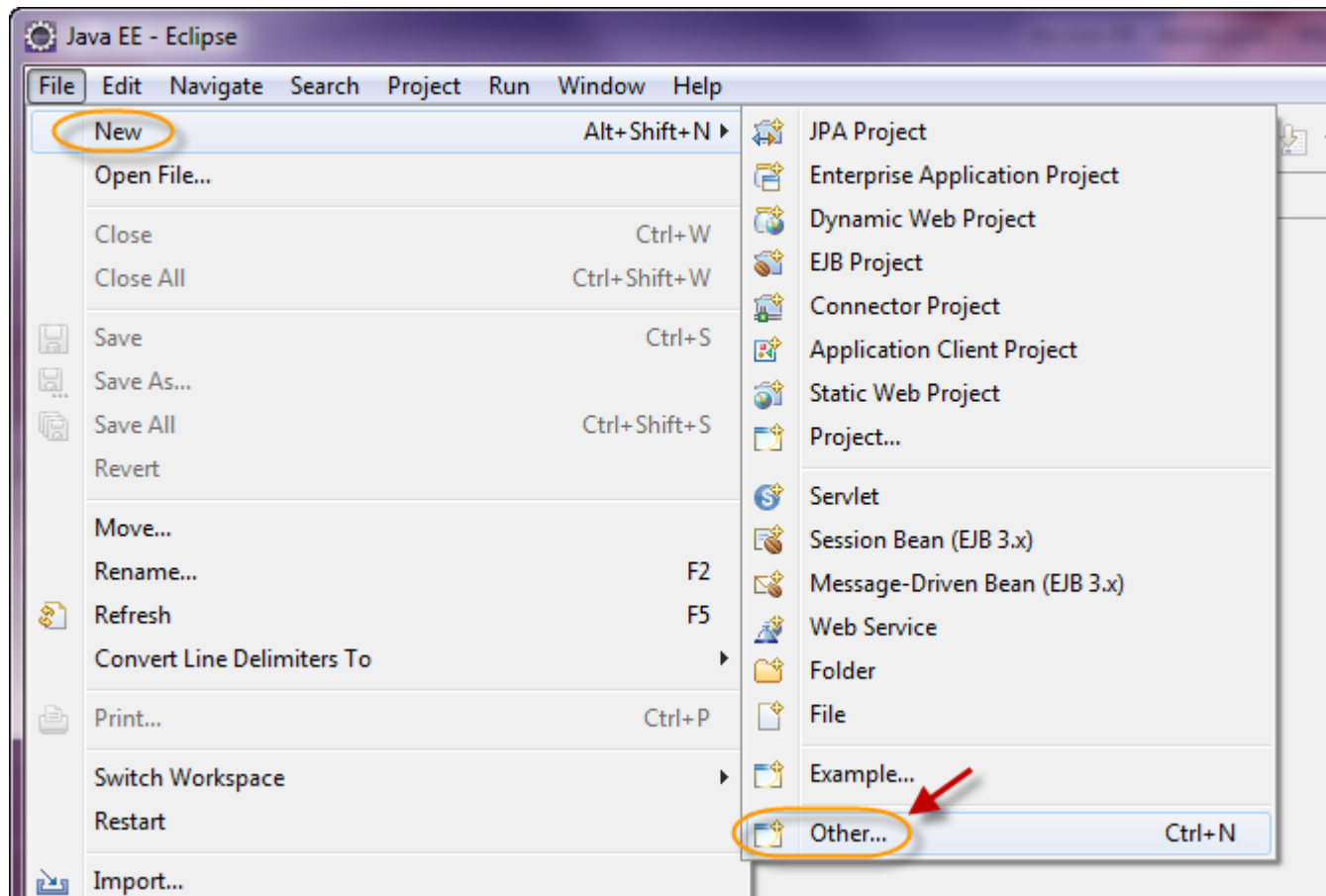
```
[INFO] --- maven-install-plugin:2.3.1:install (default-install) @ sga-jee ---  
[INFO] Installing C:\Java\Eclipse\sga-jee\target\sga-jee-1.0.jar to \\vmware-host\Shared Folders\.m2\repository\mx\com\gm\sga\sga-jee-1.0.jar  
[INFO] Installing C:\Java\Eclipse\sga-jee\pom.xml to \\vmware-host\Shared Folders\.m2\repository\mx\com\gm\sga\sga-jee\1.0\sga-jee-1.0.pom  
[INFO] Installing C:\Java\Eclipse\sga-jee\target\sga-jee-1.0-jar-with-dependencies.jar to \\vmware-host\Shared Folders\.m2\repository\mx\com\gm\sga\sga-jee-1.0-jar-with-dependencies.jar  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 35.033s
```

Con el proyecto ya instalado en el repositorio local, podemos utilizarlo como una dependencia de Maven para los proyectos que necesiten de las clases de Negocio (EJB) que hemos creado en este proyecto.

A continuación crearemos el proyecto Web, el cual utilizará esta dependencia.

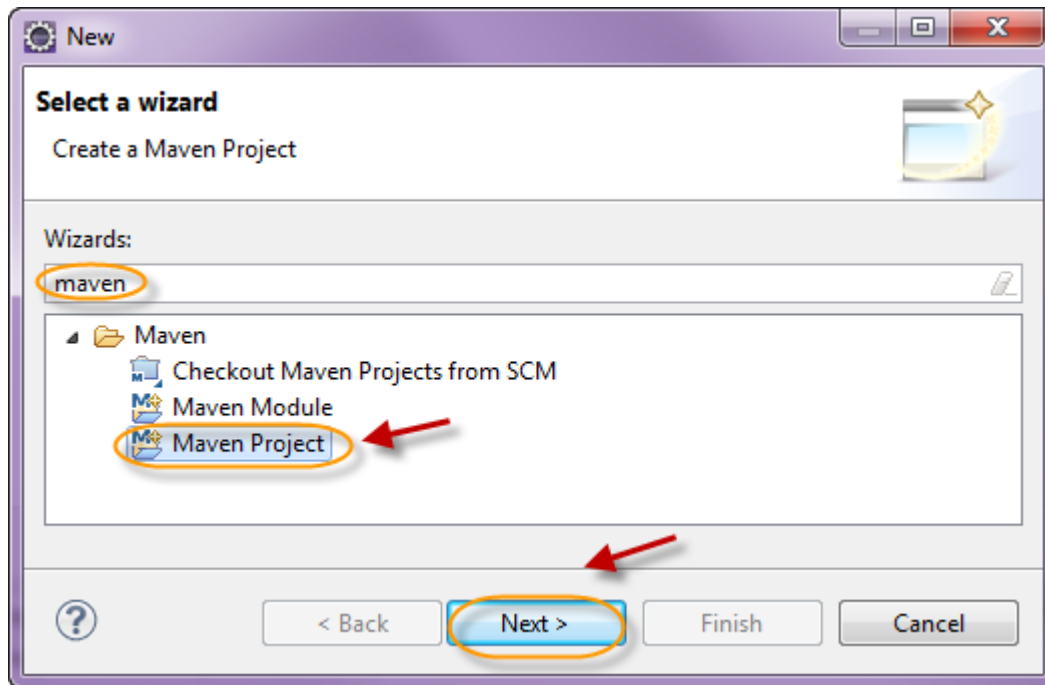
## Paso 3. Creación del proyecto Java Web

Creamos el proyecto sga-web-jsf:



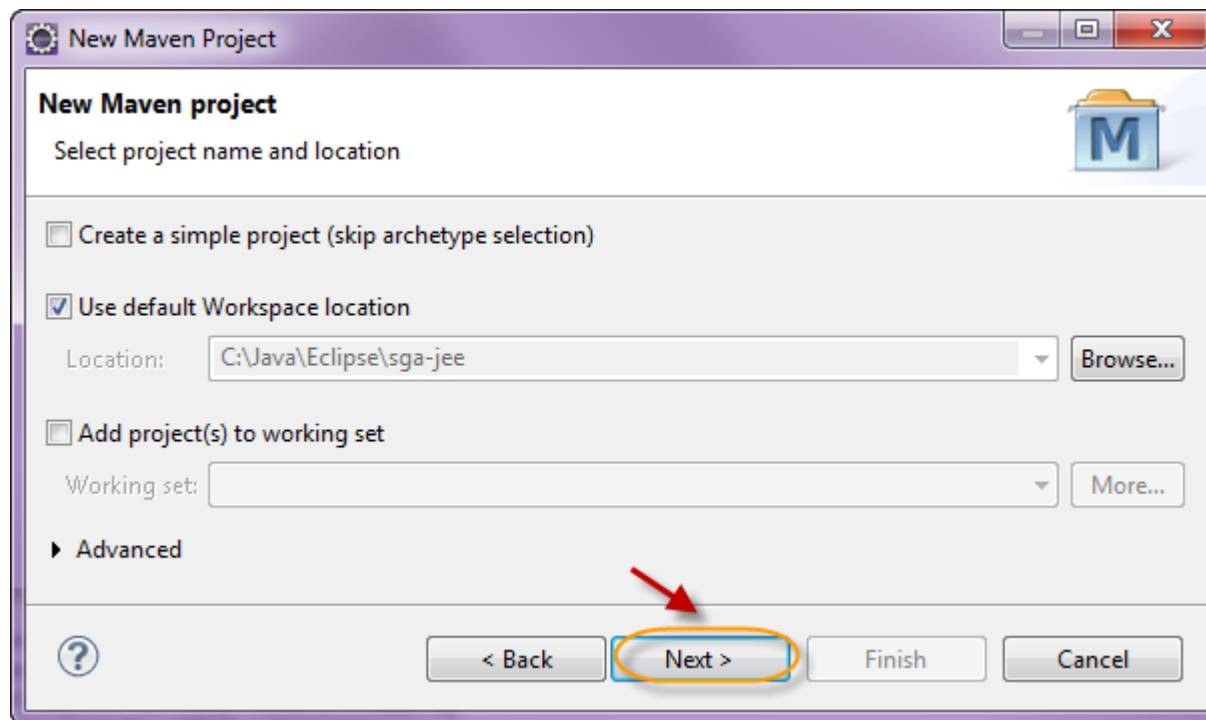
## Paso 3. Creación del proyecto Java Web (cont)

Creamos el proyecto sga-web-jsf utilizando el arquetipo web de Maven:



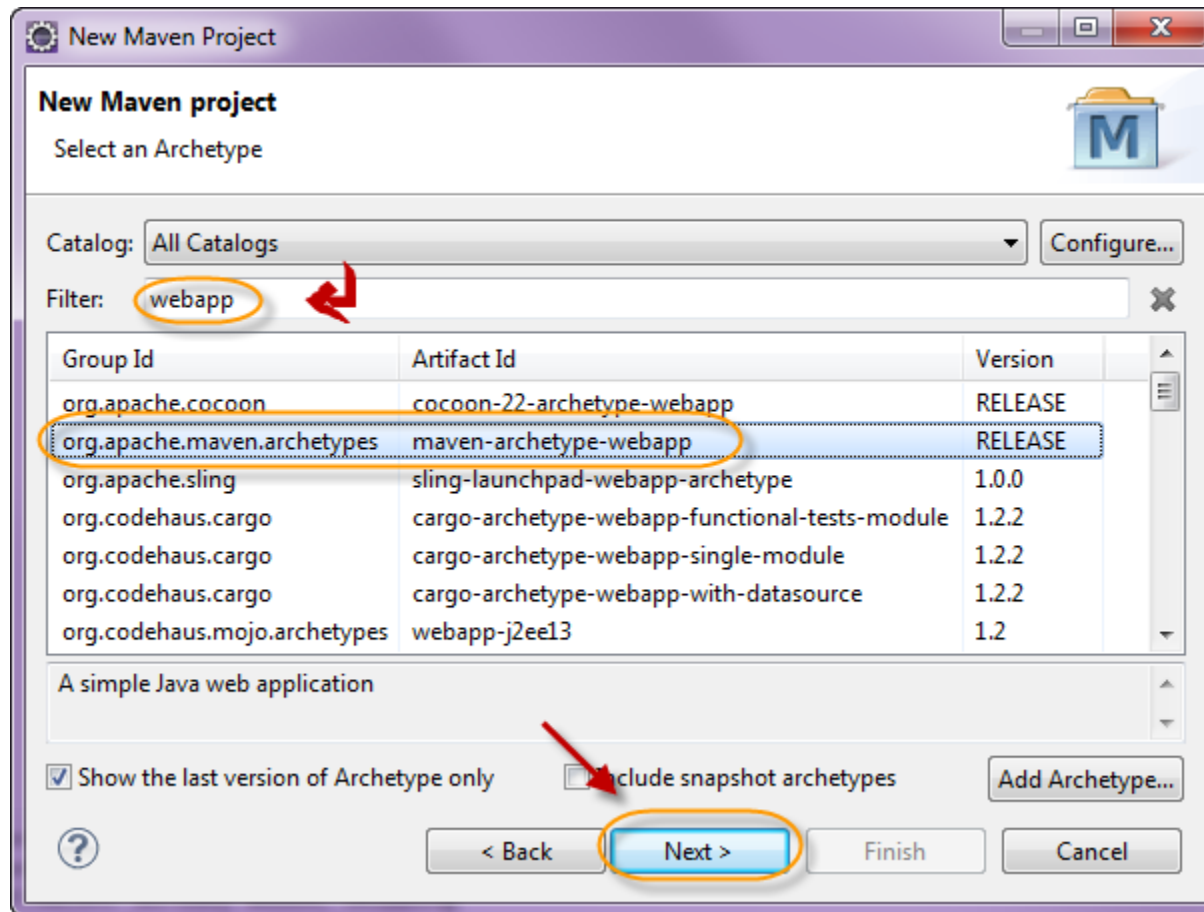
## Paso 3. Creación del proyecto Java Web (cont)

Creamos el proyecto sga-web-jsf utilizando el arquetipo web de Maven:



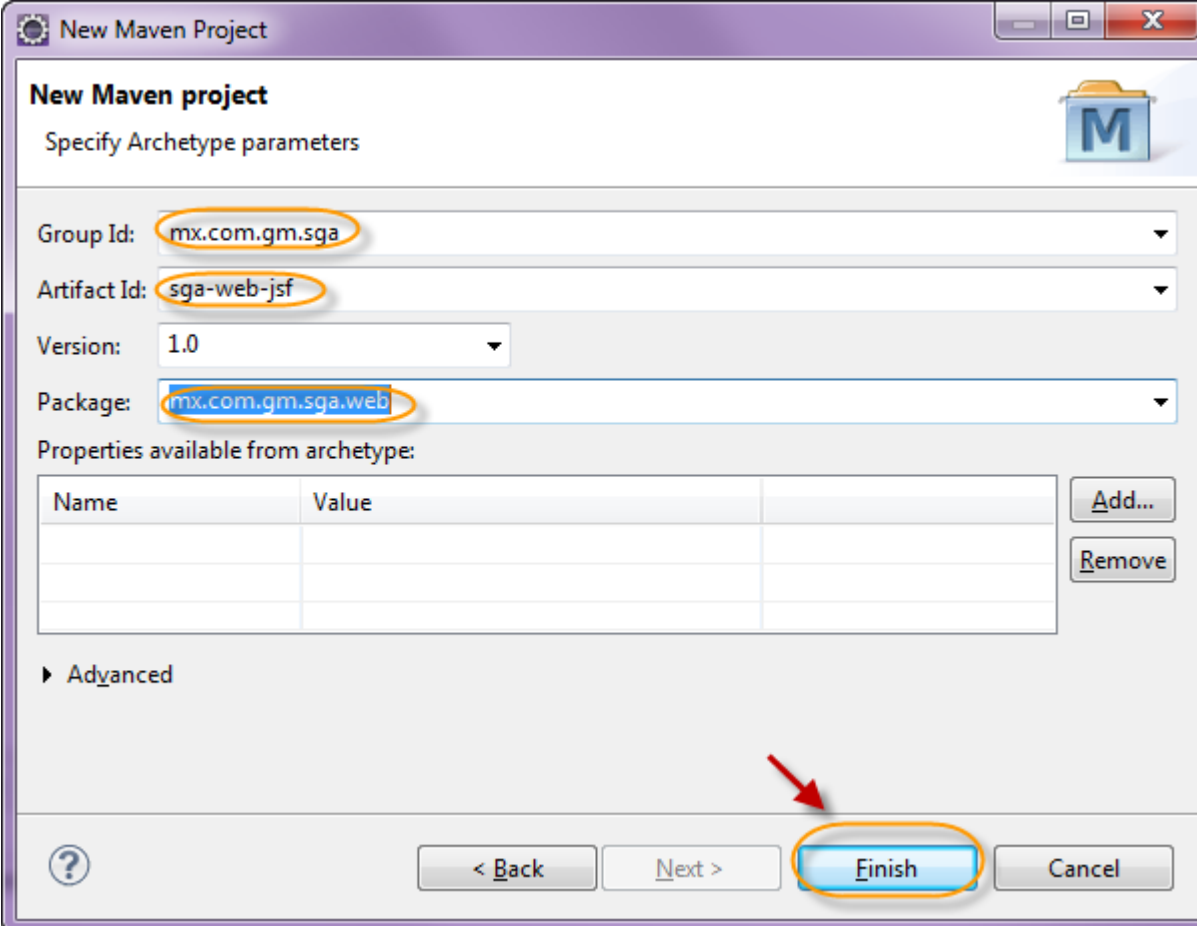
## Paso 3. Creación del proyecto Java Web (cont)

