



## Ejercicio 9

Aplicación Web con Servlets, EJB y JPA

## Objetivo del Ejercicio

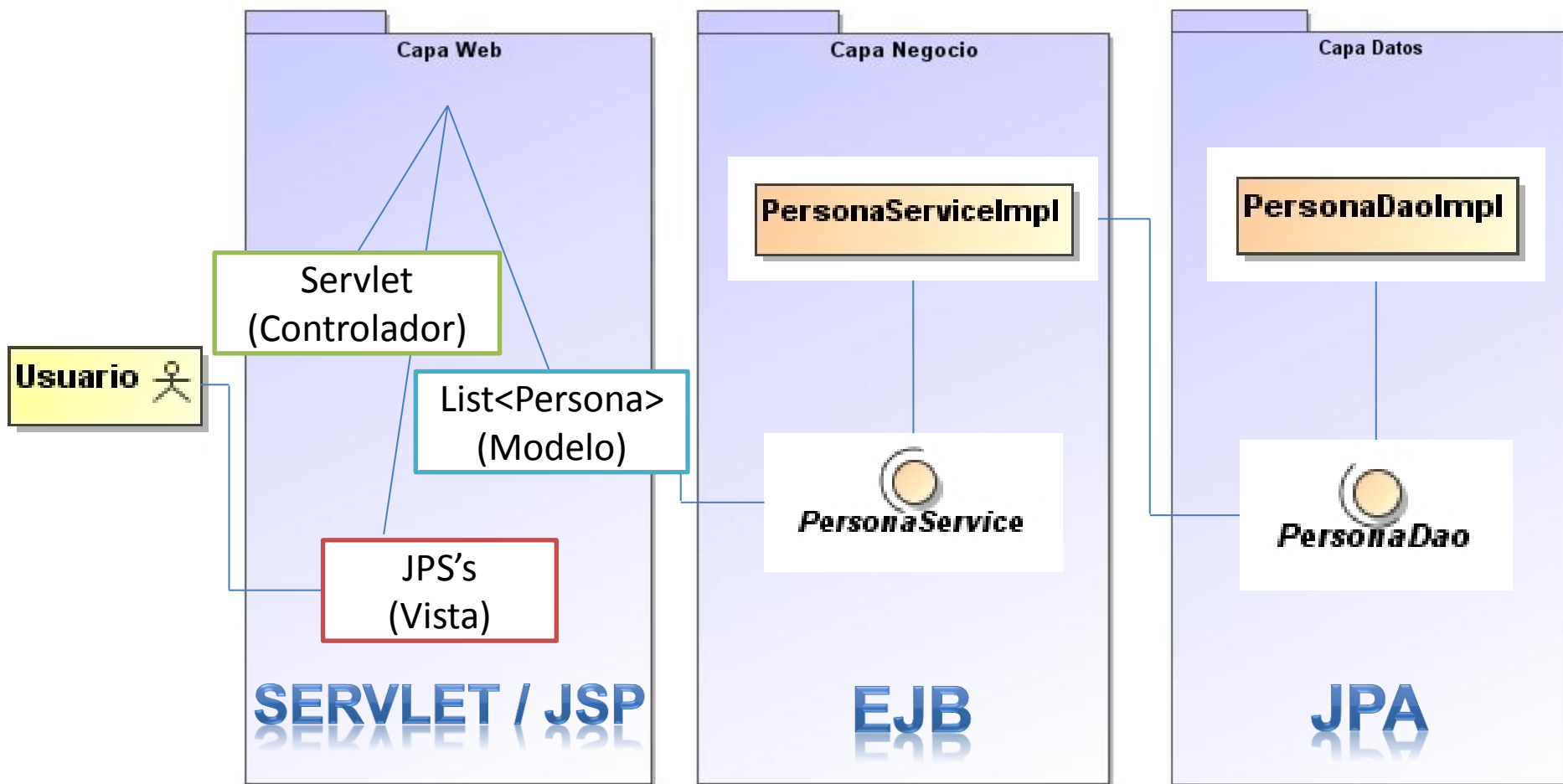
- El objetivo del ejercicio crear una aplicación Web que Listar las Personas utilizando la arquitectura de JSP, Servlets, EJB y JPA.

The screenshot shows the Eclipse IDE interface. The top bar indicates the project is 'sga-web' running on 'http://localhost:8080/sga-web/'. The left sidebar shows the project structure: 'sga-jee' > 'sga-web' > 'web' > 'Servlet.java'. The main editor area displays a JSP page titled 'listadoPersonas.jsp' with the URL 'http://localhost:8080/sga-web/Servlet'. The page content is a table titled 'Listado de Personas' with three columns: 'Nombre', 'Apellido Paterno', and 'Email'. The table contains five rows of data.

Nombre	Apellido Paterno	Email
Juan	Perez	juanperez@gmail.com
Oscar	Gomez	ogomez@mail.com
Angelica	Lara	alara@mail.com
Hugo	Sanchez	hsanchez@mail.com

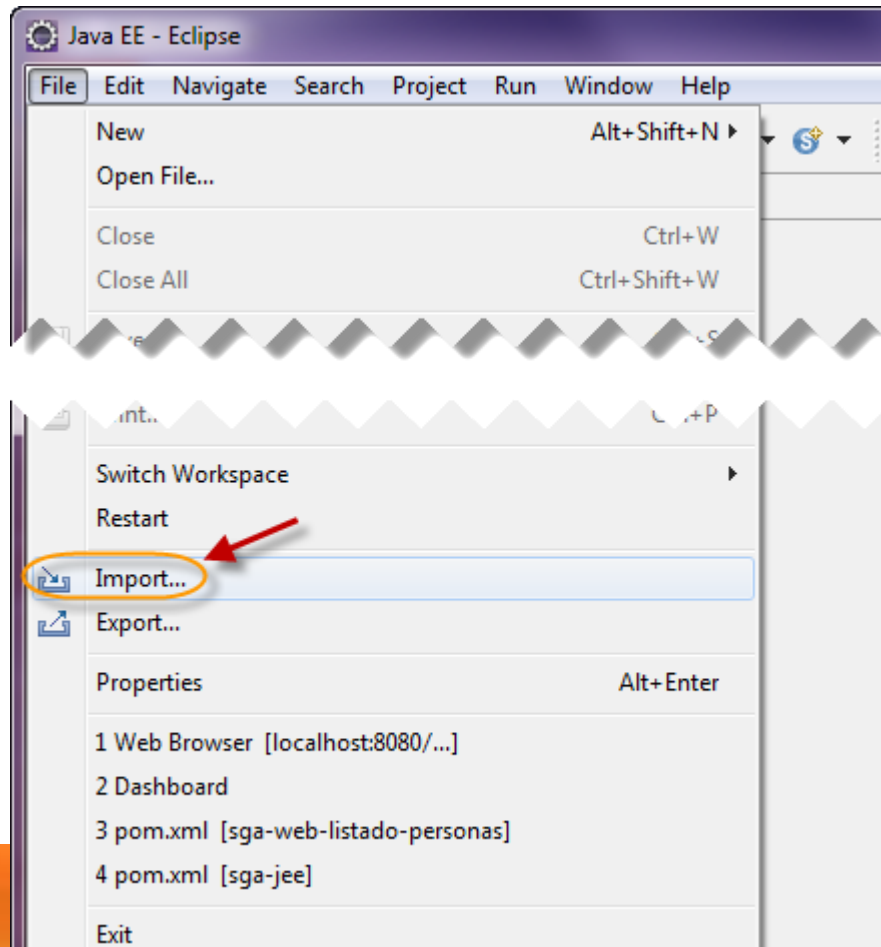
## Diagrama de Clases

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



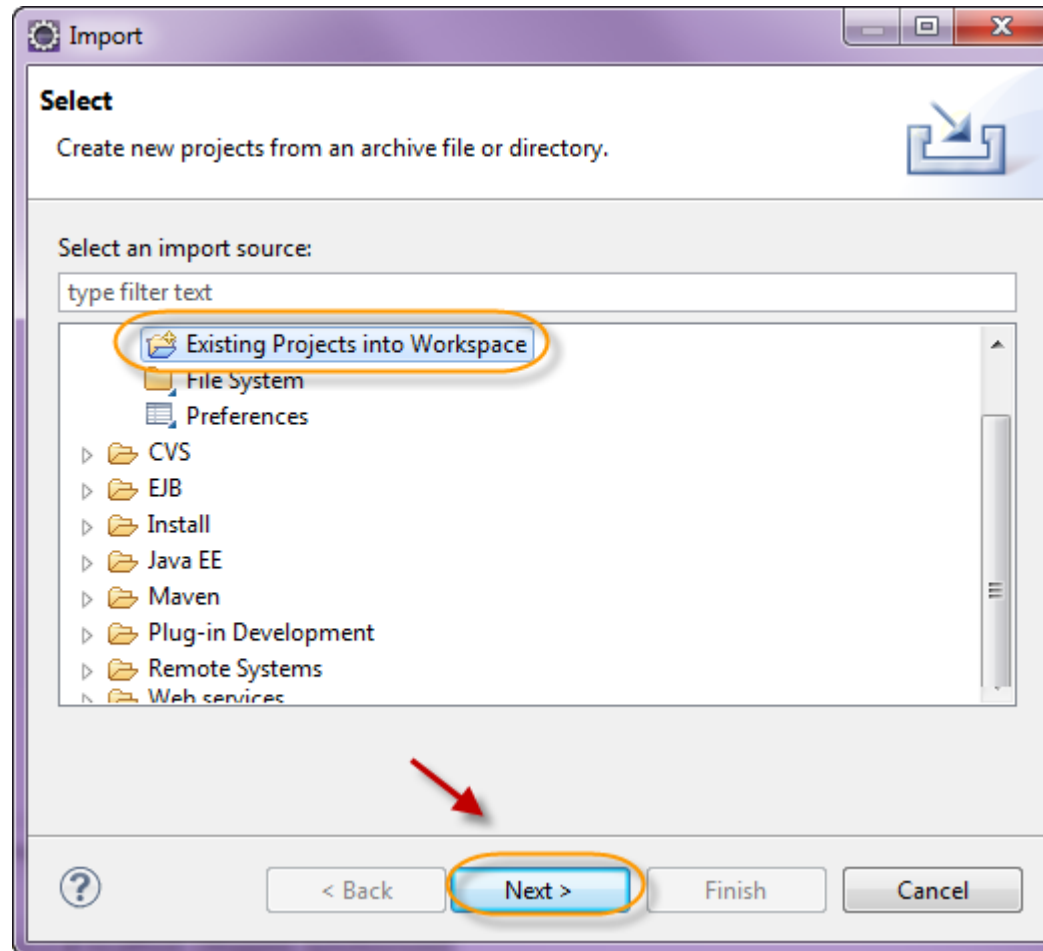
## Paso 1. Cargar el proyecto sga-jee

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. Si ya se tiene completo y cargado, se puede omitir este paso:



