



## Ejercicio 11

Creación Proyecto HolaMundo con Web Services

## Objetivo del Ejercicio

- El objetivo del ejercicio crear nuestro primer Web Service con JAX-WS y EJBs de tipo Stateless Session Bean. El resultado se muestra a continuación:

The screenshot displays a web browser window with the address bar showing the URL: `servidor:8080/ServicioSumarImplService/ServicioSumarImpl?Tester`. The page title is "Method invocation trace". The main content area is titled "sumar Method invocation" and contains the following sections:

**Method parameter(s)**

| Type | Value |
|------|-------|
| int  | 1     |
| int  | 2     |

**Method returned**

int : "3"

**SOAP Request**

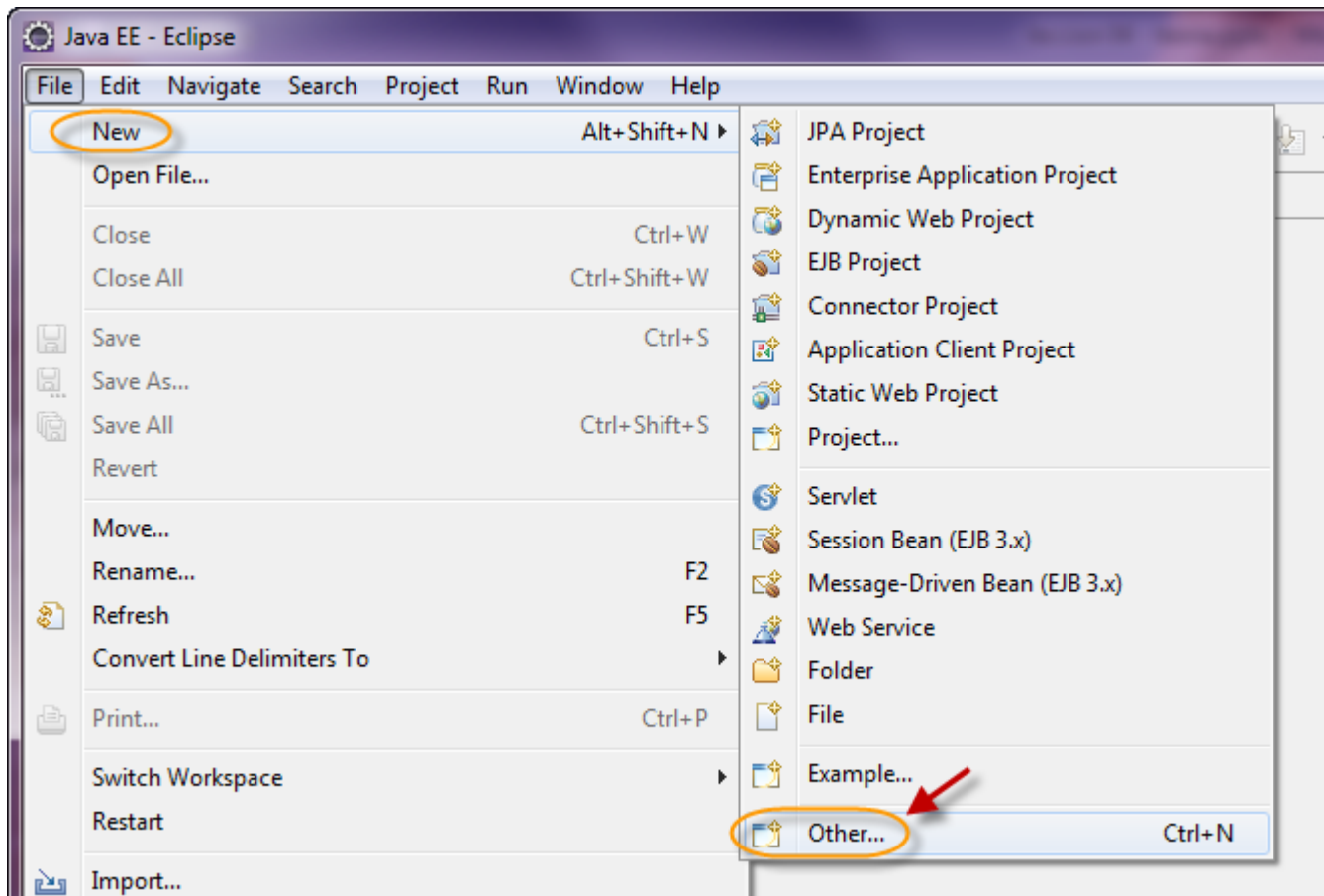
```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:sumar xmlns:ns2="http://beans.xmi.com/">
      <arg0>1</arg0>
      <arg1>2</arg1>
    </ns2:sumar>
  </S:Body>
</S:Envelope>
```

**SOAP Response**

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:sumarResponse xmlns:ns2="http://beans.xmi.com/">
      <return>3</return>
    </ns2:sumarResponse>
  </S:Body>
</S:Envelope>
```

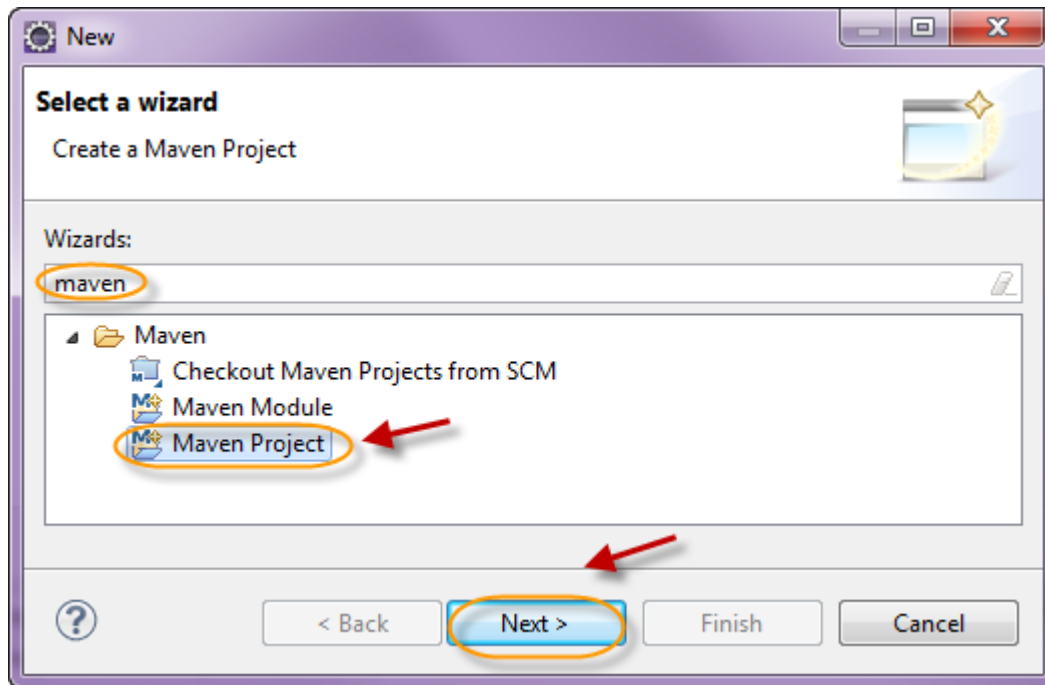
# Paso 1. Creación del proyecto SumaWS

Creamos el proyecto sumaws:



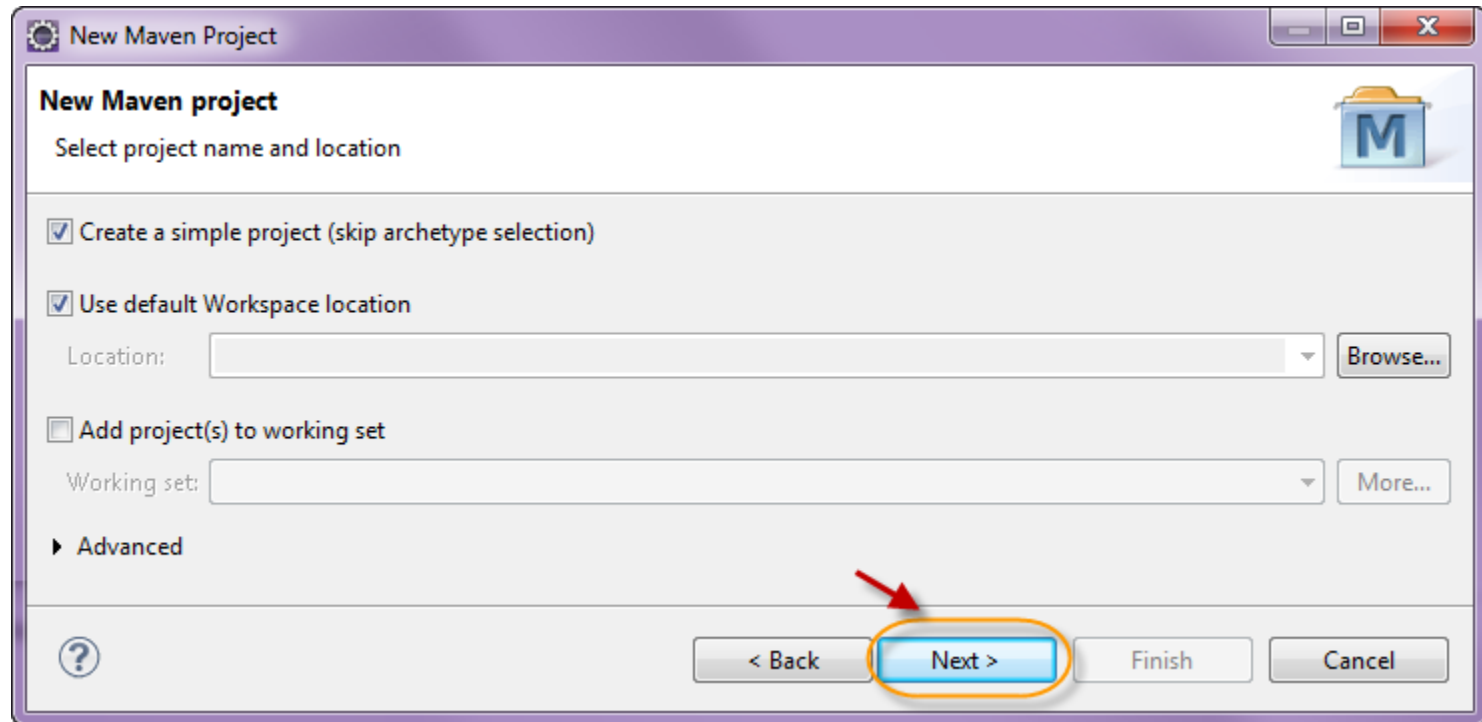
# Paso 1. Creación del proyecto SumaWS (cont)

Creamos el proyecto sumaws que es un Maven Project:



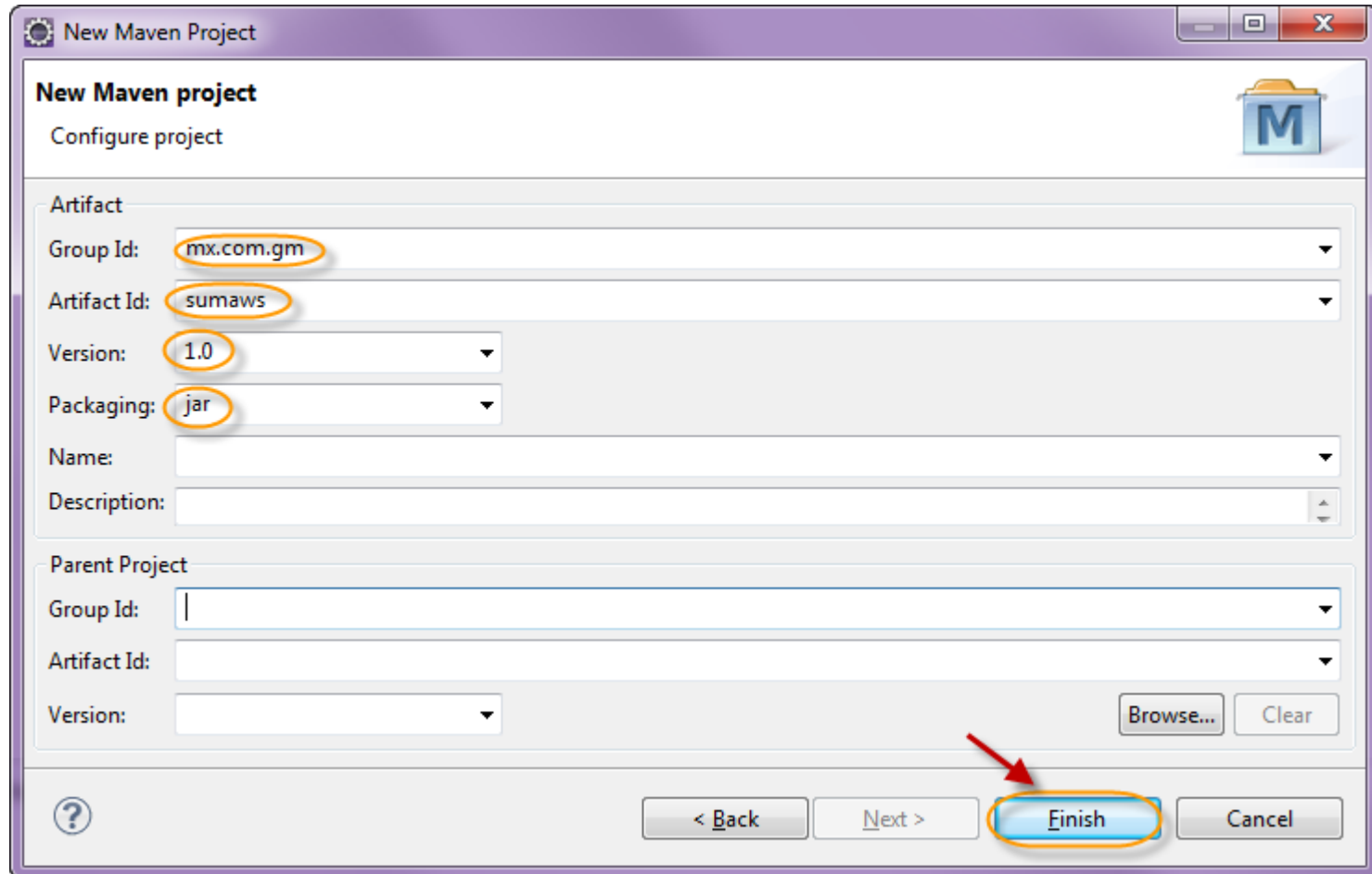
# Paso 1. Creación del proyecto SumaWS (cont)

Creamos el proyecto sumaws que es un Simple Project:



# Paso 1. Creación del proyecto SumaWS (cont)

Creamos el proyecto sumaws que es un Simple Project:



**New Maven Project**

Configure project

**Artifact**

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

**Parent Project**

Group Id:

Artifact Id:

Version:



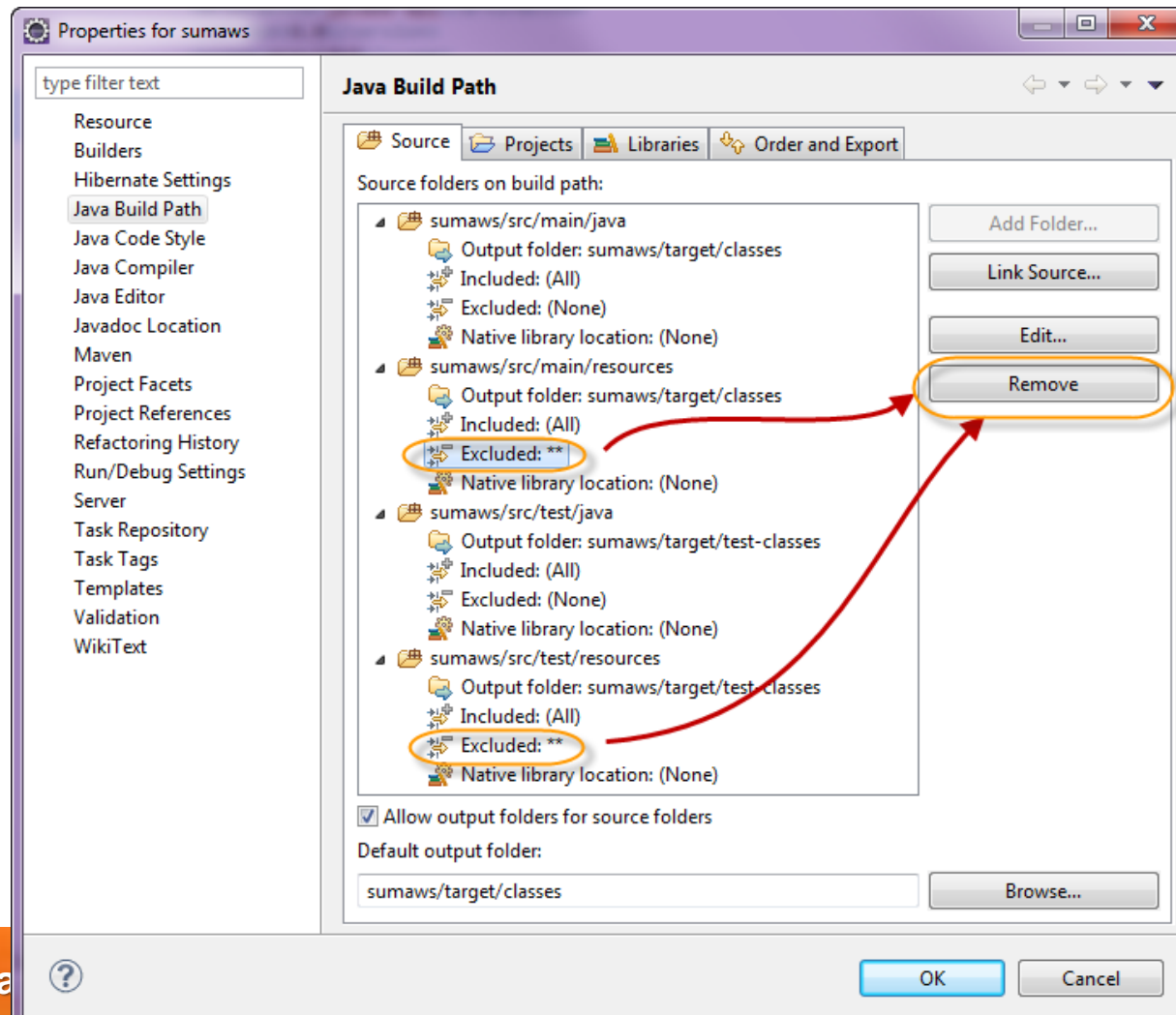
## Paso 2. Agregamos las dependencias Maven

Agregamos las dependencias y repositorios de Maven al proyecto sumaws (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>
</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>
```

