

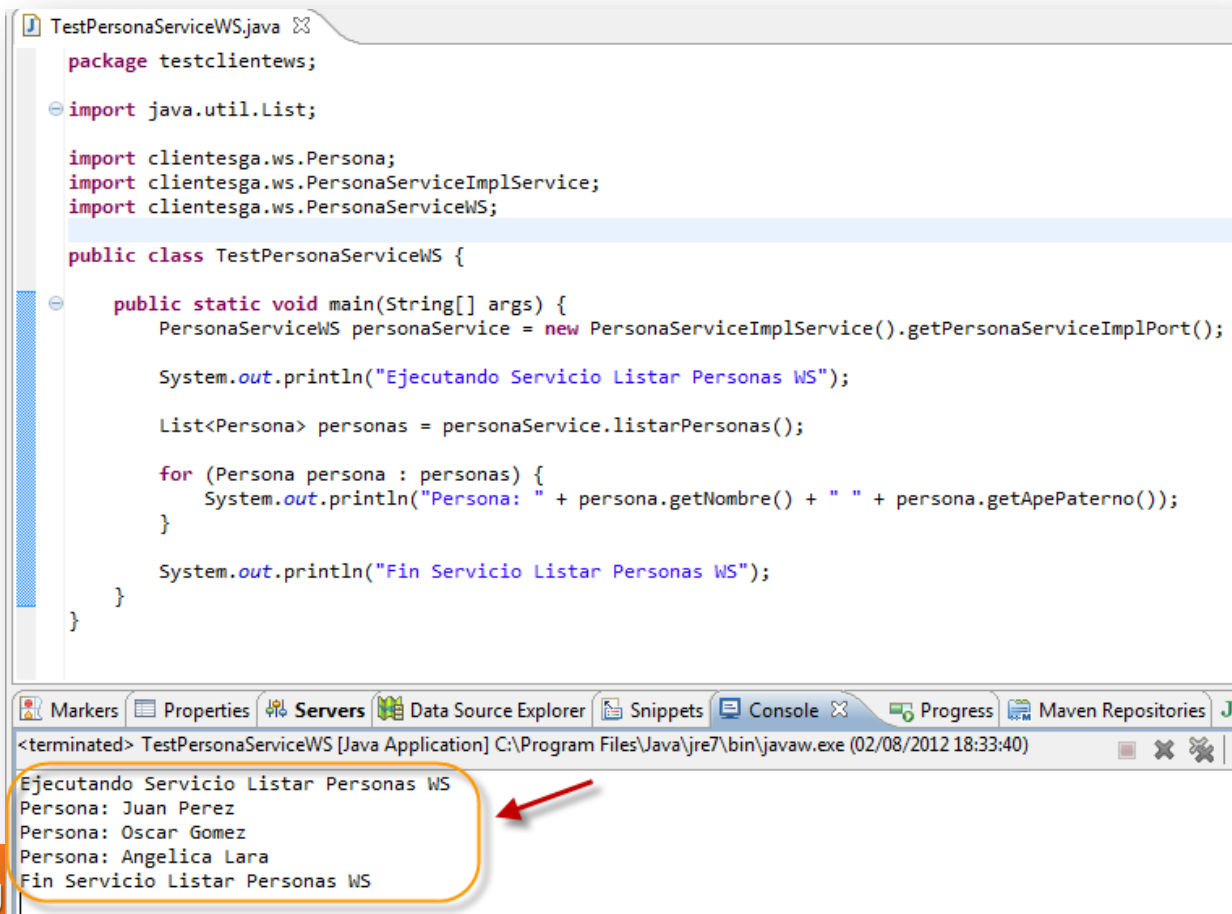


Ejercicio 11

Creación Proyecto SGA con Web
Services

Objetivo del Ejercicio

- El objetivo del ejercicio exponer el método listarPersonas del EJB del proyecto SGA como un Web Services con ayuda del API JAX-WS, así como la creación del proyecto **Cliente SGA WS**. El resultado se muestra a continuación:



The screenshot displays an IDE window titled 'TestPersonaServiceWS.java'. The code defines a package 'testclientews' and imports 'java.util.List', 'clientesga.ws.Persona', 'clientesga.ws.PersonaServiceImplService', and 'clientesga.ws.PersonaServiceWS'. A public class 'TestPersonaServiceWS' contains a 'main' method that creates a 'PersonaServiceWS' instance, calls 'listarPersonas()', and prints the results. The console output at the bottom shows the execution of the program, listing three persons: Juan Perez, Oscar Gomez, and Angelica Lara. A red arrow points to the console output.

```
TestPersonaServiceWS.java
package testclientews;

import java.util.List;

import clientesga.ws.Persona;
import clientesga.ws.PersonaServiceImplService;
import clientesga.ws.PersonaServiceWS;

public class TestPersonaServiceWS {

    public static void main(String[] args) {
        PersonaServiceWS personaService = new PersonaServiceImplService().getPersonaServiceImplPort();

        System.out.println("Ejecutando Servicio Listar Personas WS");

        List<Persona> personas = personaService.listarPersonas();

        for (Persona persona : personas) {
            System.out.println("Persona: " + persona.getNombre() + " " + persona.getApePaterno());
        }

        System.out.println("Fin Servicio Listar Personas WS");
    }
}
```

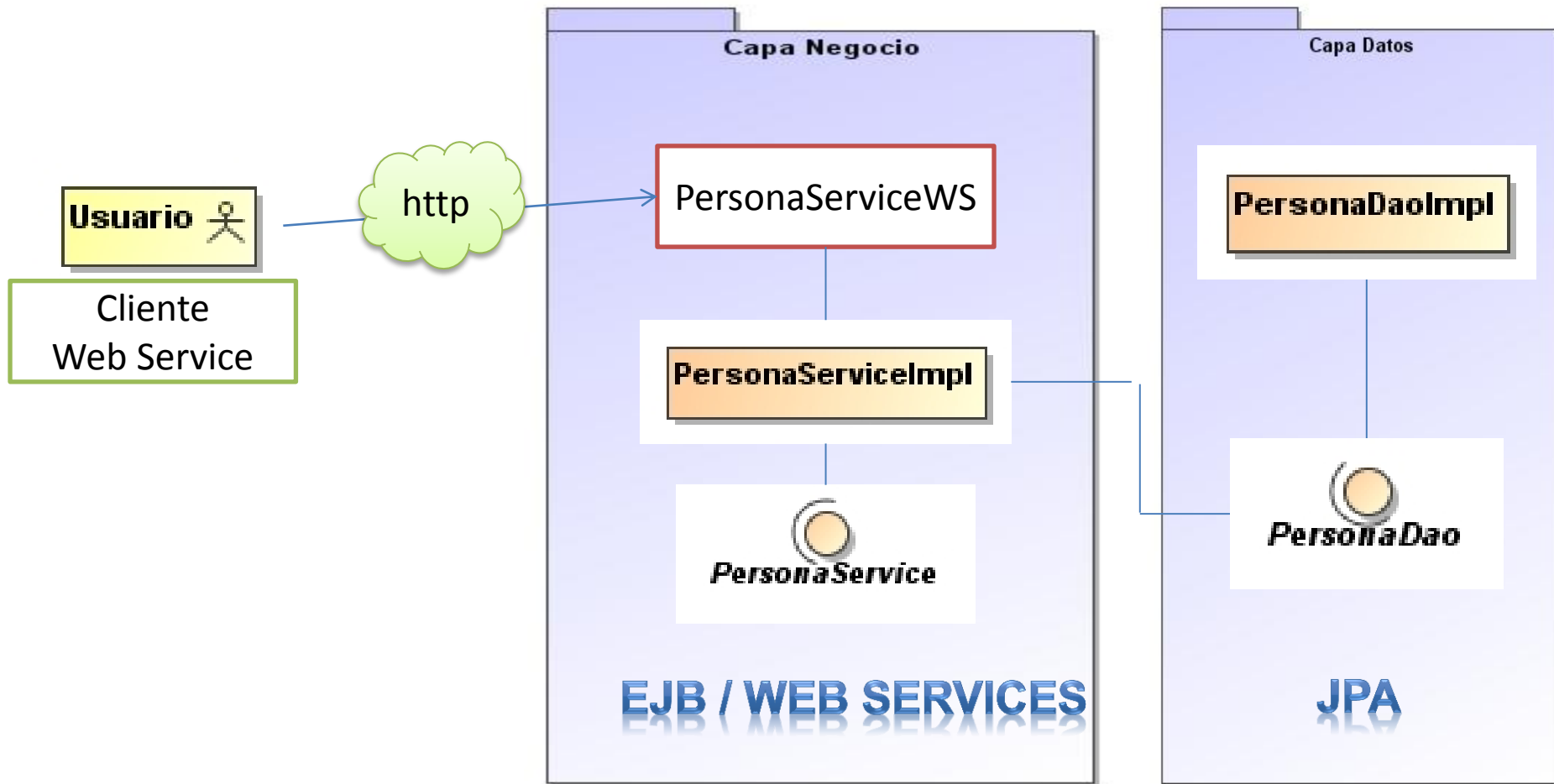
Markers Properties Servers Data Source Explorer Snippets Console Progress Maven Repositories

<terminated> TestPersonaServiceWS [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (02/08/2012 18:33:40)

Ejecutando Servicio Listar Personas WS
Persona: Juan Perez
Persona: Oscar Gomez
Persona: Angelica Lara
Fin Servicio Listar Personas WS

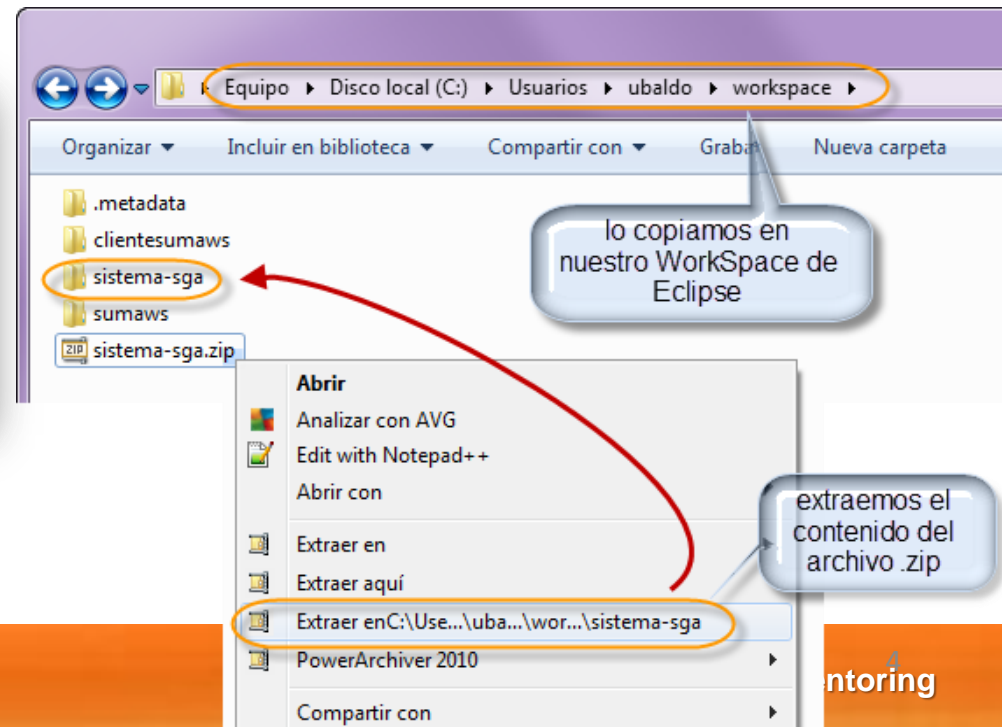
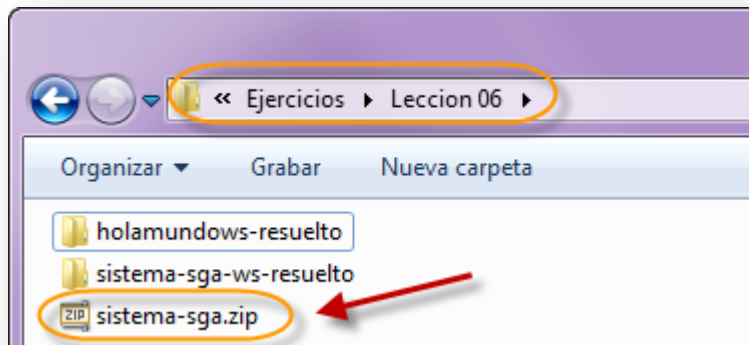
Diagrama de Clases