Creamos el proyecto sga-web-jsf utilizando el arquetipo web de Maven:



## Paso 3. Creación del proyecto Java Web (cont)

Creamos el proyecto sga-web-jsf utilizando el arquetipo web de Maven:



**New Maven Project**

Specify Archetype parameters

Group Id: **mx.com.gm.sga**

Artifact Id: **sga-web-jsf**

Version: 1.0

Package: **mx.com.gm.sga.web**

Properties available from archetype:

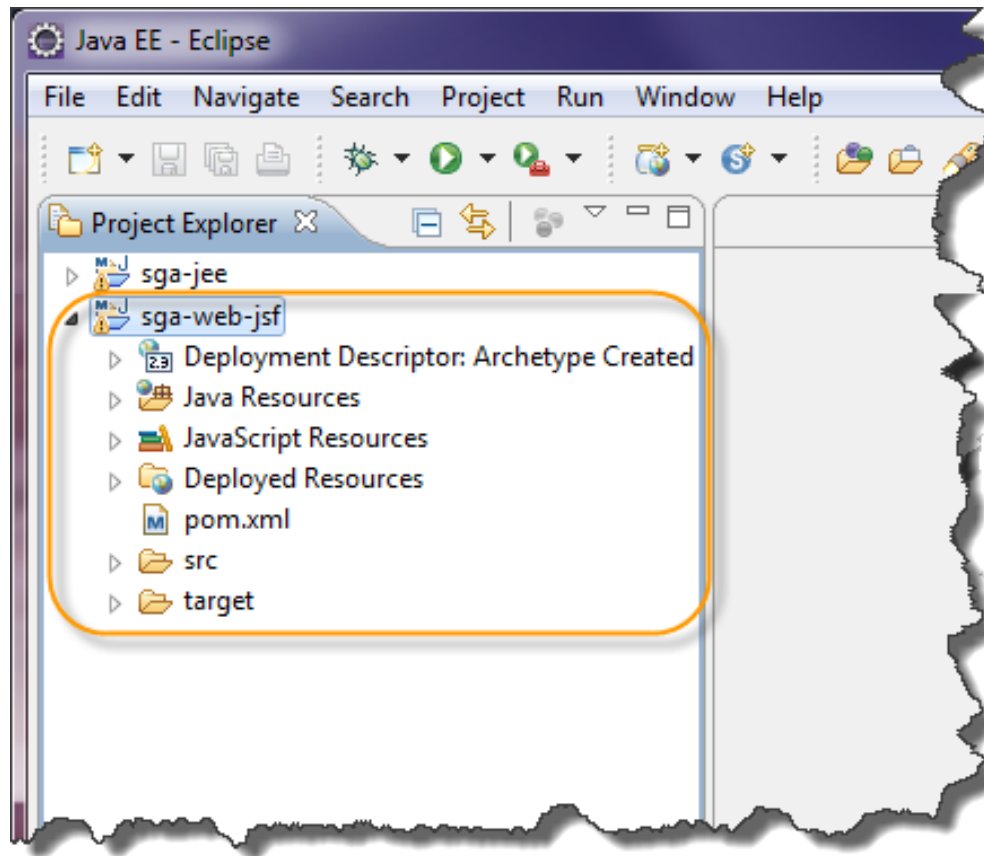
Name	Value

► Advanced

**Finish**

## Paso 3. Creación del proyecto Java Web (cont)

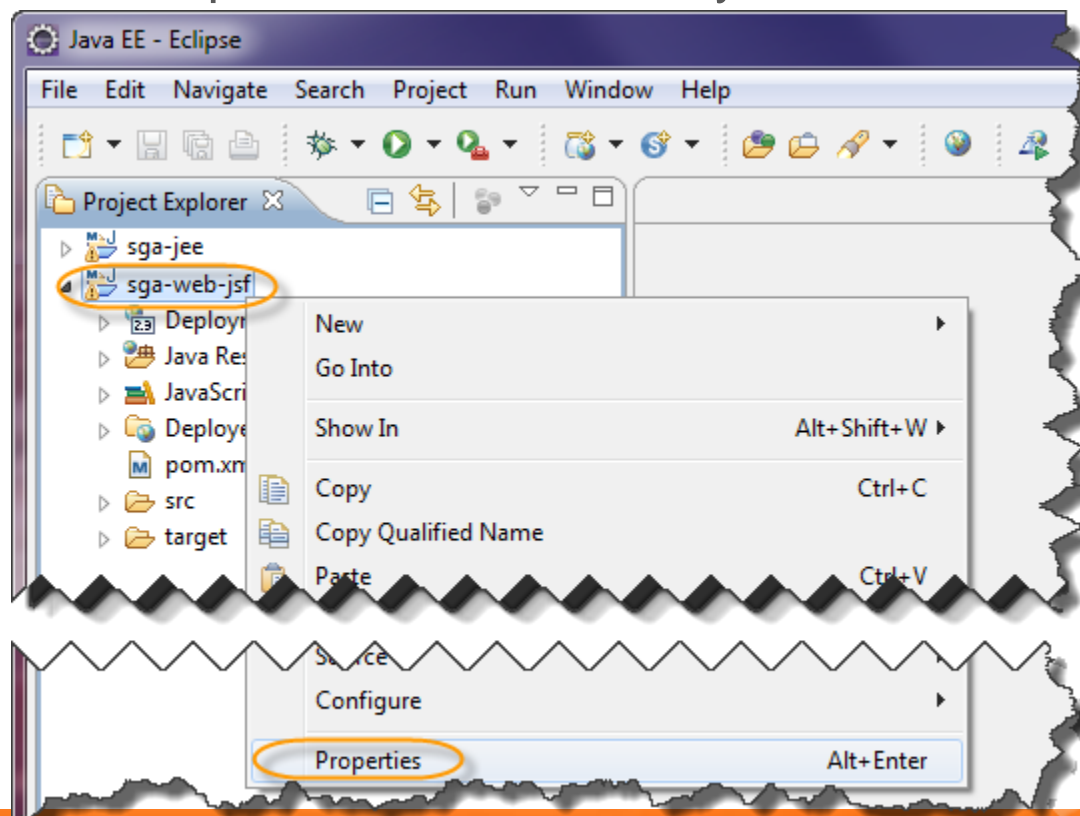
Confirmamos que la estructura creada sea la correcta:





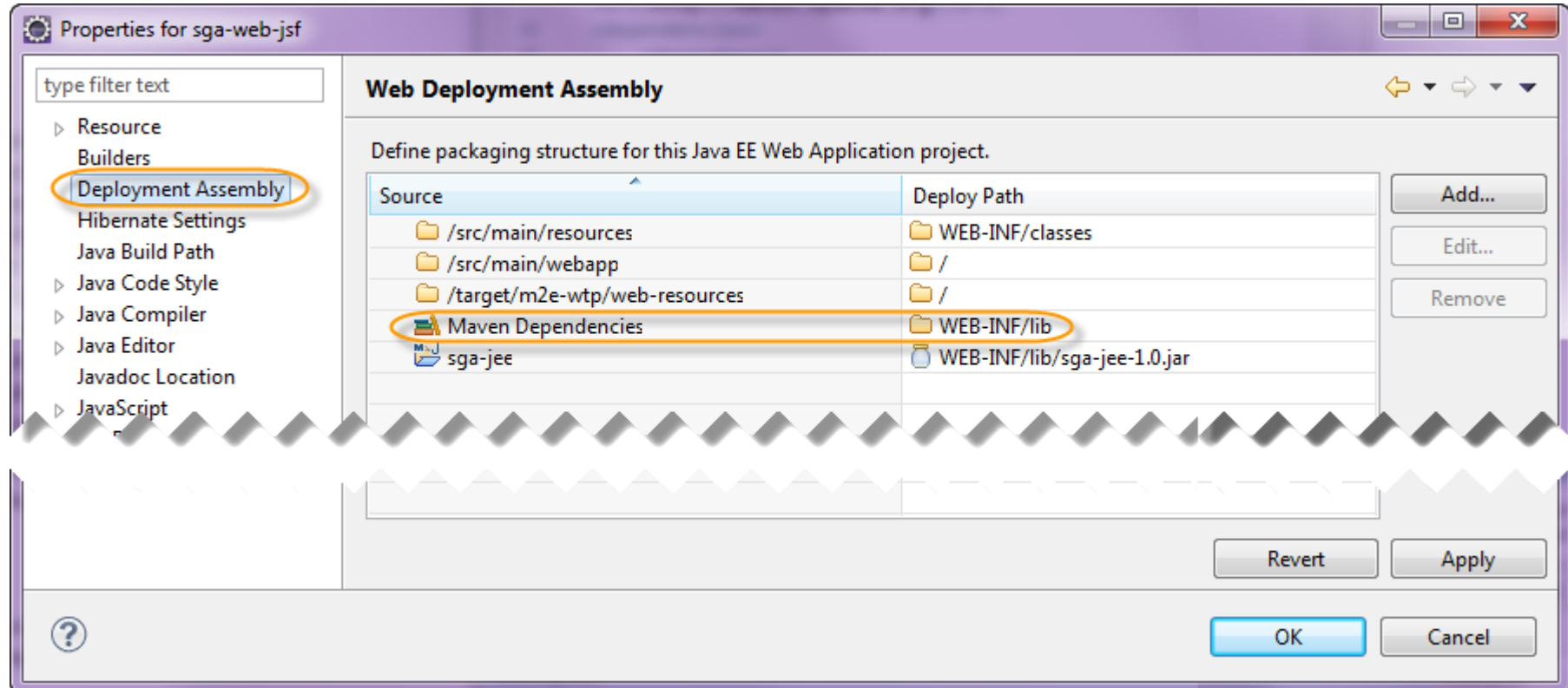
## Paso 4. Configuración del proyecto Java Web

Debido a que el arquetipo web de Maven crea un proyecto con una versión más antigua, haremos algunos cambios para actualizarlo a la última versión. Primero actualizamos los Facets de este proyecto a la versión 3, para el soporte de Servlets 3.0 y JSF 2.0.



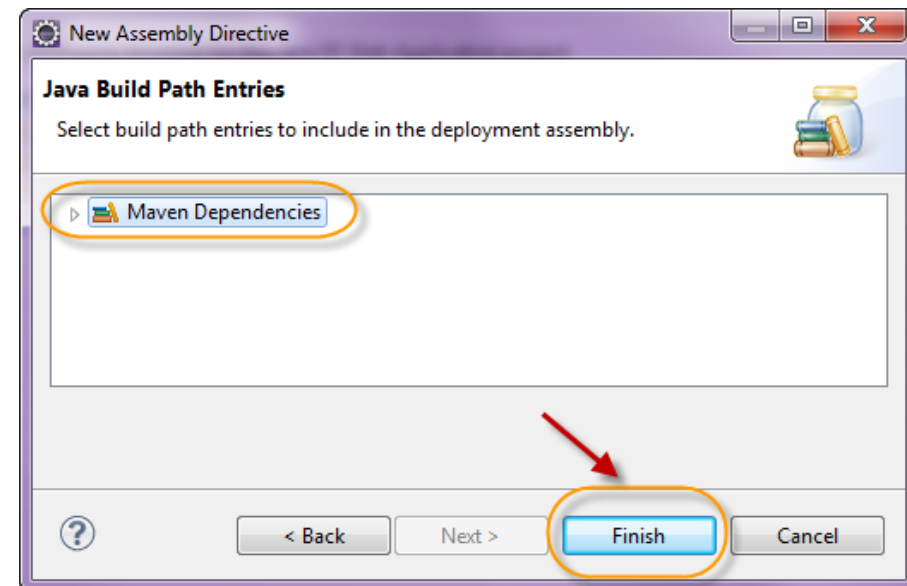
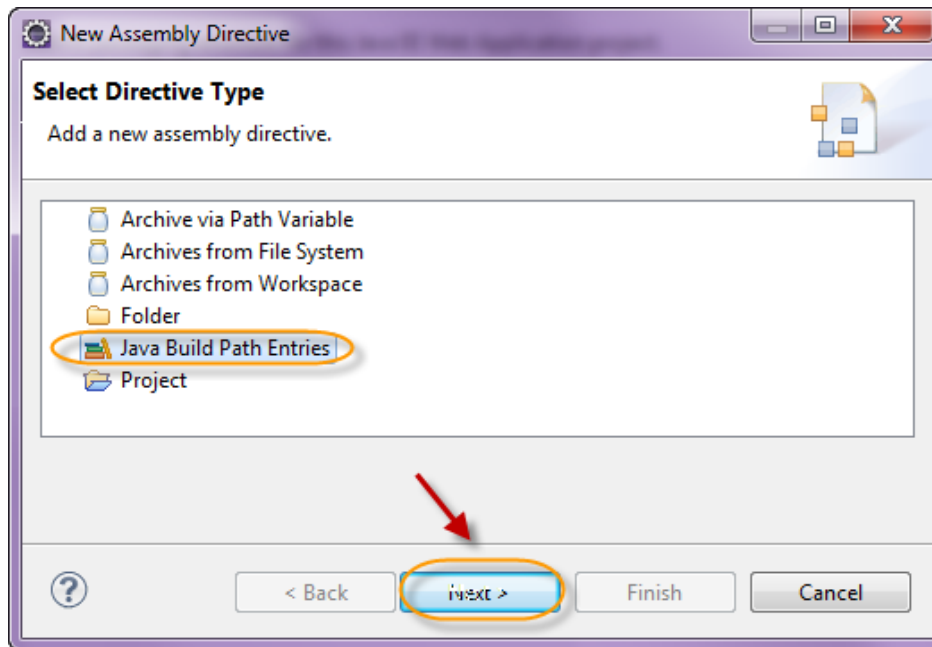
## Paso 4. Configuración del proyecto Java Web (cont)

Este paso, deberemos verificar que se incluyan las dependencias de Maven, en caso de no visualizar dichas dependencias y encontrar problemas al momento del despliegue de la aplicación y marque errores de no encontrar las clases Java del proyecto de primefaces o del proyecto sga-jee, es necesario agregar estas dependencias. Si están incluidas las dependencias de Maven NO hay que hacer nada en la sección de Deployment Assembly.