## Paso 3. Configuramos el BuildPath

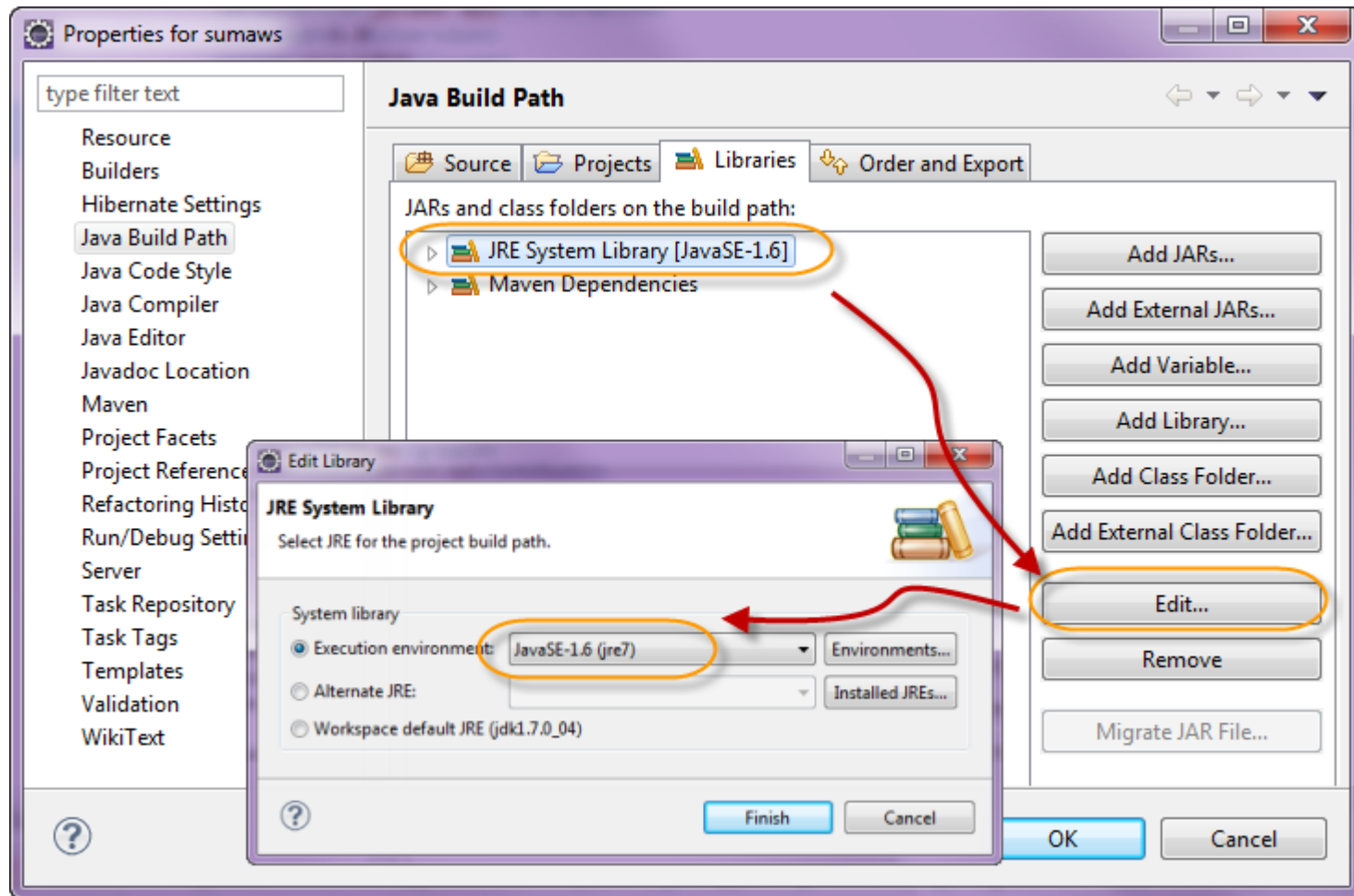
Entramos a las propiedades del proyecto y configuramos el buildpath para que no excluya ningún tipo de archivos:





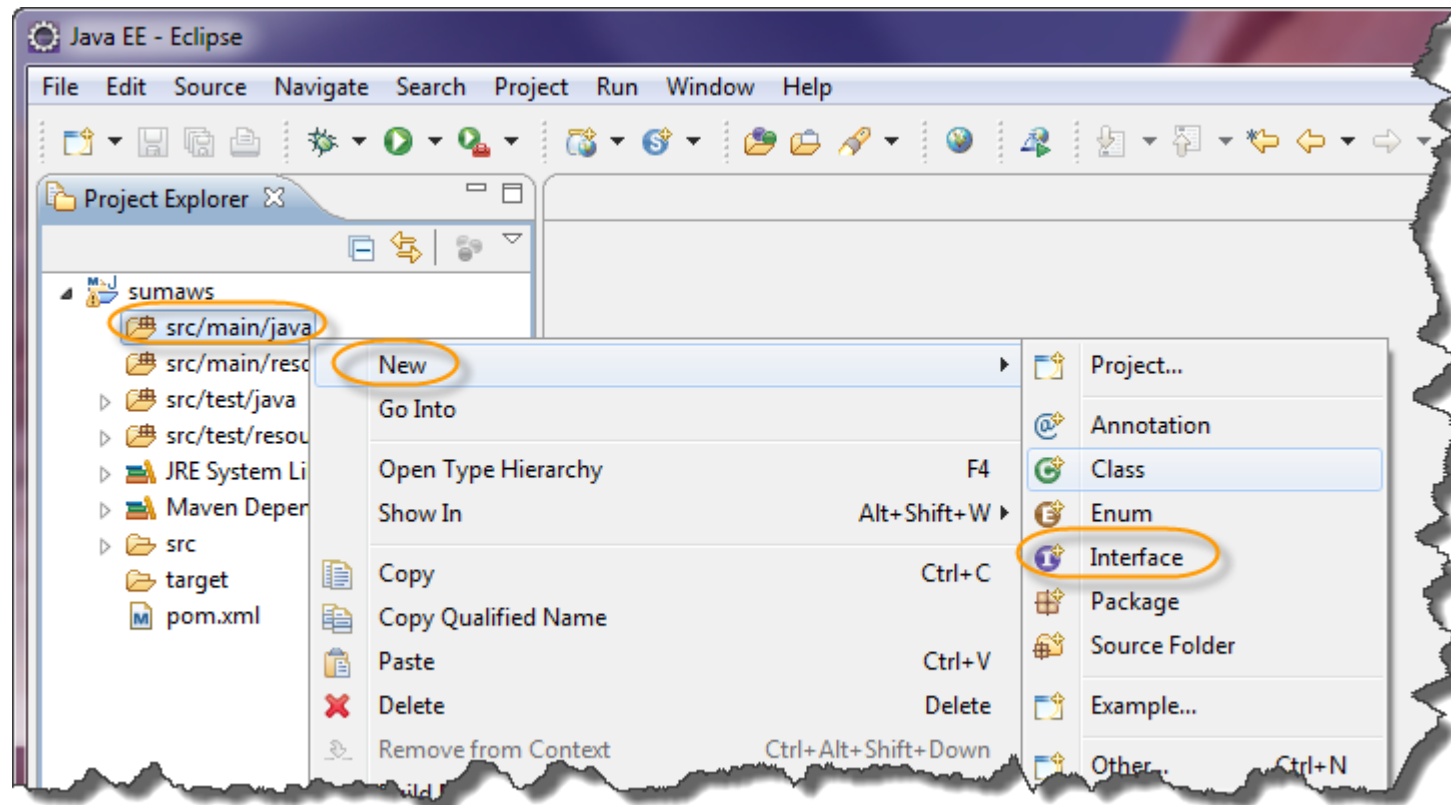
## Paso 3. Configuramos el BuildPath (cont)

Actualizamos la versión del JRE a la versión 6:



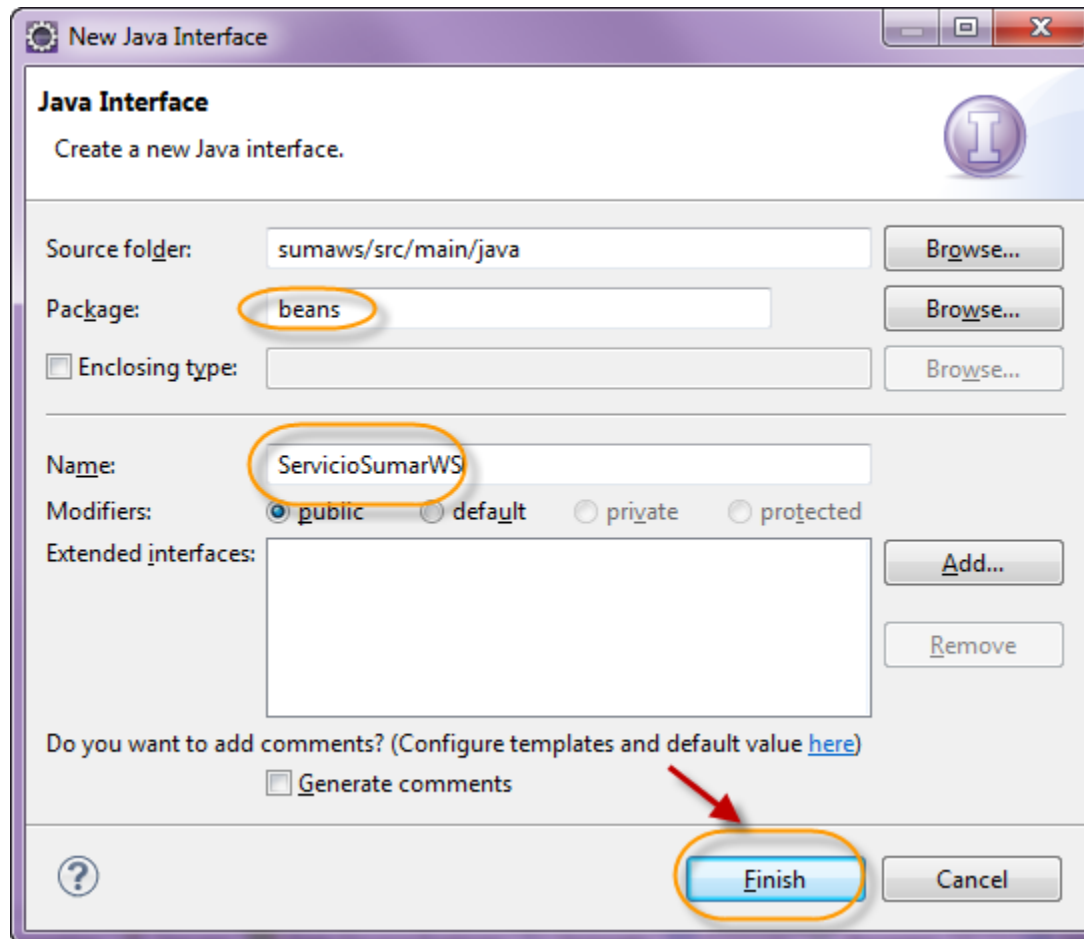
## Paso 4. Creación Interfaz ServicioSumarWS