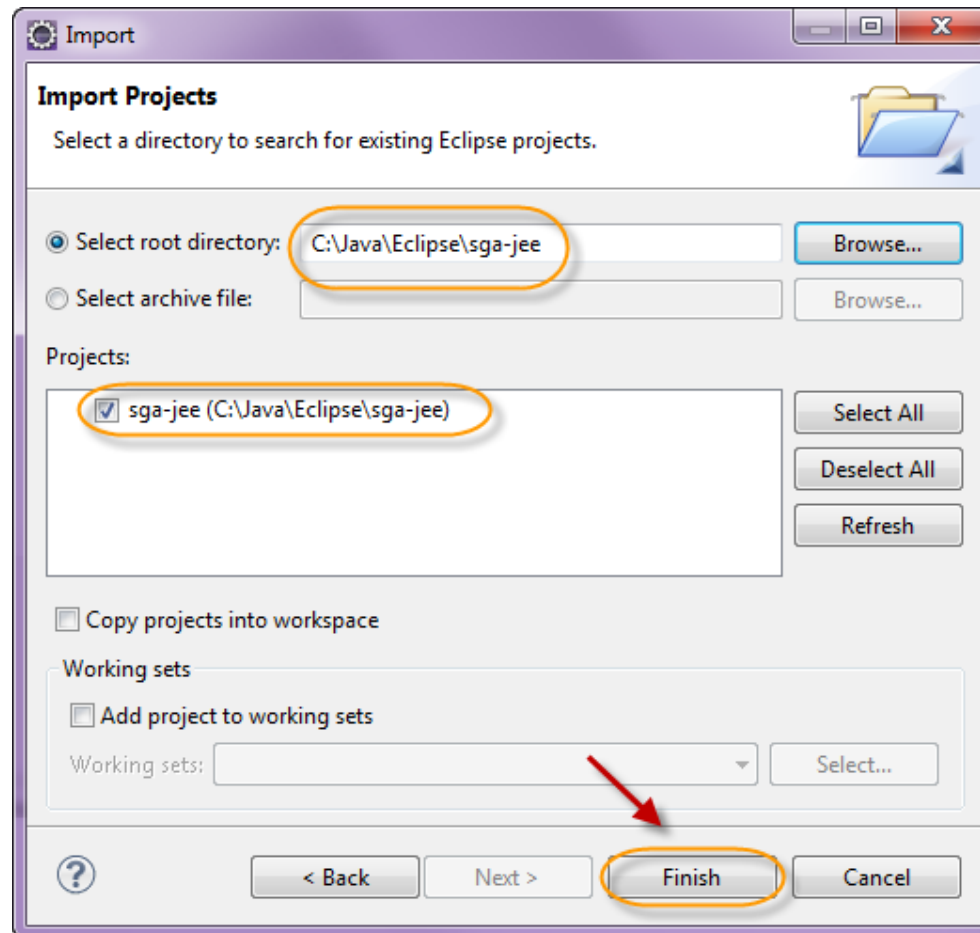
## Paso 1. Cargar el proyecto sga-jee (cont)

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



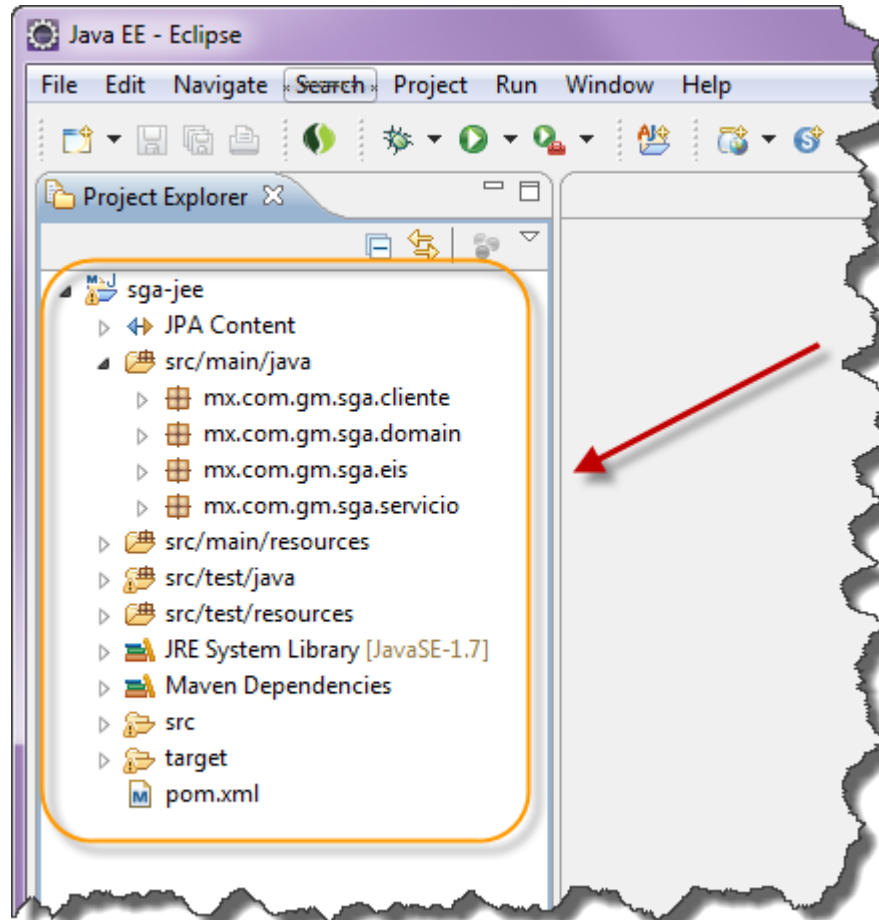
## Paso 1. Cargar el proyecto sga-jee (cont)

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)

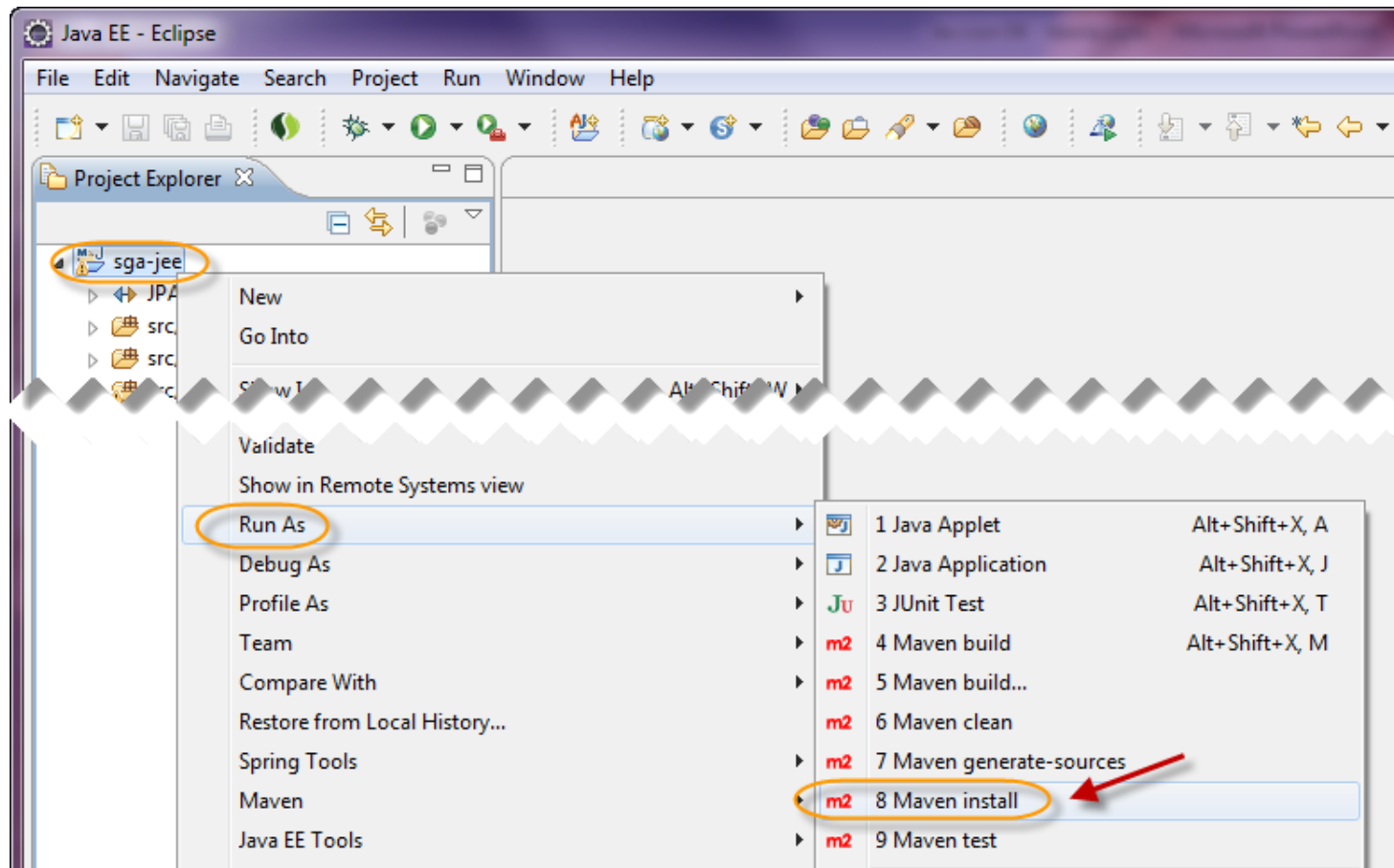
En caso de estar trabajando con el proyecto de la lección 3, se deberán desactivar TODAS las pruebas unitarias:

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



## 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.





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

Deberemos obtener un resultado similar al siguiente:

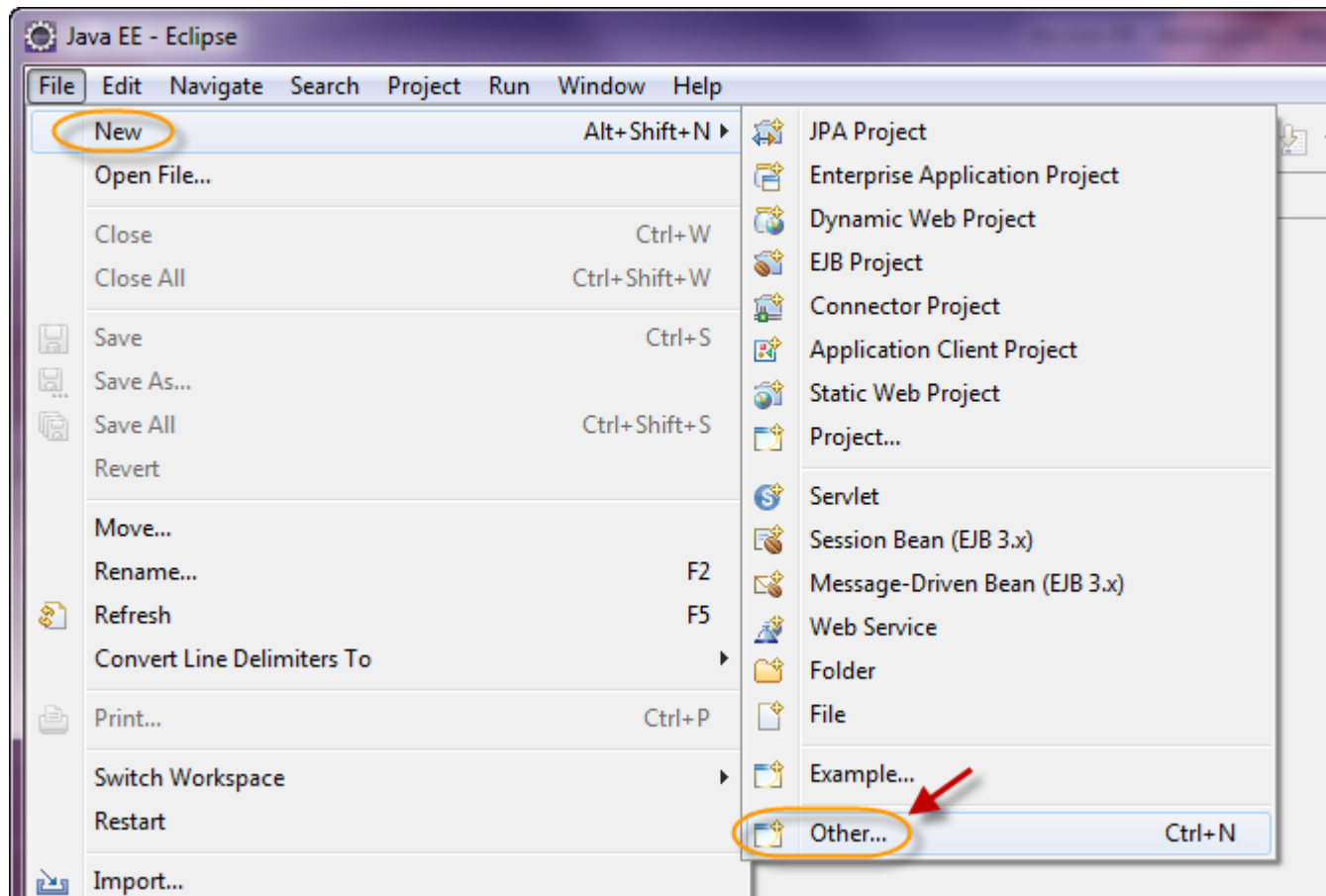
```
[INFO] --- maven-install-plugin:2.3.1:install (default-install) @ sga-jee ---  
[INFO] Installing C:\Java\Eclipse\sga-jee\target\sga-jee-1.0.jar to \\vmware-host\Shared Folders\.m2\repository\mx\com\gm\sga\sga-jee-1.0.jar  
[INFO] Installing C:\Java\Eclipse\sga-jee\pom.xml to \\vmware-host\Shared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

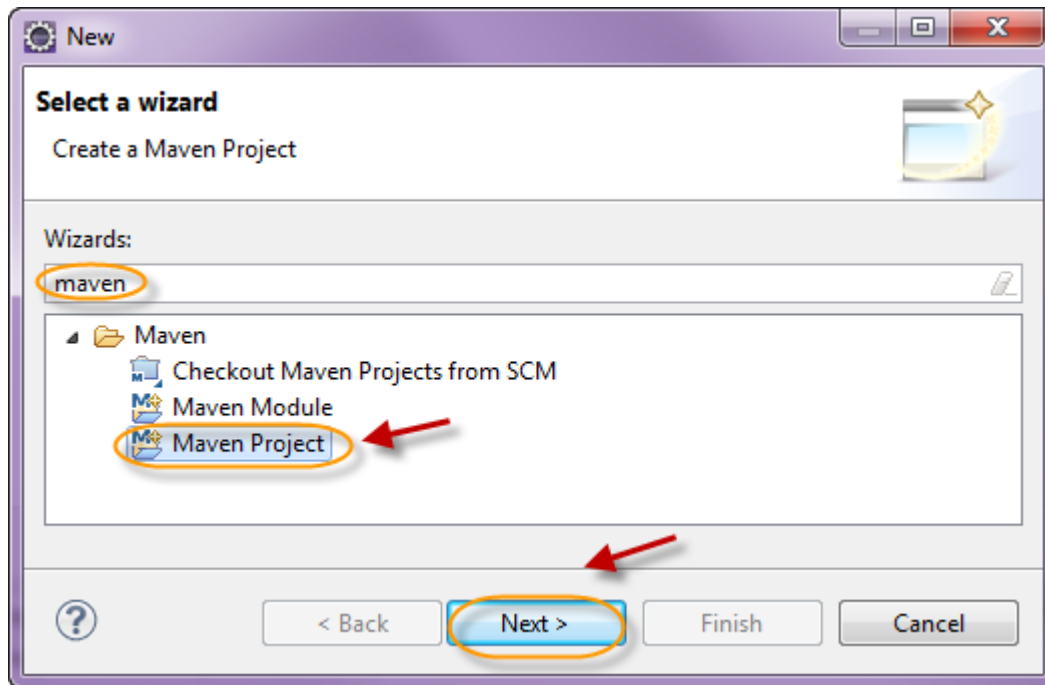
## Paso 3. Creación del proyecto Java Web

Creamos el proyecto sga-web:



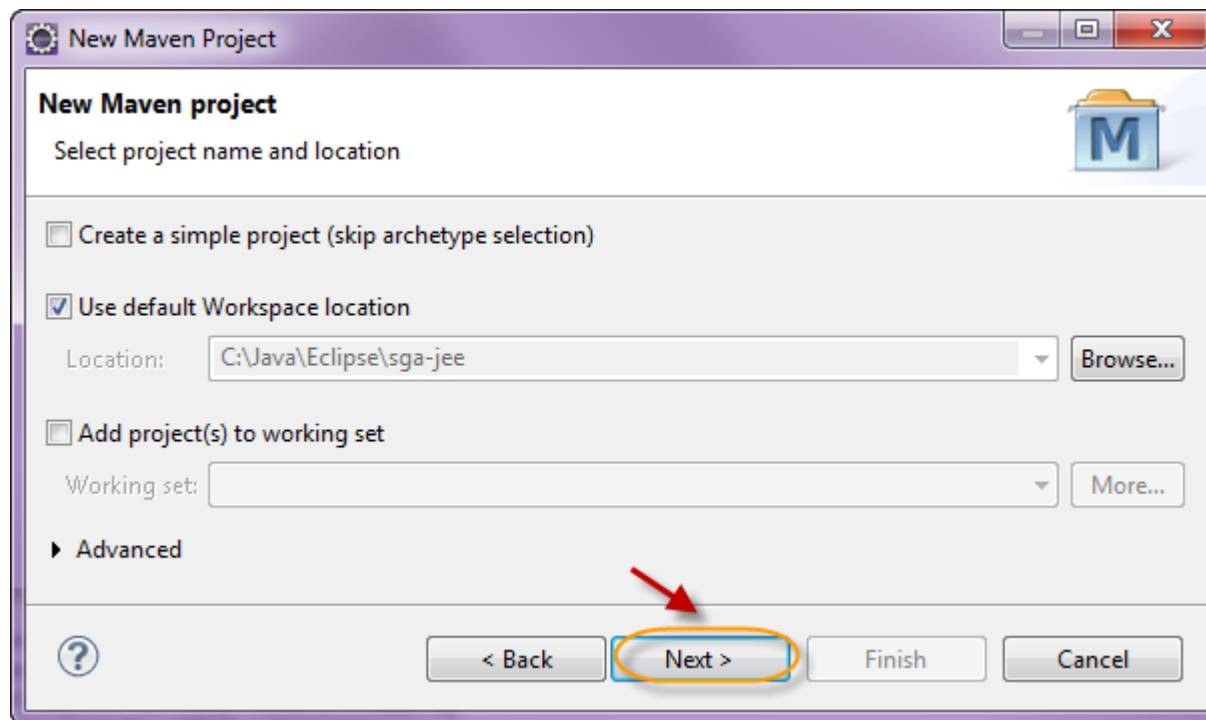
## Paso 3. Creación del proyecto Java Web (cont)

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



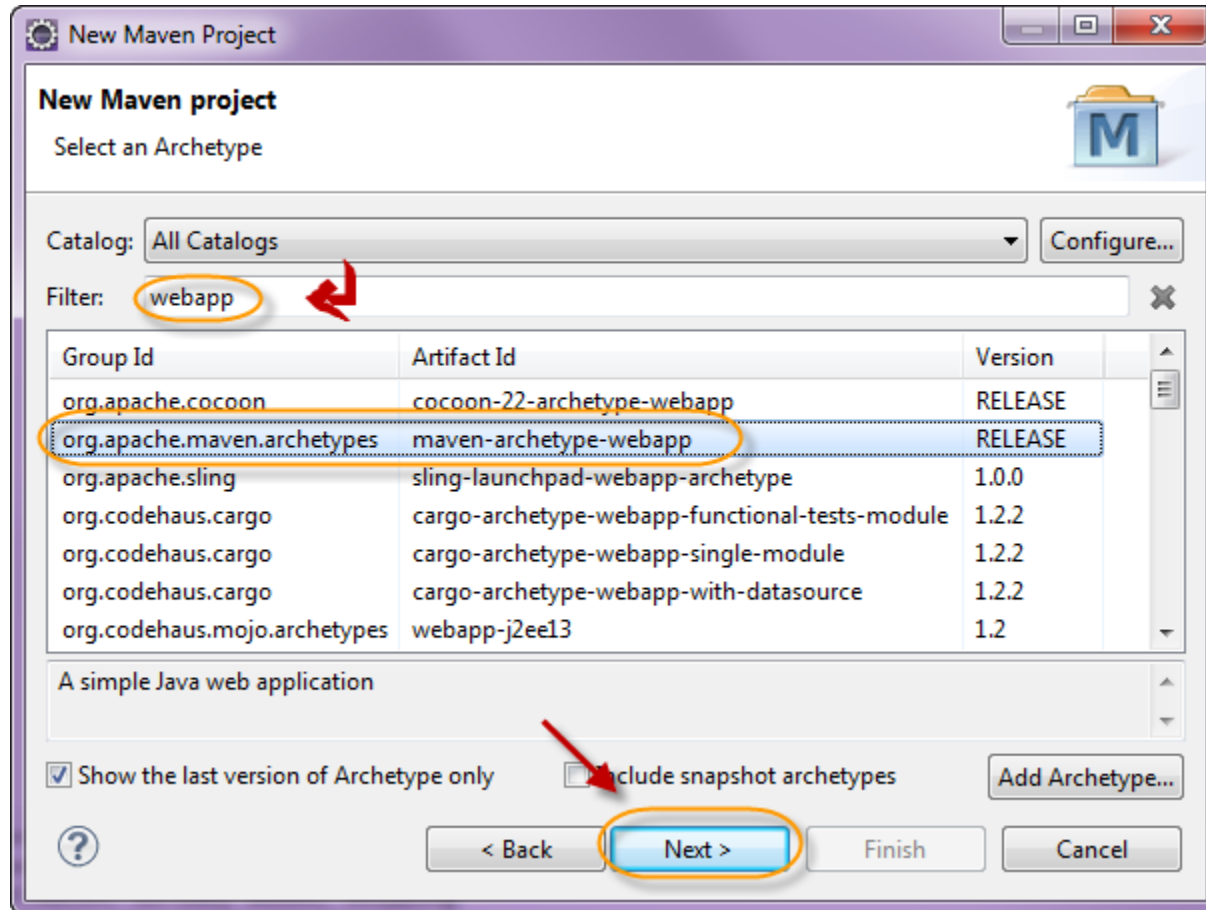
## Paso 3. Creación del proyecto Java Web (cont)

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



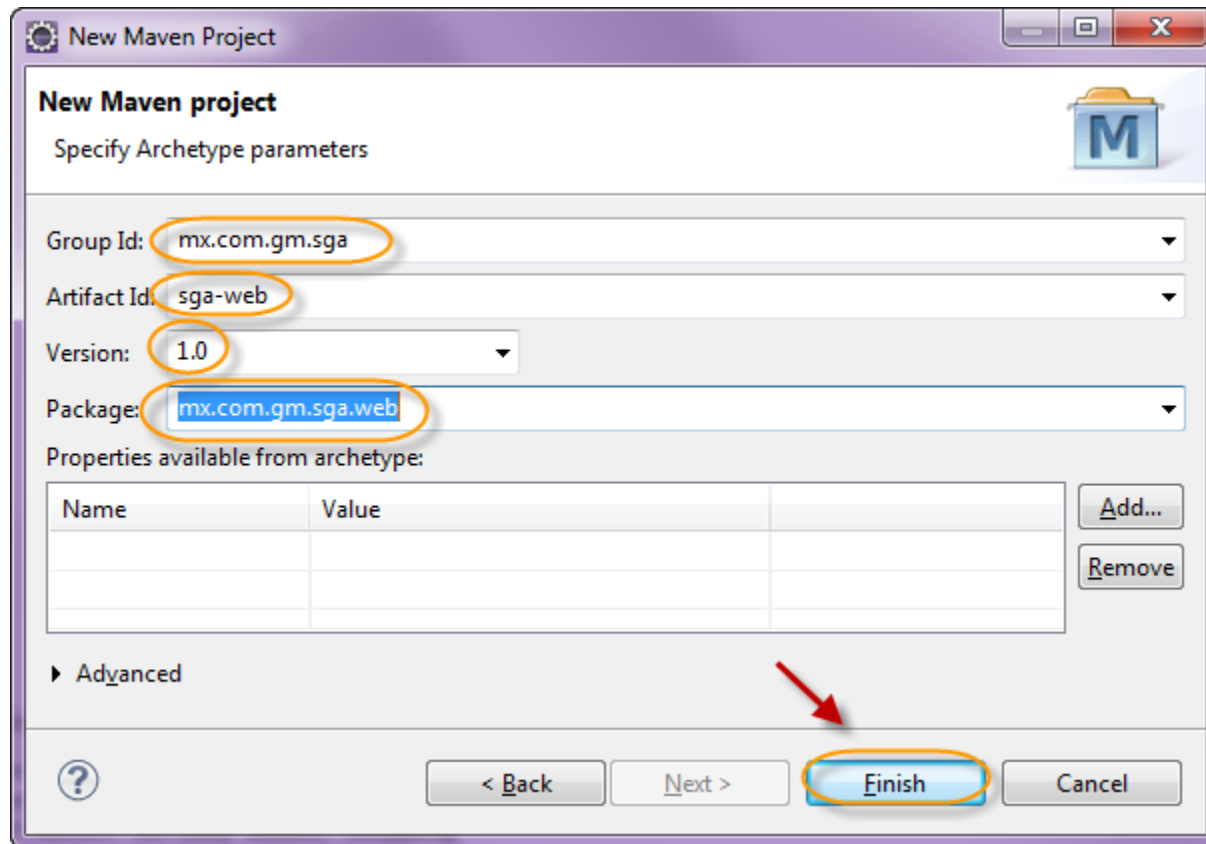
## Paso 3. Creación del proyecto Java Web (cont)

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



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

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



**New Maven Project**

Specify Archetype parameters

Group Id:

Artifact Id:

Version:

Package:

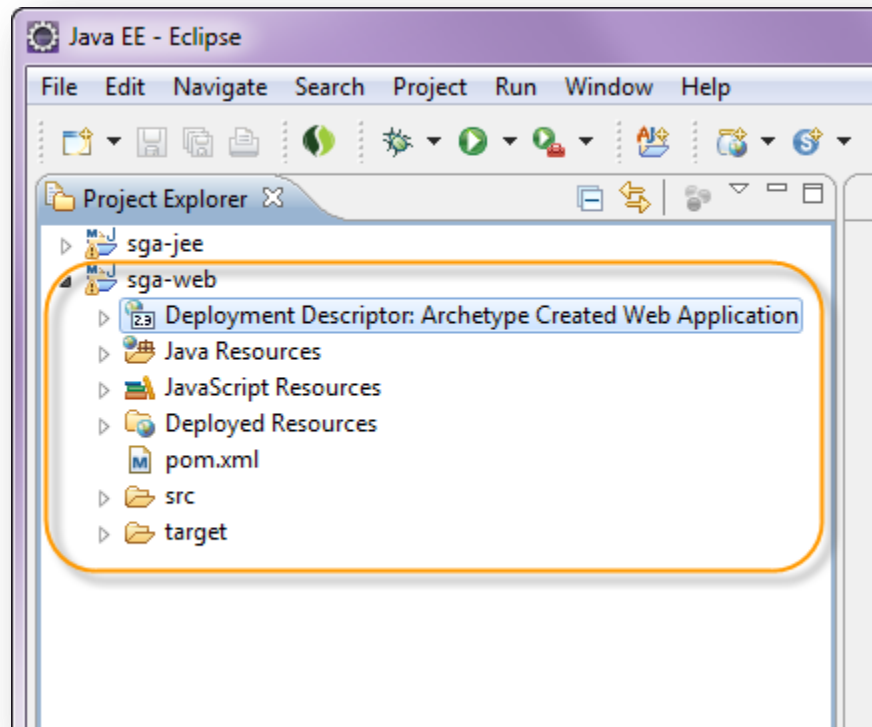
Properties available from archetype:

Name	Value

► Advanced

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

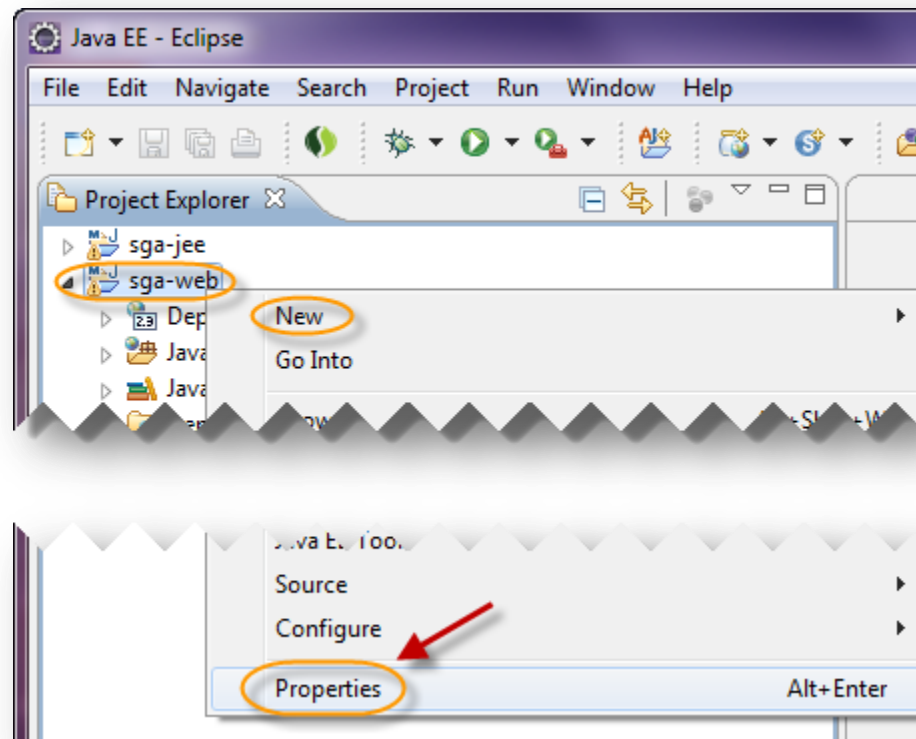
Confirmamos que la estructura creada sea la correcta:





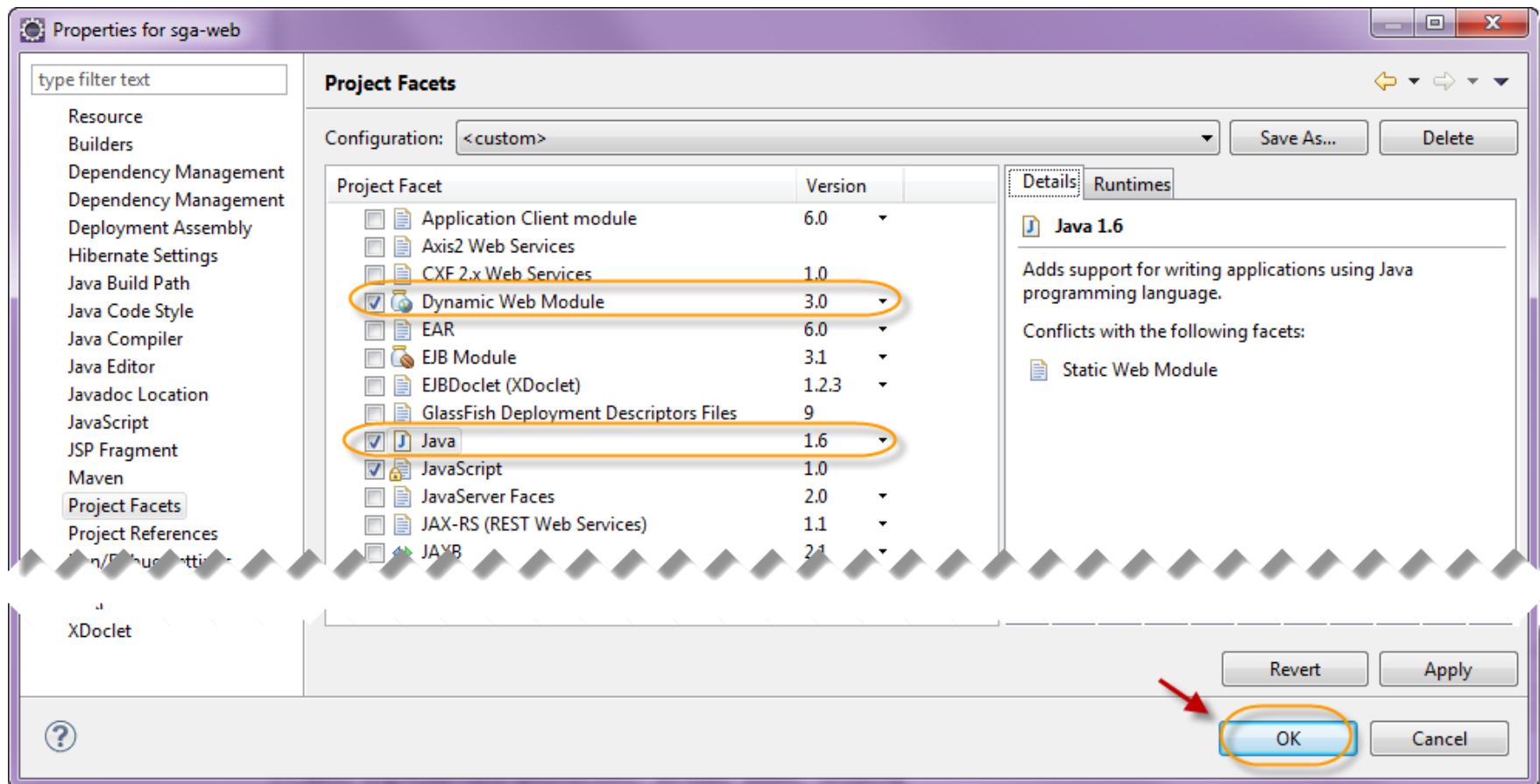
## Paso 4. Configuración del proyecto Java Web

Debido a que el arquetipo web de Maven crea un proyecto con una versión distinta, 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.



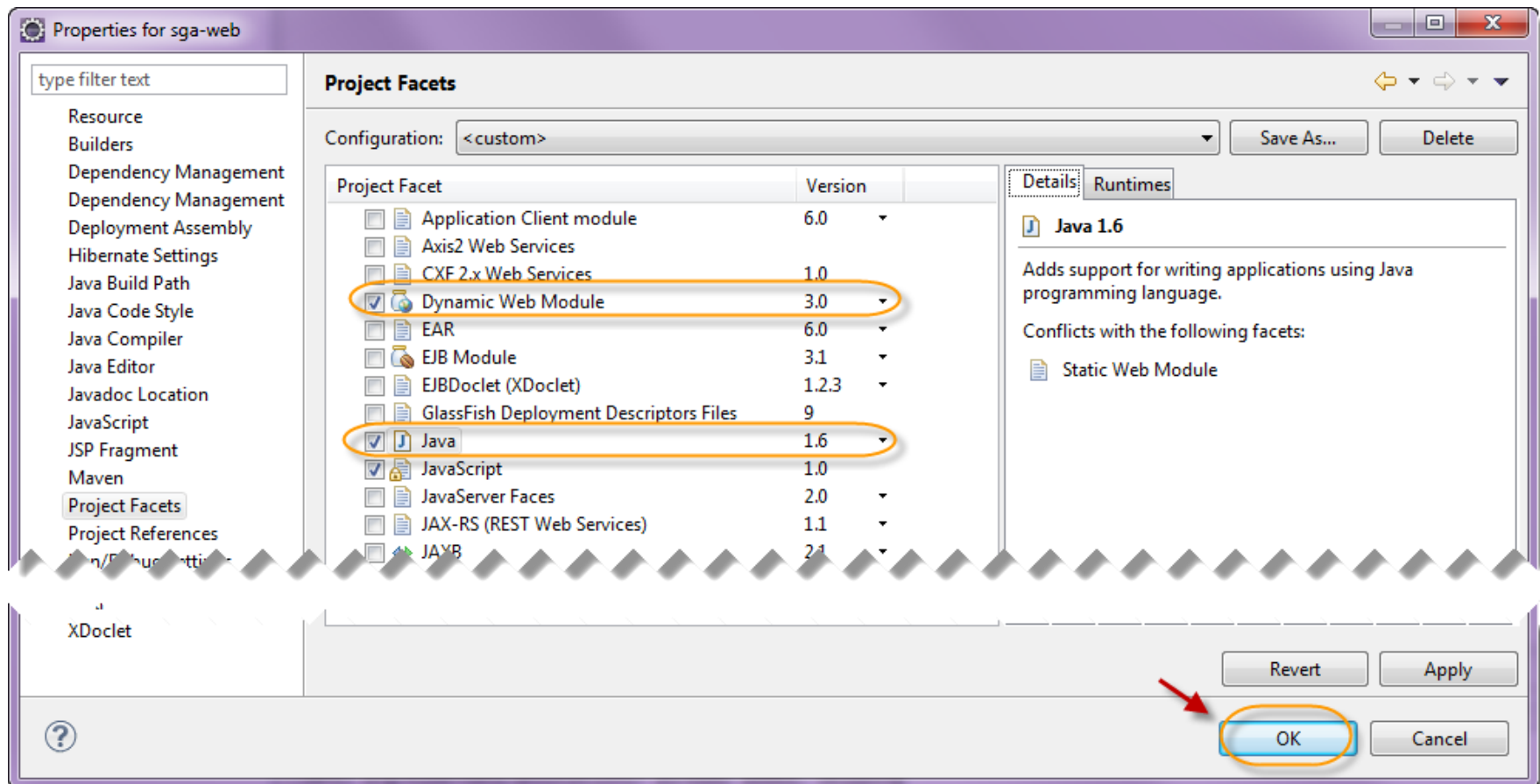
## Paso 4. Configuración del proyecto Java Web (cont)

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



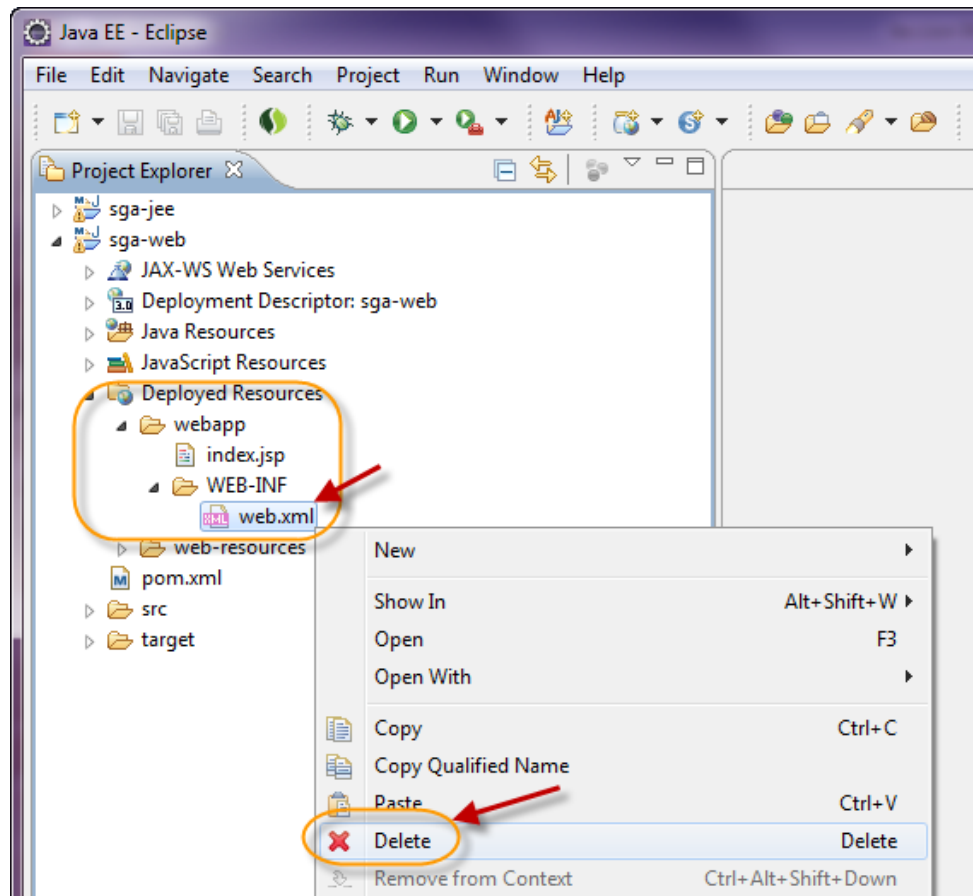
## Paso 4. Configuración del proyecto Java Web (cont)

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



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

Por último, eliminamos el archivo web.xml, ya que además de que tiene referenciada la versión 2.5, en la versión 3 es opcional, por ello lo eliminaremos.





## Paso 5. Agregamos las dependencias Maven

Agregamos las dependencias de Maven al proyecto Web, incluyendo la dependencia del proyecto sga-jee:

```
<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>
```

## Paso 5. Agregamos las dependencias Maven (cont)

Deberemos observar las nuevas dependencias, incluyendo las del proyecto sga-jee:

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays the project structure for 'sga-jee' and 'sga-web'. Under 'sga-web', the 'Maven Dependencies' folder is expanded, showing a list of JAR files. A red arrow points to the 'sga-jee' project at the bottom of the list. On the right, the 'sga-web/pom.xml' file is open, showing the XML configuration. A blue box highlights the dependency on 'sga-jee' (lines 17-21), and a purple arrow points from this dependency to the 'sga-jee' project in the Project Explorer. The pom.xml content is as follows:

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
3     http://maven.apache.org/maven-v4_0_0.xsd">
4   <groupId>mx.com.gm.sga</groupId>
5   <artifactId>sga-web</artifactId>
6   <packaging>war</packaging>
7   <version>1.0</version>
8   <name>sga-web Maven Webapp</name>
9   <url>http://maven.apache.org</url>
10  <dependencies>
11    <dependency>
12      <groupId>javax</groupId>
13      <artifactId>javaxee-api</artifactId>
14      <version>6.0</version>
15      <scope>provided</scope>
16    </dependency>
17    <dependency>
18      <groupId>mx.com.gm.sga</groupId>
19      <artifactId>sga-jee</artifactId>
20      <version>1.0</version>
21    </dependency>
22  </dependencies>
23  <build>
24    <finalName>sga-web</finalName>
25  </build>
26 </project>
```