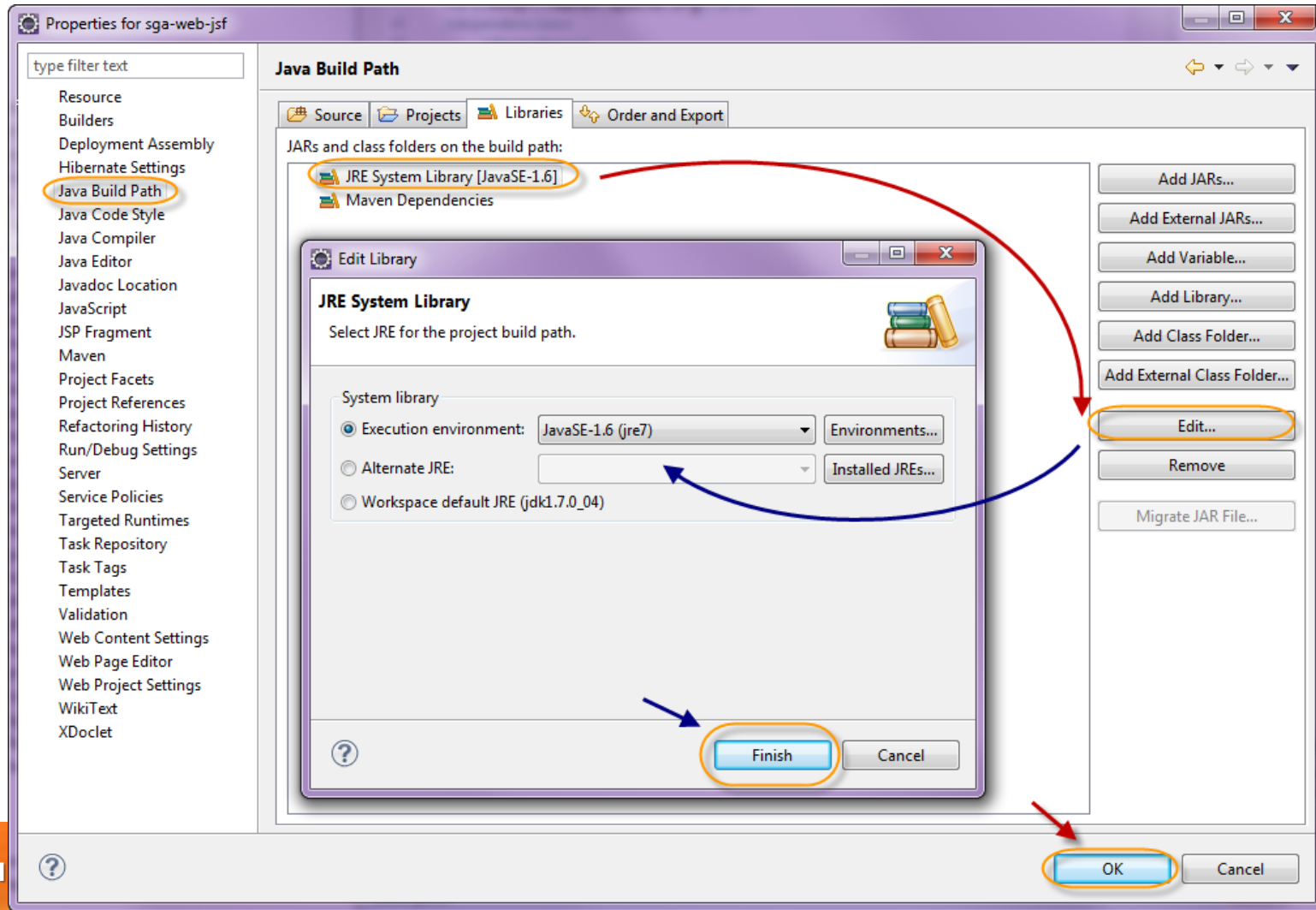
## Paso 4. Configuración del proyecto Java Web (cont)

En caso de que el paso anterior no tenga las dependencias de Maven, deberemos agregarlas como sigue, dar click en Add de la pantalla anterior y proceder como sigue:



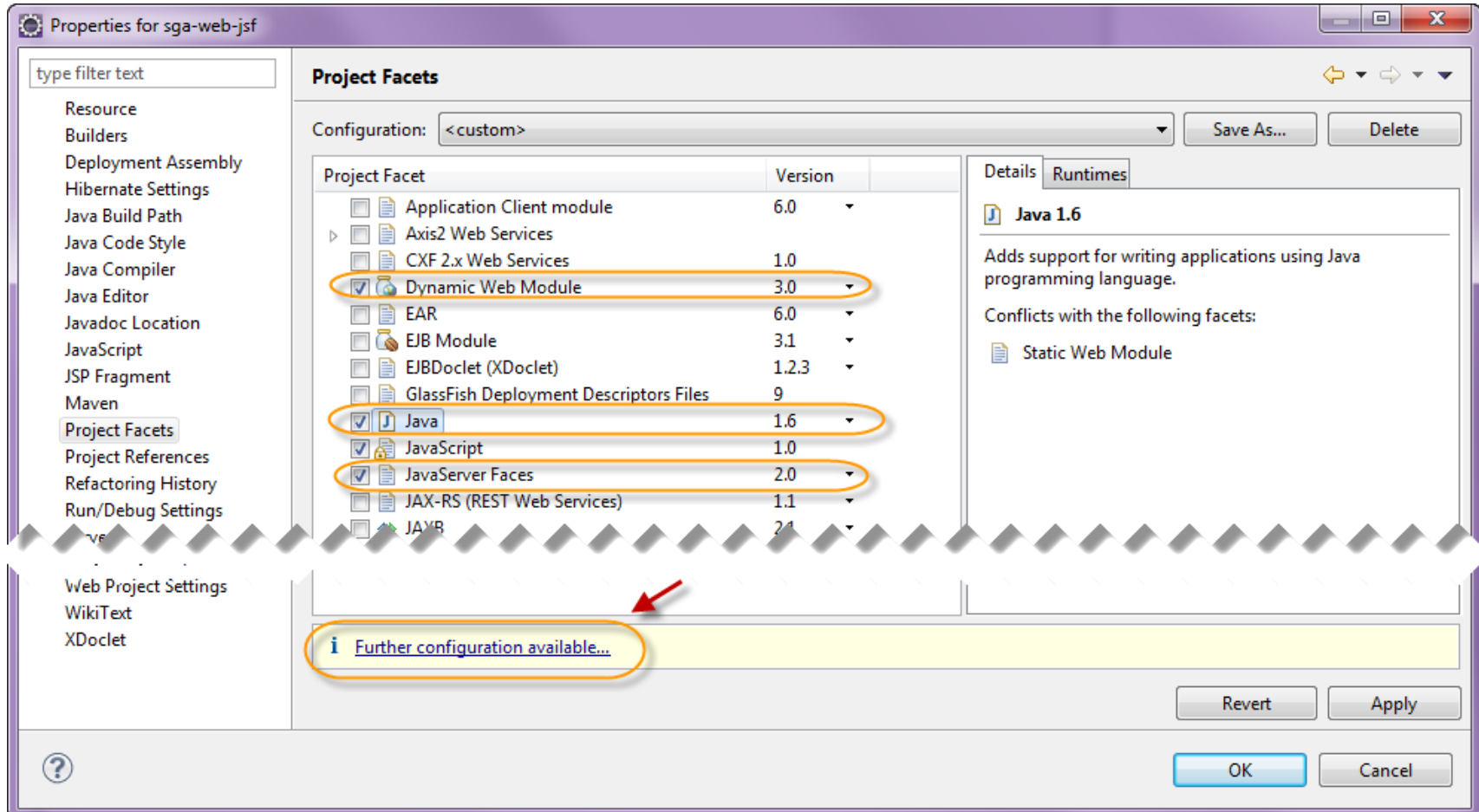
## Paso 4. Configuración del proyecto Java Web (cont)

Actualizamos el jdk a la versión 1.6 para compatibilidad con el facelet que modificaremos más adelante:



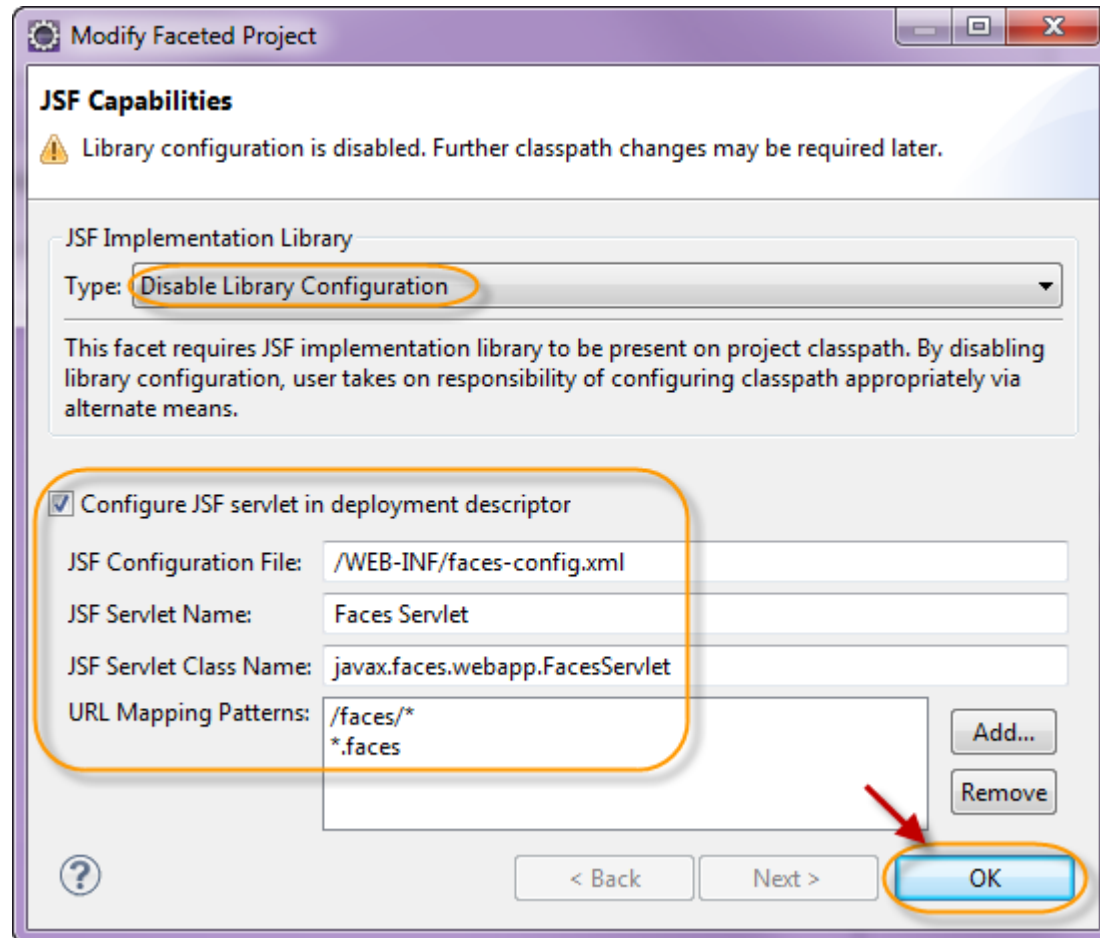
## Paso 4. Configuración del proyecto Java Web (cont)

Seleccionamos la versión 3 de Dynamic Web Module, el JDK 1.6 y JSF 2.0



## Paso 4. Configuración del proyecto Java Web (cont)

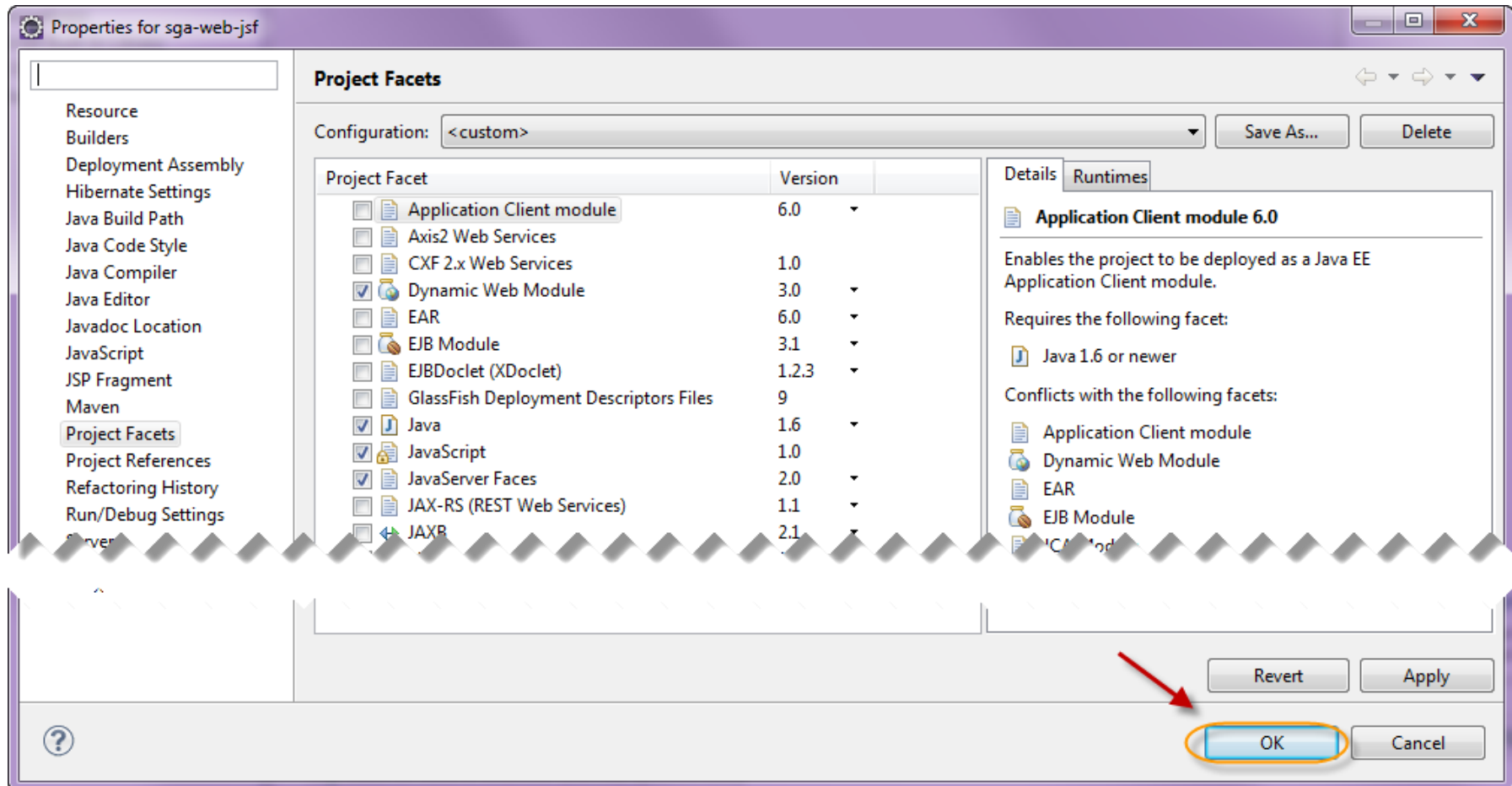
Seleccionamos la versión 3 de Dynamic Web Module y el JDK 1.6





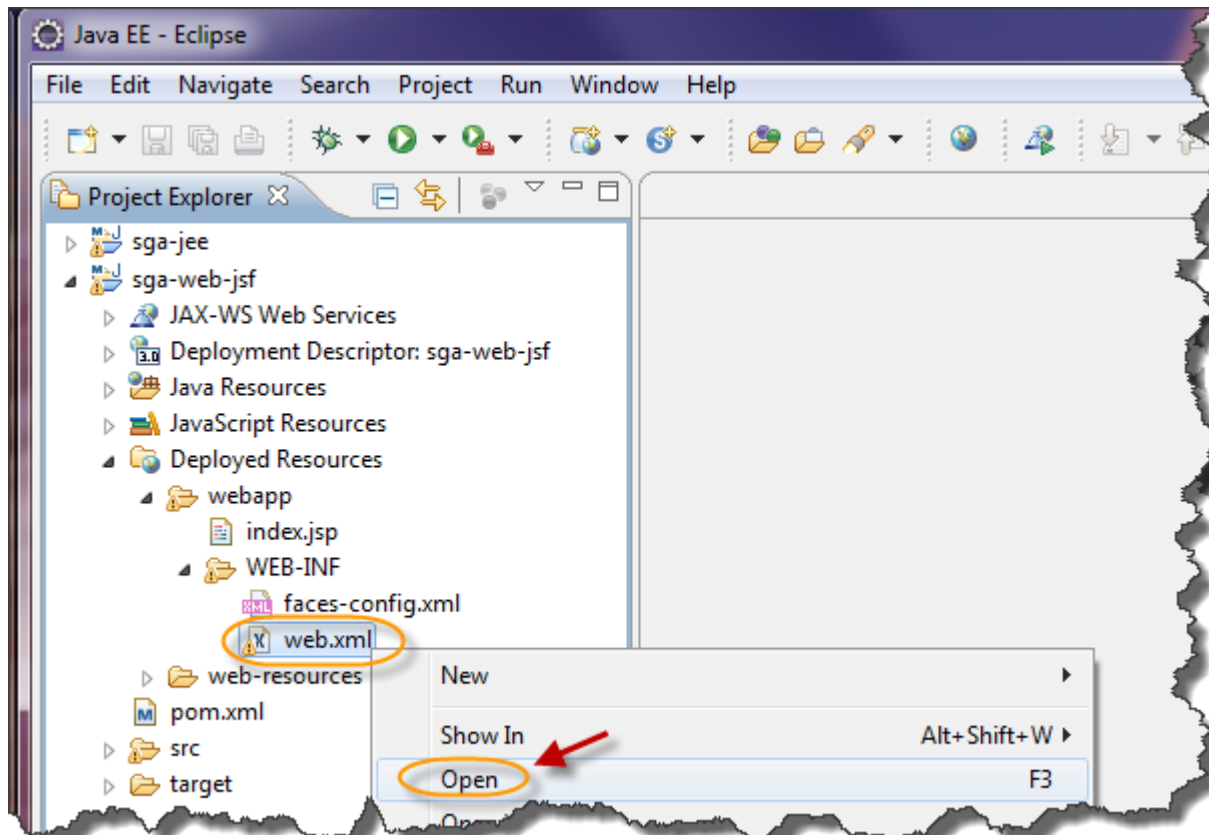
## Paso 4. Configuración del proyecto Java Web (cont)

Seleccionamos la versión 3 de Dynamic Web Module y el JDK 1.6 y JSF 2.0.