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



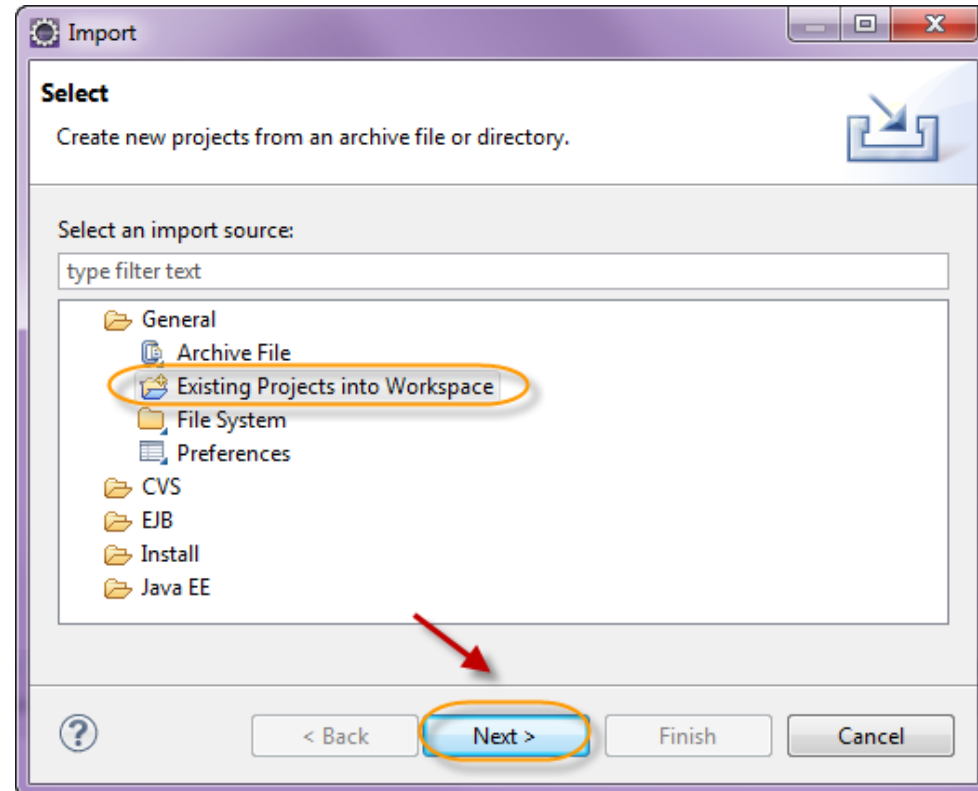
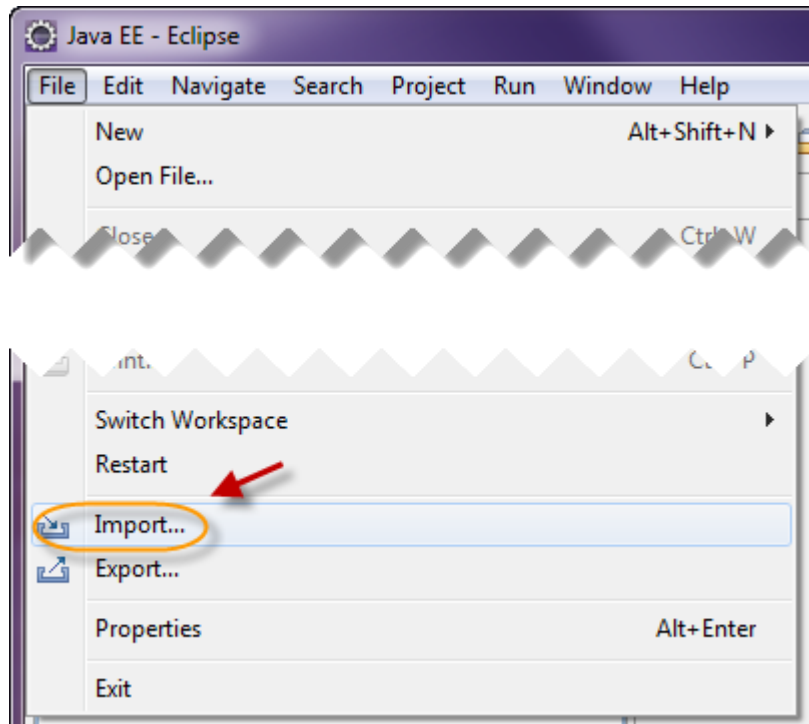
Paso 1. Importar el proyecto sistema-sga

Importamos el proyecto sistema-sga, el cual se encuentra en los ejercicios de la lección 6. La diferencia de este proyecto es que ya tiene integrado la capa de datos, capa servicio y capa web en un solo proyecto. Así que nos enfocaremos en exponer la funcionalidad del listado de personas de la capa de servicio (EJB) como un Web Service. Debemos mover el archivo **sistema-sga.zip** a nuestro WorkSpace, lo descomprimos e importamos el proyecto con Eclipse.



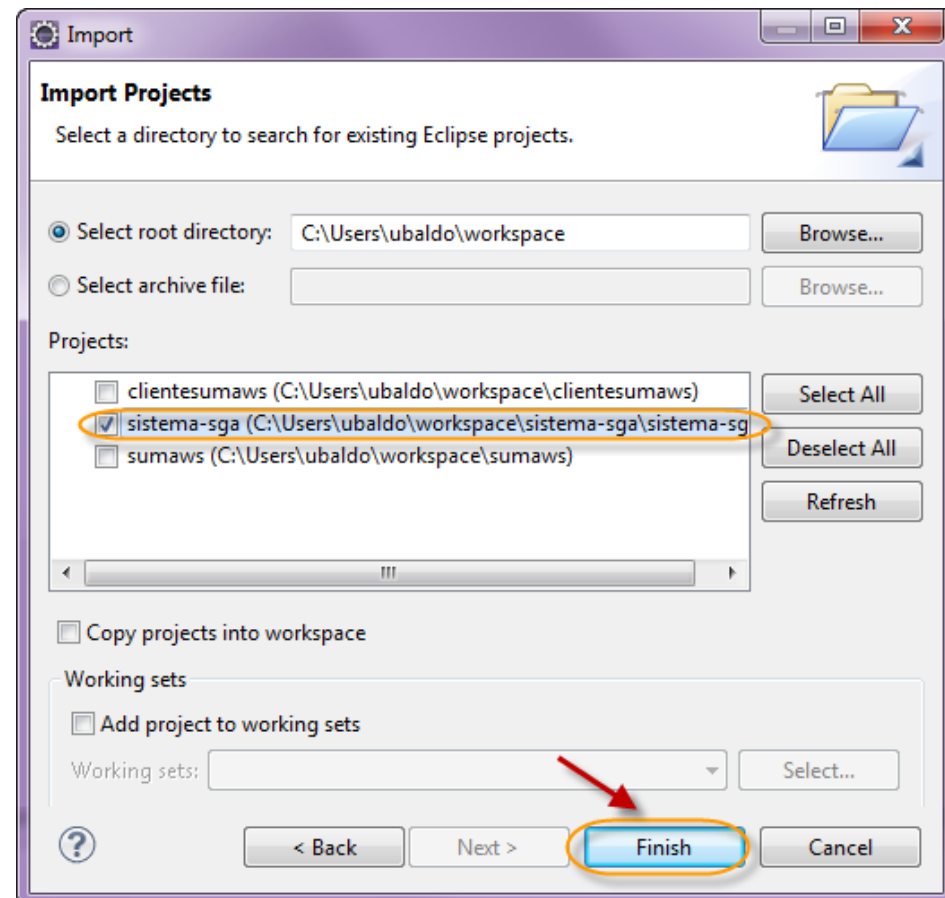
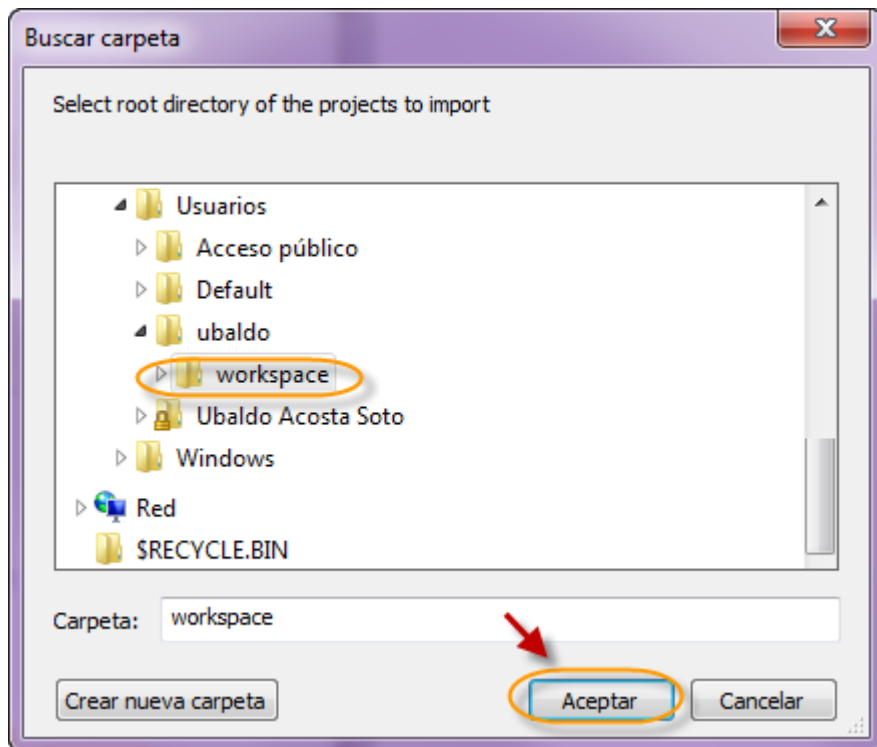
Paso 1. Importar el proyecto sistema-sga (cont)

Importamos el proyecto en nuestro IDE de Eclipse.



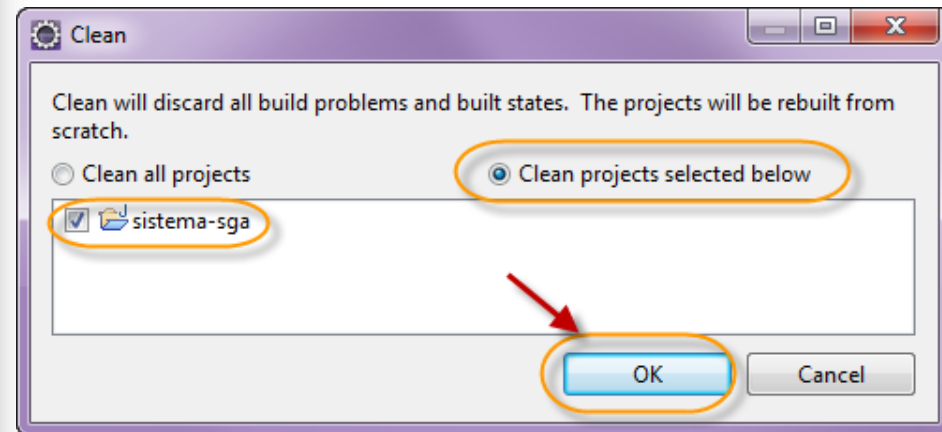
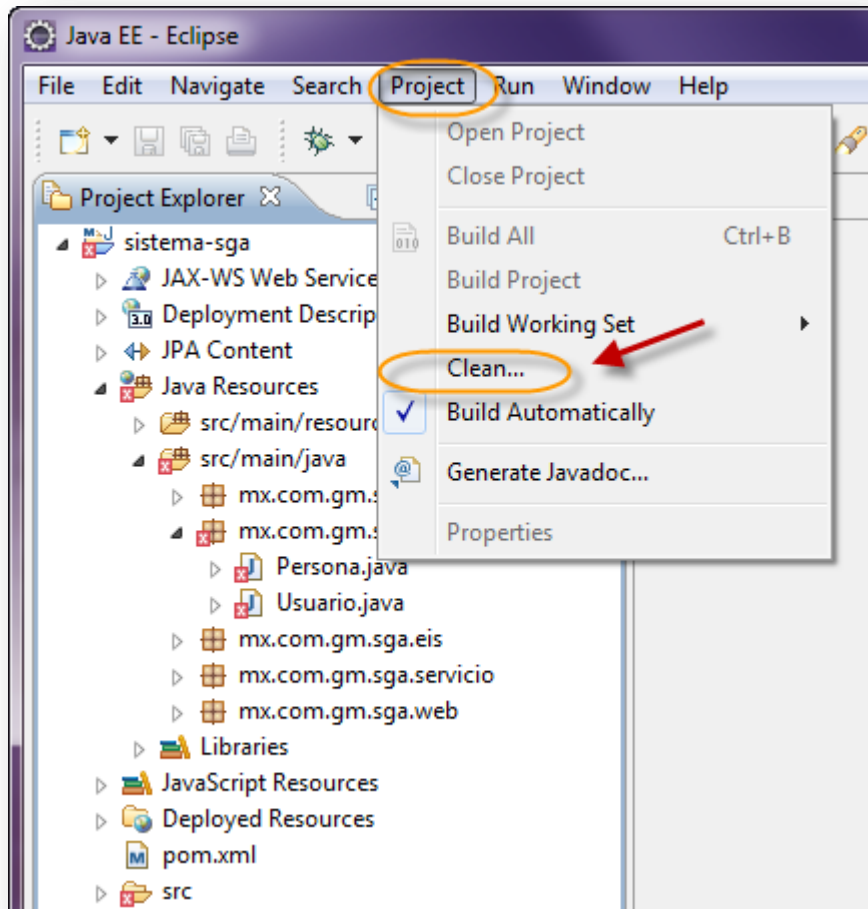
Paso 1. Importar el proyecto sistema-sga (cont)

Importamos el proyecto en nuestro IDE de Eclipse. La ruta del workspace puede variar según donde se haya generado.