Creamos la interfaz que utilizaremos para el Web Service:



## Paso 4. Creación Interfaz ServicioSumarWS (cont)

Creamos la interfaz que utilizaremos para el Web Service:





## Paso 4. Creación Interfaz ServicioSumarWS (cont)

Agregamos el siguiente contenido a la interfaz:

```
package beans;

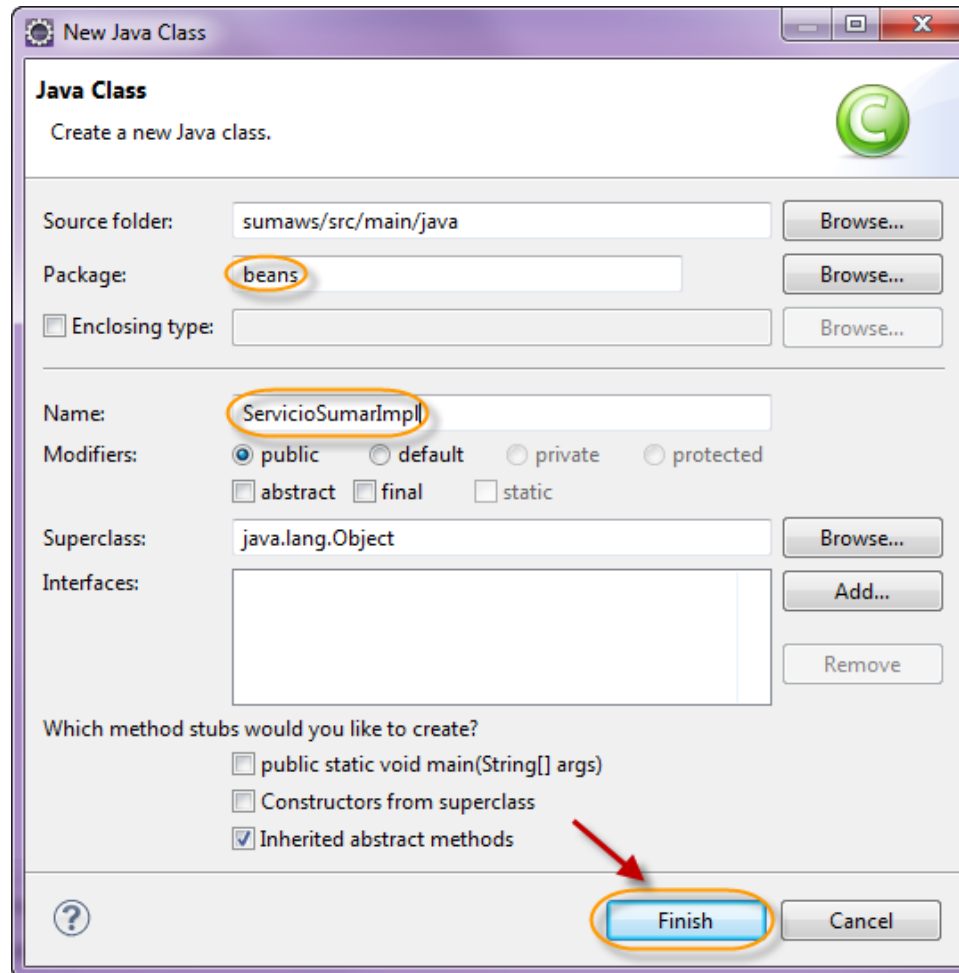
import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public interface ServicioSumarWS {

    @WebMethod
    public int sumar(int a, int b);
}
```

## Paso 5. Creación Clase ServicioSumarImpl

Creamos la clase ServicioSumarImpl:





## Paso 5. Creación Clase ServicioSumarImpl (cont)

Agregamos el contenido de la clase ServicioSumarImpl:

```
package beans;

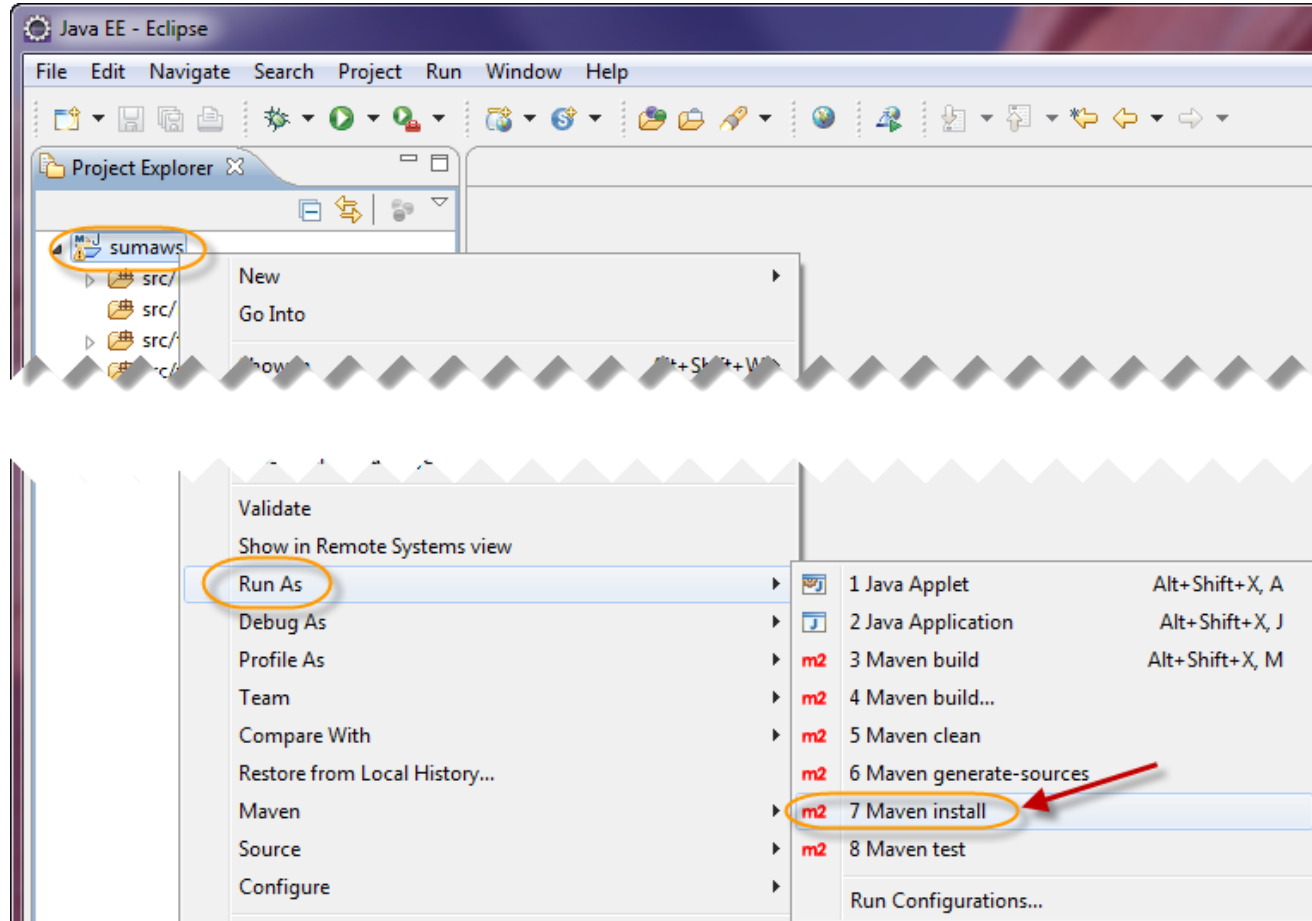
import javax.ejb.Stateless;
import javax.jws.WebService;

@Stateless
@WebService(endpointInterface = "beans.ServicioSumarWS")
public class ServicioSumarImpl implements ServicioSumarWS {

    @Override
    public int sumar(int a, int b) {
        return a + b;
    }
}
```


## Paso 6. Despliegue aplicación SumaWS

Creamos el archivo .jar como se muestra a continuación:



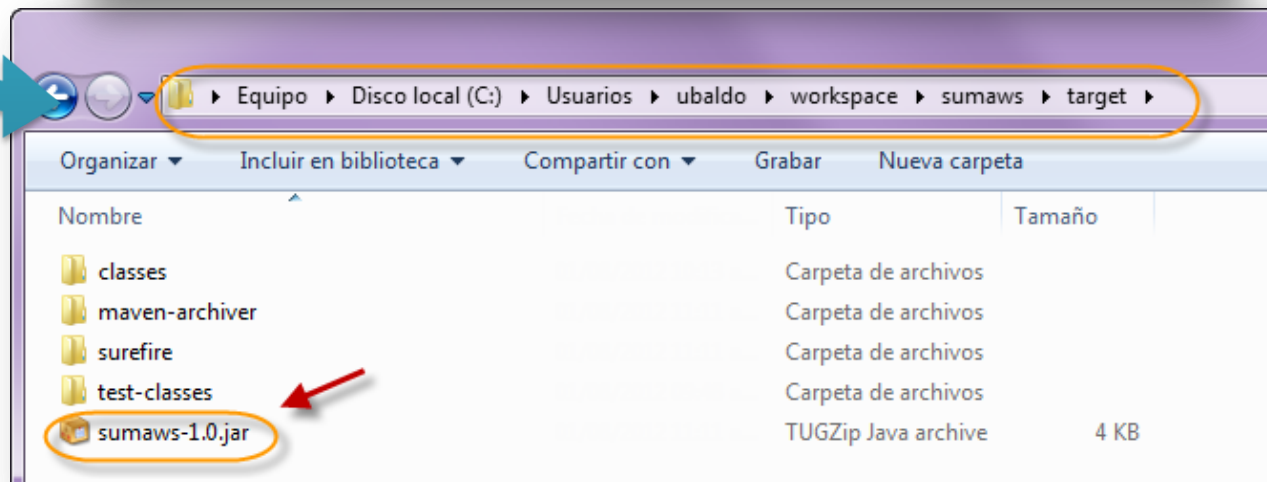
## Paso 6. Despliegue aplicación SumaWS (cont)

Seleccionamos la ruta donde se generó el archivo sumaws-1.0.jar y lo desplegamos sobre GlasshFish:



```
<terminated> C:\Program Files\Java\jre7\bin\javaw.exe (01/08/2012 11:11:25)

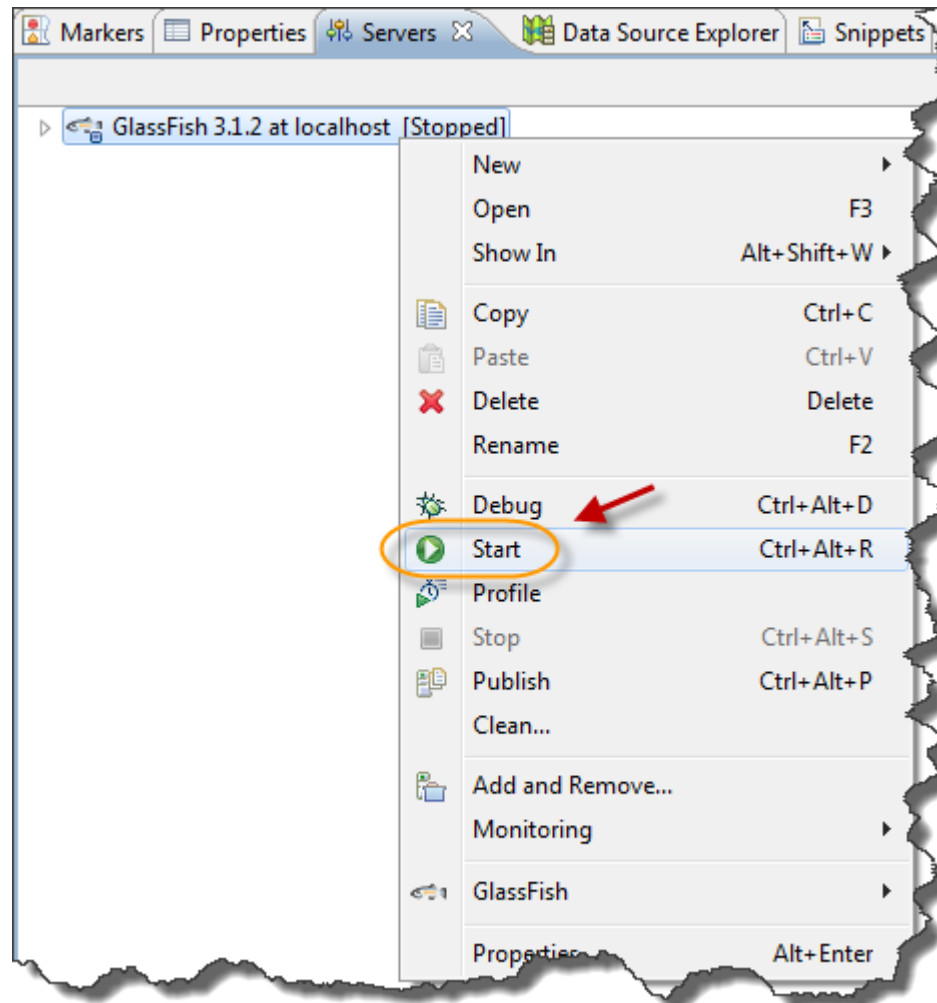
[INFO]
[INFO] --- maven-jar-plugin:2.3.1:jar (default-jar) @ sumaws ---
[INFO] Building jar: C:\Users\ubaldo\workspace\sumaws\target\sumaws-1.0.jar
[INFO]
[INFO] --- maven-install-plugin:2.3.1:install (default-install) @ sumaws ---
[INFO] Installing C:\Users\ubaldo\workspace\sumaws\target\sumaws-1.0.jar to \\vmware-host\S
[INFO] Installing C:\Users\ubaldo\workspace\sumaws\pom.xml to \\vmware-host\Shared Folders\
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 21.548s
[INFO] Finished at: Wed Aug 01 11:11:50 CDT
[INFO] Final Memory: 11M/76M
[INFO] -----
```