## Paso 5. Modificamos el archivo web.xml

Modificamos el archivo web.xml





# Paso 5. Modificamos el archivo web.xml (cont)

## Modificamos el archivo web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <context-param>
    <param-name>javax.faces.FACELETS_REFRESH_PERIOD</param-name>
    <param-value>0</param-value>
  </context-param>
  <context-param>
    <param-name>primefaces.THEME</param-name>
    <param-value>cupertino</param-value>
  </context-param>
  <error-page>
    <exception-type>javax.faces.application.ViewExpiredException</exception-type>
    <location>/faces/index.xhtml</location>
  </error-page>
  <welcome-file-list>
    <welcome-file>faces/index.xhtml</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.faces</url-pattern>
    <url-pattern>/faces/*</url-pattern>
  </servlet-mapping>
</web-app>
```



## Paso 6. Agregamos las dependencias Maven

Agregamos las dependencias y repositorios de Maven al proyecto sga-web-jsf (sustituimos cualquier dependencia existente) dentro del tag de project:

```
<dependencies>
  <dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-api</artifactId>
    <version>6.0</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>mx.com.gm.sga</groupId>
    <artifactId>sga-jee</artifactId>
    <version>1.0</version>
  </dependency>
  <dependency>
    <groupId>org.primefaces</groupId>
    <artifactId>primefaces</artifactId>
    <version>3.3.1</version>
  </dependency>
  <dependency>
    <groupId>org.primefaces.themes</groupId>
    <artifactId>cupertino</artifactId>
    <version>1.0.6</version>
  </dependency>
</dependencies>