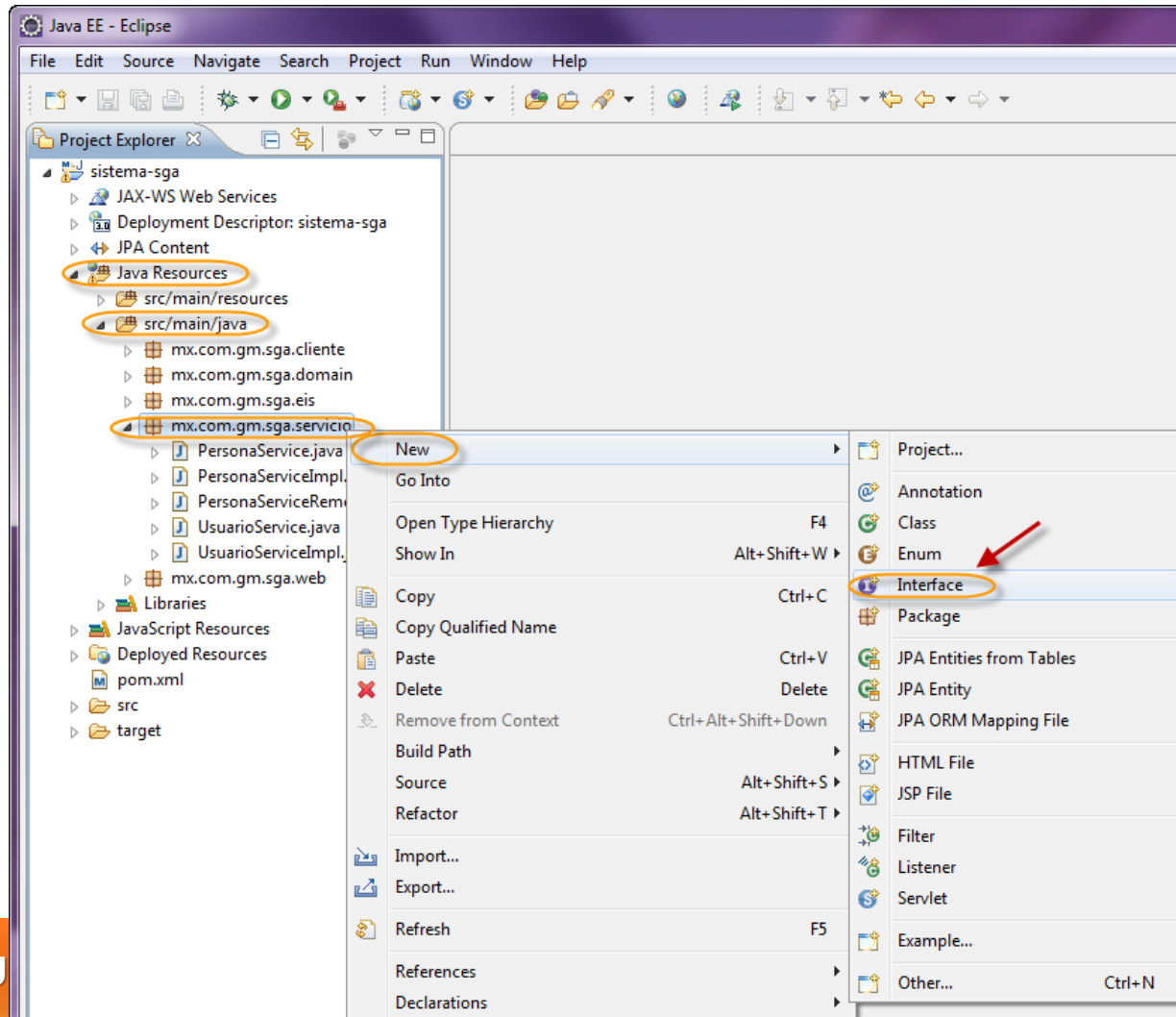
Paso 1. Importar el proyecto sistema-sga (cont)

Al cargar el proyecto pueden visualizarse algunos errores, lo que haremos es limpiar el proyecto como sigue. Con esto deberían corregirse los errores:



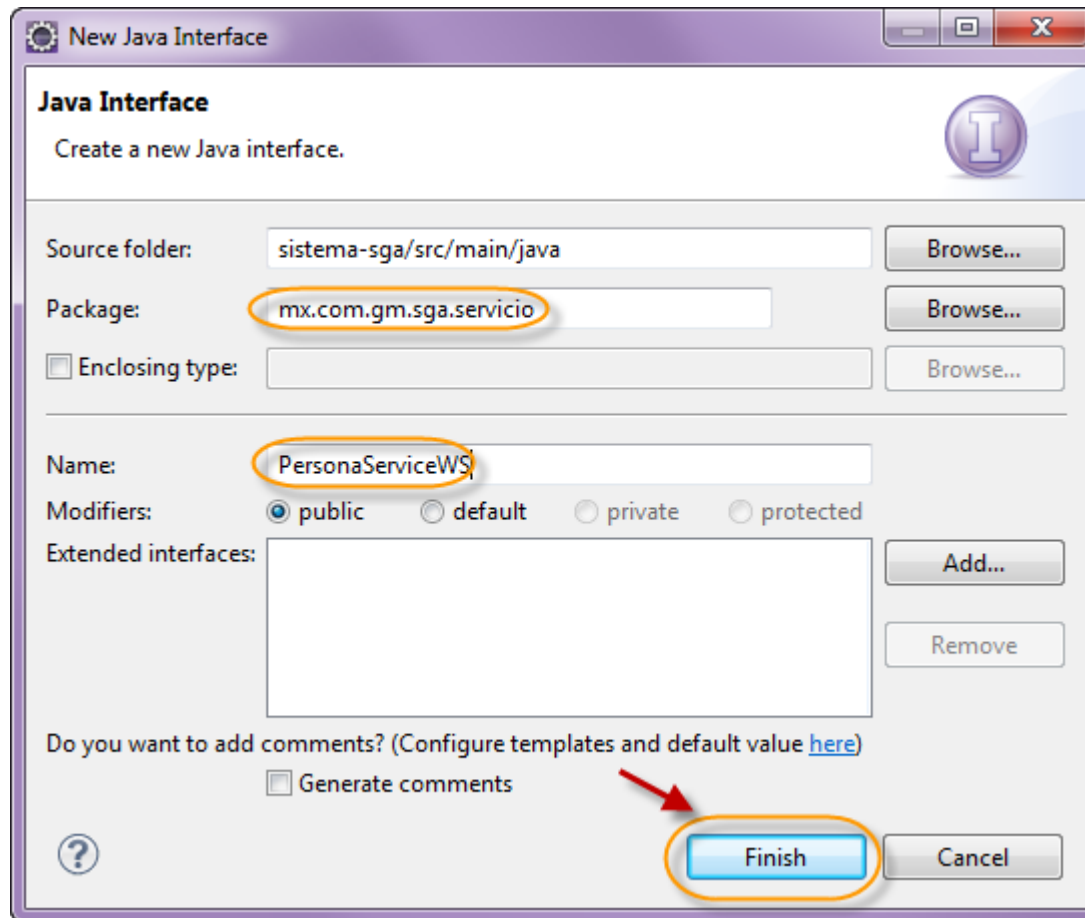
Paso 2. Creación de la interface PersonaServiceWS

Creamos la interface PersonaServiceWS para exponer únicamente el listado de personas como un Servicio Web:



Paso 2. Creación de la interface PersonaServiceWS (cont)

Creamos la interface PersonaServiceWS:





Paso 2. Creación de la interface PersonaServiceWS (cont)

Agregamos el siguiente código a la interface PersonaServiceWS:

```
package mx.com.gm.sga.servicio;

import java.util.List;
import javax.jws.WebMethod;
import javax.jws.WebService;

import mx.com.gm.sga.domain.Persona;

@WebService
public interface PersonaServiceWS {

    @WebMethod
    public List<Persona> listarPersonas();
}
```



Paso 3. Modificar clase PersonaServiceImpl

Modificamos la clase PersonaServiceImpl, agregando los siguientes cambios:

- 1) Agregar la anotación `WebService`, especificando cual es la interface a utilizar:

```
@WebService(endpointInterface = "mx.com.gm.sga.servicio.PersonaServiceWS")
```

- 2) Extender de la interface `PersonaServiceWS`, por lo que la definición de la clase queda así:

```
public class PersonaServiceImpl  
    implements PersonaServiceRemote, PersonaService, PersonaServiceWS
```

- 3) Agregamos la anotación `@Override` en el método `listarPersonas` de la clase `PersonaServiceImpl`.



Paso 4. Modificar clase Persona

Modificamos la clase de dominio Persona, agregando los siguientes cambios:

- 1) Agregar la anotación `@XmlAccessorType` a nivel de la clase, especificando cual es la interface a utilizar:

```
@XmlAccessorType(XmlAccessType.FIELD)
```

Esta anotación del API de JAXB, significa que cada uno de los atributos de la clase se utilizará en el Web Service, y se validará con el esquema XSD, el cual es parte del mensaje Web Service.

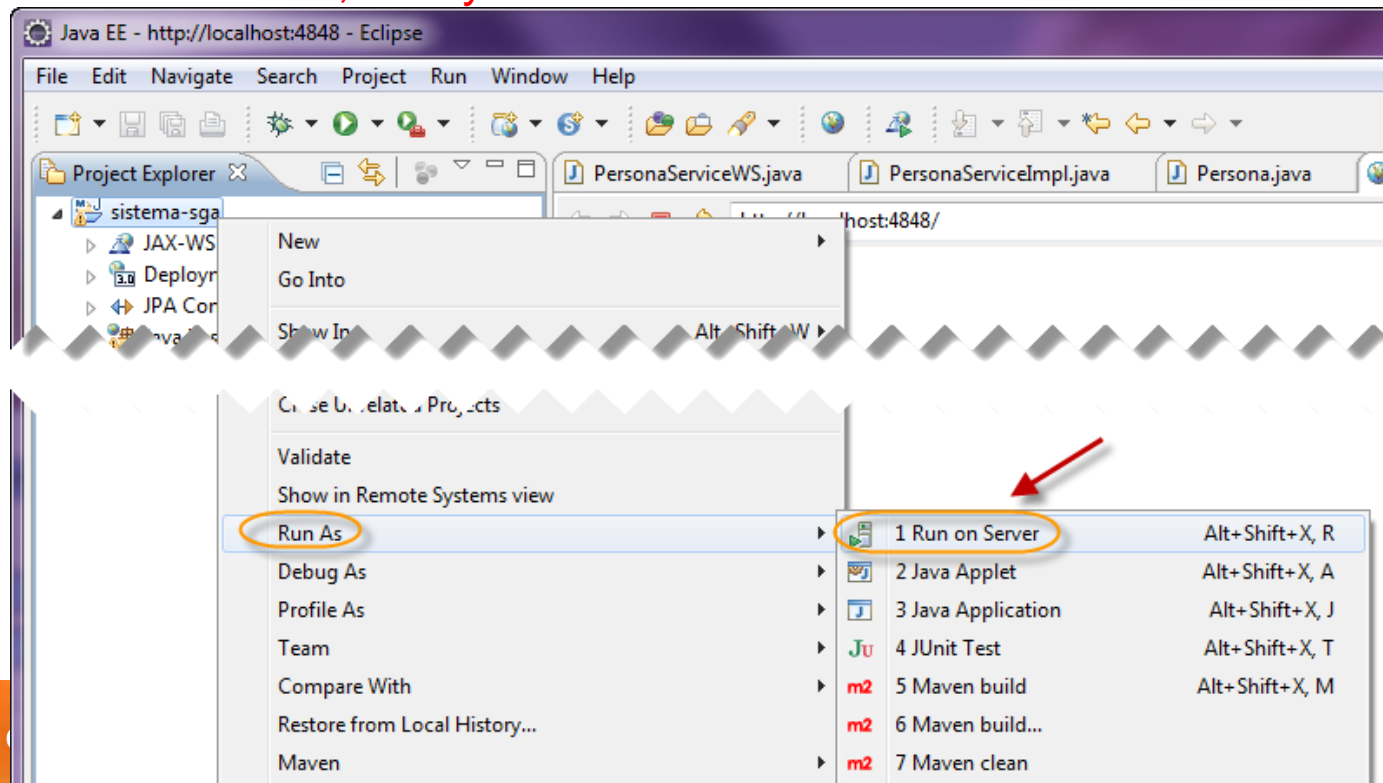
- 2) Excluir la relación con la colección de Usuarios. Para esto agregamos la anotación `@XmlTransient` al atributo usuarios de la clase Persona. Esta anotación debe ir antes de la anotación `@OneToMany` de dicho atributo:

```
@XmlTransient
```

Esta anotación la estamos agregando ya que la clase Persona tiene una relación con muchos Usuarios, y la clase Usuarios tiene una relación con una Persona, esto genera una relación circular, la cual el API de JAXB no soporta, para evitar este problema, indicamos que el atributo `private Set<Usuario> usuarios` no formará parte del envío del mensaje del Web Services. Esto no afecta a la configuración de JPA que también está incluida en la clase de dominio Persona y soporta sin problemas la navegación bidireccional.