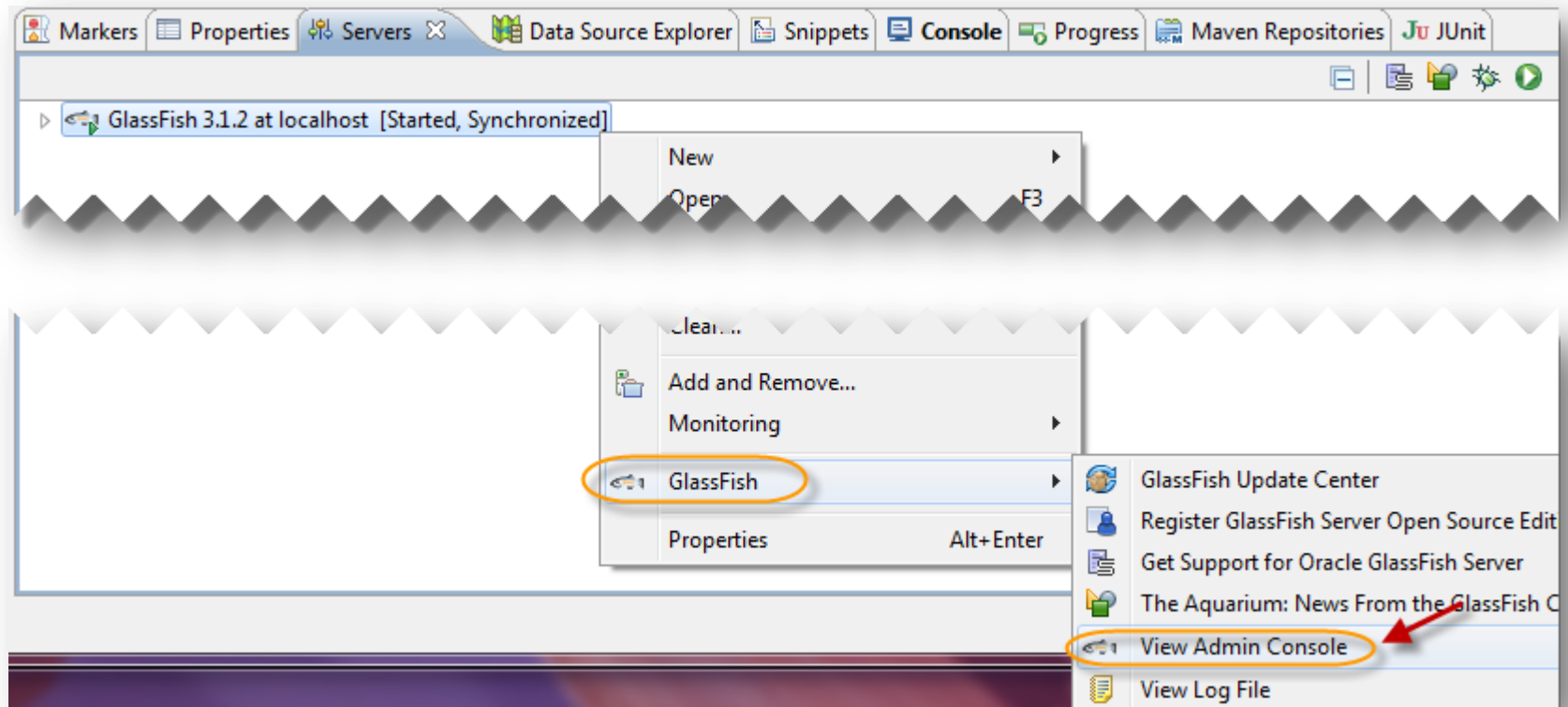
## Paso 6. Despliegue aplicación SumaWS (cont)

Levantamos el servidor de GlassFish, y desplegamos la aplicación (jar):



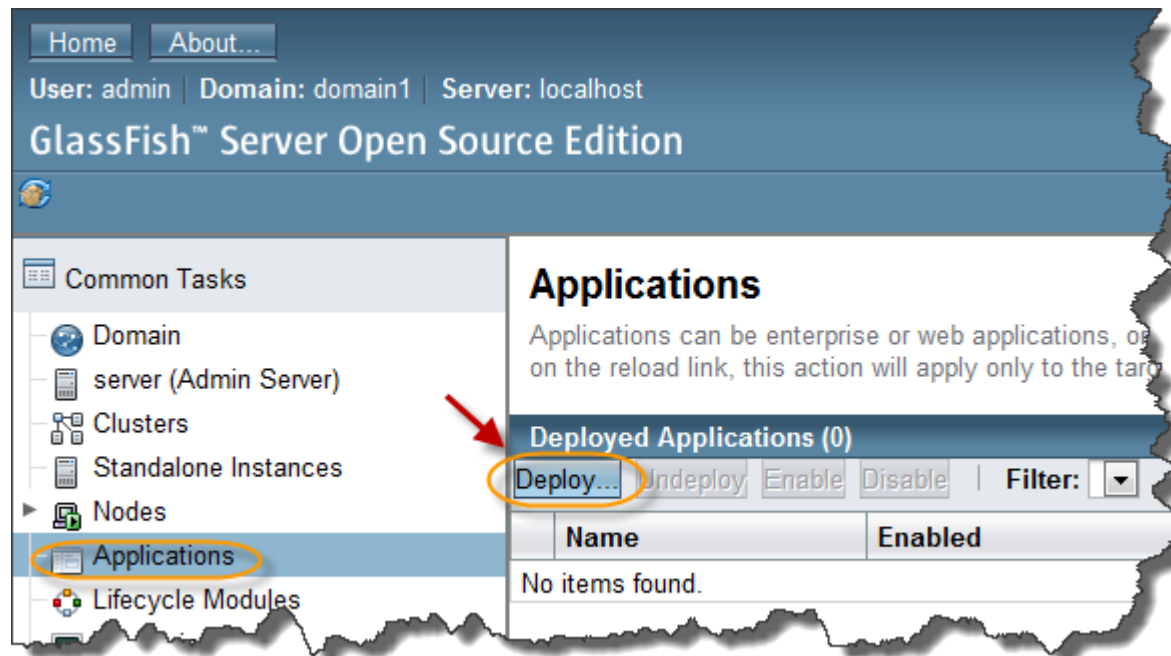
## Paso 6. Despliegue aplicación SumaWS (cont)

Entramos a la consola de GlassFish:



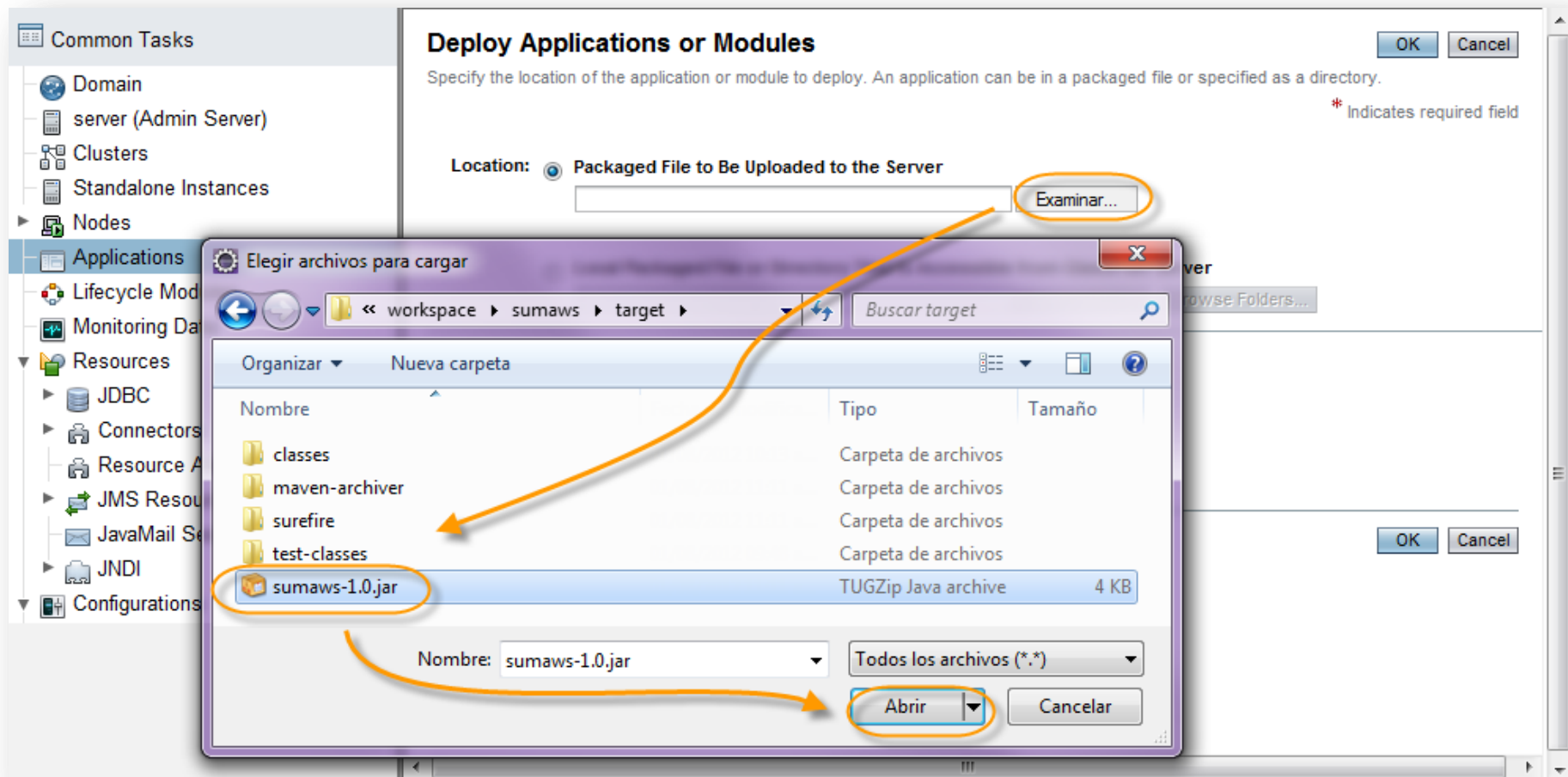
## Paso 6. Despliegue aplicación SumaWS (cont)

Entramos a la sección de aplicaciones de GlassFish. En nuestro caso hemos hecho undeploy de cualquier otra aplicación desplegada, para enfocarnos en el despliegue del Servicio Web, y así el log de GlassFish únicamente muestre información de esta nueva aplicación.



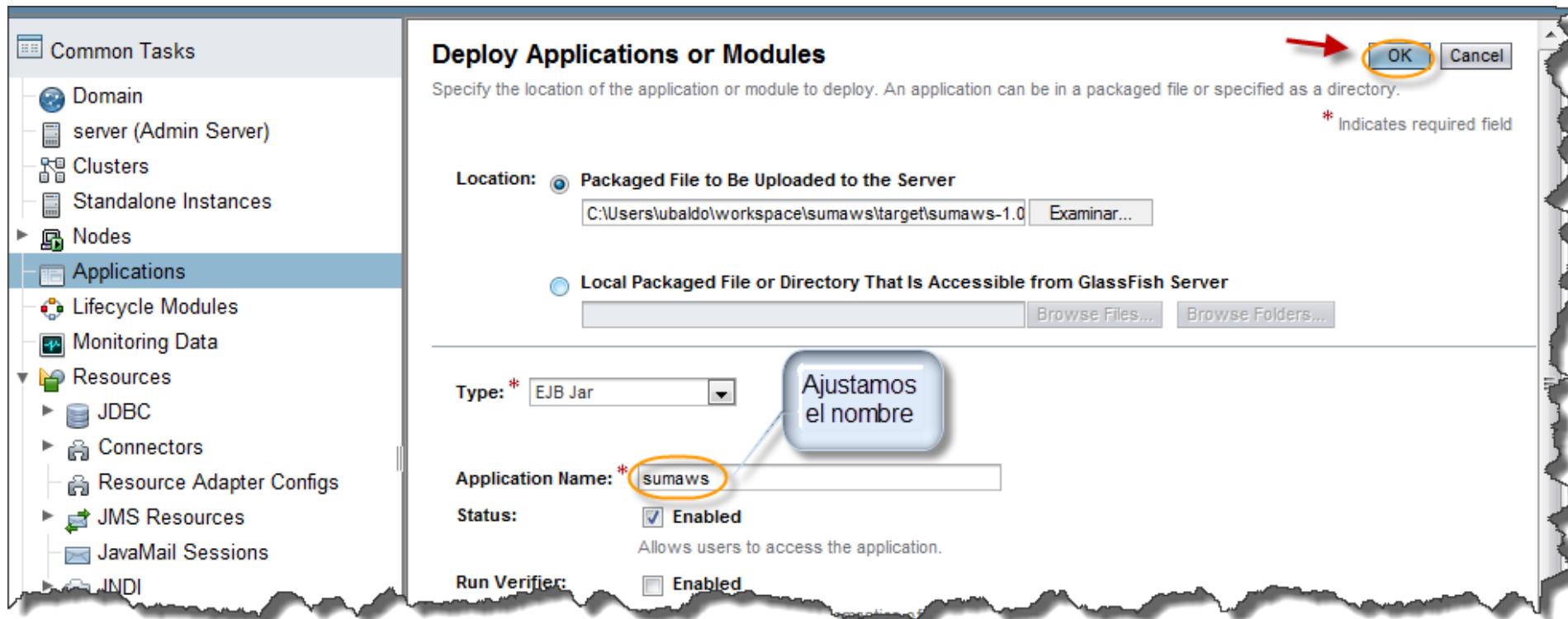
## Paso 6. Despliegue aplicación SumaWS (cont)

Con la ruta seleccionada anteriormente, seleccionamos el jar sumaws-1.0.jar:



## Paso 6. Despliegue aplicación SumaWS (cont)

Ajustamos el nombre de la aplicación a sumaws:



## Paso 7. Revisión del servicio SumaWS

Verificamos que se haya desplegado la aplicación sumaws y que incluye ejb y webservices como parte de los Engines detectados por GlassFish:



The screenshot shows the GlassFish Server Open Source Edition web console. The left sidebar contains a tree view with 'Applications' selected. The main content area shows the 'Applications' section with a table of deployed applications. The table has columns for Name, Enabled, Engines, and Action. One application, 'sumaws', is listed as deployed with engines 'ejb, webservices'. A red arrow points to the 'sumaws' name, and a yellow circle highlights the 'ejb, webservices' engines.

Home About...

User: admin | Role: domain1 | Server: localhost

GlassFish™ Server Open Source Edition

Total # of available updates : 44

Common Tasks

- Domain
- server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications**
- Lifecycle Modules
- Monitoring Data

### Applications

Applications can be enterprise or web applications, or various kinds of modules. Restart an application on the reload link, this action will apply only to the targets that the application or module is enabled on.