<repositories>
  <repository>
    <id>prime-repo</id>
    <name>PrimeFaces Maven Repository</name>
    <url>http://repository.primefaces.org</url>
    <layout>default</layout>
  </repository>
</repositories>
```

## Paso 6. Agregamos las dependencias Maven (cont)

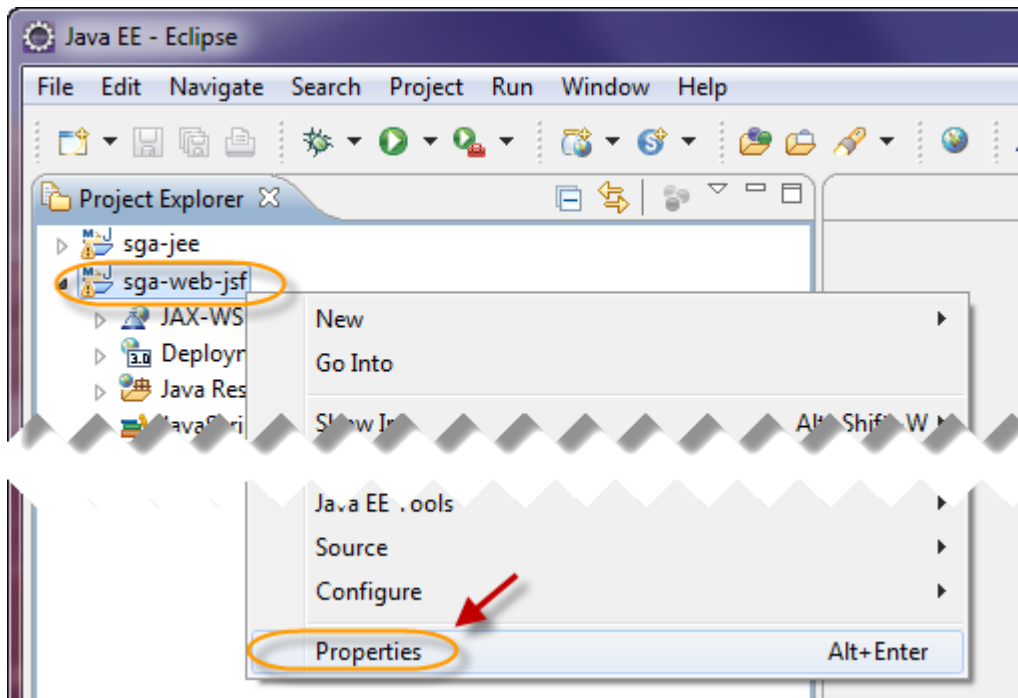
Deberemos observar las nuevas dependencias, incluyendo las del proyecto sga-jee y las nuevas librerías de PrimeFaces:

The screenshot shows the Eclipse IDE with the following components:

- Project Explorer:** Displays the project structure. The `sga-jee` project is listed as a dependency under the `sga-web-jsf` project.
- Maven Dependencies:** Lists the Maven dependencies for the `sga-web-jsf` project. The dependencies are grouped into sections: `javaee-api-6.0.jar`, `glassfish-embedded-static-shell-3.1.jar`, `junit-4.10.jar`, `hamcrest-core-1.1.jar`, `mysql-connector-java-5.1.20.jar`, `hibernate-entitymanager-4.1.4.Final.jar`, `hibernate-core-4.1.4.Final.jar`, `antlr-2.7.7.jar`, `jboss-transaction-api-1.1_spec-1.0.0.Final.jar`, `dom4j-1.6.1.jar`, `hibernate-jpa-2.0-api-1.0.1.Final.jar`, `javassist-3.15.0-GA.jar`, `hibernate-commons-annotations-4.0.1.Final.jar`, `slf4j-api-1.5.6.jar`, `slf4j-log4j12-1.5.6.jar`, `log4j-1.2.14.jar`, `primefaces-3.3.1.jar`, and `cupertino-1.0.6.jar`.
- sga-jee/pom.xml:** The XML file for the `sga-jee` project. It includes the following dependencies:
  - `javaee-api-6.0` (provided scope)
  - `sga-jee` (version 1.0)
  - `primefaces` (version 3.3.1)
  - `primefaces.themes.cupertino` (version 1.0.6)
- Repositories:** The `prime-repo` repository is defined with the URL `http://repository.primefaces.org`.

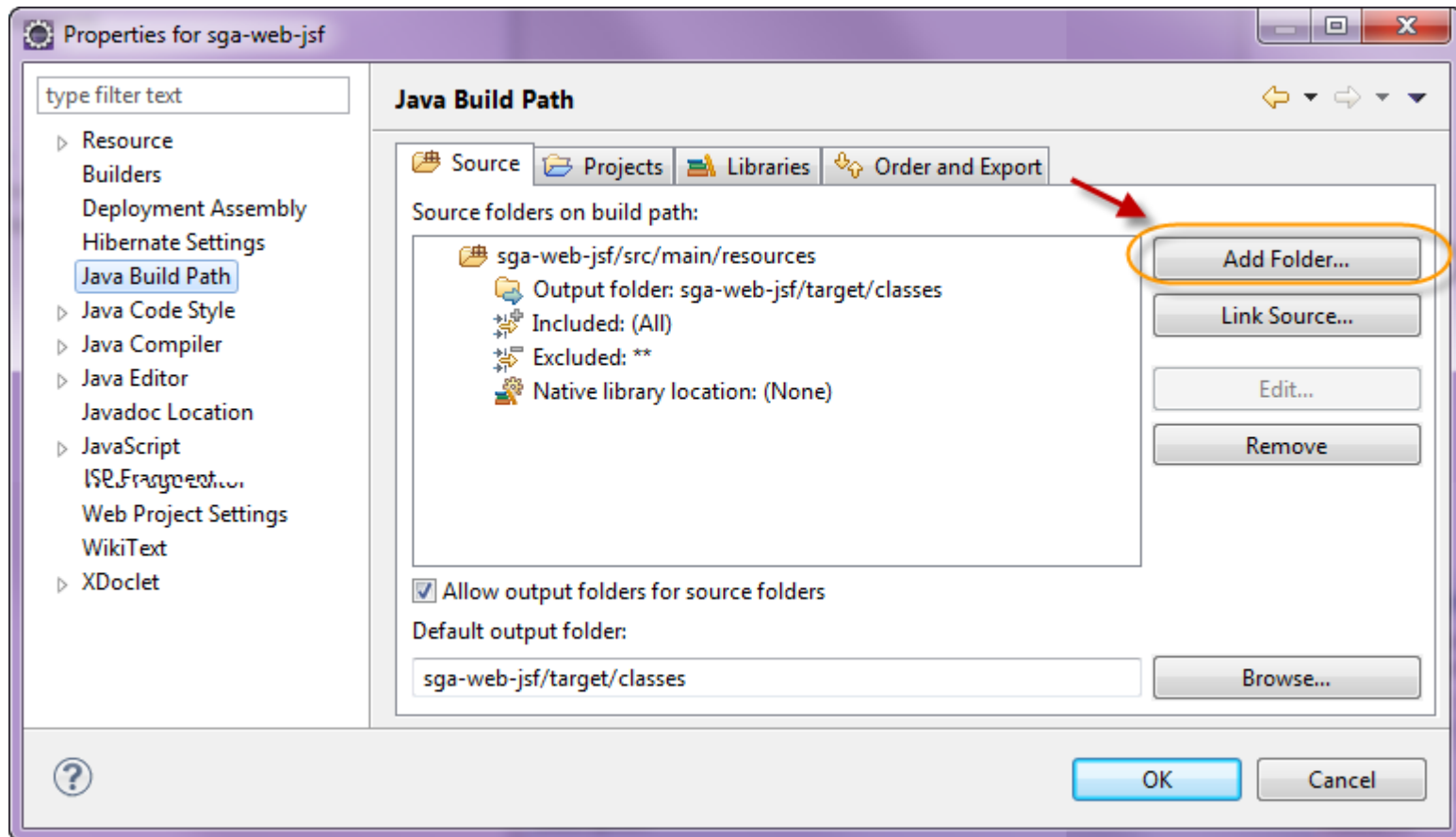
## Paso 7. Agregamos la carpeta de código Java

Modificamos la configuración del classpath de la aplicación para agregar la carpeta que contendrá la configuración del código Java:



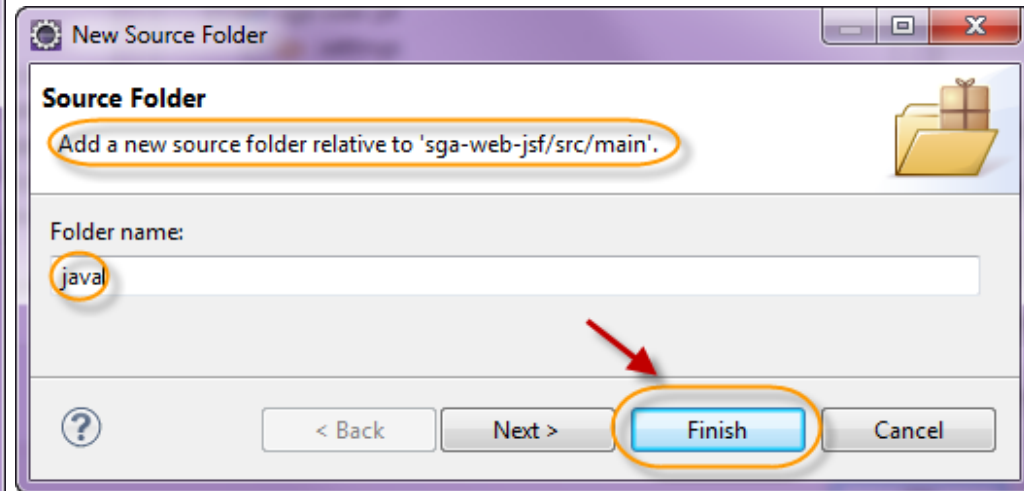
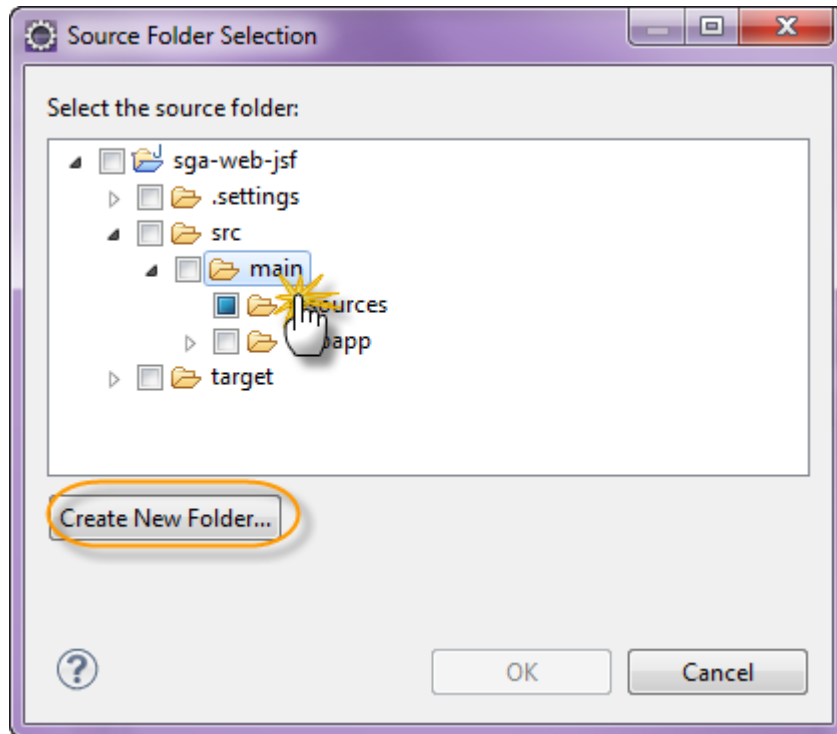
## Paso 7. Agregamos la carpeta de código Java (cont)

Modificamos la configuración del classpath de la aplicación:



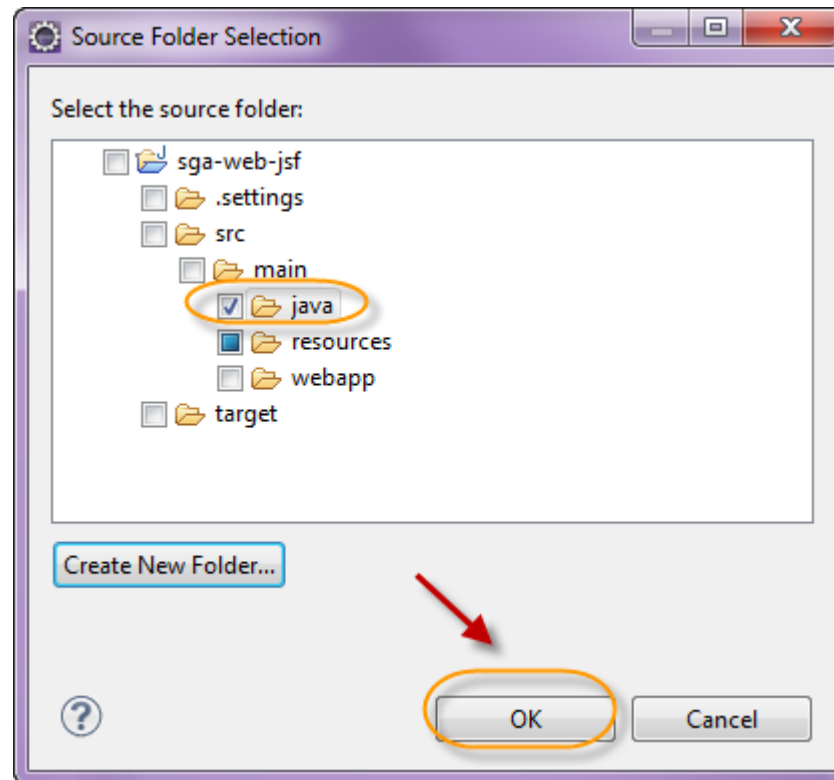
## Paso 7. Agregamos la carpeta de código Java (cont)

Modificamos la configuración del classpath de la aplicación:



## Paso 7. Agregamos la carpeta de código Java (cont)

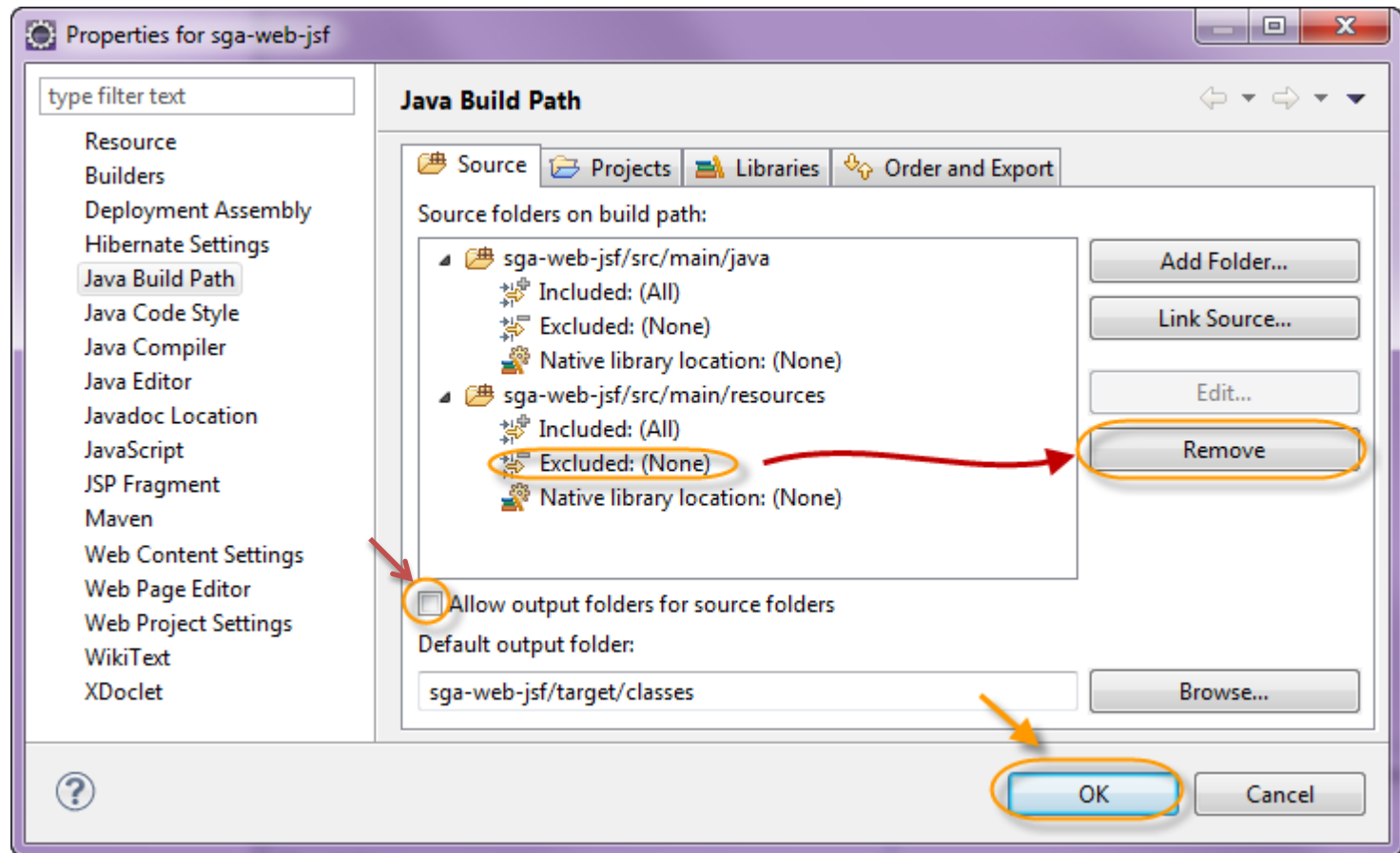
Modificamos la configuración del classpath de la aplicación:





## Paso 7. Agregamos la carpeta de código Java (cont)

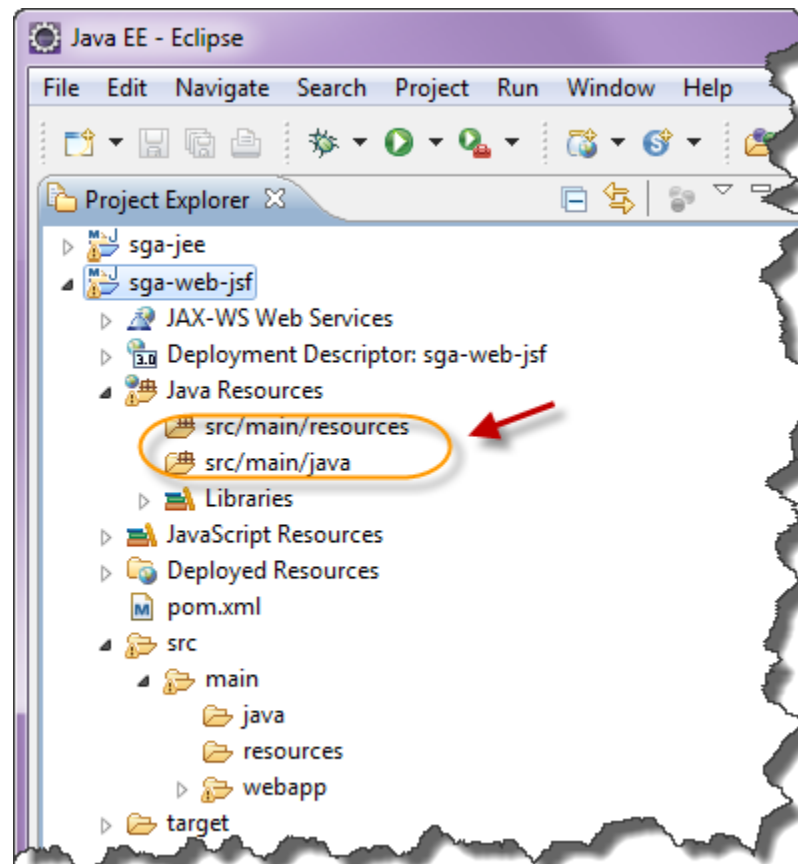
Modificamos la configuración del classpath de la aplicación:





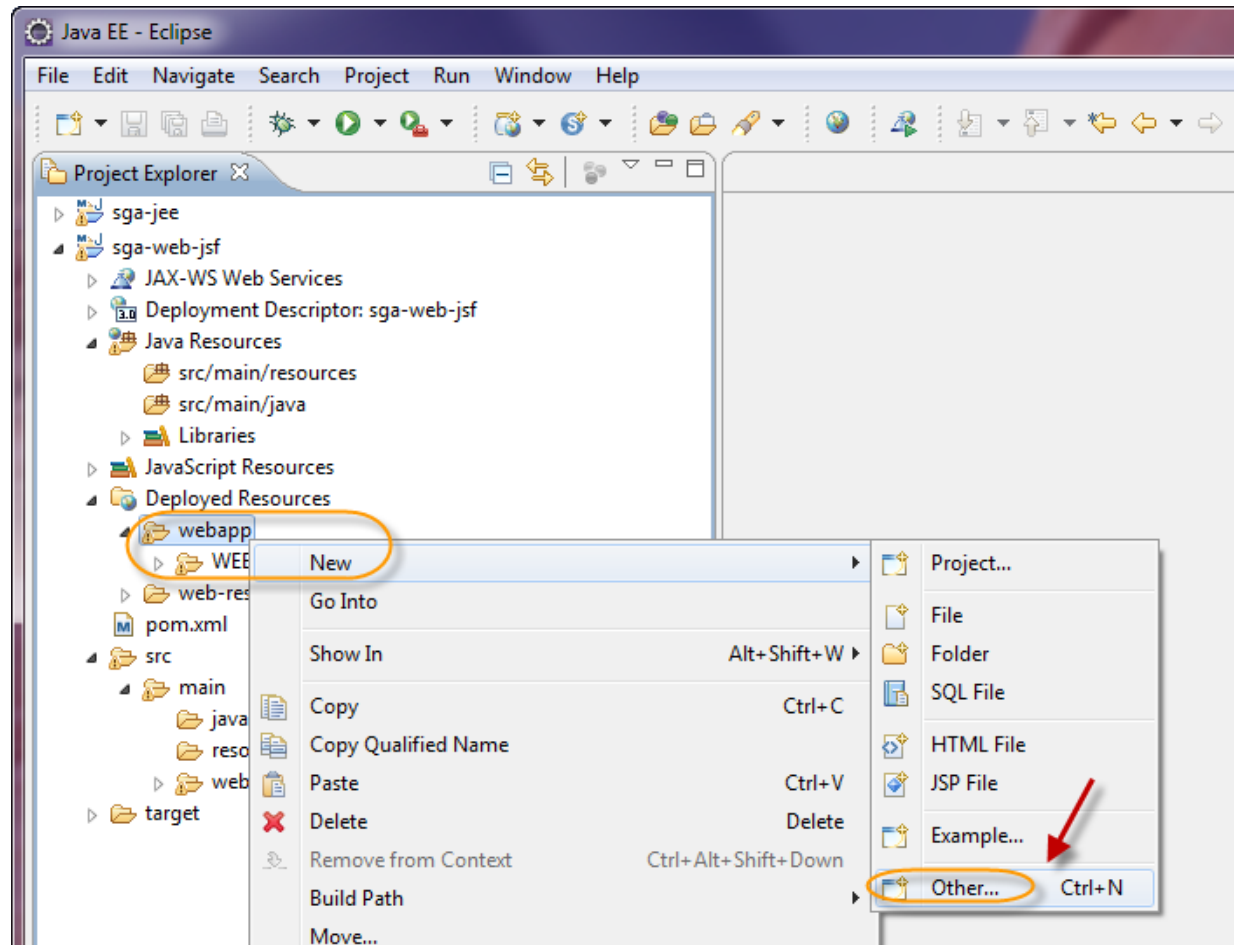
## Paso 7. Agregamos la carpeta de código Java (cont)

Este debe ser el resultado final de la configuración anterior:



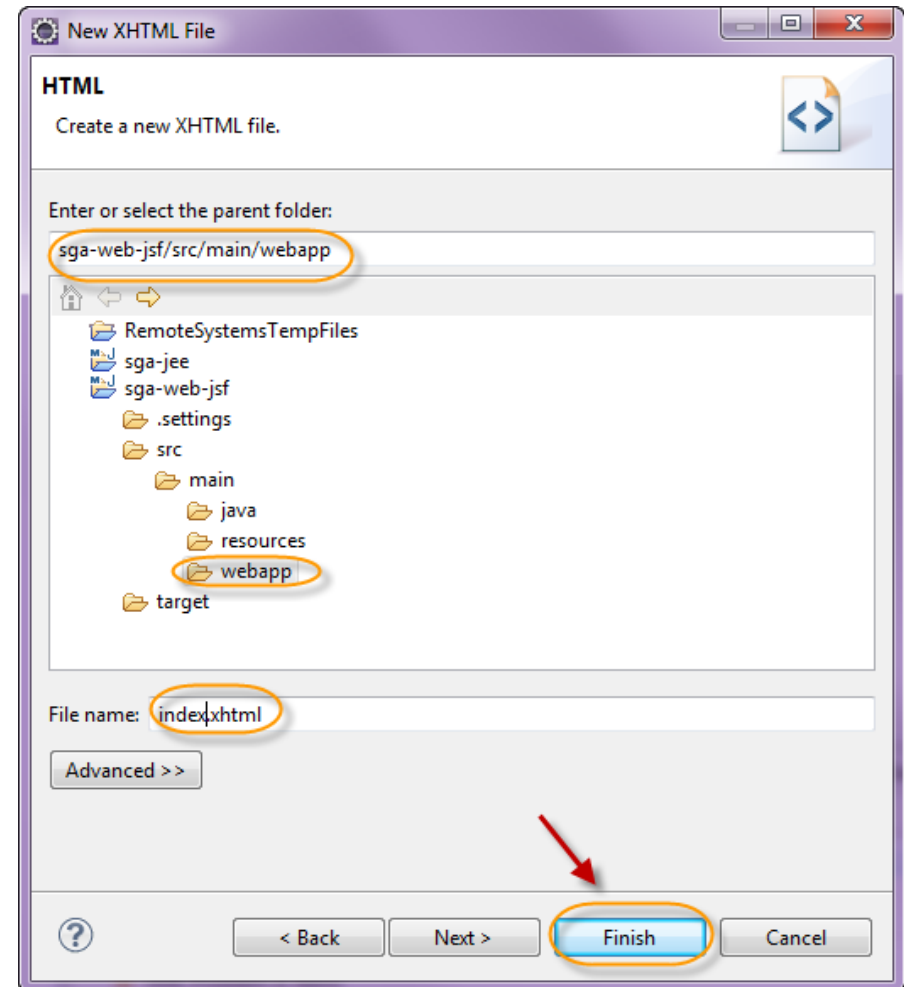
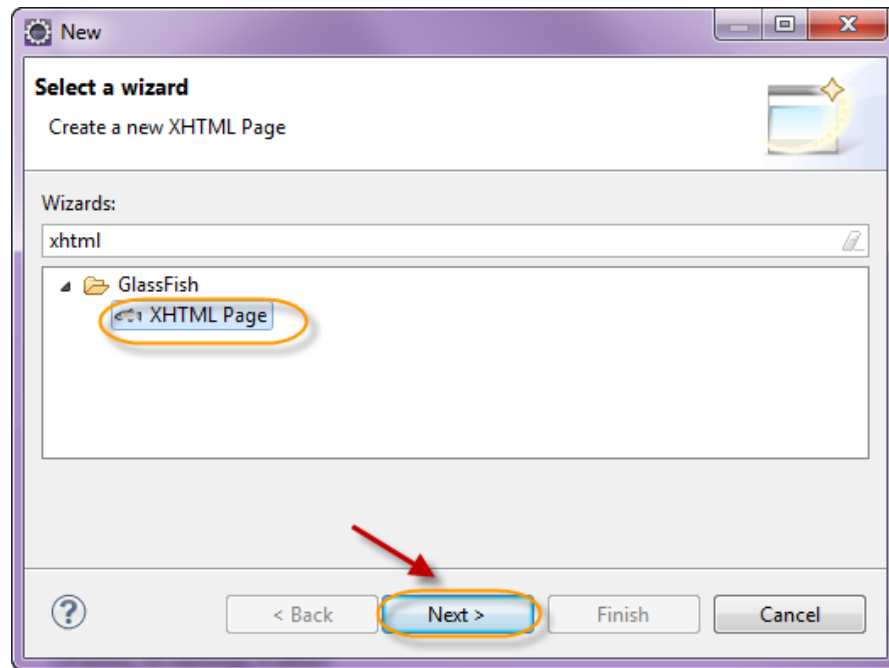
## Paso 8. Creación de index.xhtml

Creación del archivo index.xhtml



## Paso 8. Creación de index.xhtml

Creación del archivo index.xhtml



## Paso 8. Creación de index.xhtml (cont)

### Creación del archivo index.xhtml

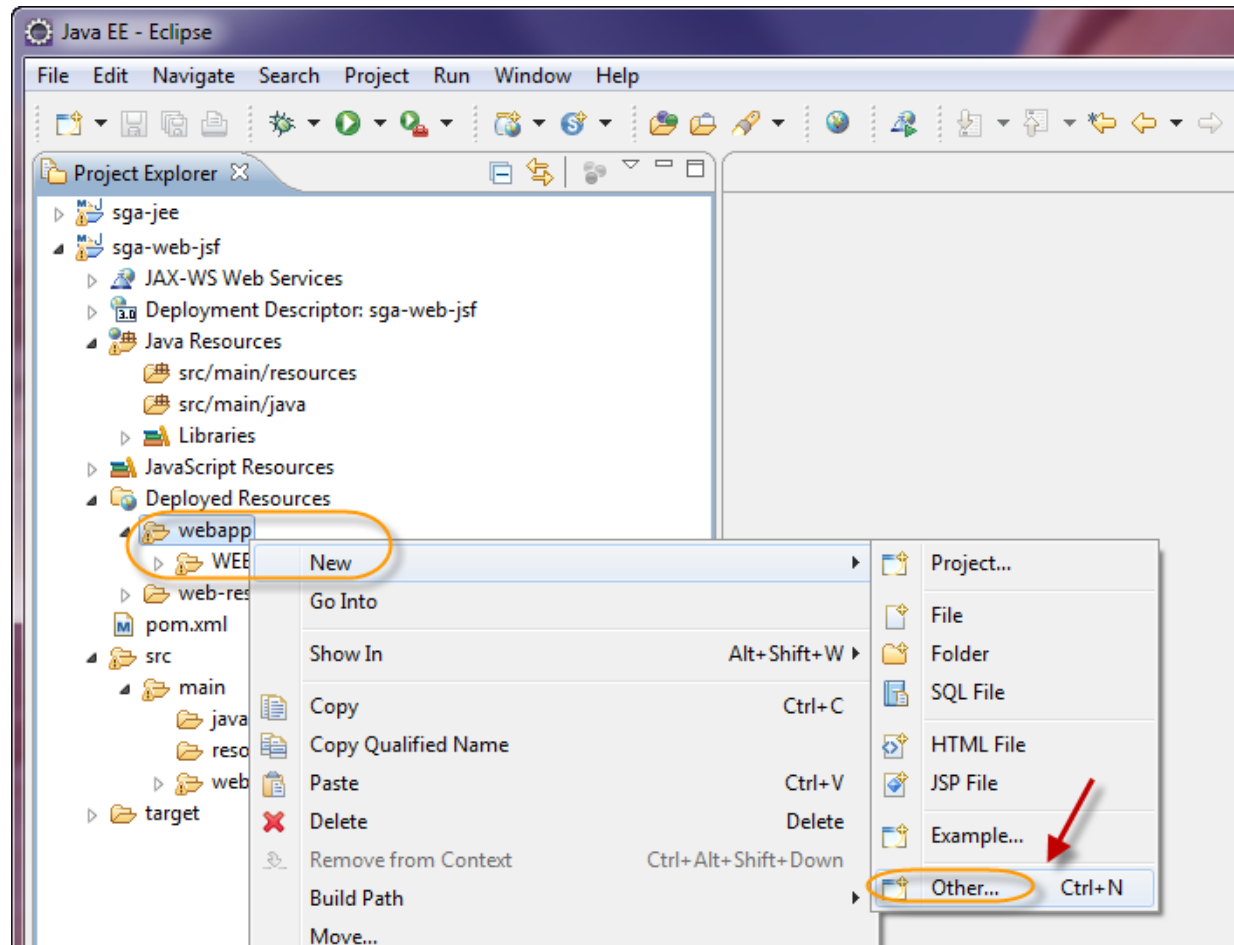
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:p="http://primefaces.org/ui">

    <h:head>
        <title>Sistema de Gestión de Alumnos</title>
    </h:head>

    <h:body>
        <h2>Sistema de Gestión de Alumnos (SGA)</h2>
        <h:form>
            <h:commandButton value="Listar Personas" action="ListarPersonas" />
        </h:form>
    </h:body>
</html>
```