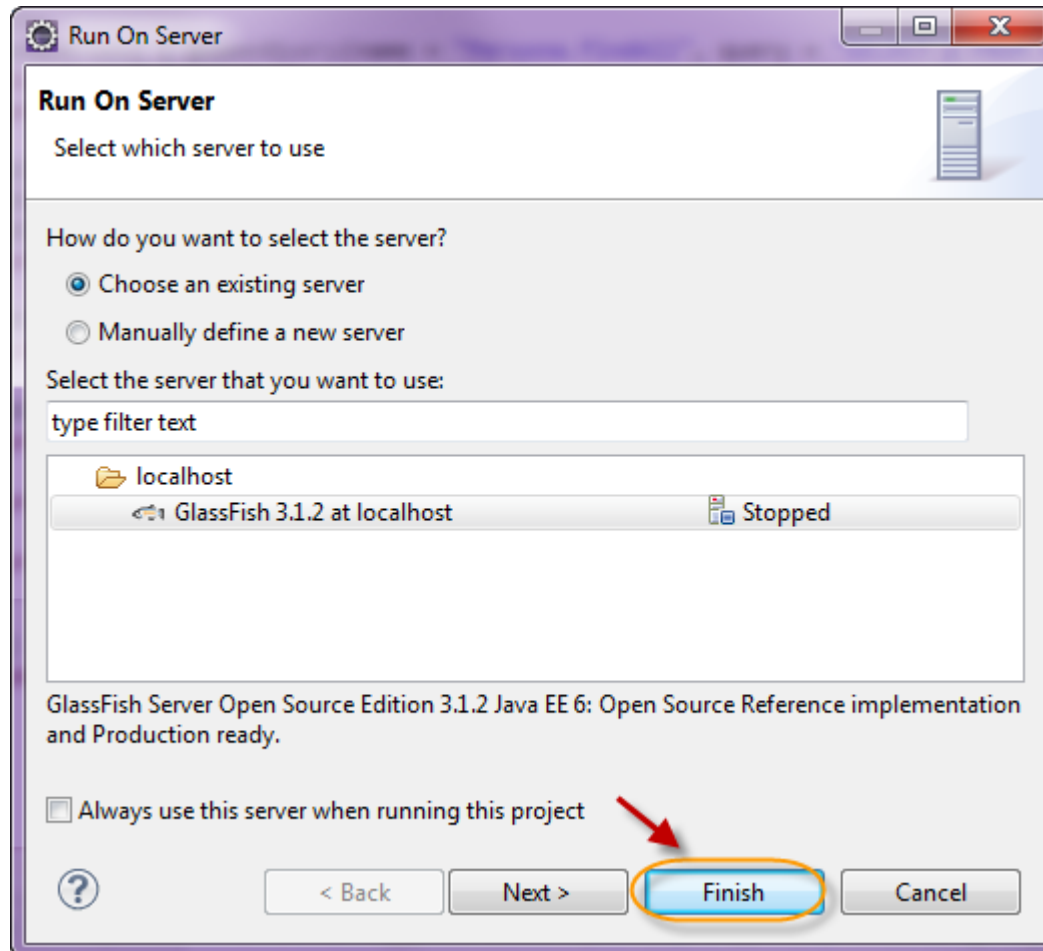
Paso 5. Despliegue aplicación sistema-sga

Una vez realizadas las modificaciones, ya está lista la configuración del Web Services, el cual expondrá el método listarPersonas del EJB de tipo Stateless. Ahora procedemos a desplegar la aplicación en GlassFish. Para desplegar la aplicación simplemente damos clic derecho -> Run As -> Run on Server. **Nota:** Antes de desplegar la aplicación debemos hacer undeploy de cualquier otra aplicación que contenga los EJB del proyecto SGA, debido a que este proyecto ya incluye todas las clases, incluyendo los EJB's:



Paso 5. Despliegue aplicación sistema-sga (cont)

Desplegamos la aplicación sobre GlassFish:



Paso 5. Despliegue aplicación sistema-sga (cont)

Observamos la consola, la cual no debe desplegar errores, y además nos debe proporcionar la URL del Web Services que se desplegó:

The screenshot shows a Java EE IDE with the following components:

- Project Explorer:** Shows the project 'sistema-sga'.
- Browser:** Displays the application at `http://localhost:8080/sistema-sga/`. The page title is 'Sistema de Gestión de Alumnos (SGA)' and it contains a button labeled 'Listar Personas'.
- Console:** Shows the following logs:
 - Advertencia: JMX MBeanServer in use: [com.sun.jmx.mbeanserver.JmxMBeanServer@2a3dceea] from index [1]
 - Información: EJB5181:Portable JNDI names for EJB UsuarioServiceImpl: [java:global/sistema-sga/UsuarioServi
 - Información: EJB5181:Portable JNDI names for EJB UsuarioDaoImpl: [java:global/sistema-sga/UsuarioDaoImpl,
 - Advertencia: Container org.glassfish.webservices.JAXWSContainer@6eb6ea5e doesn't support class com.sun.xml
 - Información: EJB5181:Portable JNDI names for EJB PersonaServiceImpl: [java:global/sistema-sga/PersonaServi
 - Información: EJB5182:Glassfish-specific (Non-portable) JNDI names for EJB PersonaServiceImpl: [mx.com.gm.s
 - Información: EJB5181:Portable JNDI names for EJB PersonaDaoImpl: [java:global/sistema-sga/PersonaDaoImpl,
 - Información: WS00019: EJB Endpoint deployed sistema-sga listening at address at `http://servidor:8080/PersonaServiceImplService/PersonaServiceImpl`**
 - Información: Inicializando Mojarra 2.1.6 (SNAPSHOT 20111206) para el contexto ''

Paso 6. Revisión del WebServices PersonaService

Con la siguiente URLs se puede ejecutar el Test del Web Services:

URL del Test:

<http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?Tester>

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:listarPersonasResponse xmlns:ns2="http://servicio.sga.gm.com.mx/">
      <return>
        <idPersona>1</idPersona>
        <apePaterno>Perez</apePaterno>
        <email>juanperez@gmail.com4</email>
        <nombre>Juan</nombre>
        <telefono>56677889</telefono>
      </return>
      <return>
        <idPersona>2</idPersona>
        <apePaterno>Gomez</apePaterno>
        <email>ogomez@mail.com</email>
        <nombre>Oscar</nombre>
        <telefono>55780109</telefono>
      </return>
      <return>
        <idPersona>22</idPersona>
        <apePaterno>Lara</apePaterno>
        <email>alara@mail.com</email>
        <nombre>Angelica</nombre>
        <telefono>1314145519</telefono>
      </return>
    </ns2:listarPersonasResponse>
  </S:Body>
</S:Envelope>
```

Paso 6. Revisión del WebService PersonaService (cont)

Con las siguientes URLs debemos poder visualizar el documento wsdl y el esquema XSD del Web Services listarPersonas:

URL del WSDL:

<http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?wsdl>

URL del XSD:

<http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?xsd=1>

WSDL

```
http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?wsdl
http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?wsdl

<?xml version="1.0" encoding="UTF-8" ?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.2-b13 (branches/2.2-6964;
2012-01-09T18:04:18+0000) JAXWS-RI/2.2.6-promoted-b20 JAXWS/2.2 svn-revision#unknown. -->
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.2-b13 (branches/2.2-6964;
2012-01-09T18:04:18+0000) JAXWS-RI/2.2.6-promoted-b20 JAXWS/2.2 svn-revision#unknown. -->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:wsap="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://servicio.sga.gm.com.mx/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:http="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://servicio.sga.gm.com.mx/" name="PersonaServiceImplService">
<types>
<xsd:schema>
<xsd:import namespace="http://servicio.sga.gm.com.mx/"
schemaLocation="http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?xsd=1" />
<xsd:schema>
</types>
<message name="listarPersonas">
<part name="parameters" element="tns:listarPersonas" />
</message>
<message name="listarPersonasResponse">
<part name="parameters" element="tns:listarPersonasResponse" />
</message>
<portType name="PersonaServiceWS">
<operation name="listarPersonas">
<input wsam:Action="http://servicio.sga.gm.com.mx/PersonaServiceWS/listarPersonasRequest"
message="tns:listarPersonas" />
<output wsam:Action="http://servicio.sga.gm.com.mx/PersonaServiceWS/listarPersonasResponse"
message="tns:listarPersonasResponse" />
</operation>
</portType>
```

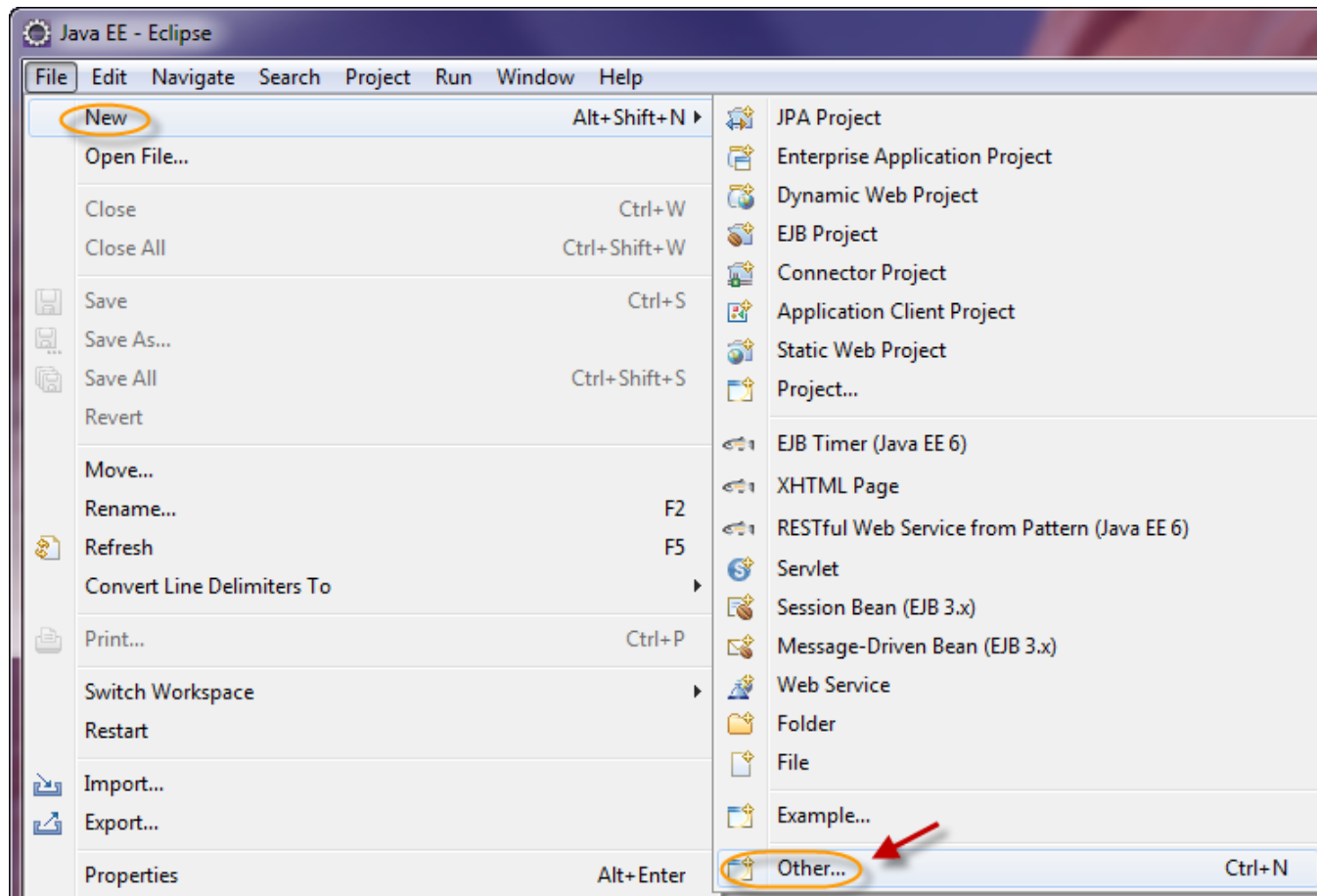
XSD