## Paso 6. Modificar el index.jsp

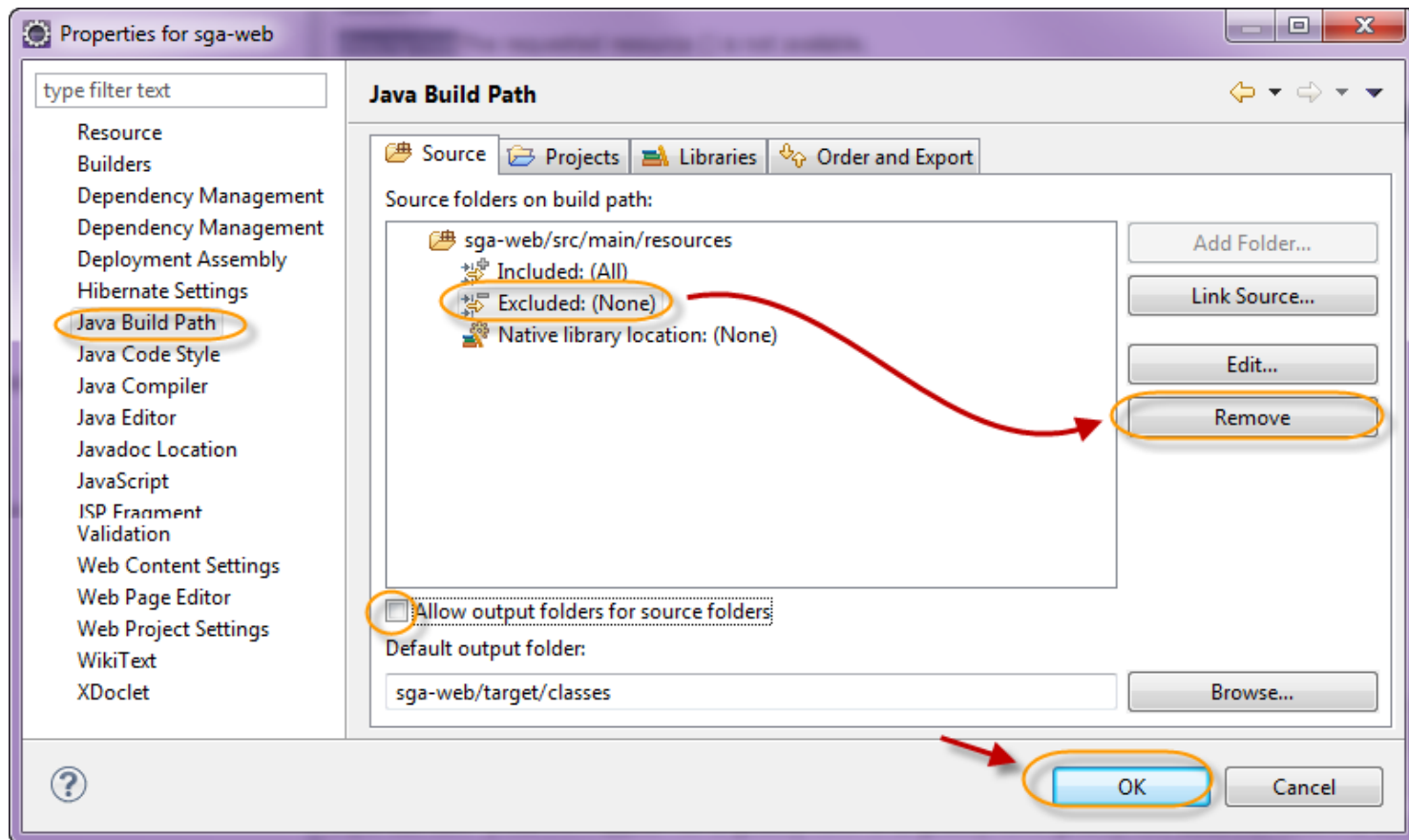
Sustituimos el código del archivo index.jsp ubicado en Deployed Resources/webpp:

```
<html>
  <body>
    <h2>Sistema de Gestión de Alumnos (SGA)</h2>
    <a href="ListarPersonas">Listado de Personas</a>
  </body>
</html>
```



## Paso 7. Creación del ServletControlador

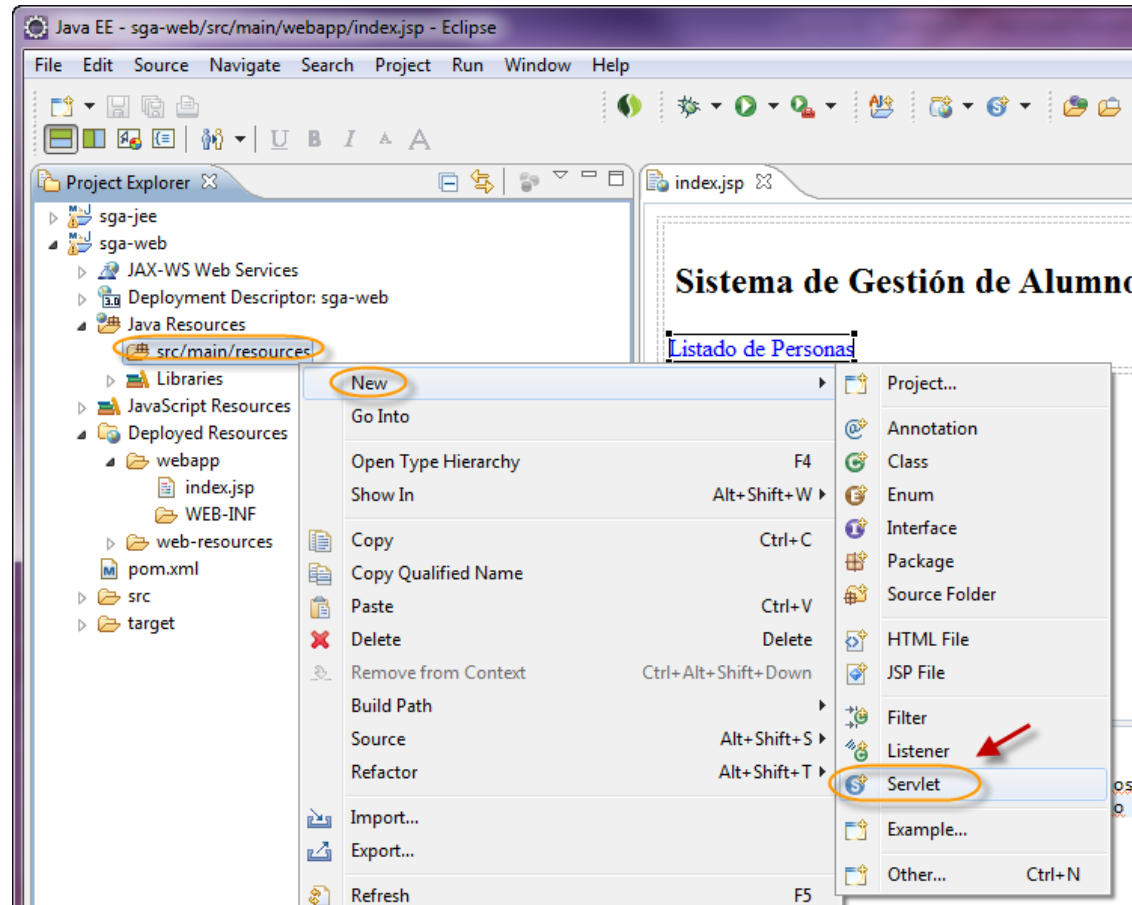
Antes de agregar el Servlet, debemos incluir todos los archivos de la carpeta de resources del proyecto. Entramos a las propiedades del proyecto e indicamos que NO excluya ningún tipo de archivo:





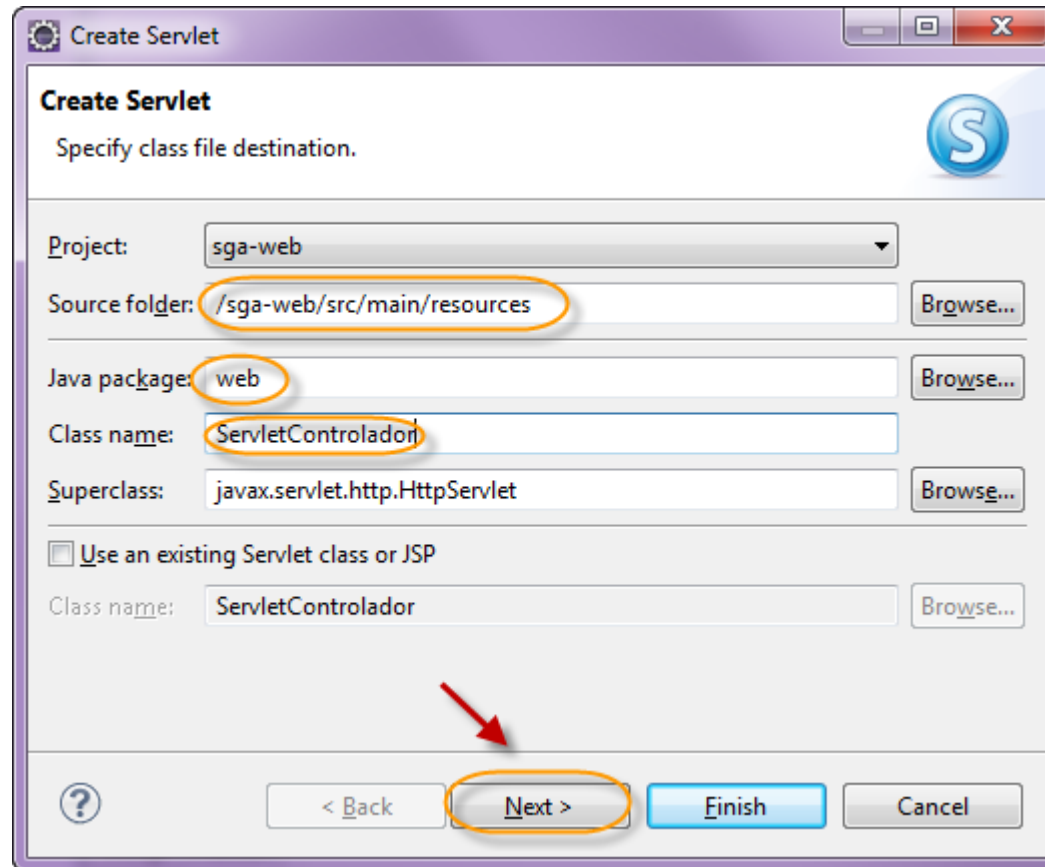
## Paso 7. Creación del ServletControlador (cont)

Creamos una clase llamada ServletControlador:



## Paso 7. Creación del Servlet Controlador (cont)

Creamos una clase llamada ServletControlador:



## Paso 7. Creación del Servlet Controlador (cont)

Creamos una clase llamada ServletControlador:

**Create Servlet**

Enter servlet deployment descriptor specific information.

Name: ServletControlador

Description:

Initialization parameters:

Name	Value	Description
------	-------	-------------

Add... Edit... Remove

URL mappings:

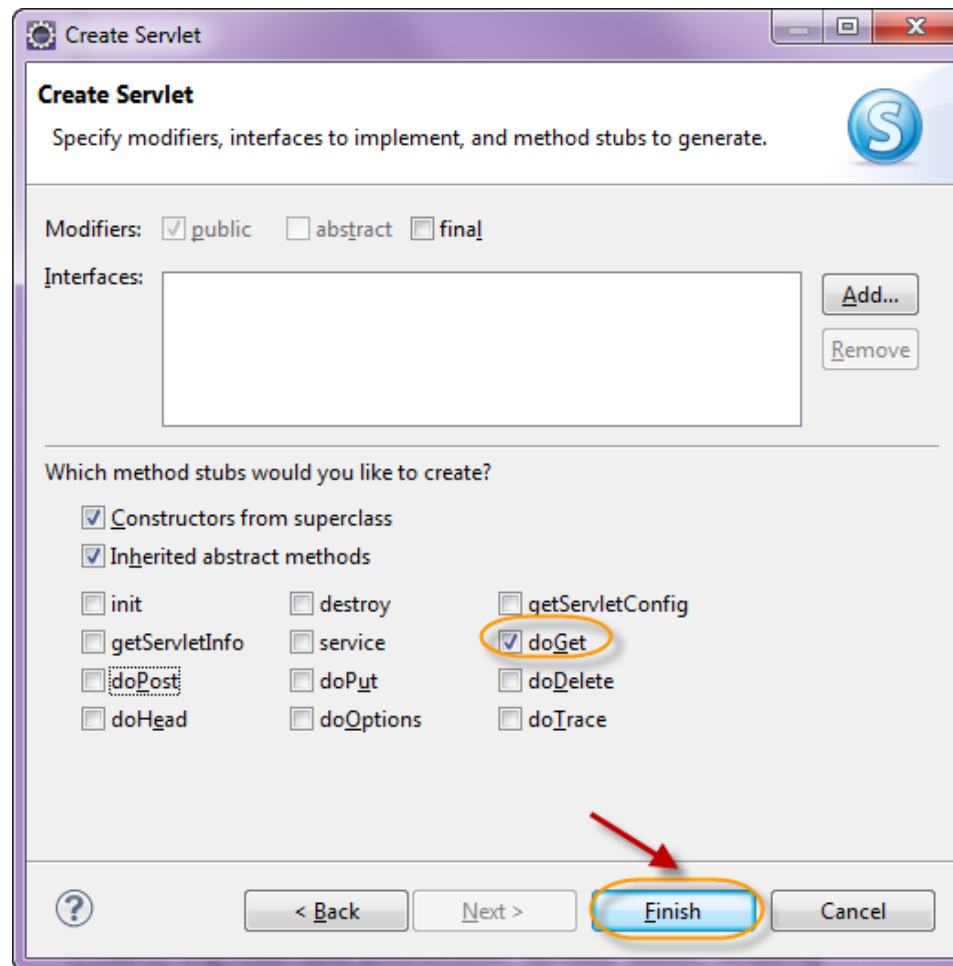
/ListarPersonas

Add... Edit... Remove

< Back **Next >** Finish Cancel

## Paso 7. Creación del Servlet Controlador (cont)

Creamos una clase llamada ServletControlador:



## Paso 7. Creación del Servlet Controlador (cont)

Reemplazar el código de la clase ServletControlador por el siguiente:

```
package web;
```

```
import java.io.IOException;
import java.util.List;
import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import mx.com.gm.sga.domain.Persona;
import mx.com.gm.sga.servicio.PersonaService;
```

```
@WebServlet("/ListarPersonas")
```

```
public class ServletControlador extends HttpServlet {
```

```
    private static final long serialVersionUID = 1L;
```

```
    @EJB
```

```
    private PersonaService personaService;
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
```

```
        List<Persona> personas = personaService.listarPersonas();
        request.setAttribute("personas", personas);
```

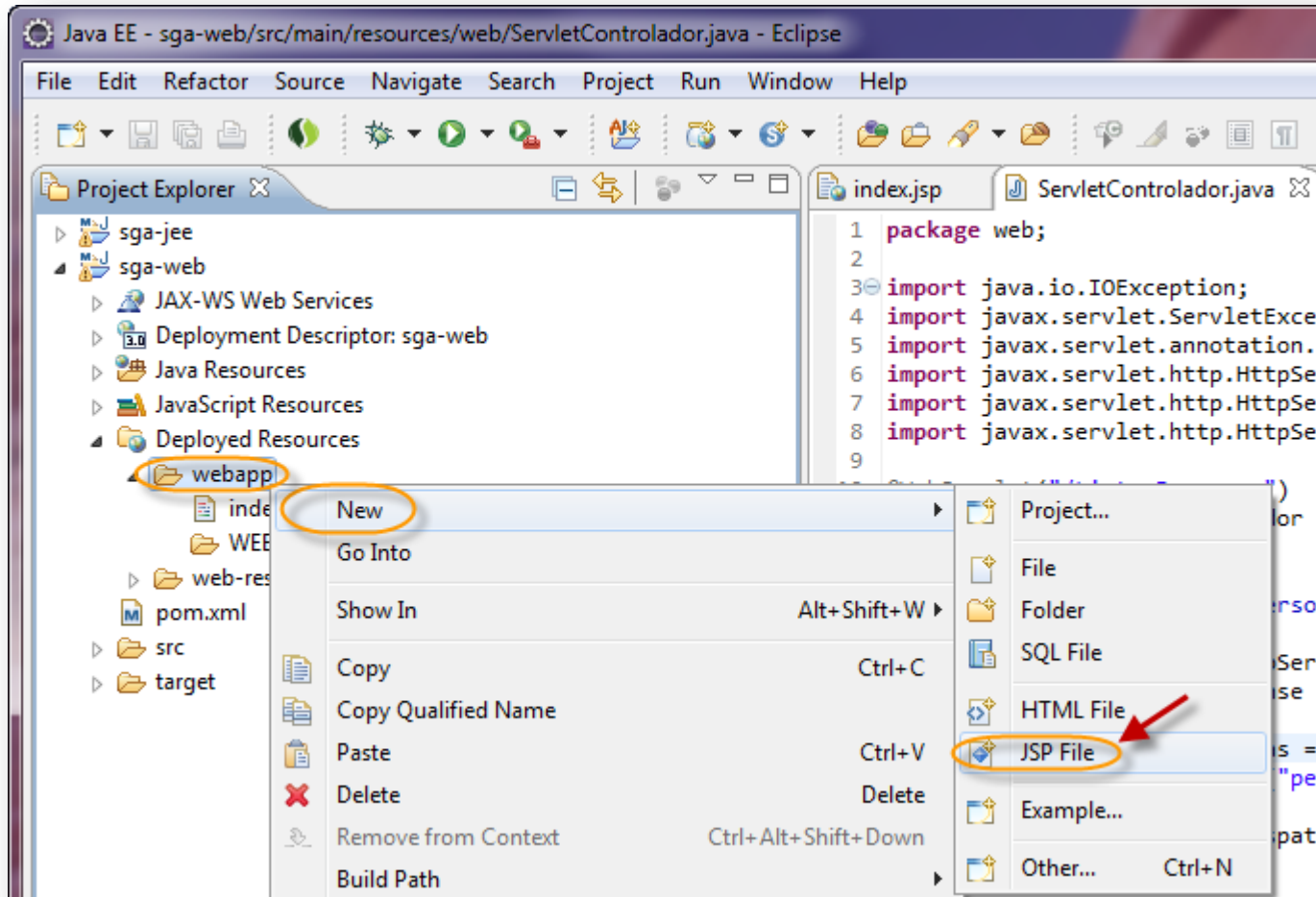
```
        request.getRequestDispatcher("listarPersonas.jsp").forward(request, response);
```

```
    }
```

```
}
```