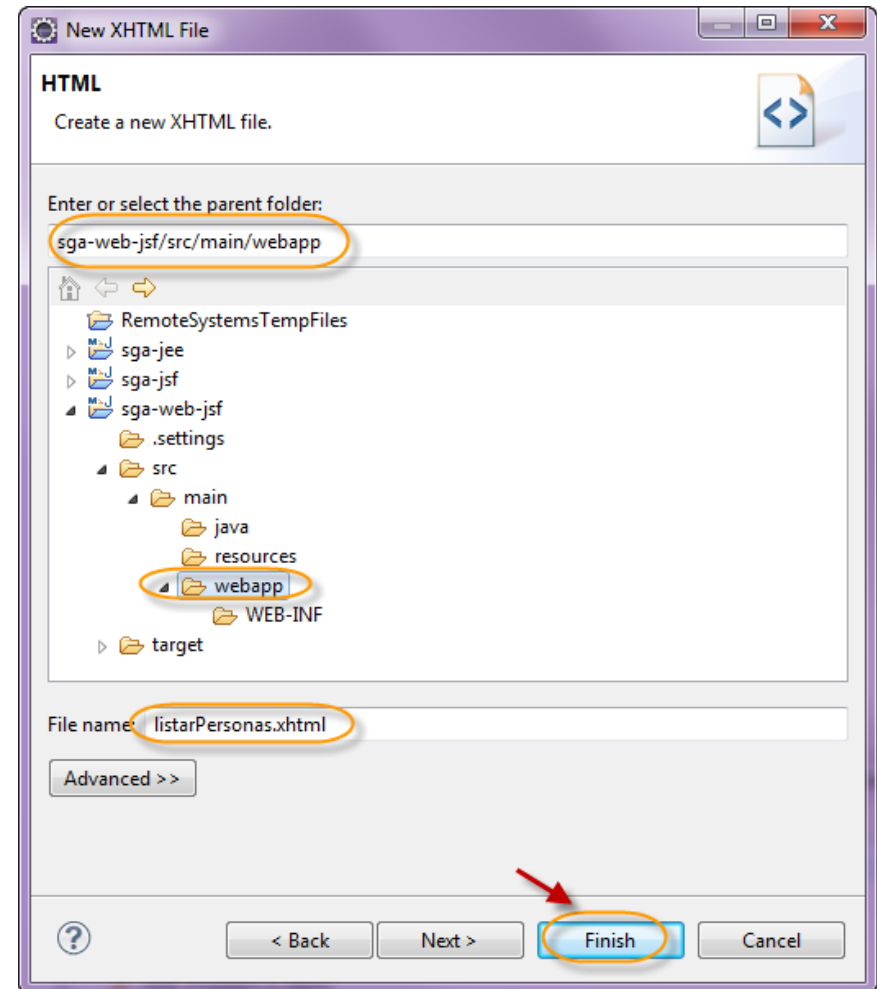
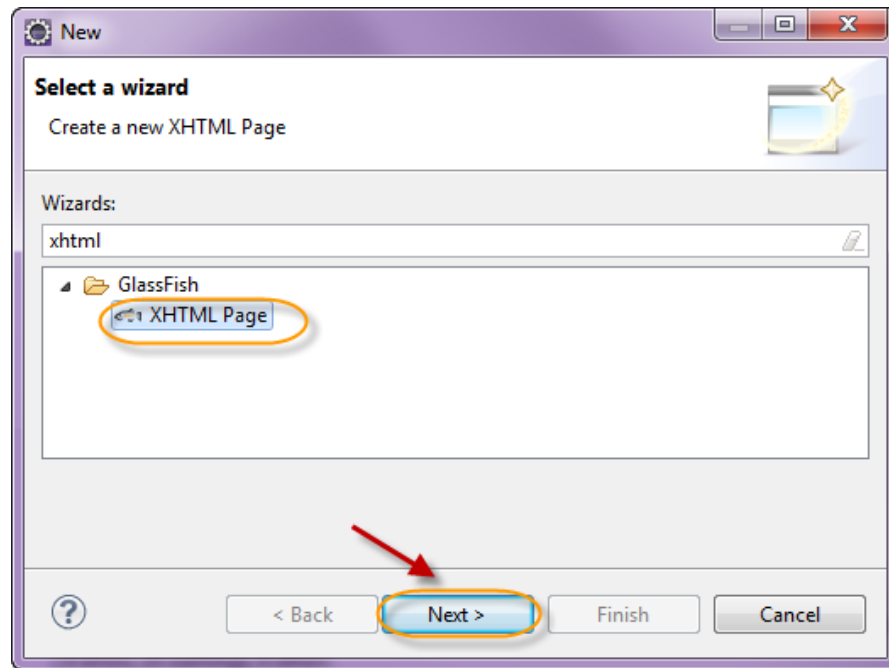
## Paso 9. Creación de listarPersonas.xhtml

Creación del archivo listarPersonas.xhtml



## Paso 9. Creación de listarPersonas.xhtml (cont)

Creación del archivo listarPersonas.xhtml



## Paso 9. Creación de listarPersonas.xhtml (cont)

```
<!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:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">

<h:head>
<title>Listar Personas</title>
</h:head>
<h:body>

<p:ajaxStatus style="width:16px;height:16px;" id="ajaxStatusPanel">
  <f:facet name="start">
    <h:graphicImage library="images" name="ajaxLoading.gif" />
  </f:facet>

  <f:facet name="complete">
    <h:outputText value="" />
  </f:facet>
</p:ajaxStatus>

<h:form prependId="false">

  <p:growl id="messages" showDetail="true" />

  <p:dataTable value="#{personaBean.personas}" var="persona"
    editable="true">

    <f:facet name="header">
      Lista de Personas
    </f:facet>

    <p:column headerText="Nombre">
      <p:cellEditor>
        <f:facet name="output">
          <h:outputText value="#{persona.nombre}" />
        </f:facet>
```

```
        <f:facet name="input">
          <p:inputText value="#{persona.nombre}" />
        </f:facet>
      </p:cellEditor>
    </p:column>
    <p:column headerText="Apellido Paterno">
      <p:cellEditor>
        <f:facet name="output">
          <h:outputText value="#{persona.apePaterno}" />
        </f:facet>
        <f:facet name="input">
          <p:inputText value="#{persona.apePaterno}" />
        </f:facet>
      </p:cellEditor>
    </p:column>
    <p:column headerText="eMail">
      <p:cellEditor>
        <f:facet name="output">
          <h:outputText value="#{persona.email}" />
        </f:facet>
        <f:facet name="input">
          <p:inputText value="#{persona.email}" />
        </f:facet>
      </p:cellEditor>
    </p:column>
    <f:facet name="footer">
      <h:commandButton value="Regresar" action="index" />
    </f:facet>

    <p:column headerText="Opciones" style="width:50px">
      <p:rowEditor />
    </p:column>

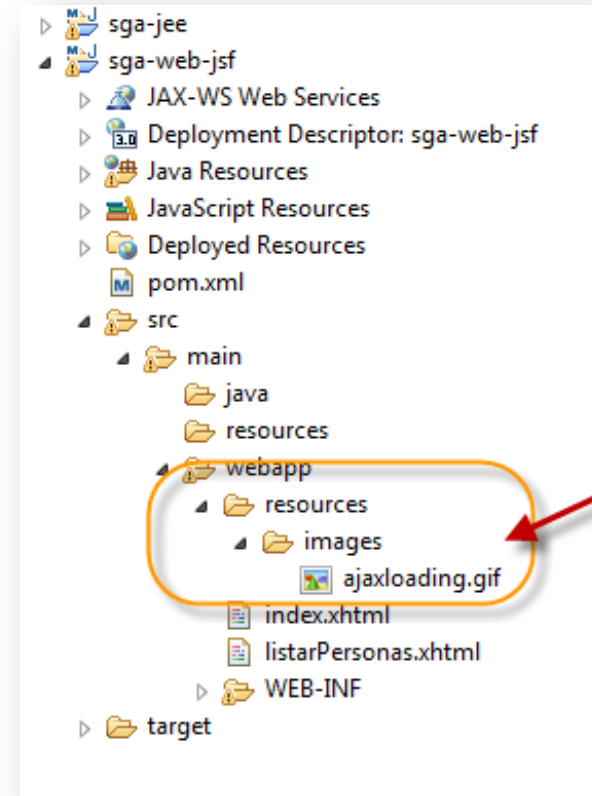
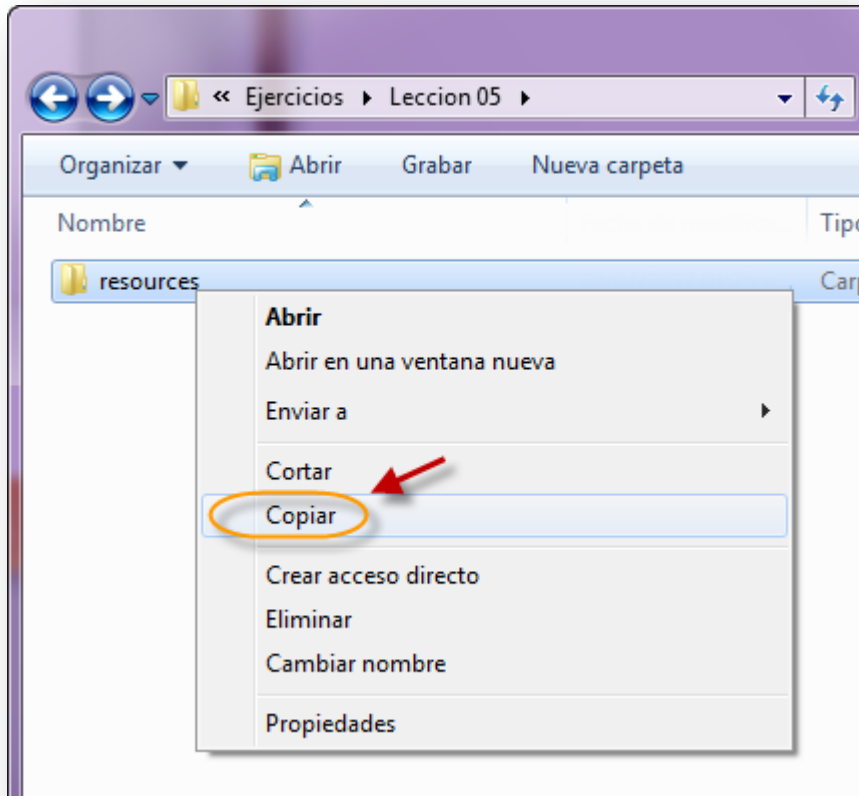
    <p:ajax event="rowEdit"
      listener="#{personaBean.editListener}" />
  </p:ajax>
</p:dataTable>

</h:form>
</h:body>

</html>
```

## Paso 10. Copiar la imagen ajaxloading.gif

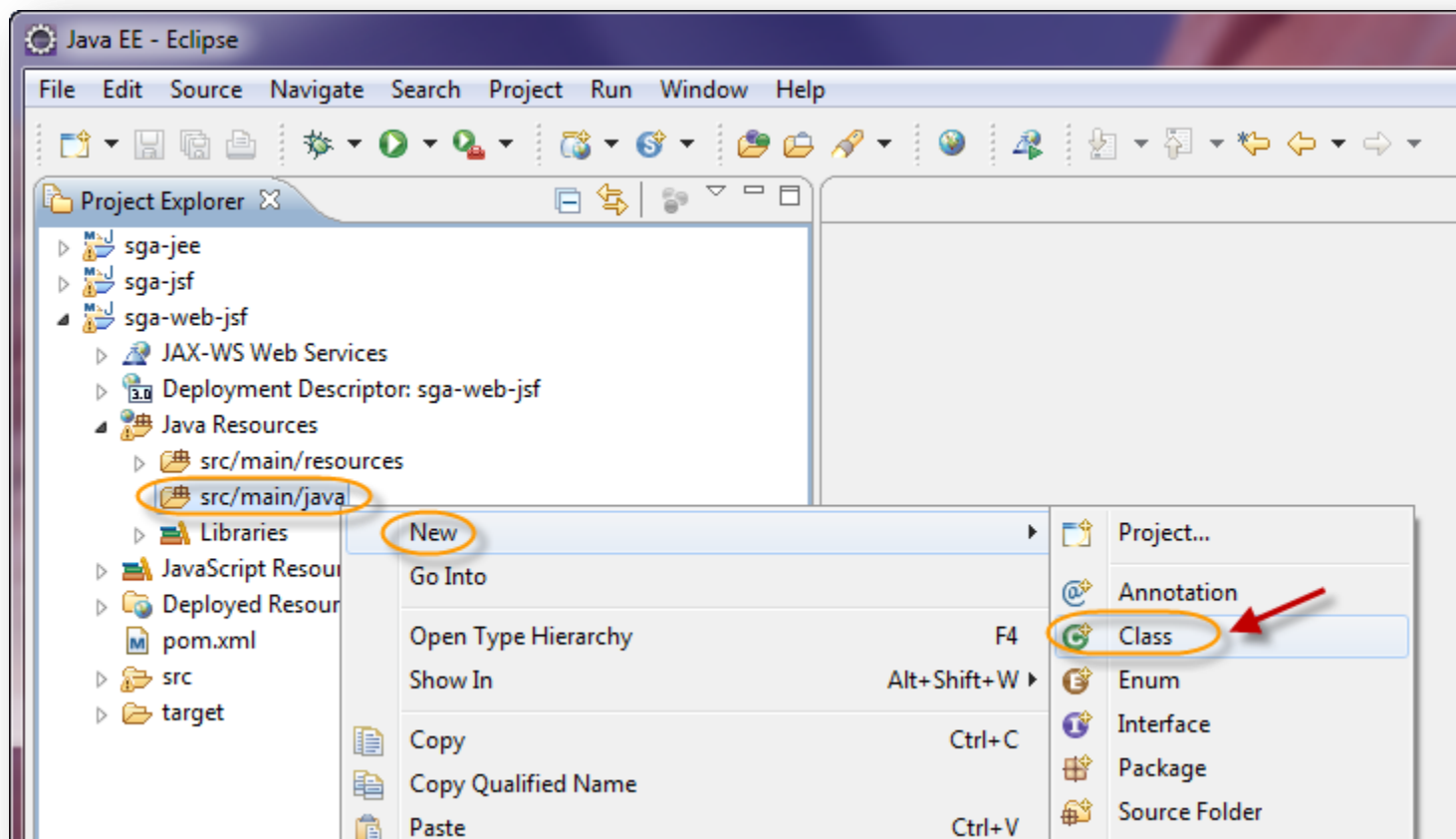
De los ejercicios de la lección 05, copiamos la carpeta resources y la pegamos en src/main/webapp/:





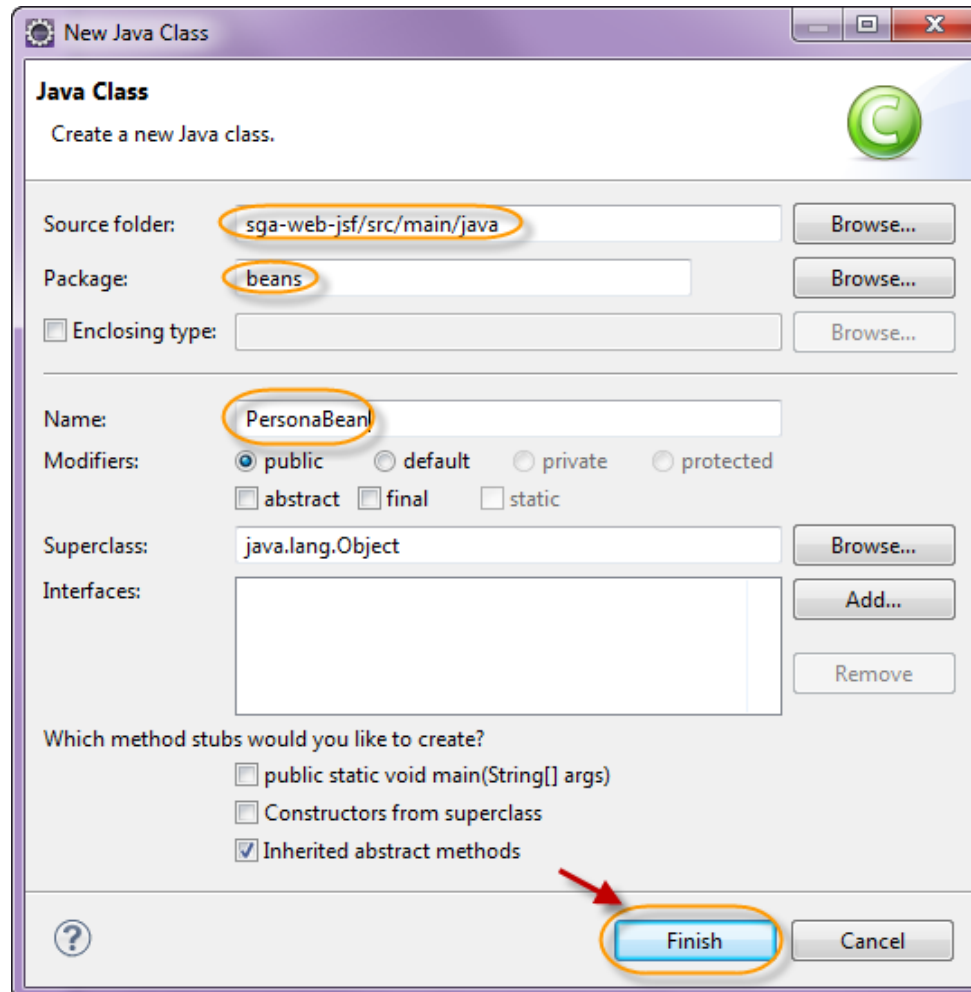
## Paso 11. Creamos la clase PersonaBean

Creamos la clase PersonaBean, la cual es un ManagedBean de JSF:



## Paso 11. Creamos la clase PersonaBean (cont)

Creamos la clase PersonaBean, la cual es un ManagedBean de JSF:





# Paso 11. Creamos la clase PersonaBean (cont)

Creamos la clase PersonaBean, la cual es un ManagedBean de JSF:

```
package beans;

import java.util.List;
import javax.annotation.PostConstruct;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import org.primefaces.event.RowEditEvent;
import mx.com.gm.sga.domain.Persona;
import mx.com.gm.sga.servicio.PersonaService;

@ManagedBean
@RequestScoped
public class PersonaBean {

    @EJB
    private PersonaService personaService;

    List<Persona> personas;

    public PersonaBean() { }

    @PostConstruct
    public void inicializar() {
        personas = personaService.listarPersonas();
    }

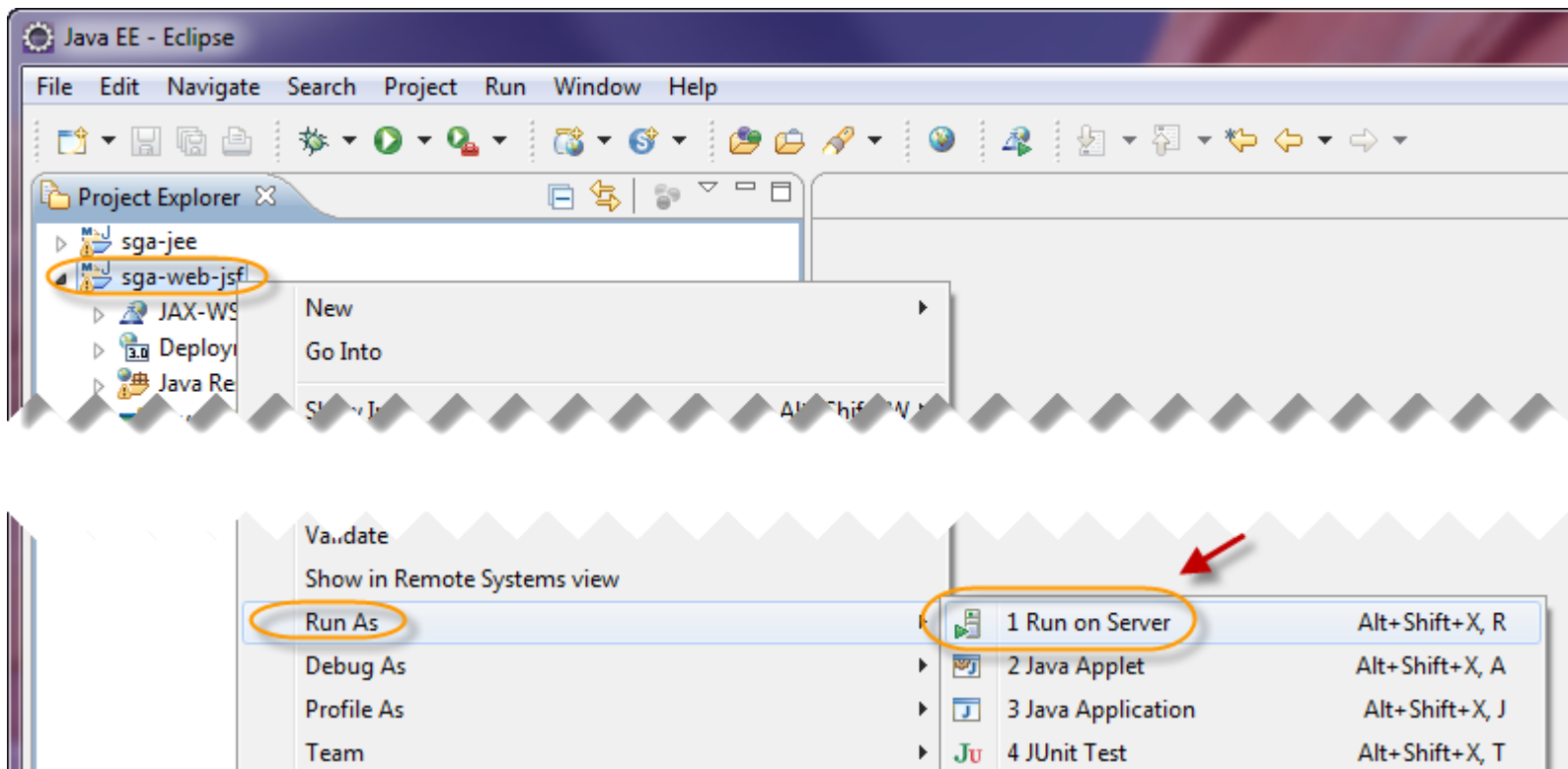
    public void editListener(RowEditEvent event) {
        Persona persona = (Persona) event.getObject();
        personaService.modificarPersona(persona);
    }

    public List<Persona> getPersonas() {
        return personas;
    }

    public void setPersonas(List<Persona> personas) {
        this.personas = personas;
    }
}
```

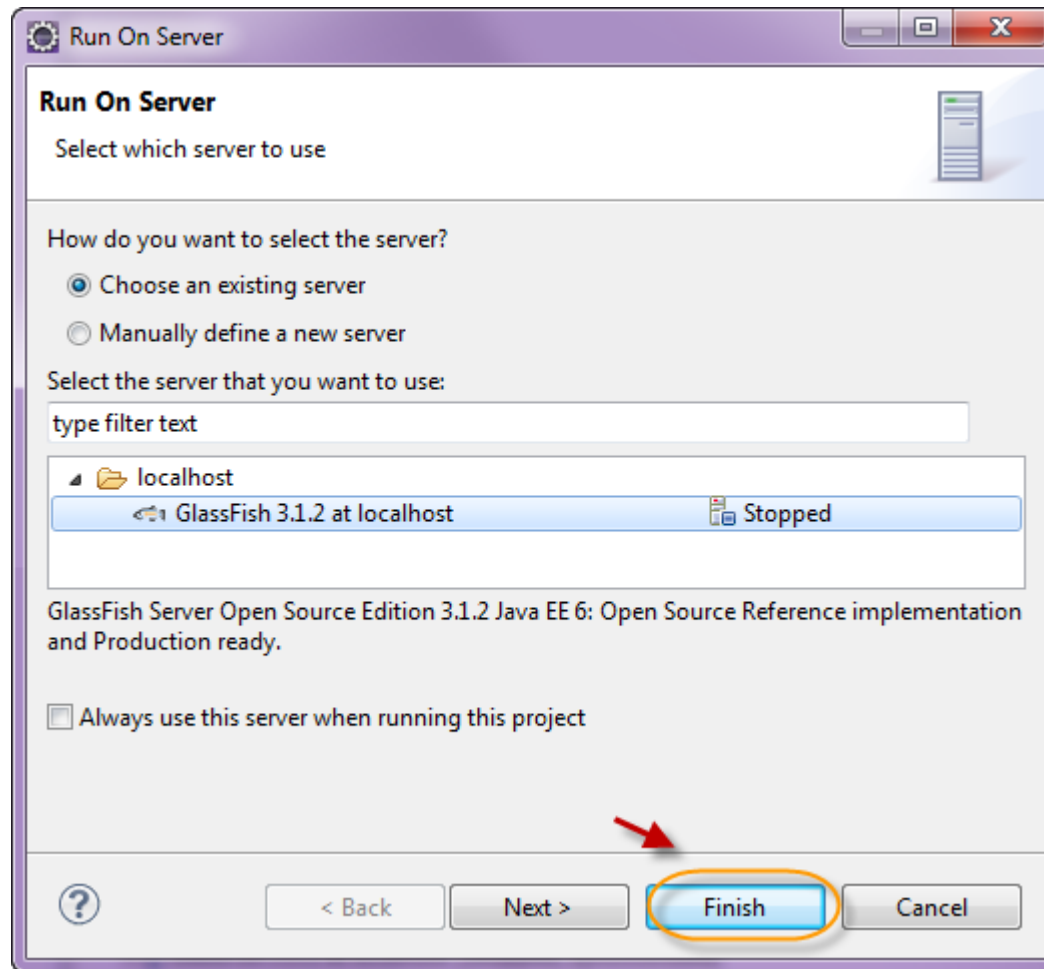
## Paso 12. Desplegar la aplicación sga-web-jsf

Ejecutamos el proyecto sga-web-jsf y lo desplegamos en GlassFish. **Nota: Si se encuentra desplegado otro proyecto que contenga los EJB a utilizar en este proyecto, es necesario hacer undeploy primero y después desplegar esta aplicación:**



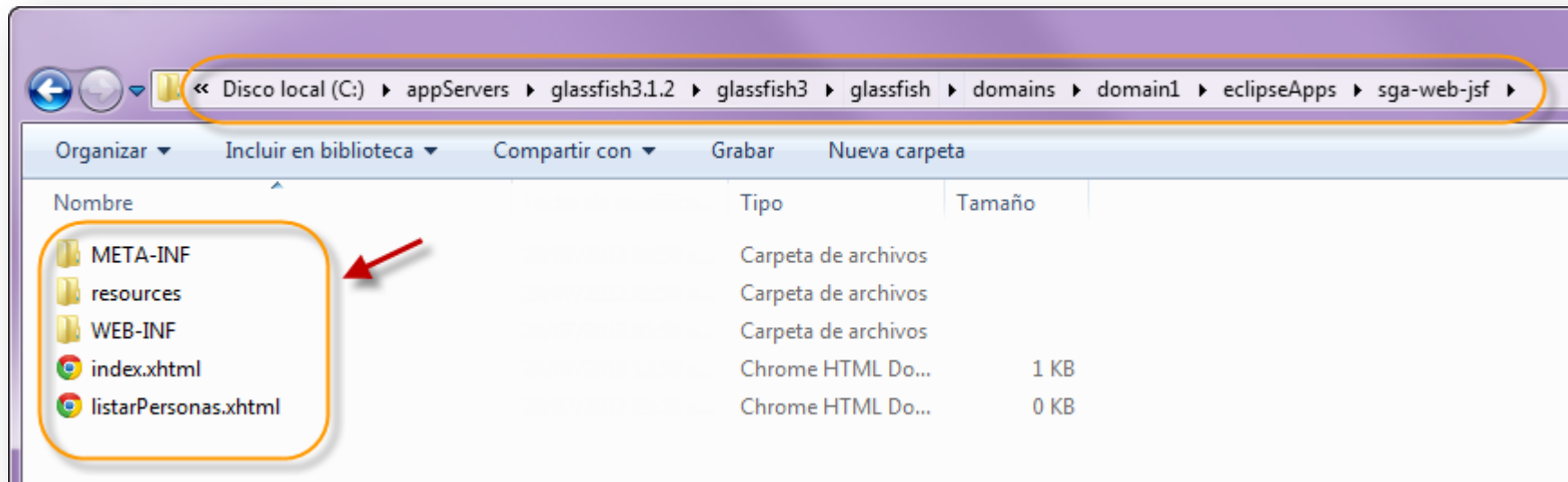
## Paso 12. Desplegar la aplicación sga-web-jsf (cont)

Ejecutamos el proyecto sga-web-jsf y lo desplegamos en GlassFish:



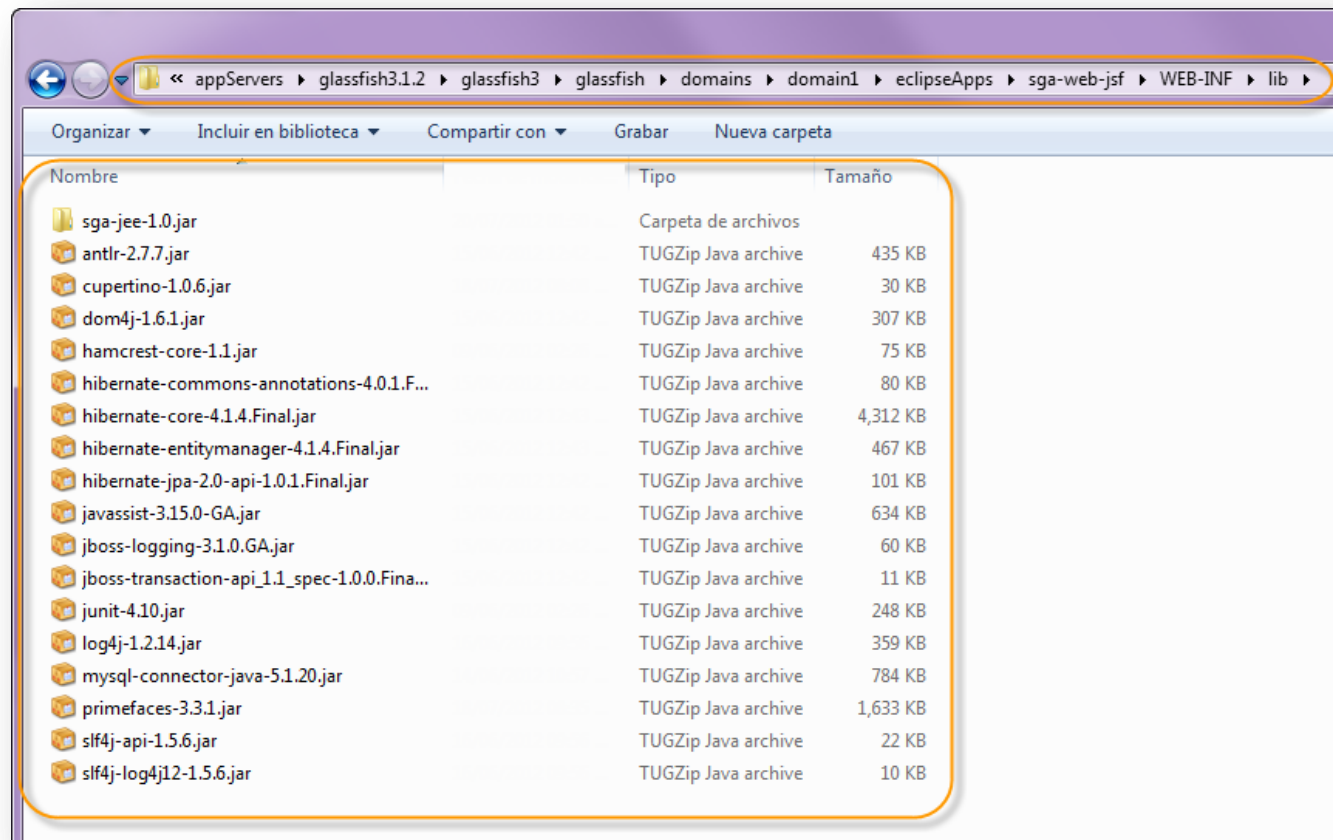
## Paso 12. Desplegar la aplicación sga-web-jsf (cont)