```
http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?xsd=1
http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?xsd=1

<?xml version="1.0" encoding="UTF-8" ?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.2-b13 (bra
2012-01-09T18:04:18+0000) JAXWS-RI/2.2.6-promoted-b20 JAXWS/2.2 svn-revision#unknown. -->
<xs:schema xmlns:tns="http://servicio.sga.gm.com.mx/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://servicio.sga.gm.com.mx/">
<xs:element name="listarPersonas" type="tns:listarPersonas" />
<xs:element name="listarPersonasResponse" type="tns:listarPersonasResponse" />
<xs:complexType name="listarPersonas">
<xs:sequence />
</xs:complexType>
<xs:complexType name="listarPersonasResponse">
<xs:sequence />
</xs:complexType>
<xs:complexType name="persona">
<xs:sequence>
<xs:element name="return" type="tns:persona" minOccurs="0" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
<xs:complexType name="persona">
<xs:sequence>
<xs:element name="idPersona" type="xs:int" />
<xs:element name="apeMaterno" type="xs:string" minOccurs="0" />
<xs:element name="apePaterno" type="xs:string" minOccurs="0" />
<xs:element name="email" type="xs:string" minOccurs="0" />
<xs:element name="nombre" type="xs:string" minOccurs="0" />
<xs:element name="telefono" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:schema>
```

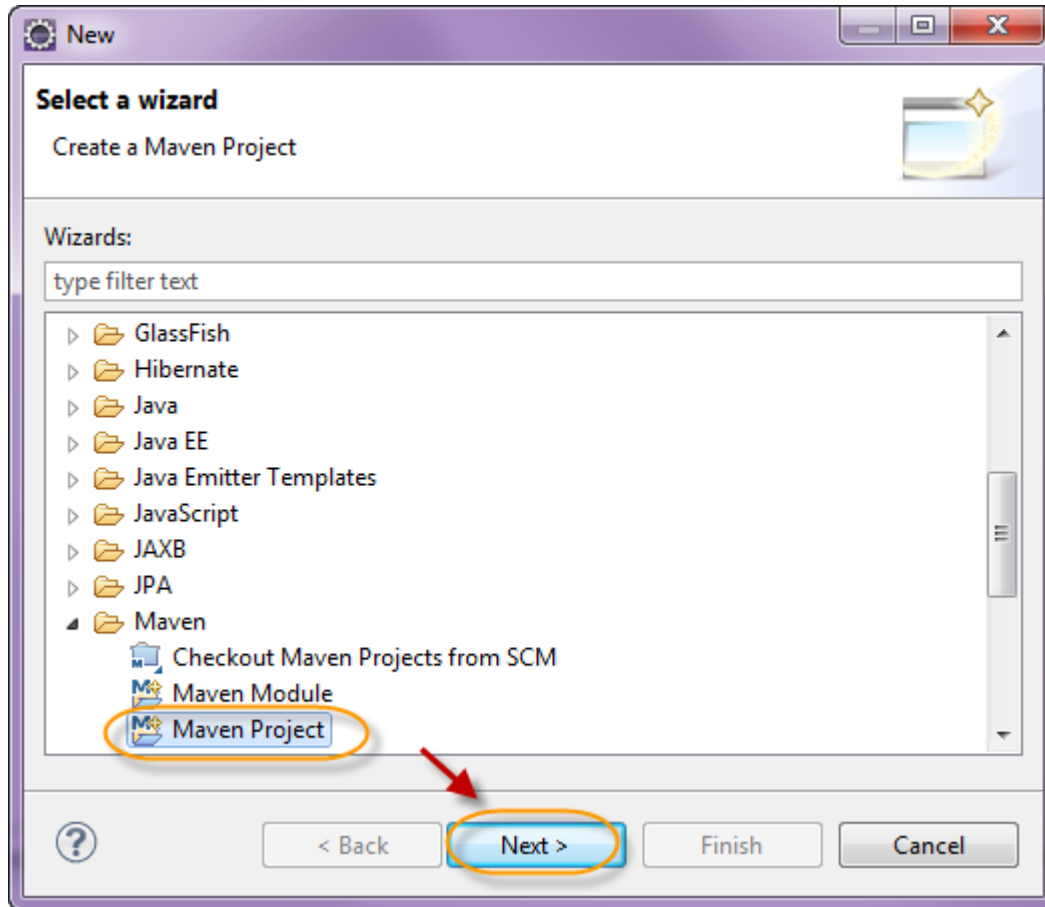
Paso 7. Creación del Cliente SGA-ClienteWS

Una vez desplegada la aplicación Sistema-SGA y con el servidor GlassFish ya iniciado, procedemos a crear el cliente del Servicio Web. Para ello vamos a crear un nuevo proyecto llamado SGA-ClienteWS:



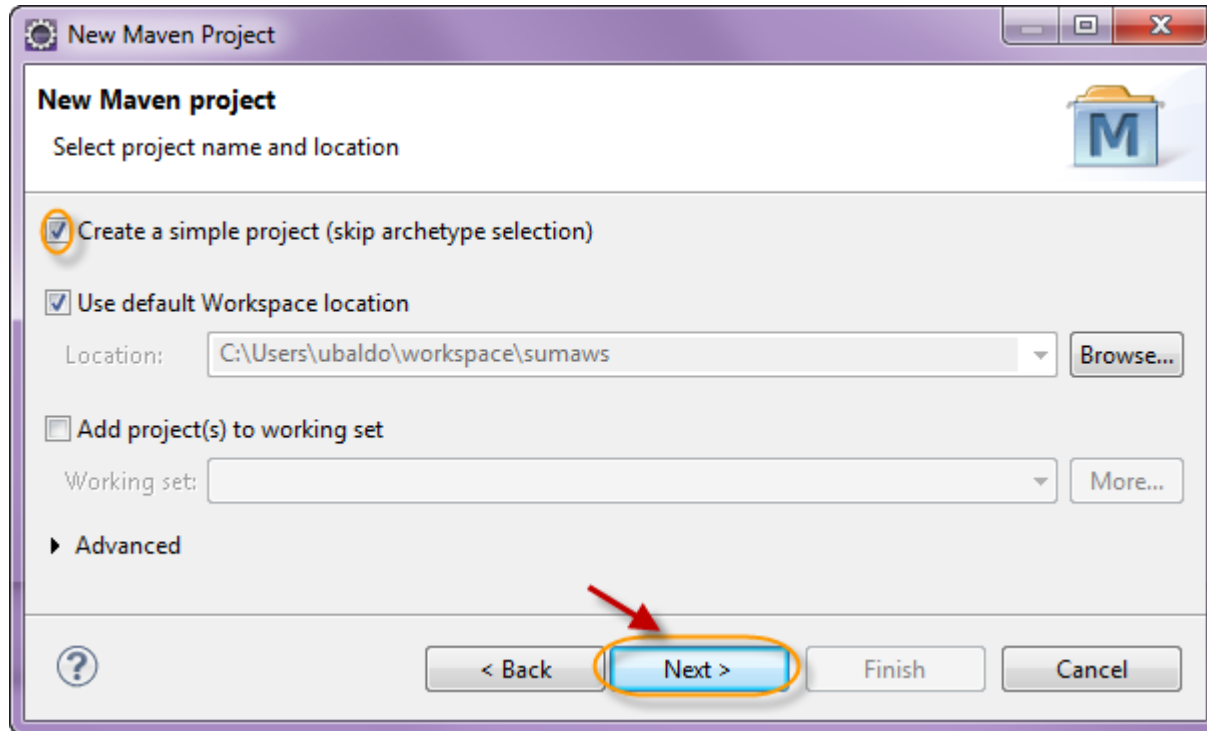
Paso 7. Creación del Cliente SGA-ClienteWS (cont)

Creamos un nuevo proyecto llamado SGA-ClienteWS utilizando Maven:



Paso 7. Creación del Cliente SGA-ClienteWS (cont)

Creamos un nuevo proyecto llamado SGA-ClienteWS utilizando Maven:



Paso 7. Creación del Cliente SGA-ClienteWS (cont)

Creamos un nuevo proyecto llamado SGA-ClienteWS utilizando Maven:

New Maven Project
Configure project

Artifact

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:



Paso 8. Configuración del archivo pom.xml (cont)

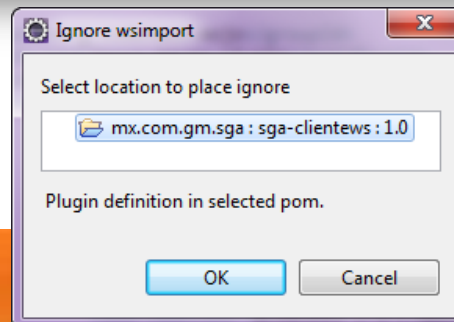
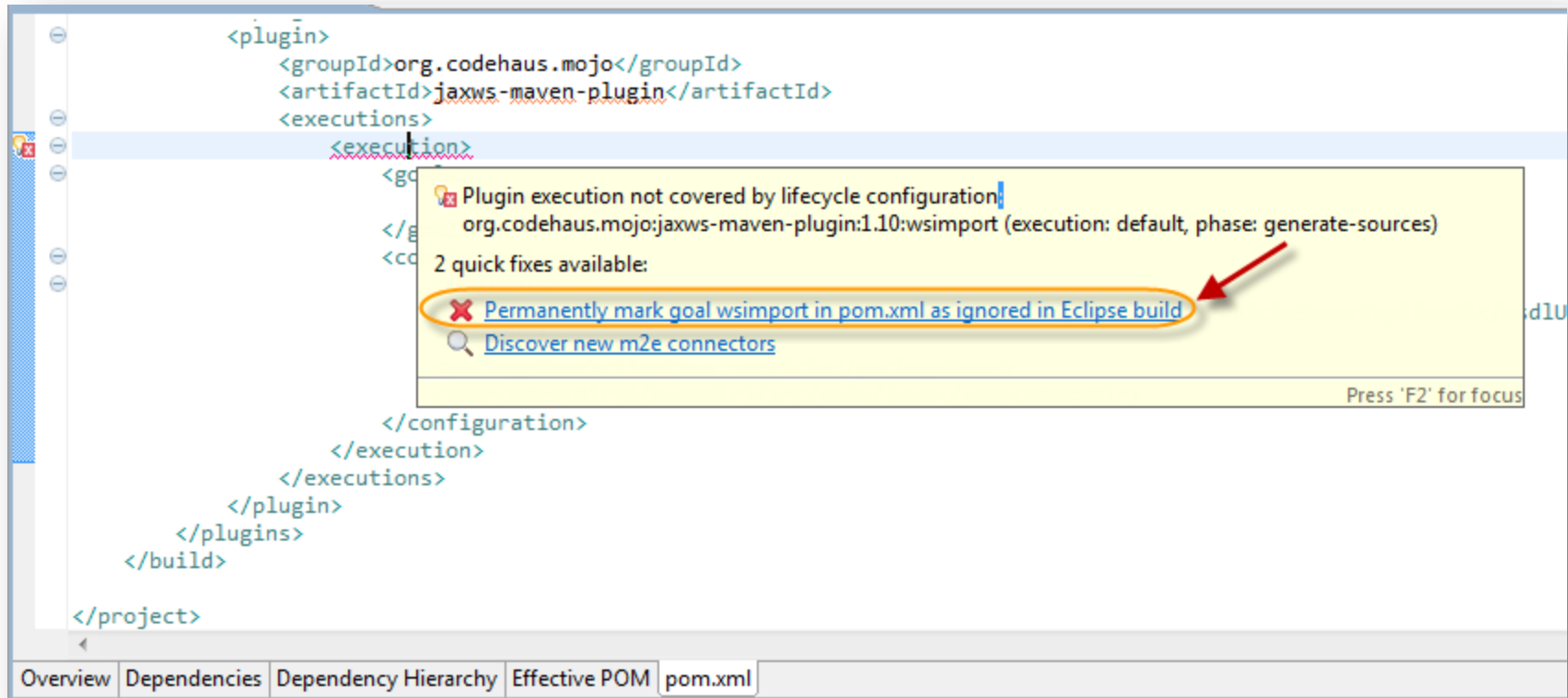
Sustituimos el contenido del archivo pom.xml por el siguiente:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>mx.com.gm.sga</groupId>
  <artifactId>sga-clientews</artifactId>
  <version>1.0</version>
  <dependencies>
    <dependency>
      <groupId>javax</groupId>
      <artifactId>javaee-api</artifactId>
      <version>6.0</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
  <pluginRepositories>
    <pluginRepository>
      <id>maven2-repository.dev.java.net</id>
      <name>Java.net Repository for Maven</name>
      <url>http://download.java.net/maven/glassfish/</url>
    </pluginRepository>
  </pluginRepositories>
</project>
```