Deployed Applications (1)

Deploy... Undeploy Enable Disable Filter: ▼

| Name   | Enabled | Engines          | Action            |
|--------|---------|------------------|-------------------|
| sumaws | ✓       | ejb, webservices | Redeploy   Reload |

## Paso 7. Revisión del servicio SumaWS (cont)

Al dar clic sobre la aplicación nos muestra en la columna de Action un link llamado View Endpoint:

GlassFish™ Server Open Source Edition

Total # of available updates : 44

Common Tasks

- Domain
  - server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications**
  - sumaws**
- Lifecycle Modules
- Monitoring Data
- Resources
  - JDBC
  - Connectors
  - Resource Adapter Configs
  - JMS Resources
  - JavaMail Sessions
  - JNDI
- Configurations

**General** **Descriptor**

### Edit Application

Modify an existing application or module.

**Name:** sumaws

**Status:** ☒ Enabled

**Description:**

**Location:** \${com.sun.aas.instanceRootURI}/applications/sumaws/

**Libraries:**

**Modules and Components (2)**

| Module Name | Engines            | Component Name    | Type                 | Action                        |
|-------------|--------------------|-------------------|----------------------|-------------------------------|
| sumaws      | [ejb, webservices] | -----             | -----                |                               |
| sumaws      |                    | ServicioSumarImpl | StatelessSessionBean | <a href="#">View Endpoint</a> |

Save Cancel



## Paso 7. Revisión del servicio SumaWS (cont)

Al dar clic sobre el link de View Endpoint nos despliega la siguiente pantalla, son dos links para probar el Web Service: 1) Enviando un mensaje SOAP de prueba y 2) Un link para obtener el URL del documento WSDL. El URL del WSDL servirá para generar el cliente del Web Services:

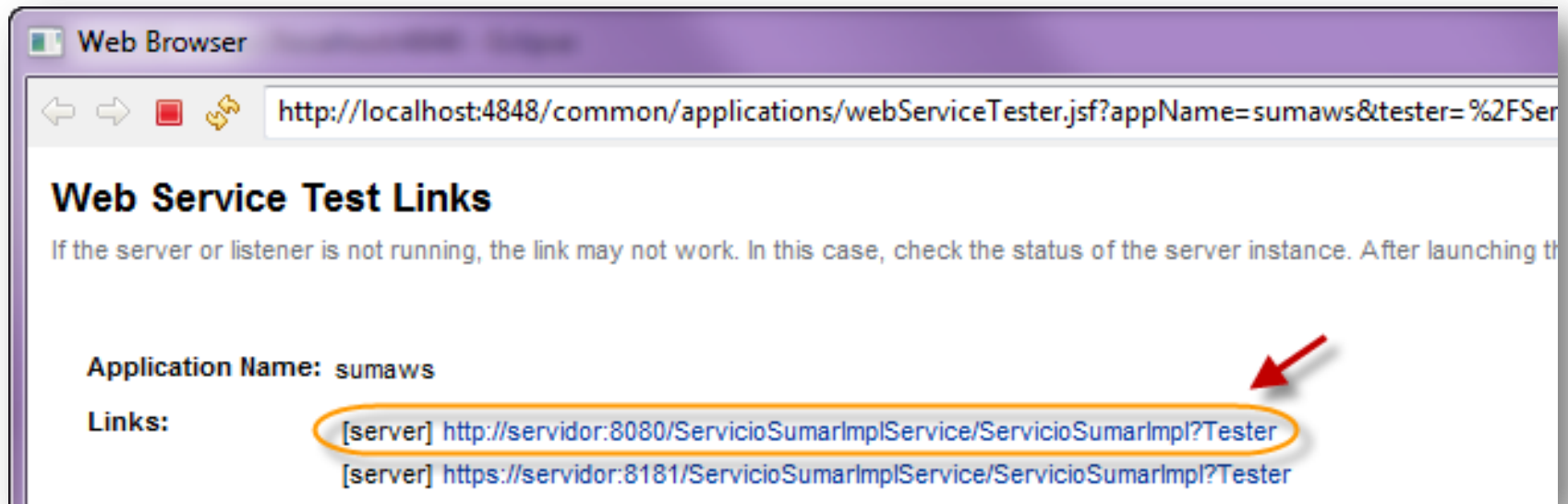
The screenshot shows the GlassFish Server Open Source Edition interface. The left sidebar contains a tree view with 'Applications' expanded, showing the 'sumaws' application selected. The main content area displays 'Web Service Endpoint Information' for the 'sumaws' service. The information includes the Application Name, Tester, WSDL, Endpoint Name, Service Name, Port Name, Deployment Type, Implementation Type, Implementation Class Name, Endpoint Address URI, and Namespace. The Tester and WSDL links are highlighted with an orange rounded rectangle.

| Property                   | Value   |
|----------------------------|---|
| Application Name:          | sumaws  |
| Tester:                    | <a href="/ServicioSumarImplService/ServicioSumarImpl?Tester">/ServicioSumarImplService/ServicioSumarImpl?Tester</a> |
| WSDL:                      | <a href="/ServicioSumarImplService/ServicioSumarImpl?wsdl">/ServicioSumarImplService/ServicioSumarImpl?wsdl</a>     |
| Endpoint Name:             | ServicioSumarImpl   |
| Service Name:              | ServicioSumarImplService  |
| Port Name:                 | ServicioSumarImplPort   |
| Deployment Type:           | 109   |
| Implementation Type:       | EJB   |
| Implementation Class Name: | beans.ServicioSumarImpl   |
| Endpoint Address URI:      | /ServicioSumarImplService/ServicioSumarImpl   |
| Namespace:                 | http://beans/   |



## Paso 7. Revisión del servicio SumaWS (cont)

Si damos clic en el link de Tester nos despliega la siguiente pantalla:



## Paso 7. Revisión del servicio SumaWS (cont)

Si damos clic en la prueba de Web Service utilizando el protocolo http obtenemos la siguiente pantalla. Proporcionamos los valores 1 y 2 para realizar la prueba de la suma y damos clic en el botón del método **sumar**.

Web Browser

http://servidor:8080/ServicioSumarImplService/ServicioSumarImpl?Tester

### ServicioSumarImplService Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

**Methods :**

`public abstract int beans.ServicioSumarWS.sumar(int,int)`

sumar ( 1 , 2 )

## Paso 7. Revisión del servicio SumaWS (cont)

El resultado de la llamada al Web Services sumar tiene como resultado:

The screenshot shows a web browser window with the URL `http://servidor:8080/ServicioSumaImplService/ServicioSumaImpl?Tester`. The page displays the results of a SOAP method invocation for the `sumar` service.

**sumar Method invocation**

Method parameter(s)

| Type | Value |
|------|-------|
| int  | 1     |
| int  | 2     |

**Method returned**

int : "3"

**SOAP Request**

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:sumar xmlns:ns2="http://beans/">
      <arg0>1</arg0>
      <arg1>2</arg1>
    </ns2:sumar>
  </S:Body>
</S:Envelope>
```

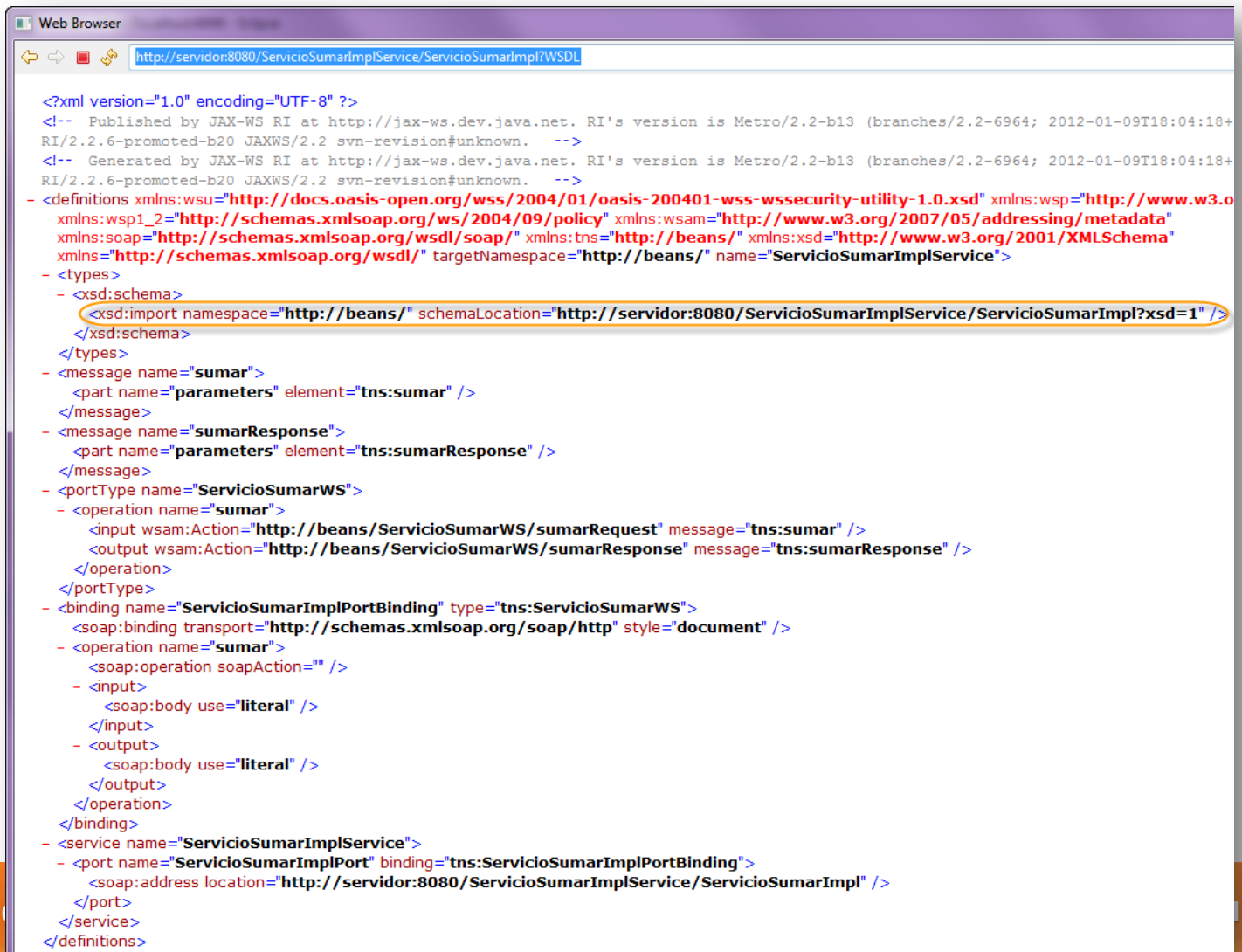
**SOAP Response**

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:sumarResponse xmlns:ns2="http://beans/">
      <return>3</return>
    </ns2:sumarResponse>
  </S:Body>
</S:Envelope>
```

## Paso 7. Revisión del servicio SumaWS (cont)

Por último, el documento WSDL generado se encuentra en la siguiente URL:

<http://servidor:8080/ServicioSumarImplService/ServicioSumarImpl?WSDL>

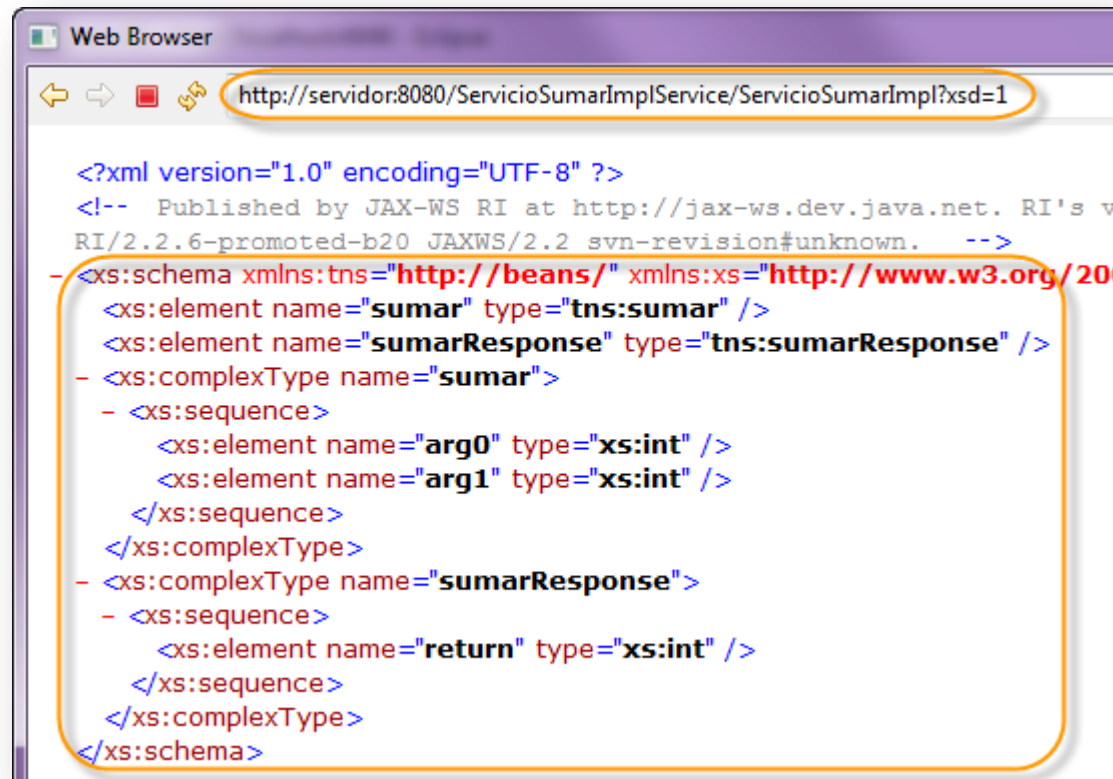


The screenshot shows a web browser window with the address bar displaying the URL: `http://servidor:8080/ServicioSumarImplService/ServicioSumarImpl?WSDL`. The main content area displays the WSDL XML document. The document is an XML Schema Definition (XSD) for a service named `ServicioSumarImplService`. It includes various namespaces and defines a port type `ServicioSumarWS` with two operations: `sumar` and `sumarResponse`. The `sumar` operation has an input parameter `parameters` of type `tns:sumar`. The `sumarResponse` operation has an output parameter `parameters` of type `tns:sumarResponse`. The document also defines a binding `ServicioSumarImplPortBinding` for the `sumar` operation, which uses a SOAP binding with a document style. The `sumar` operation is defined with a SOAP action of `http://beans/ServicioSumarWS/sumarRequest` and a message of `tns:sumar`. The `sumarResponse` operation is defined with a SOAP action of `http://beans/ServicioSumarWS/sumarResponse` and a message of `tns:sumarResponse`. The document also defines a service `ServicioSumarImplService` with a port `ServicioSumarImplPort` that binds to the `ServicioSumarImplPortBinding` and has a SOAP address location of `http://servidor:8080/ServicioSumarImplService/ServicioSumarImpl`.

```
<?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+
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+
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:wsp="http://www.w3.o
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://beans/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://beans/" name="ServicioSumarImplService">
- <types>
- <xsd:schema>
- <xsd:import namespace="http://beans/" schemaLocation="http://servidor:8080/ServicioSumarImplService/ServicioSumarImpl?xsd=1" />
</xsd:schema>
</types>
- <message name="sumar">
<part name="parameters" element="tns:sumar" />
</message>
- <message name="sumarResponse">
<part name="parameters" element="tns:sumarResponse" />
</message>
- <portType name="ServicioSumarWS">
- <operation name="sumar">
<input wsam:Action="http://beans/ServicioSumarWS/sumarRequest" message="tns:sumar" />
<output wsam:Action="http://beans/ServicioSumarWS/sumarResponse" message="tns:sumarResponse" />
</operation>
</portType>
- <binding name="ServicioSumarImplPortBinding" type="tns:ServicioSumarWS">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="sumar">
<soap:operation soapAction="" />
- <input>
<soap:body use="literal" />
</input>
- <output>
<soap:body use="literal" />
</output>
</operation>
</binding>
- <service name="ServicioSumarImplService">
- <port name="ServicioSumarImplPort" binding="tns:ServicioSumarImplPortBinding">
<soap:address location="http://servidor:8080/ServicioSumarImplService/ServicioSumarImpl" />
</port>
</service>
</definitions>
```