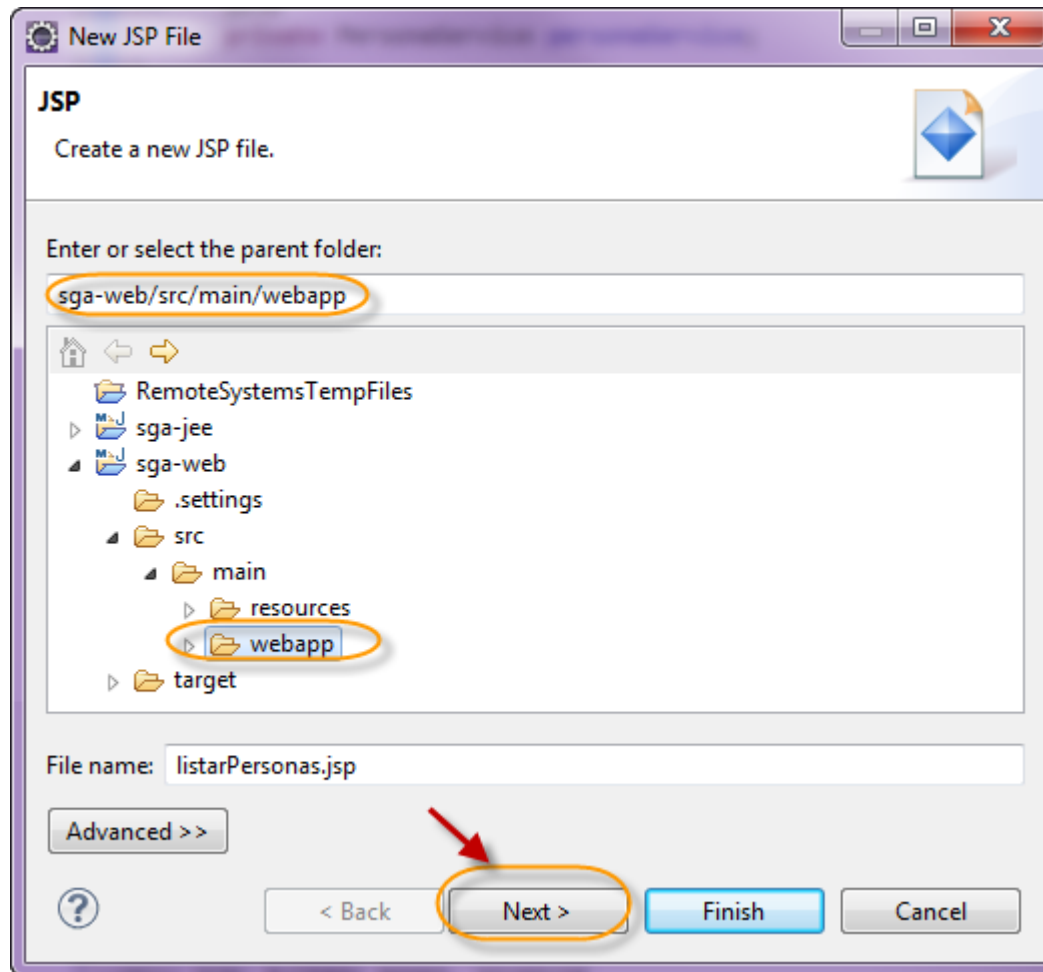
## Paso 8. Creación del JSP listarPersonas

Crear el jsp llamado listarPersonas.jsp:



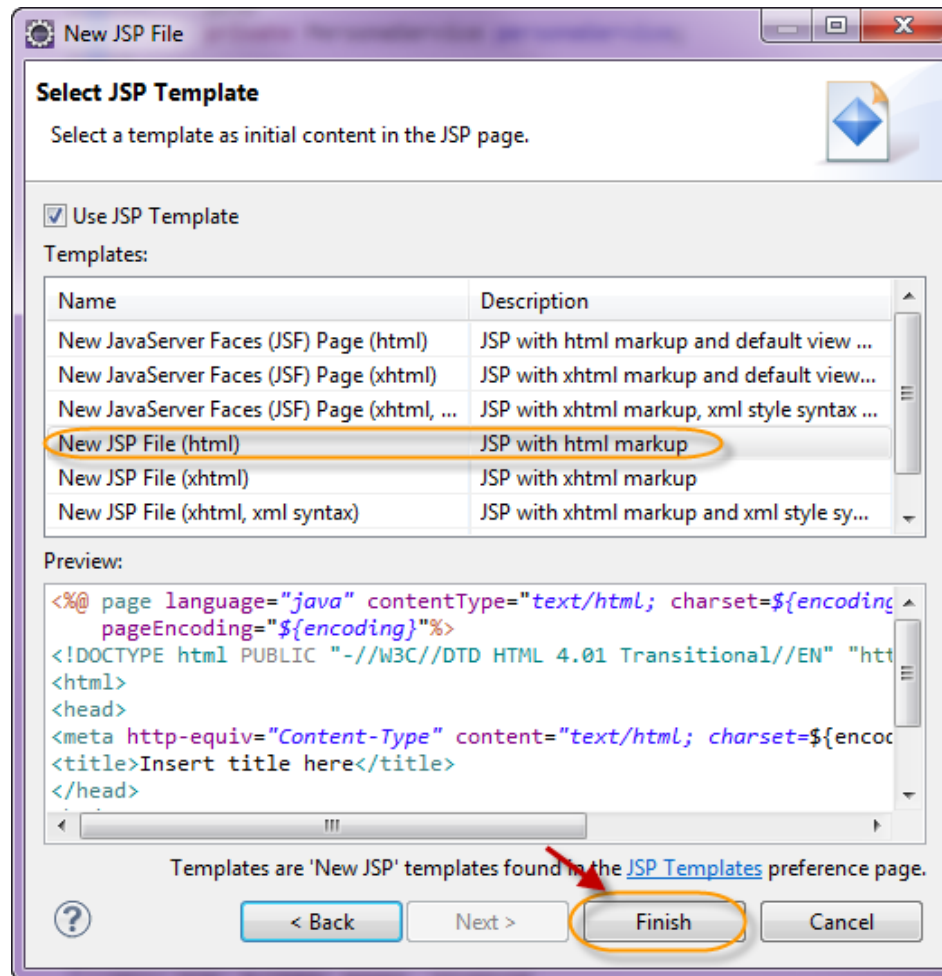
## Paso 8. Creación del JSP listarPersonas (cont)

Crear el jsp llamado listarPersonas.jsp:



## Paso 8. Creación del JSP listarPersonas (cont)

Crear el jsp llamado listarPersonas.jsp:







## Paso 8. Creación del JSP listarPersonas (cont)

Reemplazar el código del JSP con el siguiente:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
  <head>
    <title>Listado Personas</title>
  </head>
<body>

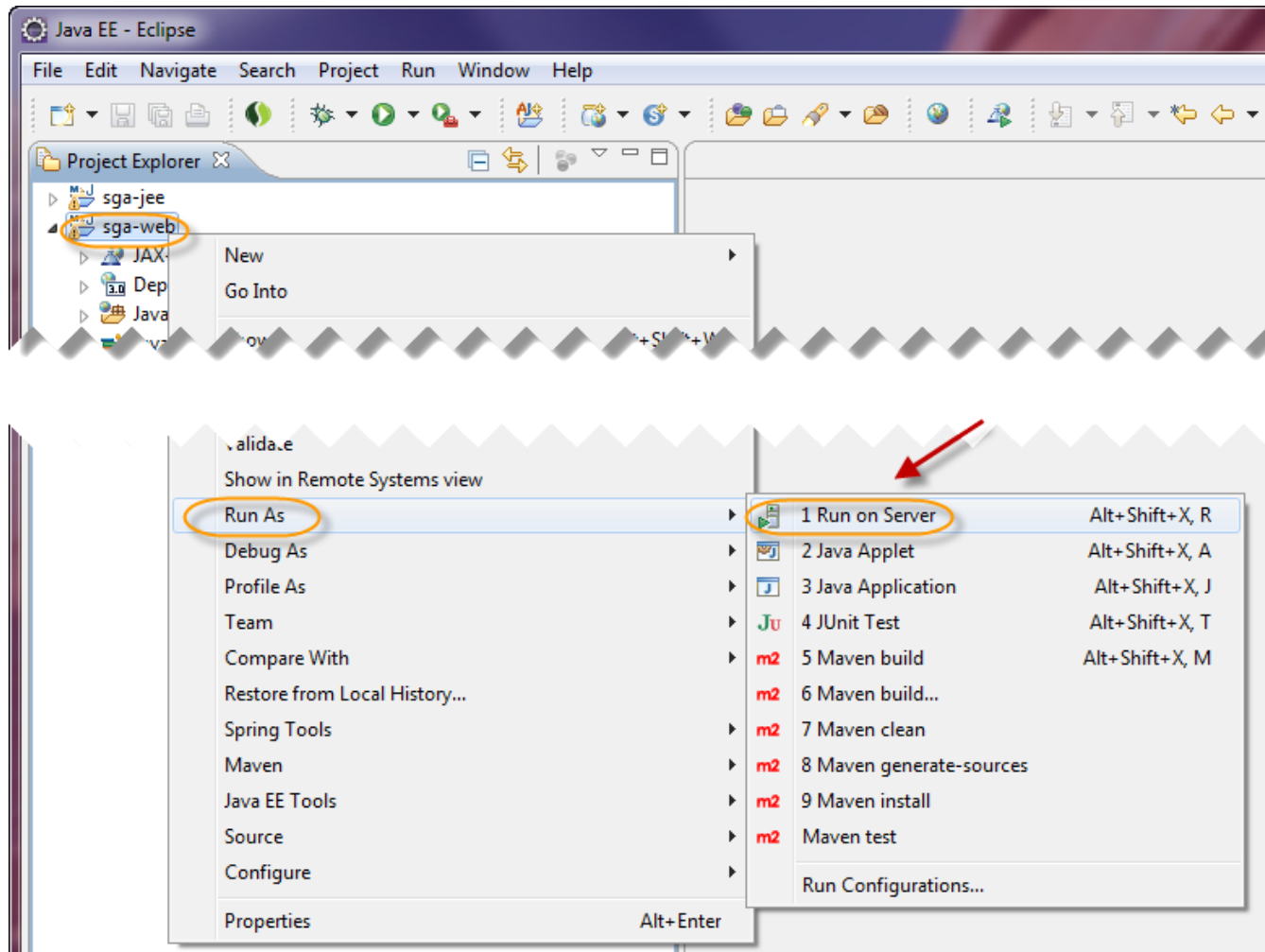
  <h1>Listado de Personas</h1>

  <table border="1">
    <tr>
      <th>Nombre</th>
      <th>Apellido Paterno</th>
      <th>Email</th>
    </tr>