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.6</source>
        <target>1.6</target>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>jaxws-maven-plugin</artifactId>
      <executions>
        <execution>
          <goals>
            <goal>wsimport</goal>
          </goals>
          <configuration>
            <wsdlUrls>
              <wsdlUrl>http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?wsdl</wsdlUrl>
            </wsdlUrls>
            <packageName>clientesga.ws</packageName>
            <sourceDestDir>${basedir}/src/main/java</sourceDestDir>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
</project>
```

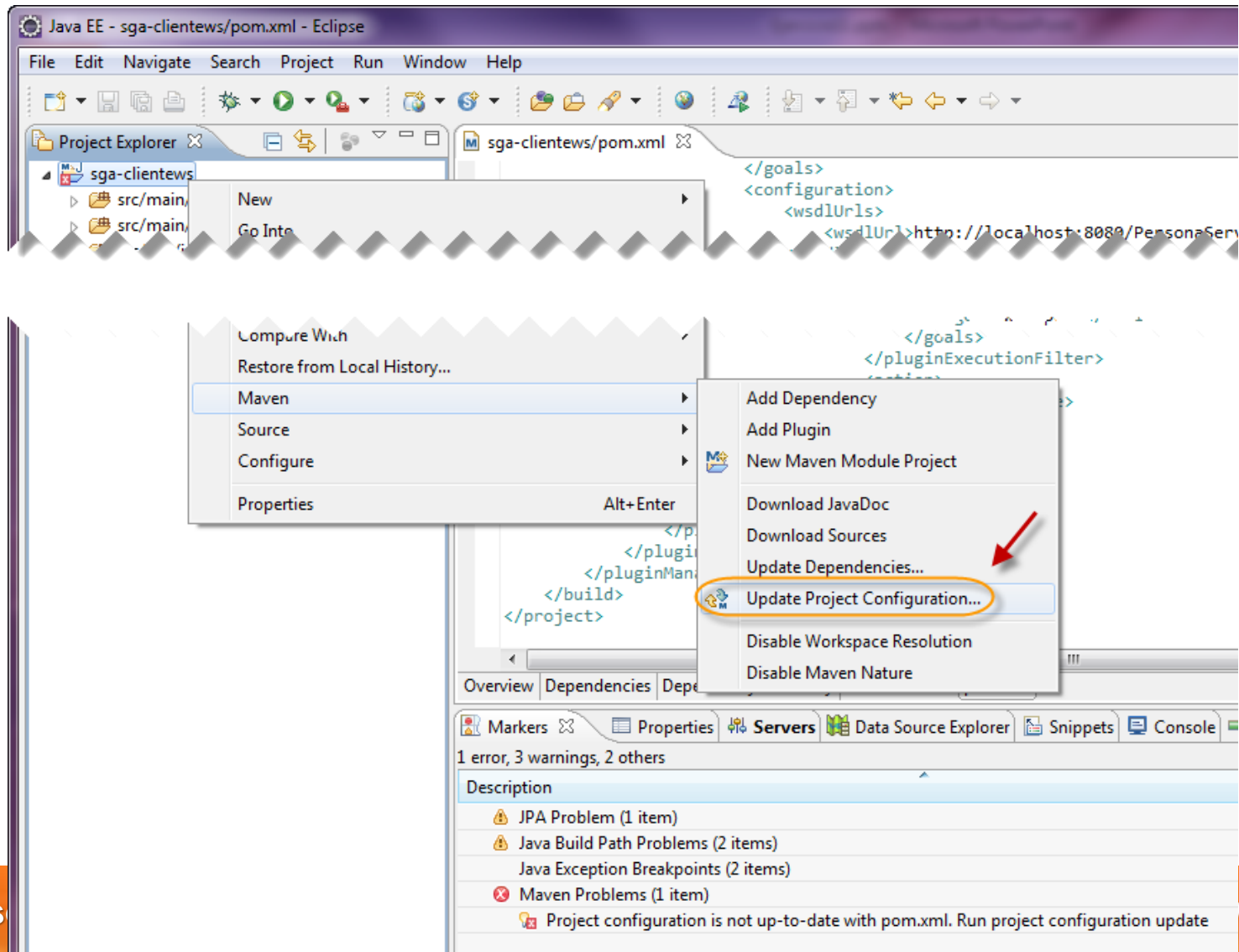

Paso 8. Configuración del archivo pom.xml (cont)

Omitimos el warning que manda el archivo pom.xml como sigue:



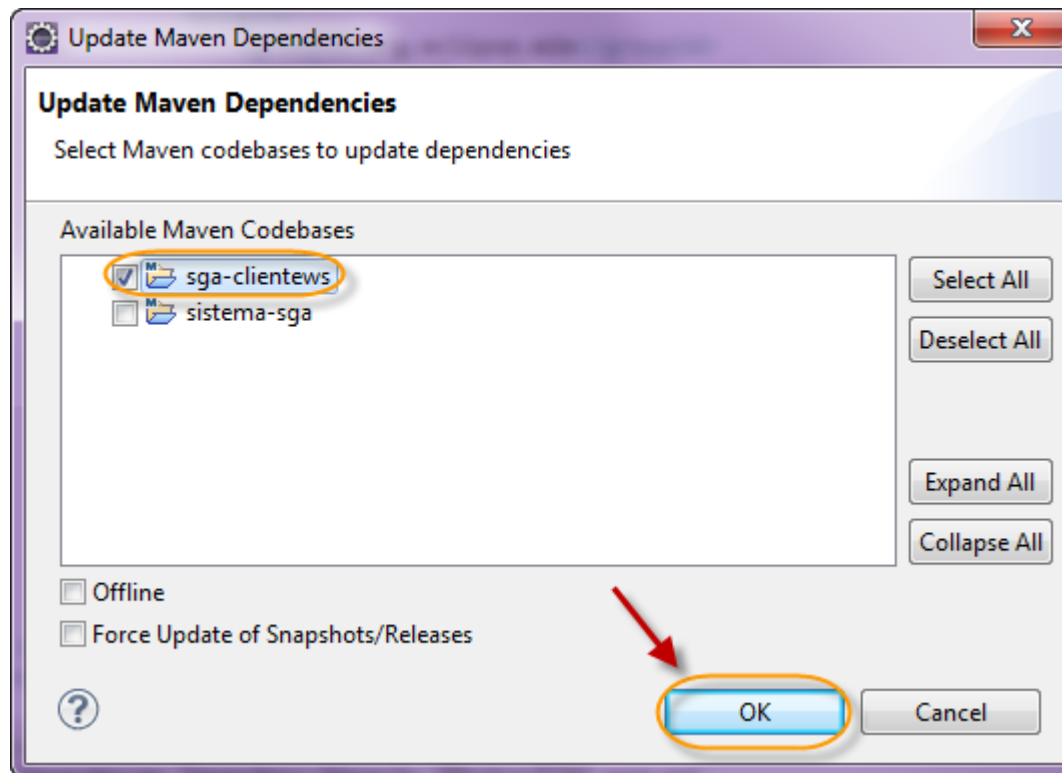
Paso 9. Actualización proyecto Maven

Una vez modificado el archivo pom.xml, debemos actualizarlo como sigue:



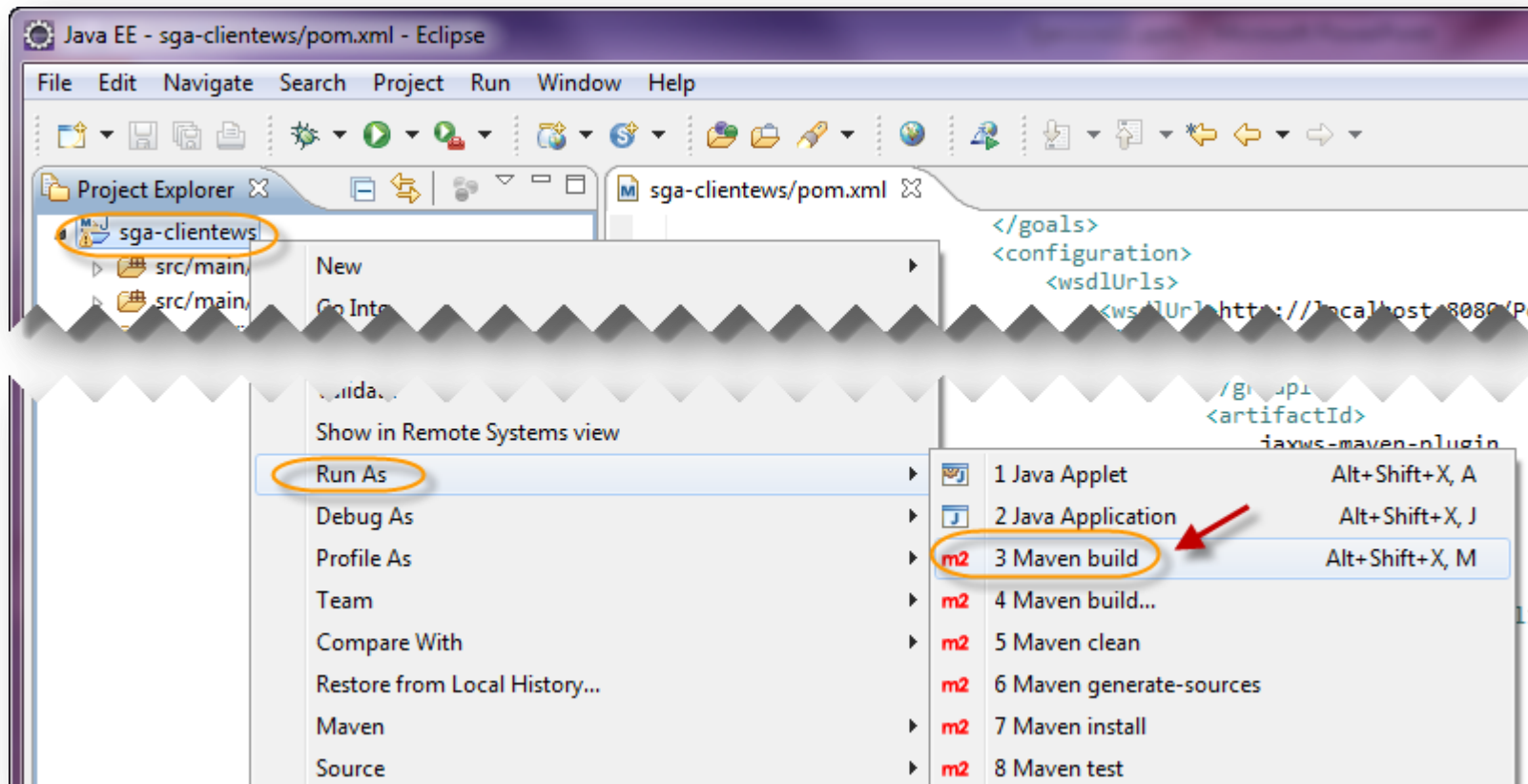
Paso 9. Actualización proyecto Maven (cont)

Con esto ya se debería eliminar cualquier error o warning del proyecto:



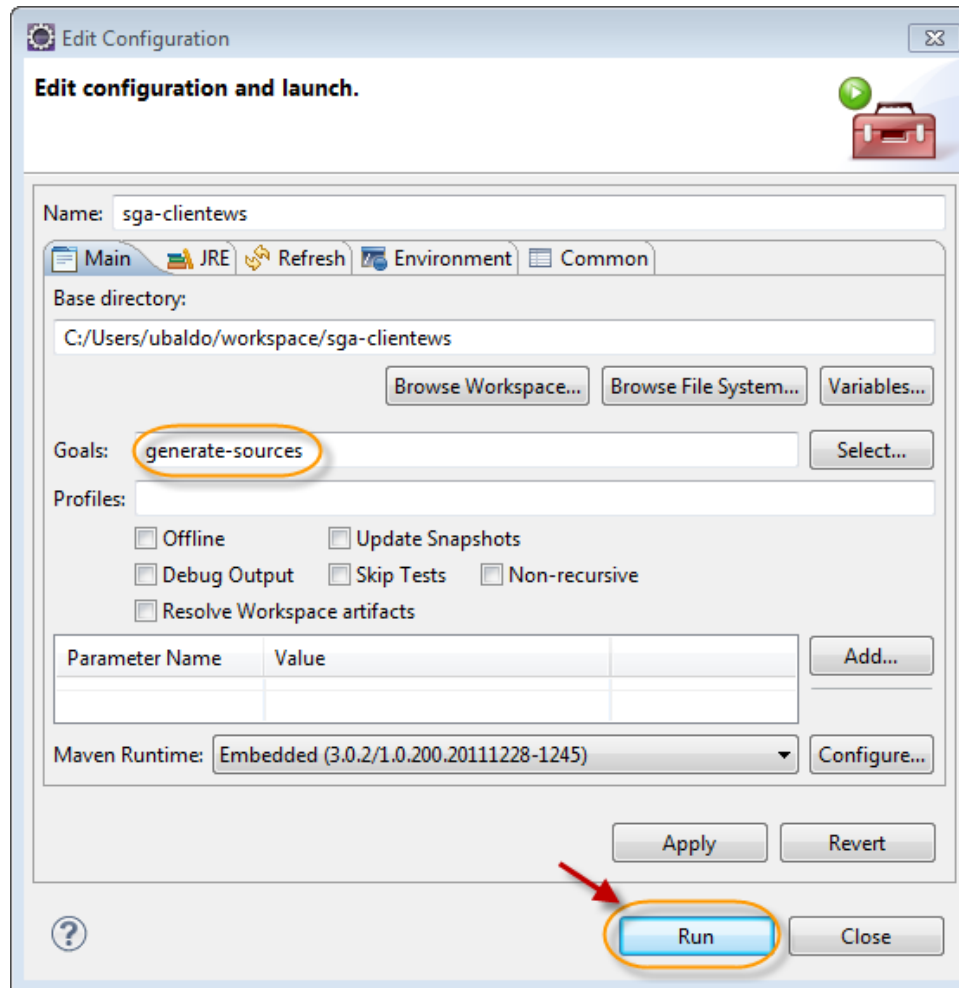
Paso 10. Generación del Cliente SGA-ClienteWS

Una vez ya configurado el proyecto Maven, ejecutamos la aplicación e indicamos que genere el código utilizando el WSDL del archivo pom.xml:



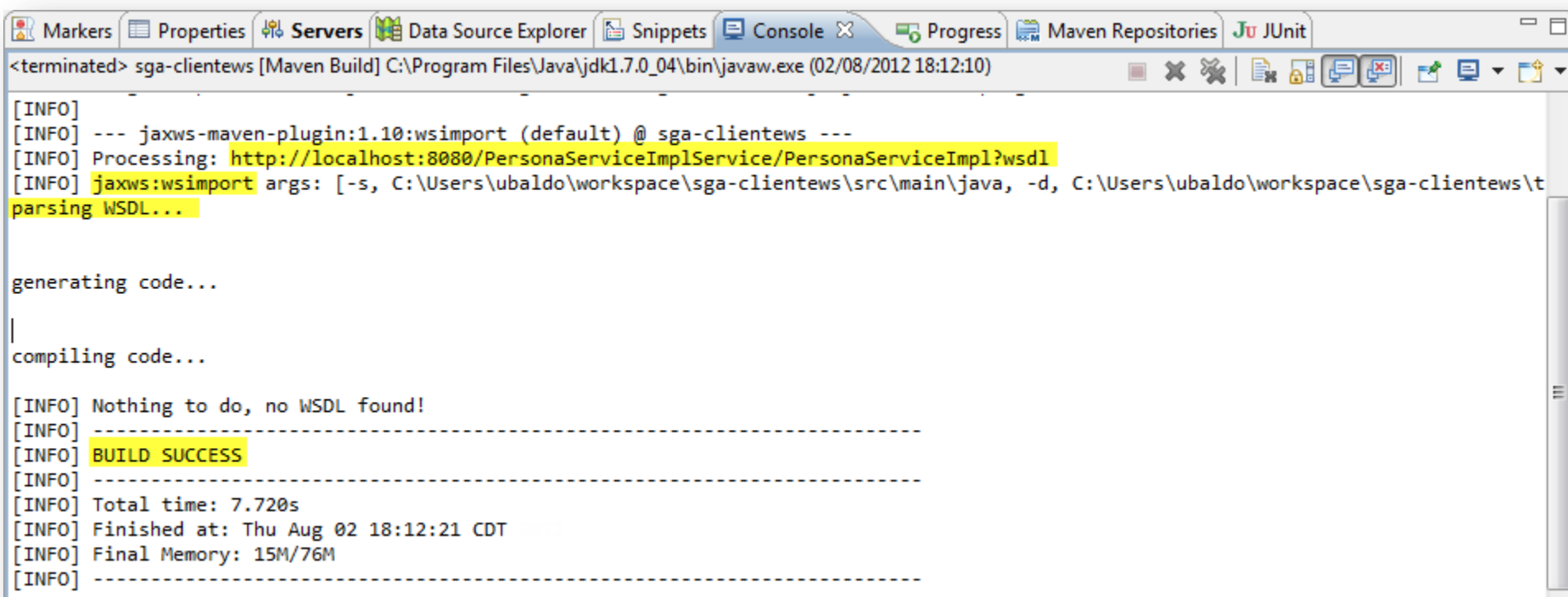
Paso 10. Generación del Cliente SGA-ClienteWS (cont)

Escribimos generate-sources en el campo de **Goals**, el cual es parte del ciclo de vida de Maven. Con esto se generará el código Java del Cliente:



Paso 10. Generación del Cliente SGA-ClienteWS (cont)

Al ejecutar deberos recibir una salida similar a la siguiente, si recibimos el mensaje ***Nothing todo, no WSDL found***, es normal, siempre y cuando se genere el código Java del cliente.