## Paso 7. Revisión del servicio SumaWS (cont)

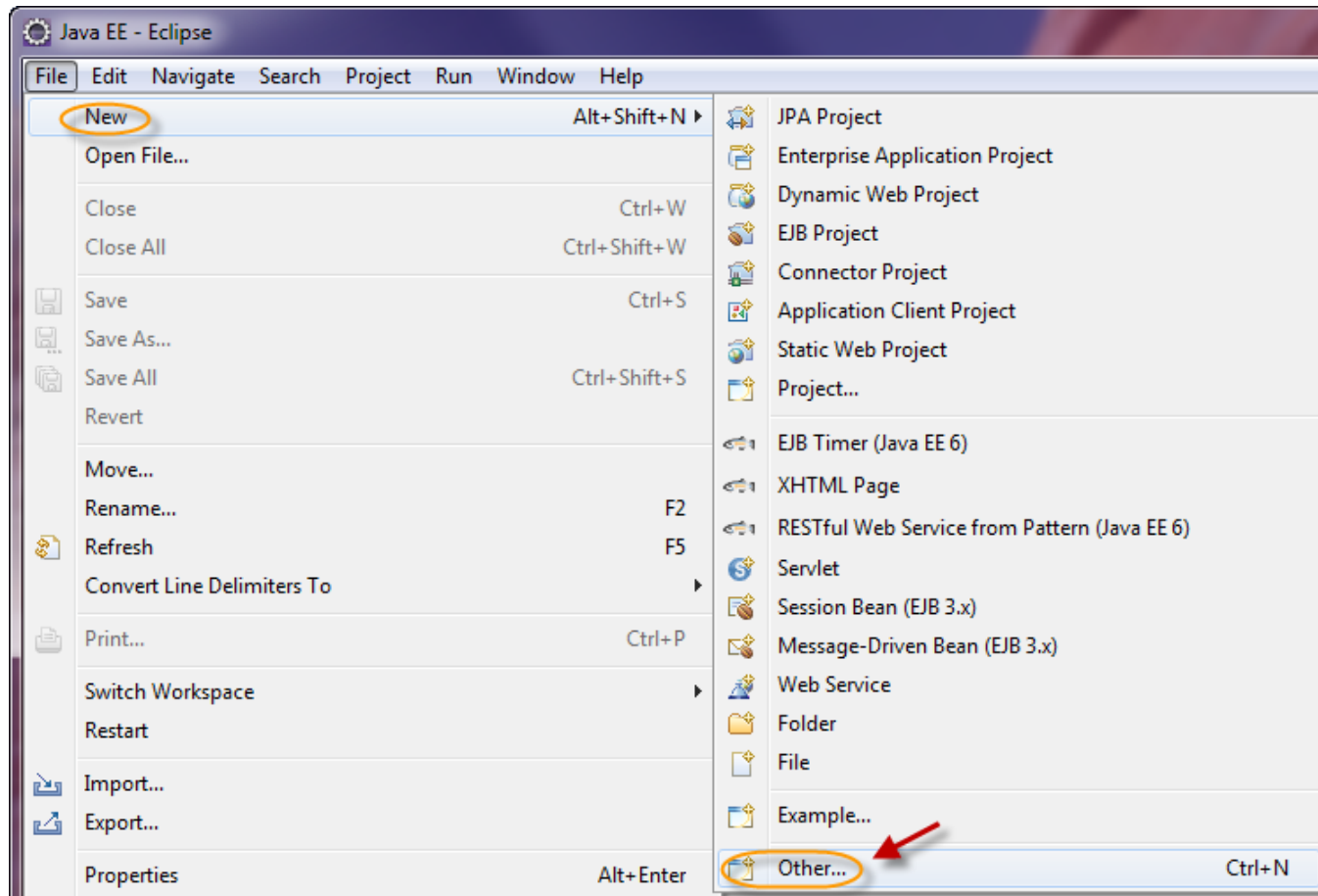
Por último, el documento XSD generado se encuentra en la siguiente URL:  
<http://servidor:8080/ServicioSumarImplService/ServicioSumarImpl?xsd=1>



```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's v
RI/2.2.6-promoted-b20 JAXWS/2.2 svn-revision#unknown. -->
- <xs:schema xmlns:tns="http://beans/" xmlns:xs="http://www.w3.org/20
  <xs:element name="sumar" type="tns:sumar" />
  <xs:element name="sumarResponse" type="tns:sumarResponse" />
- <xs:complexType name="sumar">
  - <xs:sequence>
    <xs:element name="arg0" type="xs:int" />
    <xs:element name="arg1" type="xs:int" />
  </xs:sequence>
  </xs:complexType>
- <xs:complexType name="sumarResponse">
  - <xs:sequence>
    <xs:element name="return" type="xs:int" />
  </xs:sequence>
  </xs:complexType>
</xs:schema>
```

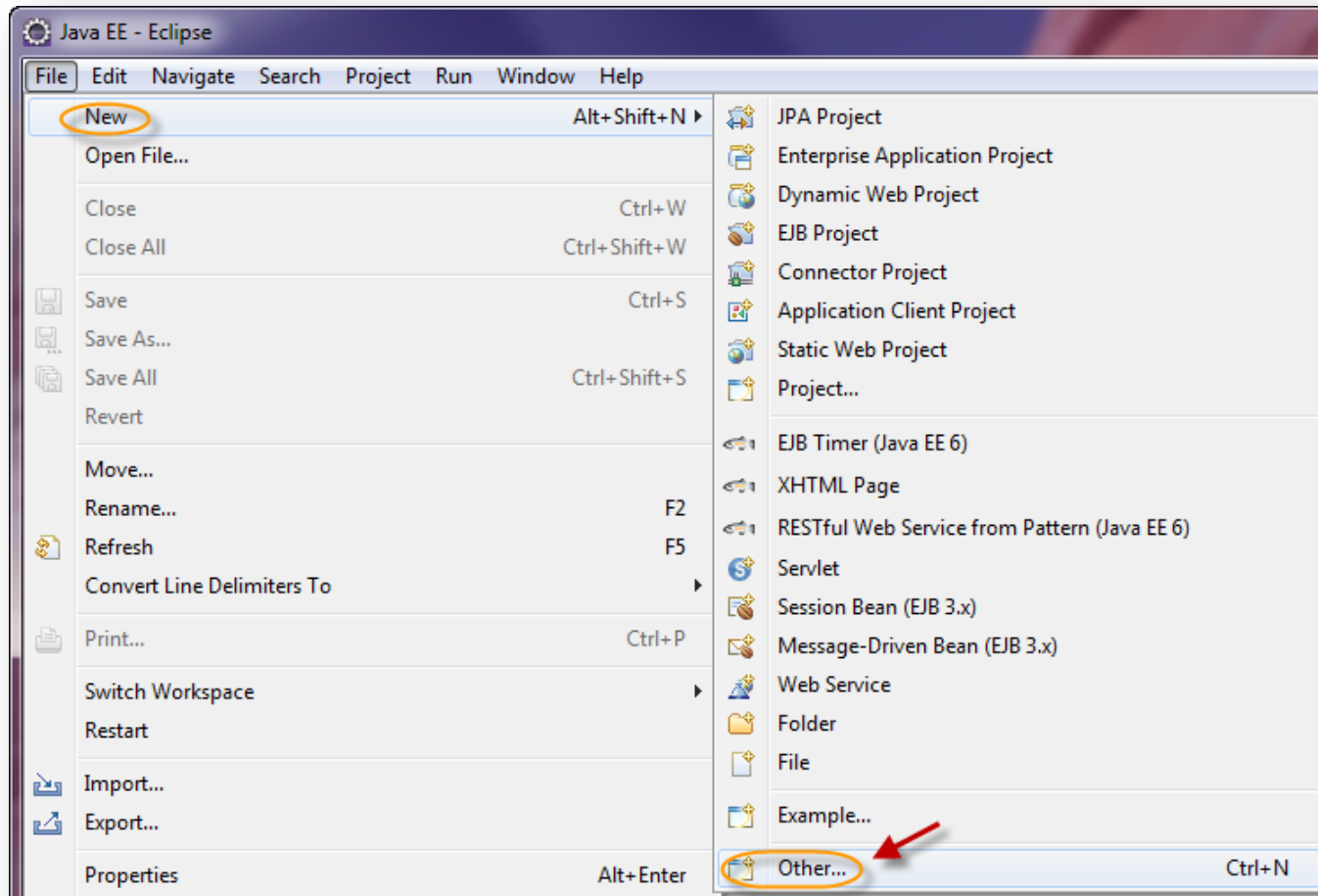
## Paso 8. Creación del Cliente SumaWS

Una vez desplegado el servicio SumaWS y con el servidor GlassFish ya iniciado, procedemos a crear el cliente del Servicio Web. Para ello vamos a crear un nuevo proyecto llamado ClienteSumaWS:



## Paso 8. Creación del Cliente SumaWS (cont)

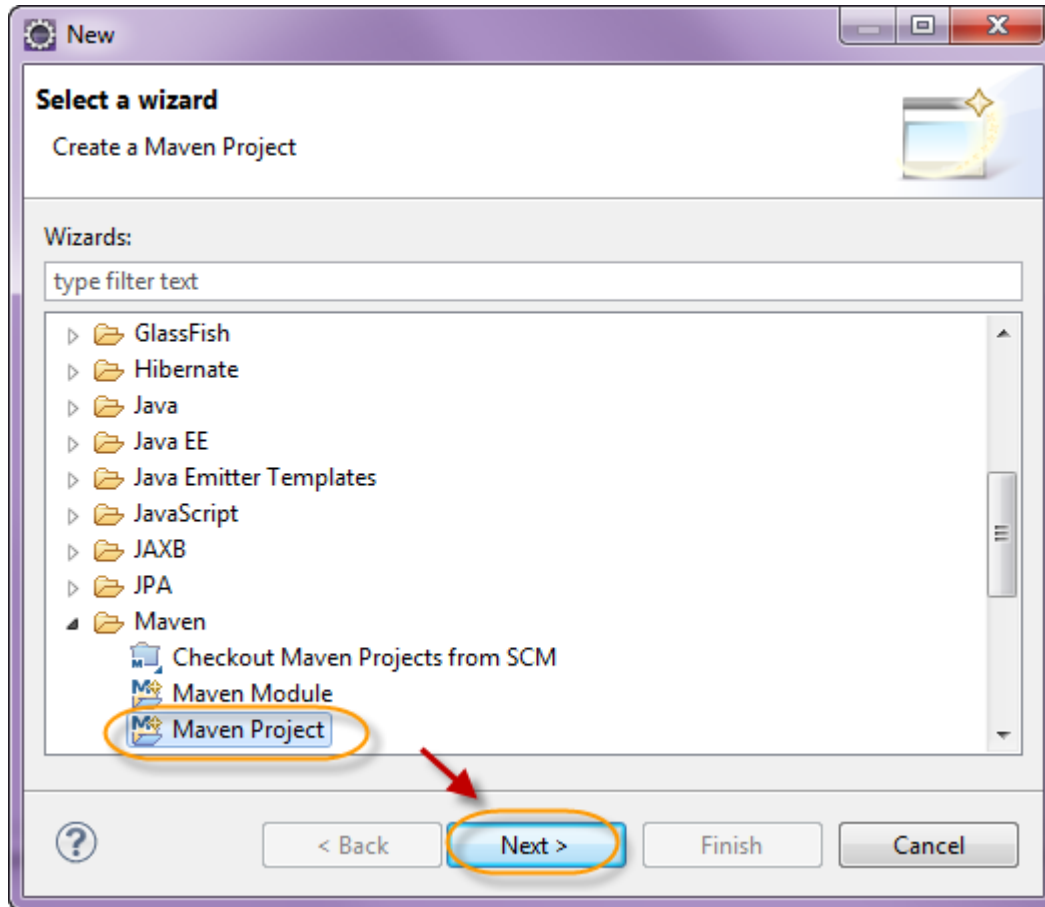
Creamos un nuevo proyecto llamado ClienteSumaWS utilizando Maven:





## Paso 8. Creación del Cliente SumaWS (cont)

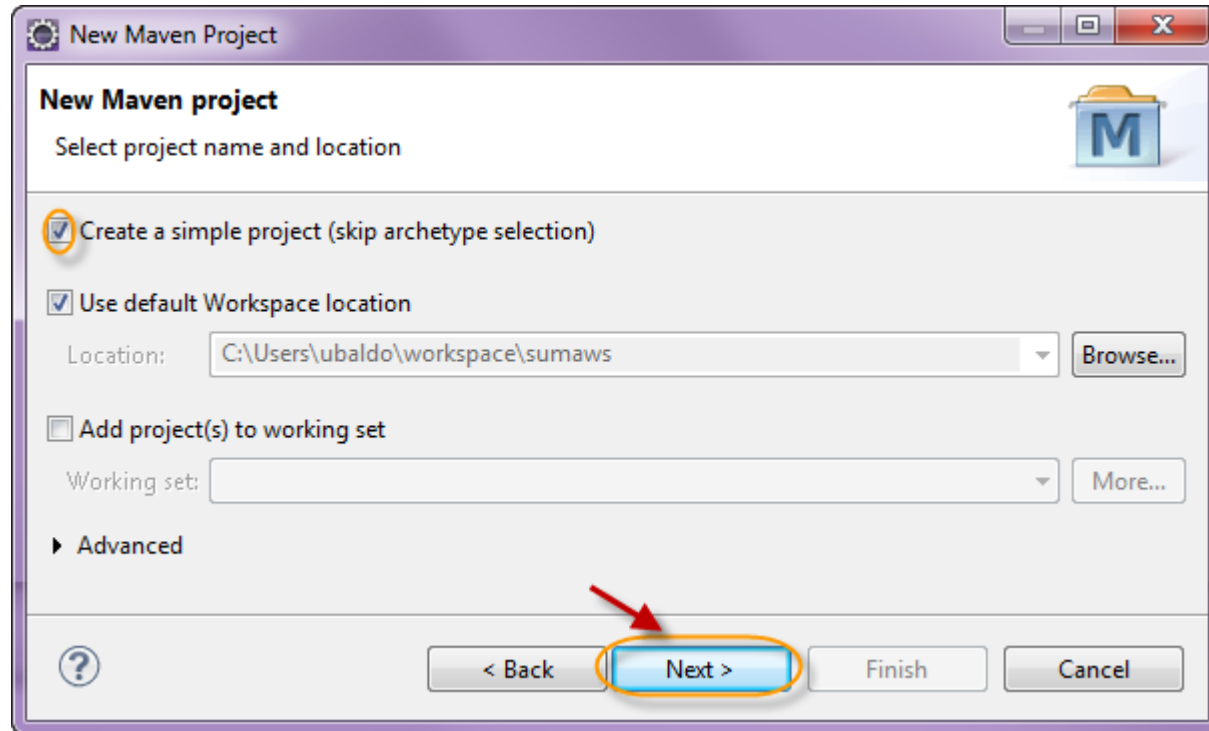
Creamos un nuevo proyecto llamado ClienteSumaWS utilizando Maven:





## Paso 8. Creación del Cliente SumaWS (cont)

Creamos un nuevo proyecto llamado ClienteSumaWS utilizando Maven:



## Paso 8. Creación del Cliente SumaWS (cont)

Creamos un nuevo proyecto llamado ClienteSumaWS utilizando Maven:

**New Maven Project**

Configure project

**Artifact**

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

**Parent Project**

Group Id:

Artifact Id:

Version:

## Paso 9. 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</groupId>
  <artifactId>clientesumaws</artifactId>
  <version>1.0</version>

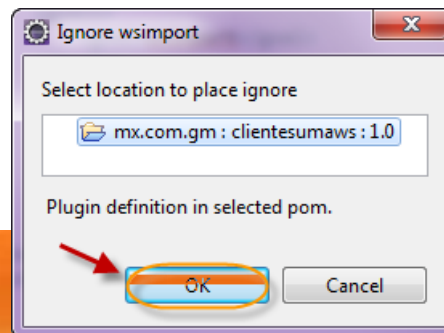
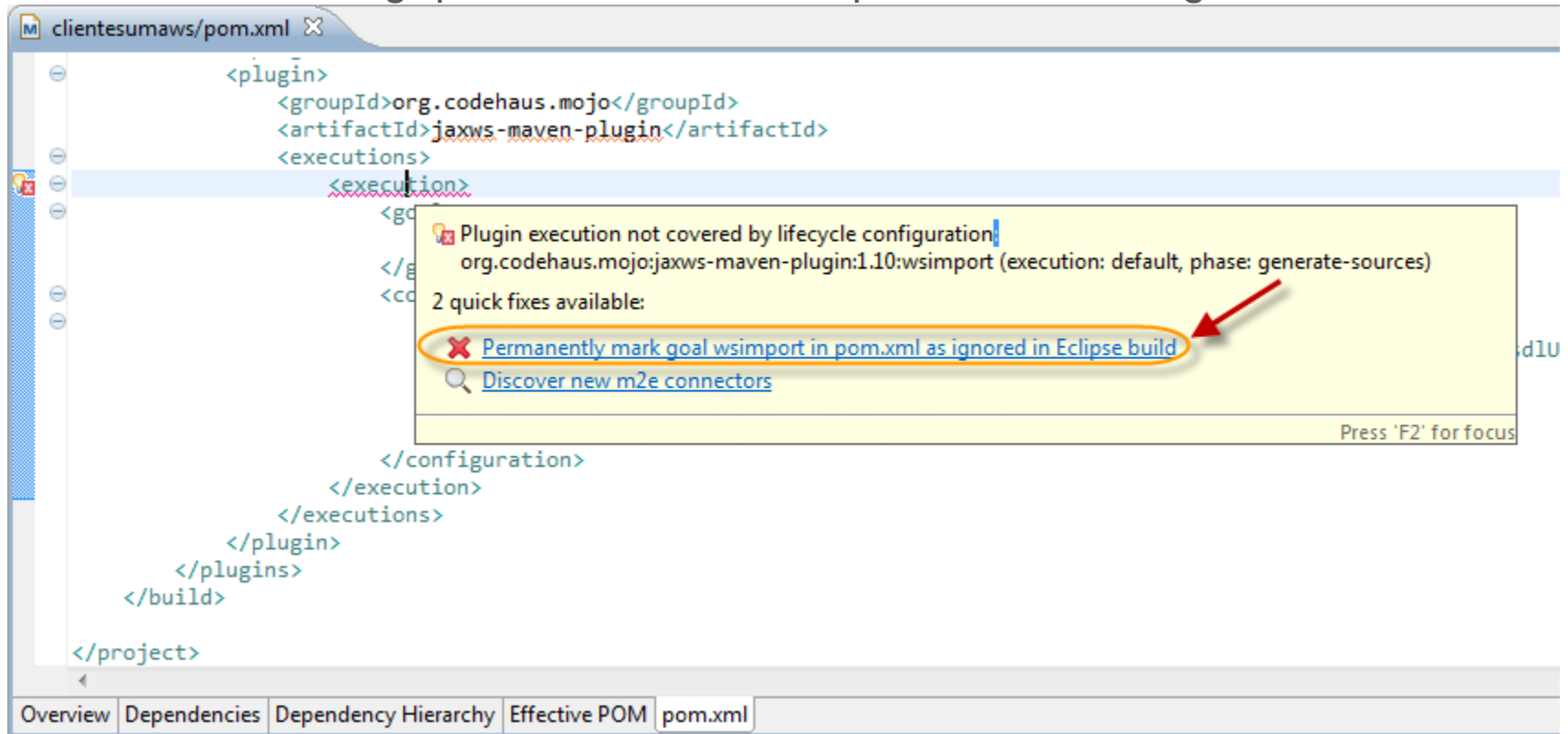
  <dependencies>
    <dependency>
      <groupId>javax</groupId>
      <artifactId>javaee-api</artifactId>
      <version>6.0</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
  <pluginRepositories>
    <pluginRepository>
      <id>maven2-repository.dev.java.net</id>
      <name>Java.net Repository for Maven</name>
      <url>http://download.java.net/maven/glassfish/</url>
    </pluginRepository>
  </pluginRepositories>
```

```
<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/ServicioSumarImplService/ServicioSumarImpl?wsdl</wsdlUrl>
            </wsdlUrls>
            <packageName>clientews.servicio</packageName>
            <sourceDestDir>${basedir}/src/main/java</sourceDestDir>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>