Verificamos que la ruta que nos proporcione Glassfish en la consola, permita ver el despliegue del proyecto y sus librerías:



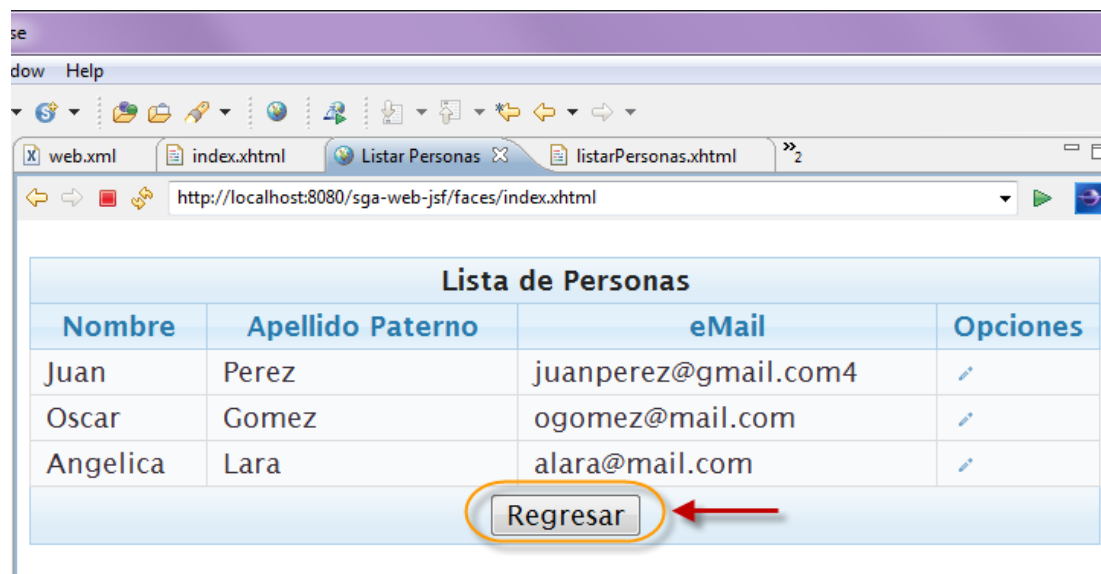
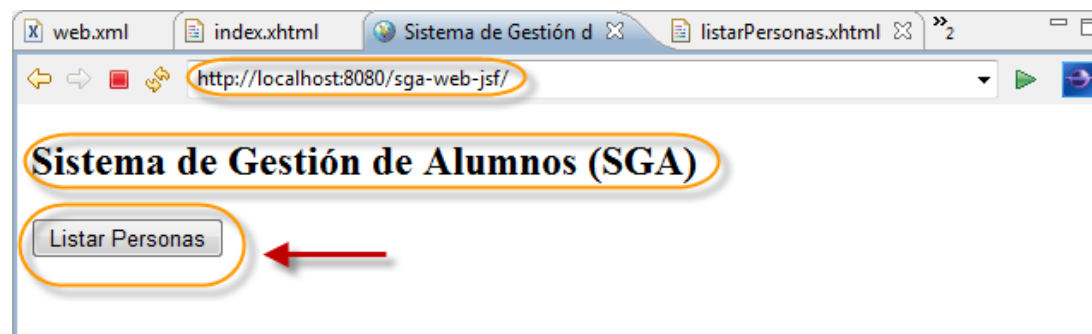
## Paso 12. Desplegar la aplicación sga-web-jsf (cont)

Con esto garantizamos que el proyecto se ha desplegado correctamente y con todas las dependencias:



## Paso 12. Desplegar la aplicación sga-web-jsf (cont)

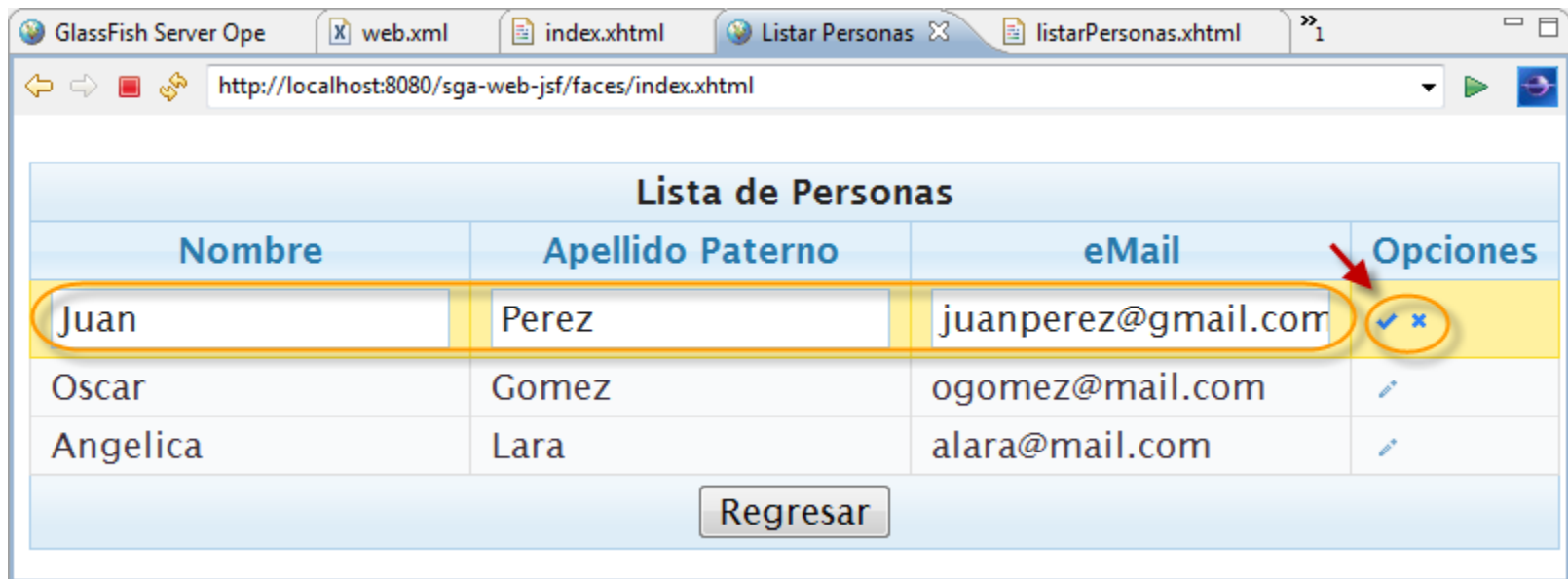
Ejecutamos el proyecto sga-web-jsf y lo desplegamos en GlassFish:





## Paso 12. Desplegar la aplicación sga-web (cont)

También es posible editar el línea el registro, debido a la funcionalidad de PrimeFaces que se ha agregado, además se observa el gif de ajaxloading.gif al guardar el registro y un look & feel personalizado. Estas son tan solo algunas mejoras que nos ofrece PrimeFaces.



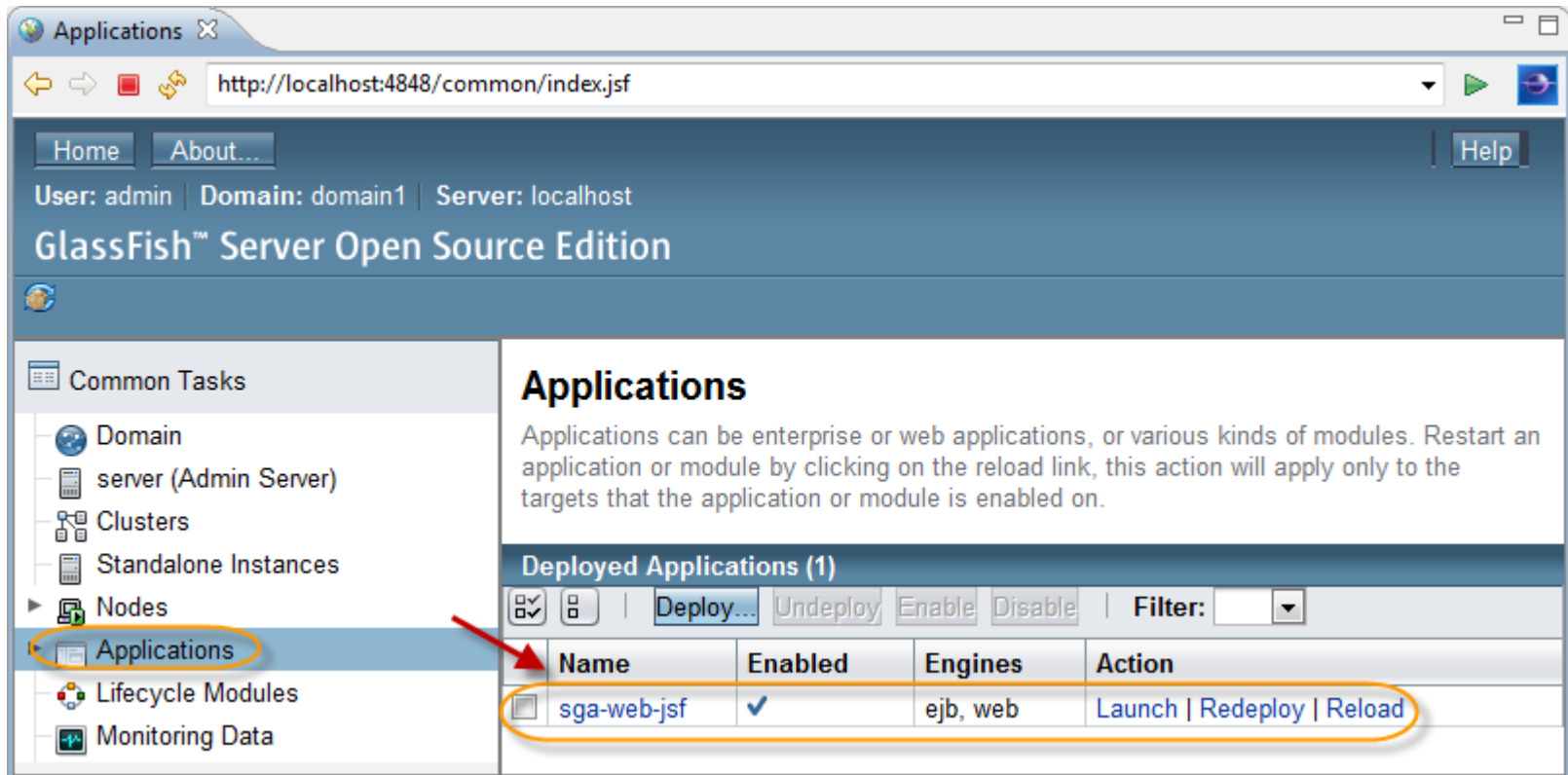
The screenshot shows a web browser window with the URL `http://localhost:8080/sga-web-jsf/faces/index.xhtml`. The page displays a table titled "Lista de Personas". The table has four columns: "Nombre", "Apellido Paterno", "eMail", and "Opciones". The first row is highlighted in yellow and contains the data: "Juan", "Perez", "juanperez@gmail.com", and a checkmark and an "x" icon. A red arrow points to the "Opciones" column of the first row, and a yellow circle highlights the "x" icon. The second row contains "Oscar", "Gomez", and "ogomez@mail.com". The third row contains "Angelica", "Lara", and "alara@mail.com". Below the table is a "Regresar" button.

Nombre	Apellido Paterno	eMail	Opciones
Juan	Perez	juanperez@gmail.com	✓ x
Oscar	Gomez	ogomez@mail.com	✎
Angelica	Lara	alara@mail.com	✎

Regresar

## Paso 13. Verificar Componentes Desplegados

Verificamos en la consola de GlassFish que efectivamente se hayan desplegado los componentes web y ejb, verificándolo en la consola:



The screenshot shows the GlassFish Server Open Source Edition console. The left sidebar has a tree view with 'Applications' selected. The main content area is titled 'Applications' and contains a table of 'Deployed Applications (1)'. The table has columns: Name, Enabled, Engines, and Action. The application 'sga-web-jsf' is listed with 'Enabled' checked and 'Engines' as 'ejb, web'. The 'Action' column for this application shows 'Launch | Redeploy | Reload'. A red arrow points to the 'Applications' tab in the sidebar, and a yellow circle highlights the 'sga-web-jsf' row in the table.

Name	Enabled	Engines	Action
sga-web-jsf	✓	ejb, web	Launch   Redeploy   Reload

## Paso 13. Verificar Componentes Desplegados (cont)

Verificamos en la consola de GlassFish que efectivamente se hayan desplegado los componentes web y ejb:

The screenshot shows the GlassFish Administration Console. On the left, the 'Applications' tree is expanded, showing 'sga-web-jsf'. The main panel is titled 'Edit Application' and has two tabs: 'General' and 'Descriptor'. The 'General' tab is active, showing the following details:

- Name:** sga-web-jsf
- Status:** ☒ Enabled
- Description:** (empty text field)
- Location:** \${com.sun.aas.instanceRootURI}/eclipseApps/sga-web-jsf/
- Libraries:** (empty list)

Below the details, there is a table titled 'Modules and Components (8)' which lists the deployed components:

Module Name	Engines	Component Name	Type	Action
sga-web-jsf	[ejb, jpa, web]	-----	-----	Launch
sga-web-jsf		PersonaDaolImpl	StatelessSessionBean	
sga-web-jsf		Faces Servlet	Servlet	
sga-web-jsf		default	Servlet	
sga-web-jsf		jsp	Servlet	
sga-web-jsf		UsuarioServiceImpl	StatelessSessionBean	
sga-web-jsf		PersonaServiceImpl	StatelessSessionBean	
sga-web-jsf		UsuarioDaolImpl	StatelessSessionBean	



## Ejercicio y Conclusión

Se deja como ejercicio agregar los casos de Agregar y Eliminar una persona. En caso de requerir apoyo se les incluye la solución en los ejercicios resueltos de la lección 5.

### Conclusión:

Con este ejercicio pudimos observar cómo integrar la capa Web con la capa de Servicio, esta última ya estaba integrada con la capa de datos. De esta manera hemos integrado las 3 capas en una sola aplicación.

La tecnología JSF, en combinación de AJAX y extensiones como PrimeFaces, iceFaces o RichFaces permiten mejorar rápidamente nuestras aplicaciones Web y Empresariales, además la configuración e integración con los EJB se ha simplificado enormemente, al grado de poder eliminar el archivo faces-config.xml. En la versión JSF 2.0 todavía no es posible prescindir del archivo web.xml ya que contiene la configuración de JSF, sin embargo en versiones futuras será opcional el uso de este archivo.

Este proyecto ya es una excelente base para sus propios proyectos Web y Empresariales utilizando las tecnologías JSF/AJAX/PrimeFaces + EJB + JPA.



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

*Pasión por la tecnología Java*

Experiencia y Conocimiento para tu vida