```
<terminated> sga-clientews [Maven Build] C:\Program Files\Java\jdk1.7.0_04\bin\javaw.exe (02/08/2012 18:12:10)

[INFO]
[INFO] --- jaxws-maven-plugin:1.10:wsimport (default) @ sga-clientews ---
[INFO] Processing: http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?wsdl
[INFO] jaxws:wsimport args: [-s, C:\Users\ubaldo\workspace\sga-clientews\src\main\java, -d, C:\Users\ubaldo\workspace\sga-clientews\t
parsing WSDL...

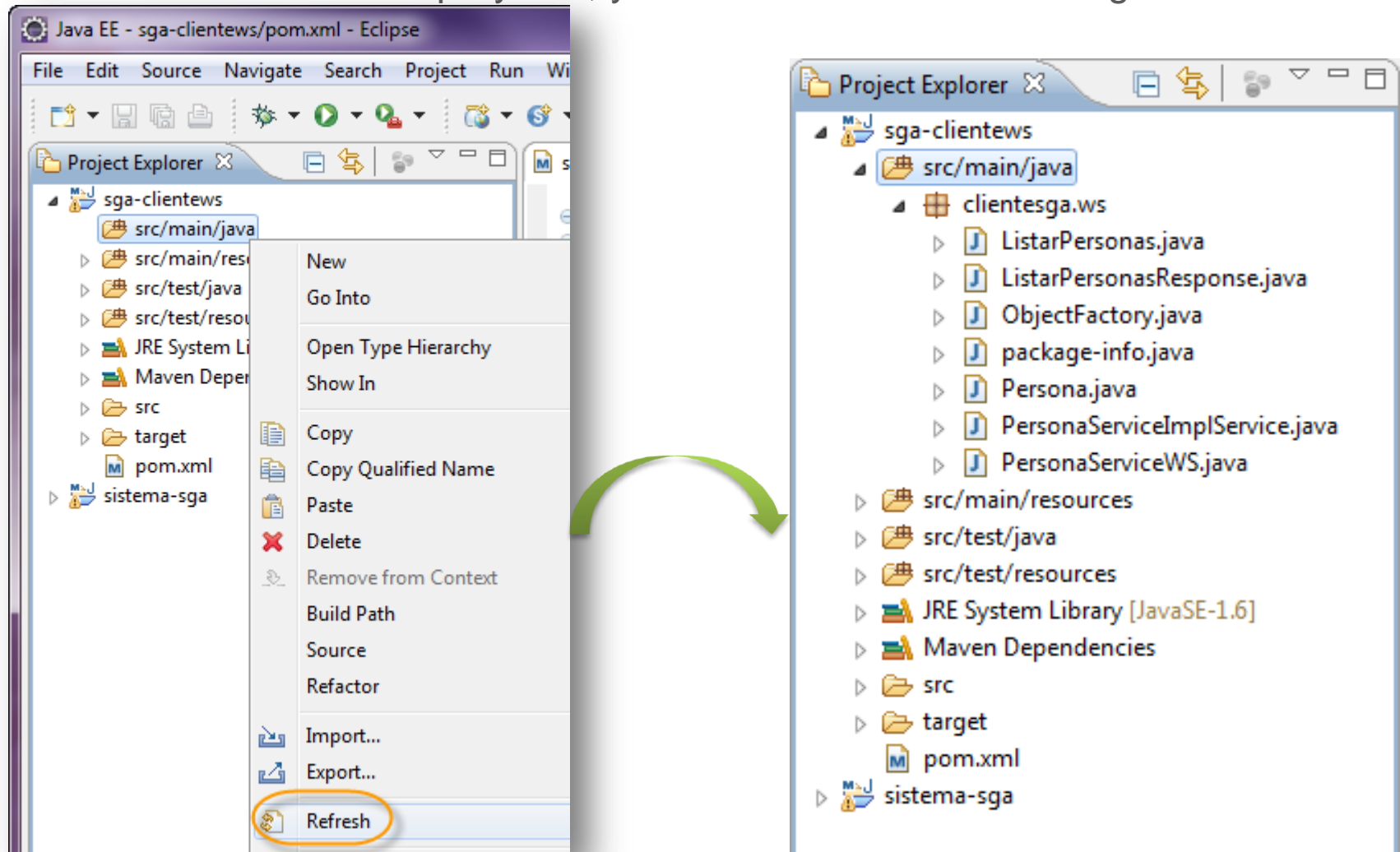
generating code...

compiling code...