    <c:forEach var="persona" items="${personas}">
      <tr>
        <td>${persona.nombre}</td>
        <td>${persona.apePaterno}</td>
        <td>${persona.email}</td>
      </tr>
    </c:forEach>
  </table>
  <br>
  <a href="index.jsp">Regresar al Inicio</a>
</body>
</html>
```

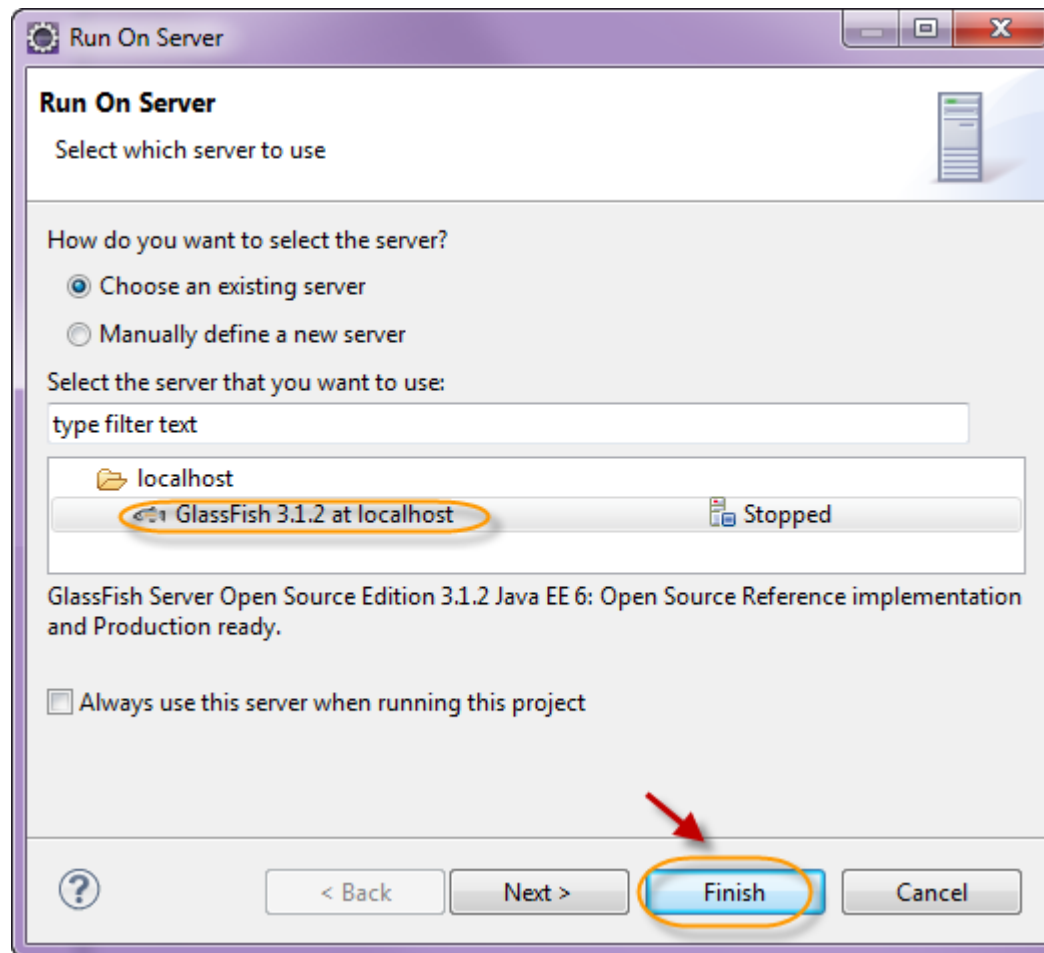
## Paso 9. Desplegar la aplicación sga-web

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



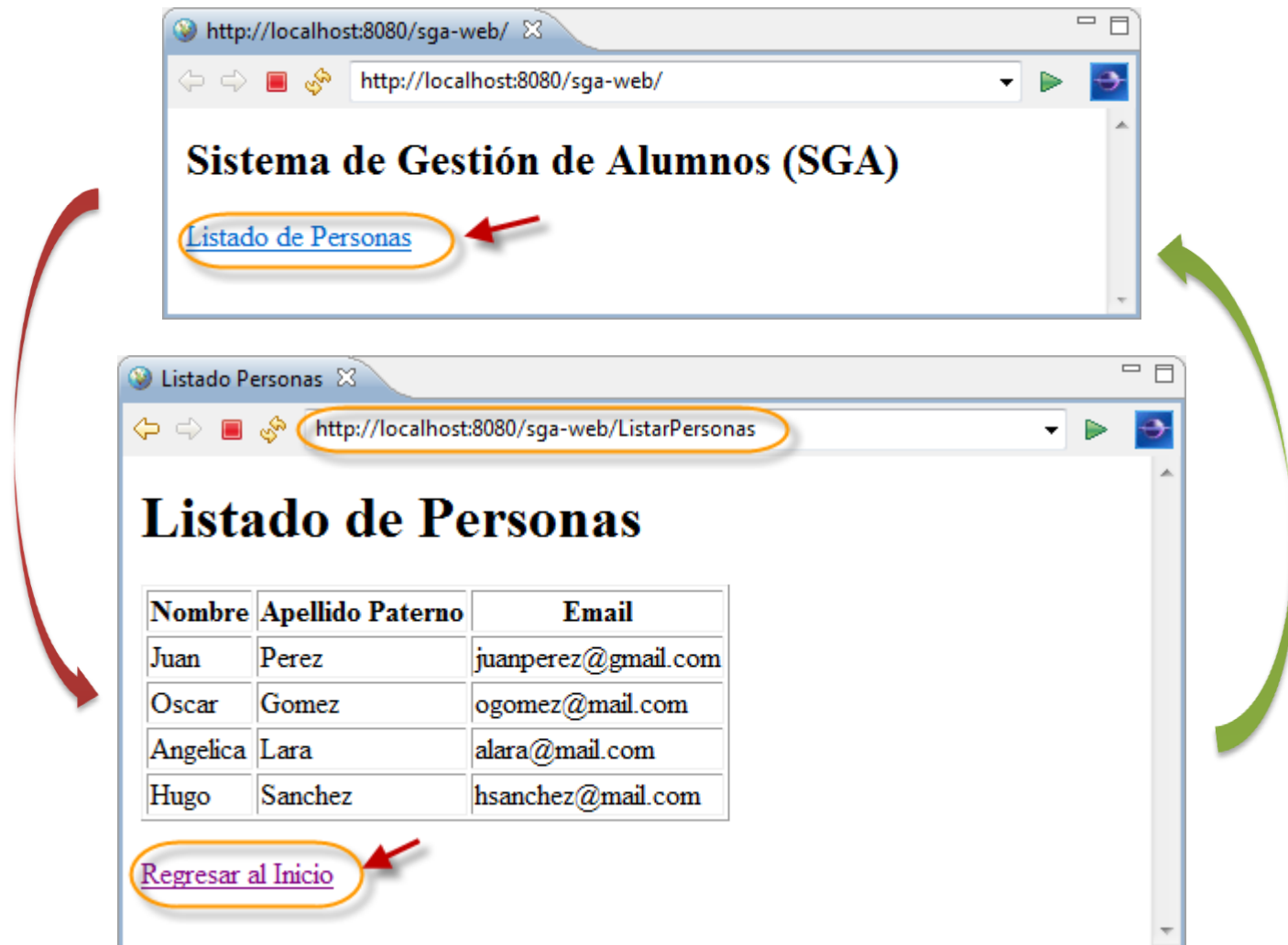
## Paso 9. Desplegar la aplicación sga-web (cont)

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



## Paso 9. Desplegar la aplicación sga-web (cont)

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



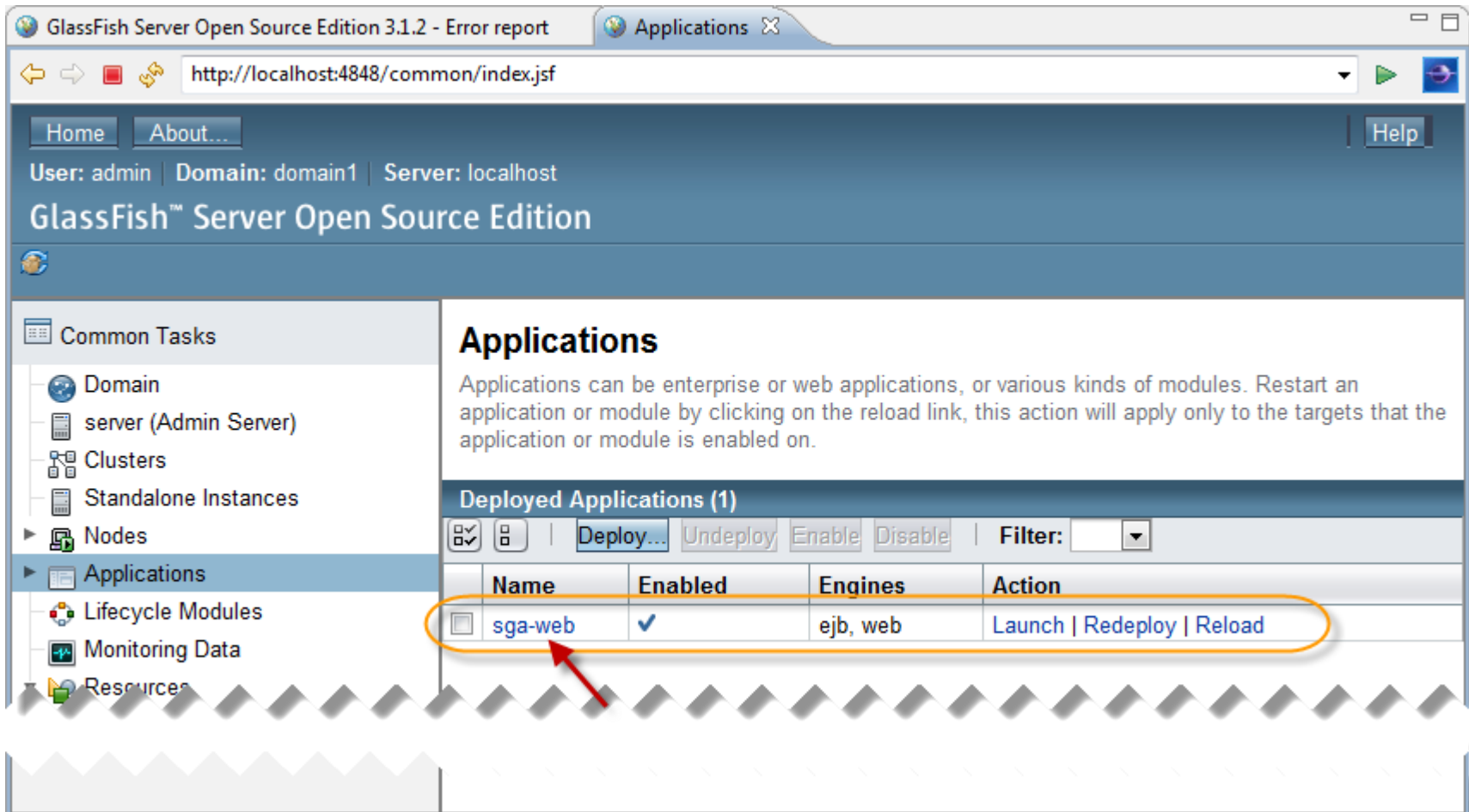
## Paso 9. Desplegar la aplicación sga-web (cont)

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



## Paso 10. Verificar Componentes Desplegados

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



The screenshot shows the GlassFish Server Open Source Edition 3.1.2 console. The 'Applications' tab is selected, displaying a table of deployed applications. The table has columns for Name, Enabled, Engines, and Action. The application 'sga-web' is listed as deployed and enabled, with engines 'ejb, web'. A red arrow points to the 'sga-web' entry in the table.

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

## Paso 10. Verificar Componentes Desplegados (cont)

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

GlassFish Server Open Source Edition 3.1.2 - Error report

http://localhost:4848/common/index.jsf

GlassFish™ Server Open Source Edition

Common Tasks

- Domain
  - server (Admin Server)
  - Clusters
  - Standalone Instances
  - Nodes
  - Applications
    - sga-web
  - Lifecycle Modules
  - Monitoring Data
  - Resources
    - JDBC
    - Connectors
    - Resource Adapter Configs
    - JMS Resources
    - JavaMail Sessions
    - JNDI
  - Configurations
    - default-config
    - server-config
  - Update Tool

General Descriptor

### Edit Application

Modify an existing application or module.

Save Cancel

Name: sga-web

Status: ☒ Enabled

Description:

Location: \${com.sun.aas.instanceRootURI}/eclipseApps/sga-web/

Libraries:

Modules and Components (7)				
Module Name	Engines	Component Name	Type	Action
sga-web	[ejb, jpa, web]	-----	-----	Launch
sga-web		PersonaDaolImpl	StatelessSessionBean	
sga-web		default	Servlet	
sga-web		jsp	Servlet	
sga-web		UsuarioServicImpl	StatelessSessionBean	
sga-web		PersonaServicImpl	StatelessSessionBean	
sga-web		UsuarioDaolImpl	StatelessSessionBean	



## Ejercicio y Conclusión

Se deja como ejercicio agregar los casos de Agregar, Editar y Eliminar una persona. En caso de requerir apoyo se les incluye la solución en los ejercicios resueltos de la lección 4.

### 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.

Los Servlets 3.0 ha facilitado la configuración e integración con los EJB debido al soporte de anotaciones que brindan, y con ello podemos realizar tareas de manera muy simple, sin necesidad de archivos de configuración xml.

En la siguiente lección integraremos la tecnología JSF con EJB y JPA de manera muy similar a como lo hemos hecho en este ejercicio.





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

*Pasión por la tecnología Java*

Experiencia y Conocimiento para tu vida