[INFO] Nothing to do, no WSDL found!
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.720s
[INFO] Finished at: Thu Aug 02 18:12:21 CDT
[INFO] Final Memory: 15M/76M
[INFO] -----
```

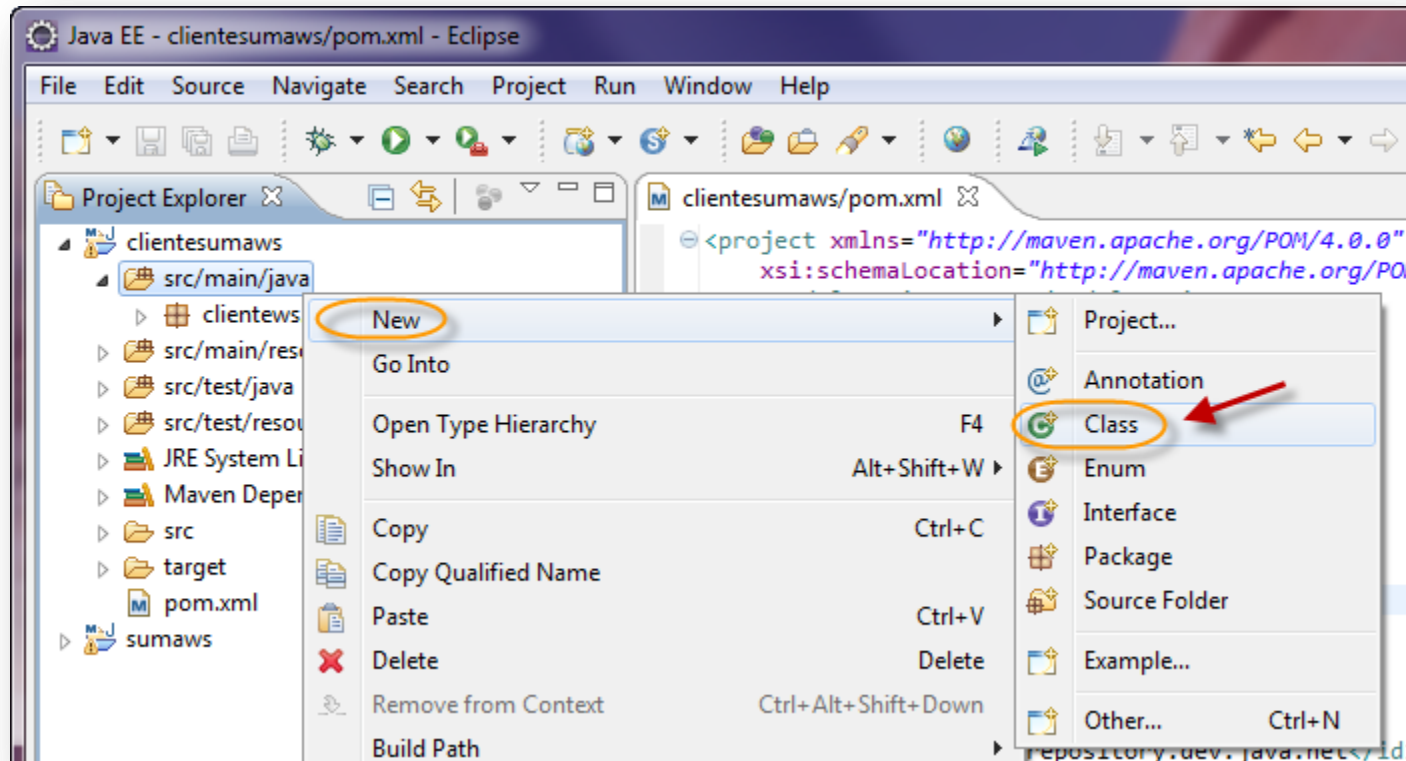
Paso 10. Generación del Cliente SGA-ClienteWS (cont)

Al hacer refresh sobre el proyecto, ya debería mostrarse el código del cliente:



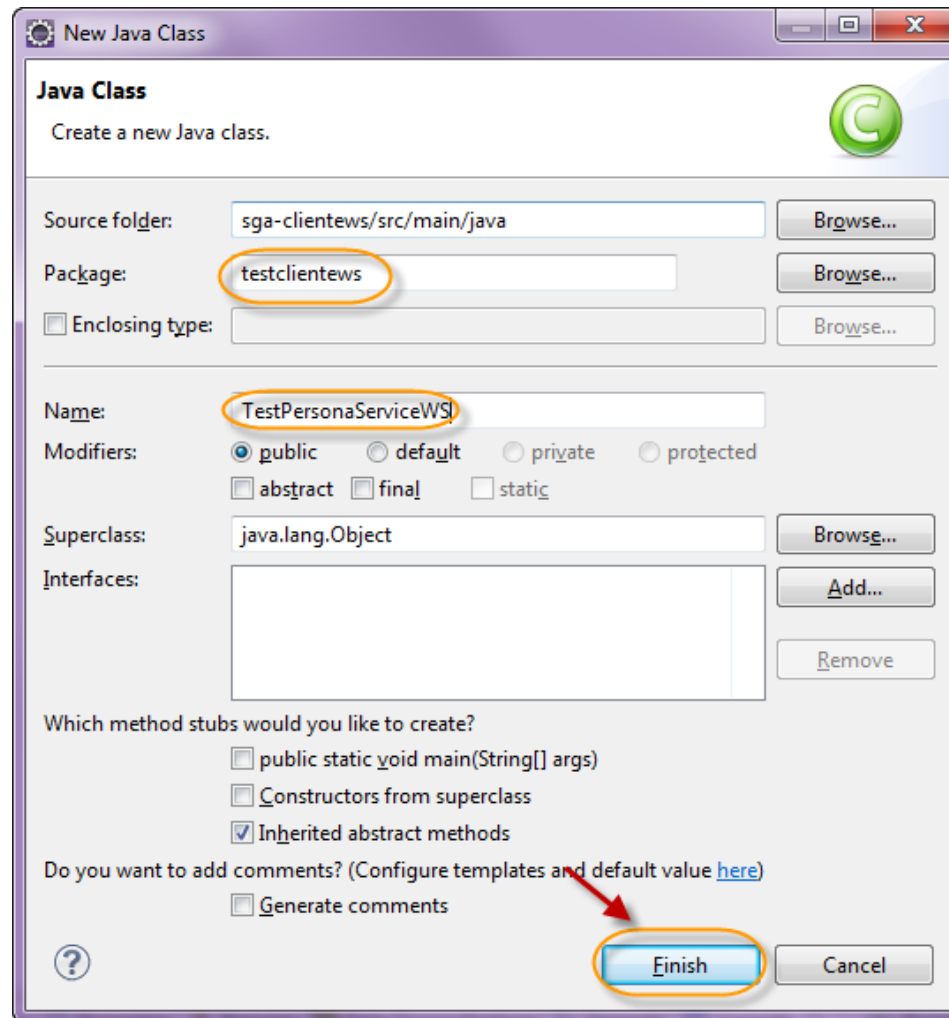
Paso 11. Creación de la clase TestPersonaServiceWS

Creamos la clase TestPersonaServiceWS.java:



Paso 11. Creación de la clase TestPersonaServiceWS(cont)

Creamos la clase TestPersonaServiceWS.java:



Paso 11. Creación de la clase TestPersonaServiceWS(cont)

Agregamos el siguiente código a la clase TestPersonaServiceWS.java:

```
package testclientews;

import java.util.List;
import clientesga.ws.Persona;
import clientesga.ws.PersonaServiceImplService;
import clientesga.ws.PersonaServiceWS;

public class TestPersonaServiceWS {

    public static void main(String[] args) {
        PersonaServiceWS personaService = new PersonaServiceImplService().getPersonaServiceImplPort();

        System.out.println("Ejecutando Servicio Listar Personas WS");

        List<Persona> personas = personaService.listarPersonas();

        for (Persona persona : personas) {
            System.out.println("Persona: " + persona.getNombre() + " " + persona.getApePaterno());
        }

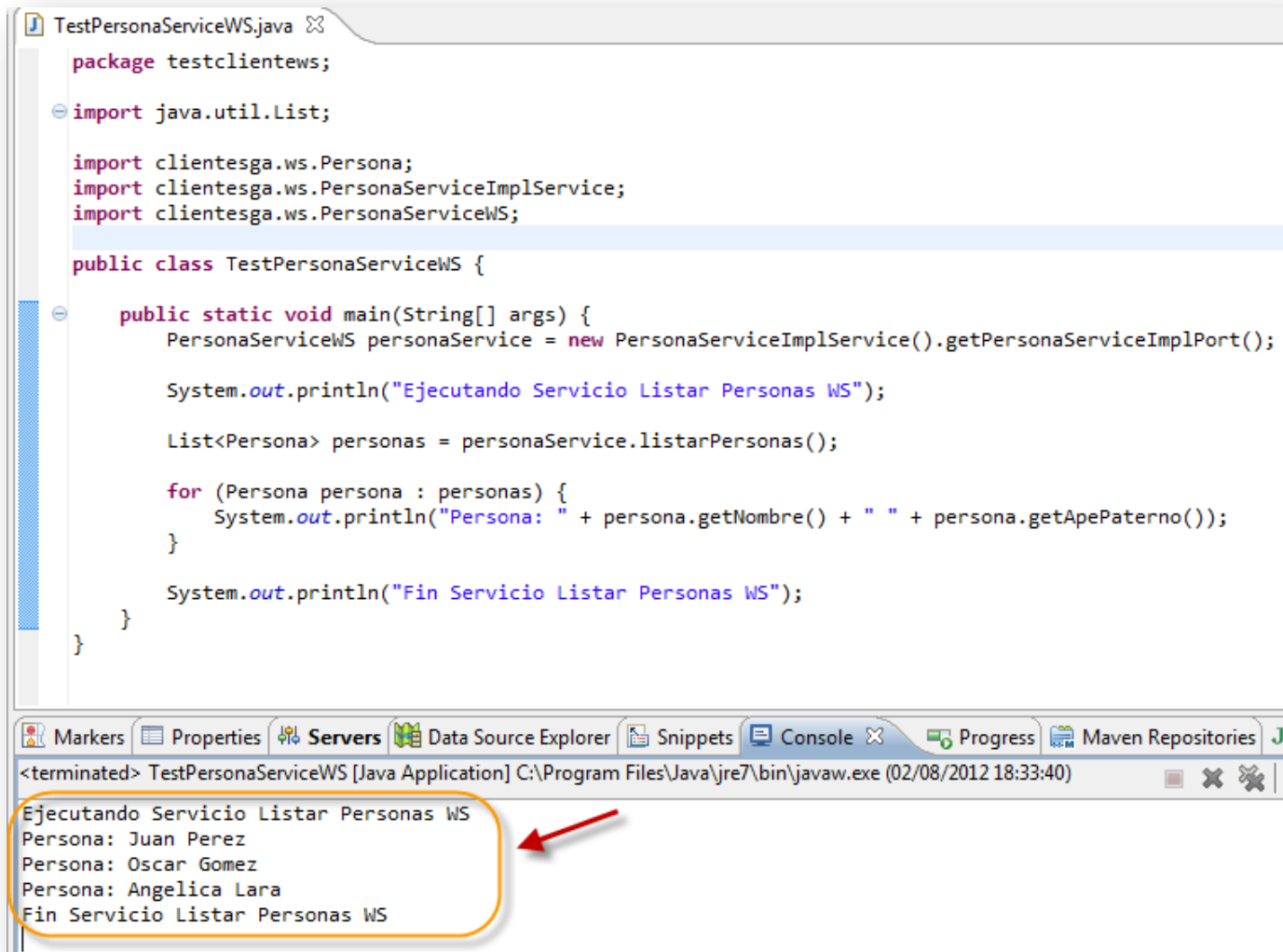
        System.out.println("Fin Servicio Listar Personas WS");
    }
}
```

Ejecutamos la clase `TestServicioSumarWS.java`:



Paso 12. Ejecución clase TestServicioSumarWS (cont)

Ejecutamos la clase TestServicioSumarWS.java:



```
TestPersonaServiceWS.java
package testclientews;

import java.util.List;

import clientesga.ws.Persona;
import clientesga.ws.PersonaServiceImplService;
import clientesga.ws.PersonaServiceWS;

public class TestPersonaServiceWS {

    public static void main(String[] args) {
        PersonaServiceWS personaService = new PersonaServiceImplService().getPersonaServiceImplPort();

        System.out.println("Ejecutando Servicio Listar Personas WS");

        List<Persona> personas = personaService.listarPersonas();

        for (Persona persona : personas) {
            System.out.println("Persona: " + persona.getNombre() + " " + persona.getApePaterno());
        }

        System.out.println("Fin Servicio Listar Personas WS");
    }
}
```

<terminated> TestPersonaServiceWS [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (02/08/2012 18:33:40)

Ejecutando Servicio Listar Personas WS
Persona: Juan Perez
Persona: Oscar Gomez
Persona: Angelica Lara
Fin Servicio Listar Personas WS



Conclusión

- ✔ Con este ejercicio pudimos observar cómo exponer los métodos del EJB's de PersonaService utilizando Web Services con el API de JAX-WS.
- ✔ Observamos cómo hacer un test del Web Service una vez desplegado sobre el servidor GlassFish.
- ✔ Además, revisamos cómo crear las clases Java del Cliente del Web Service a partir del documento WSDL, todo esto con ayuda del comando ***wsimport***, el cual es parte del JDK de Java.
- ✔ Con esto hemos visto el proceso completo de cómo crear un SOAP Web Service y el cliente respectivo.
- ✔ Además, vimos cómo excluir algunos atributos a enviar utilizando JAXB en nuestra clase de dominio Persona.
- ✔ Por último y lo más importante, es que utilizamos una arquitectura de 3 capas y vimos cómo exponer la lógica de negocio como un servicio Web a cualquier cliente interesado en la información del sistema.



Laboratorio

Se deja como laboratorio agregar los siguientes métodos como parte de la interfaz del Web Service:

- ✓ registrarPersona
- ✓ modificarPersona
- ✓ eliminarPersona

Se debe modificar además el cliente para hacer un test de cada método y visualizar vía la aplicación web que se hayan registrado, modificado o eliminado correctamente los objetos de tipo Persona.



www.globalmentoring.com.mx

Pasión por la tecnología Java

Experiencia y Conocimiento para tu vida