</project>
```

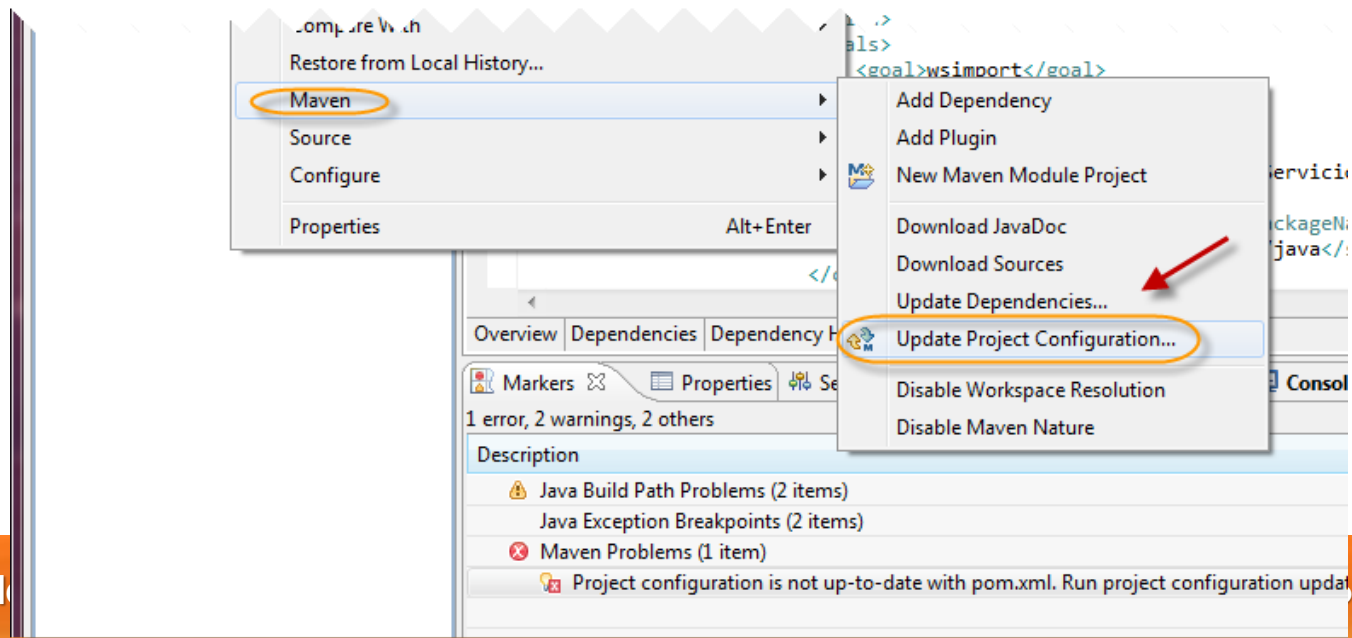
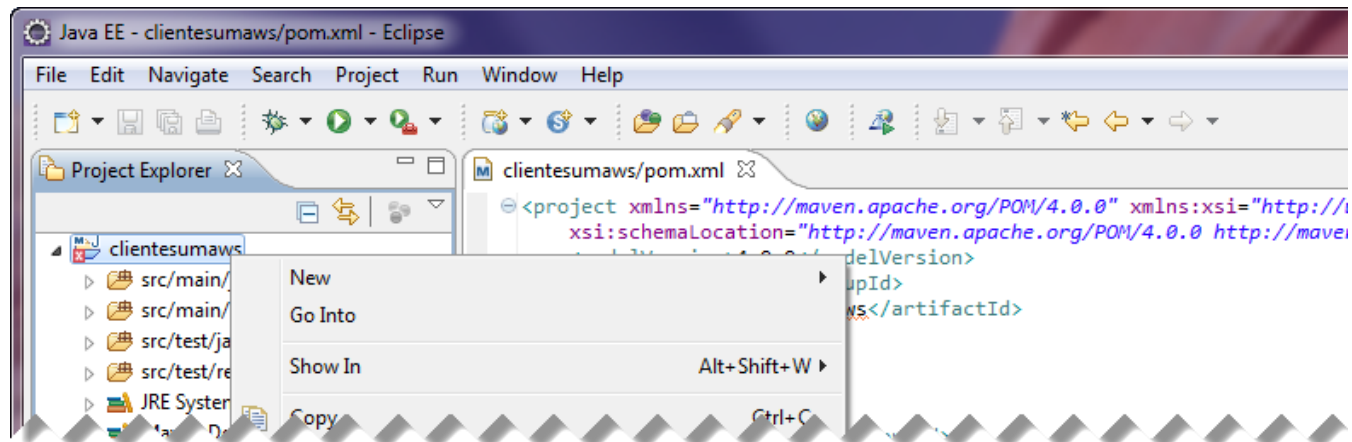
## Paso 9. Configuración del archivo pom.xml (cont)

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



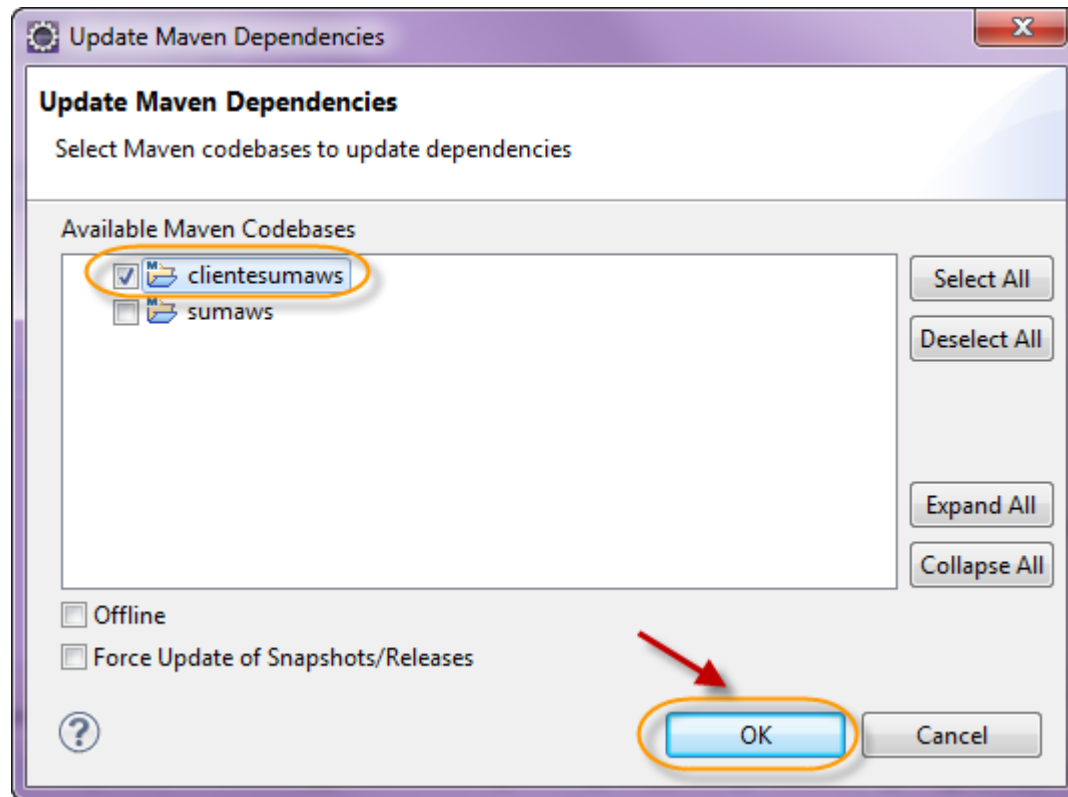
## Paso 10. Actualización proyecto Maven

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



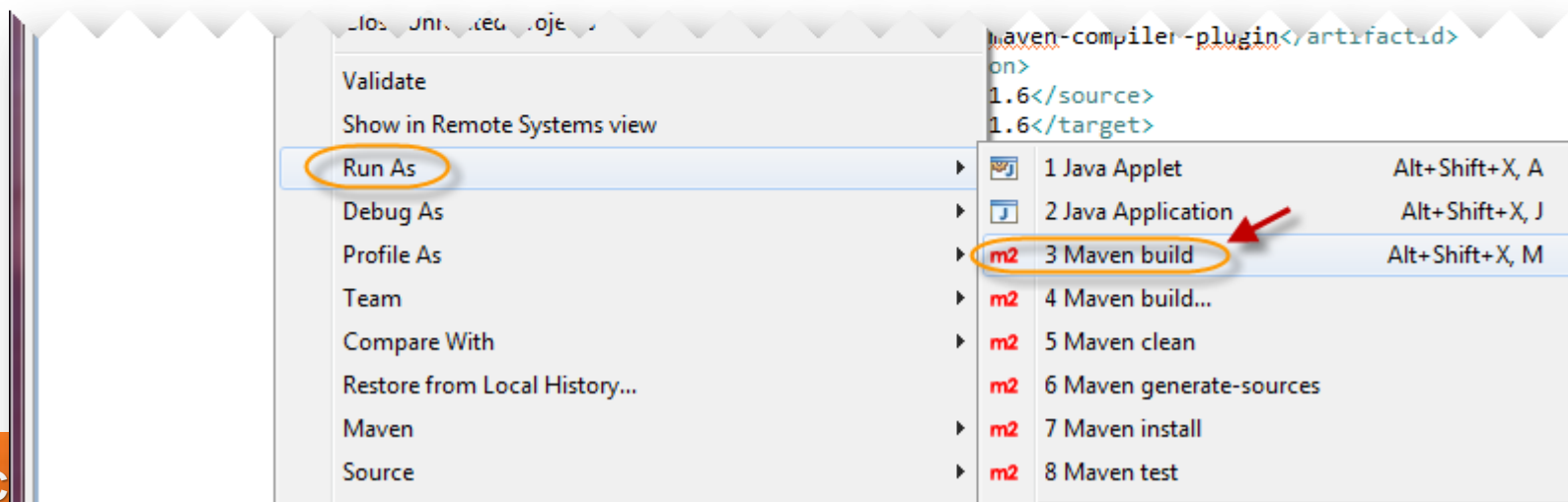
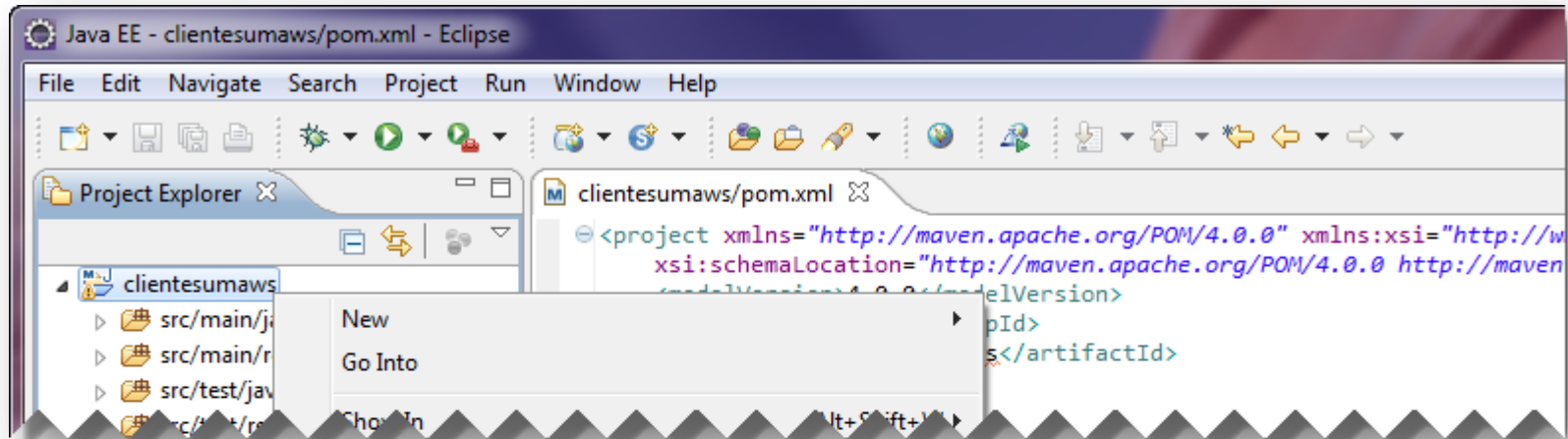
## Paso 10. Actualización proyecto Maven (cont)

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



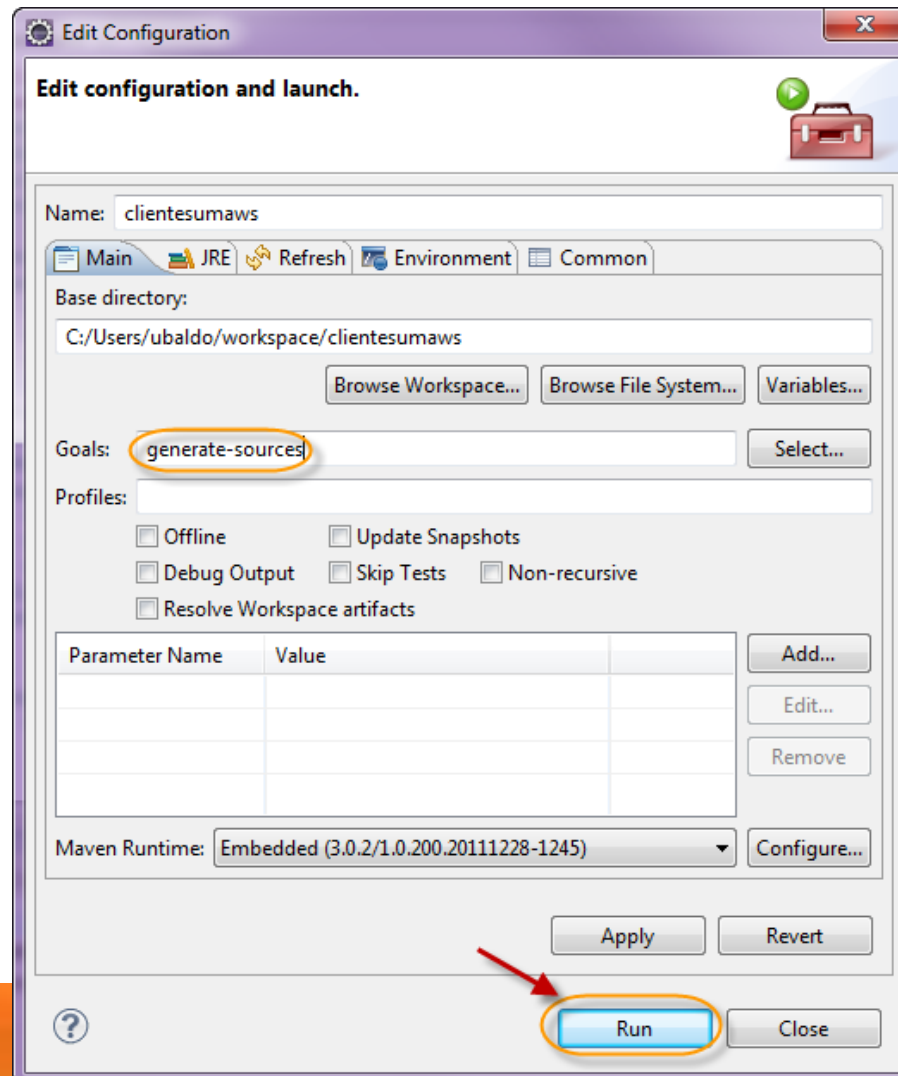
# Paso 11. Generación del Cliente SumaWS

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



## Paso 11. Generación del Cliente SumaWS (cont)

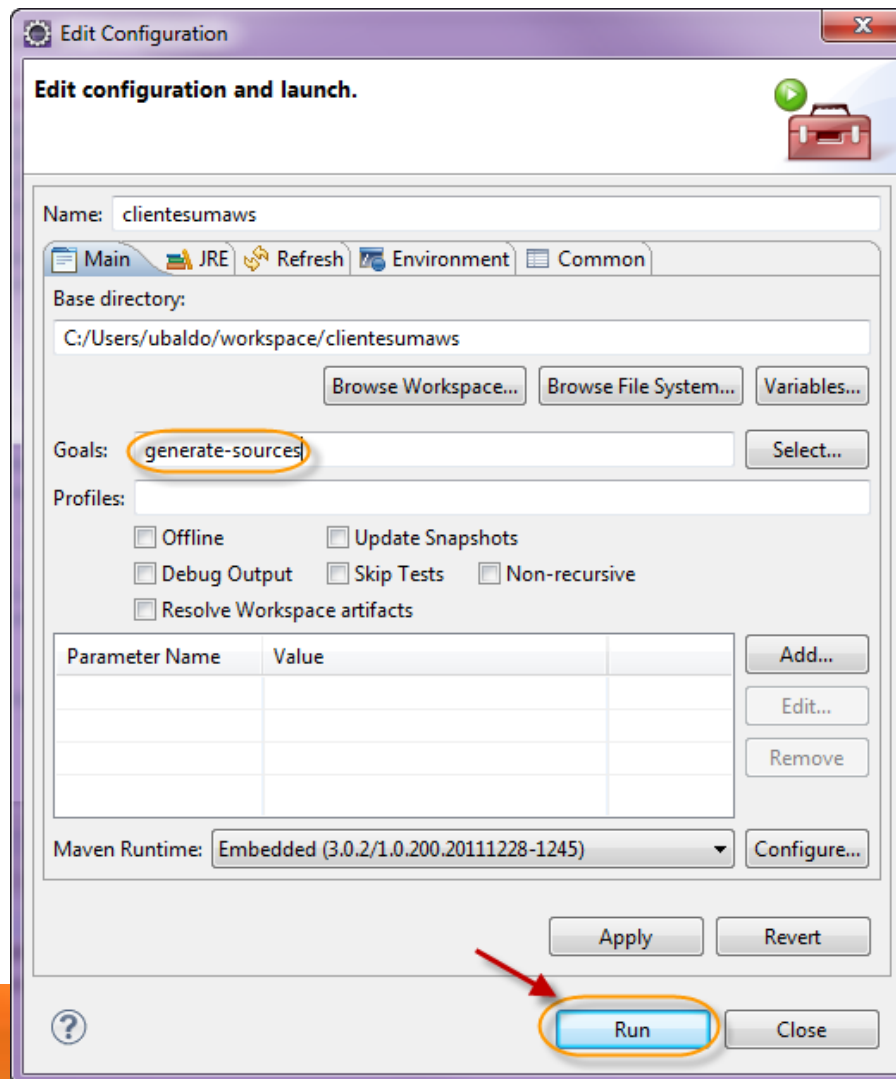
Escribimos generate-sources en el campo de **Goals**, el cual es parte del ciclo de vida de Maven. Con esto se generará el código Java del Cliente:





## Paso 11. Generación del Cliente SumaWS (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 11. Generación del Cliente SumaWS (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.

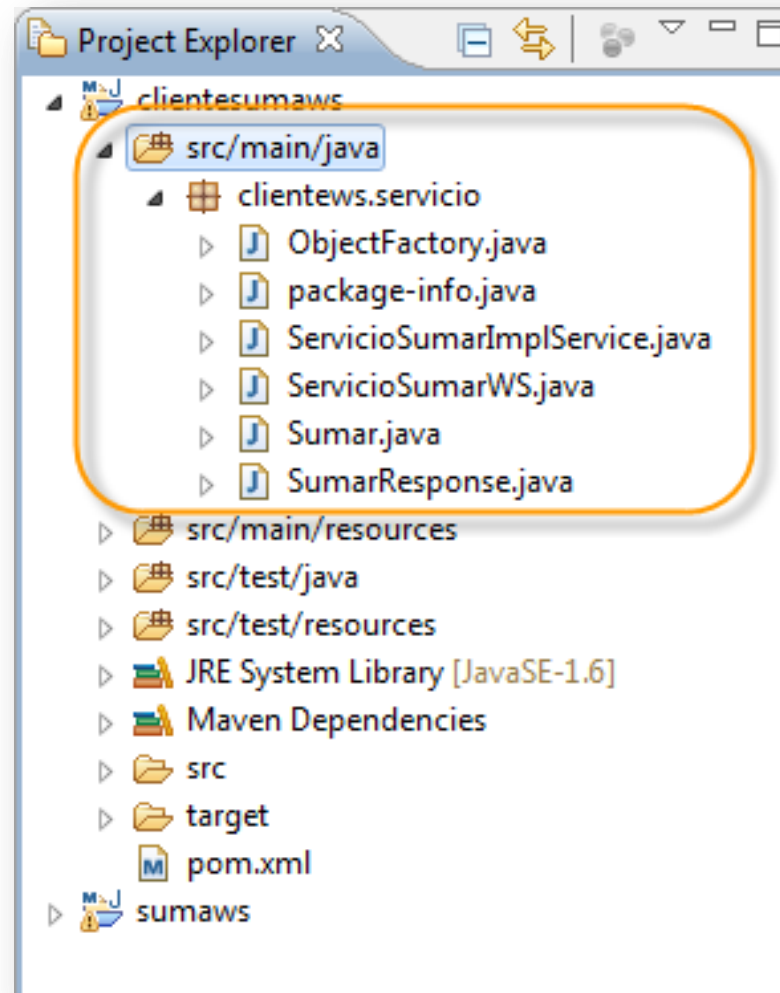
```
[INFO] -----
[INFO] Building clientesumaws 1.0
[INFO] -----
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Downloading: http://download.java.net/maven/glassfish/org/codehaus/mojo/jaxws-maven-plugin/maven-metadata.xml
[INFO] --- jaxws-maven-plugin:1.10:wsimport (default) @ clientesumaws ---
[INFO] Processing: http://servidor:8080/ServicioSumarImplService/ServicioSumarImpl?wsdl
[INFO] jaxws:wsimport args: [-s, C:\Users\ubaldo\workspace\clientesumaws\src\main\java, -d, C:\Users\ubaldo\work
parsing WSDL...

generating code...
|
compiling code...

[INFO] Nothing to do, no WSDL found!
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.511s
[INFO] Finished at: Wed Aug 01 18:03:47 CDT 2012
[INFO] Final Memory: 15M/76M
[INFO] -----
```

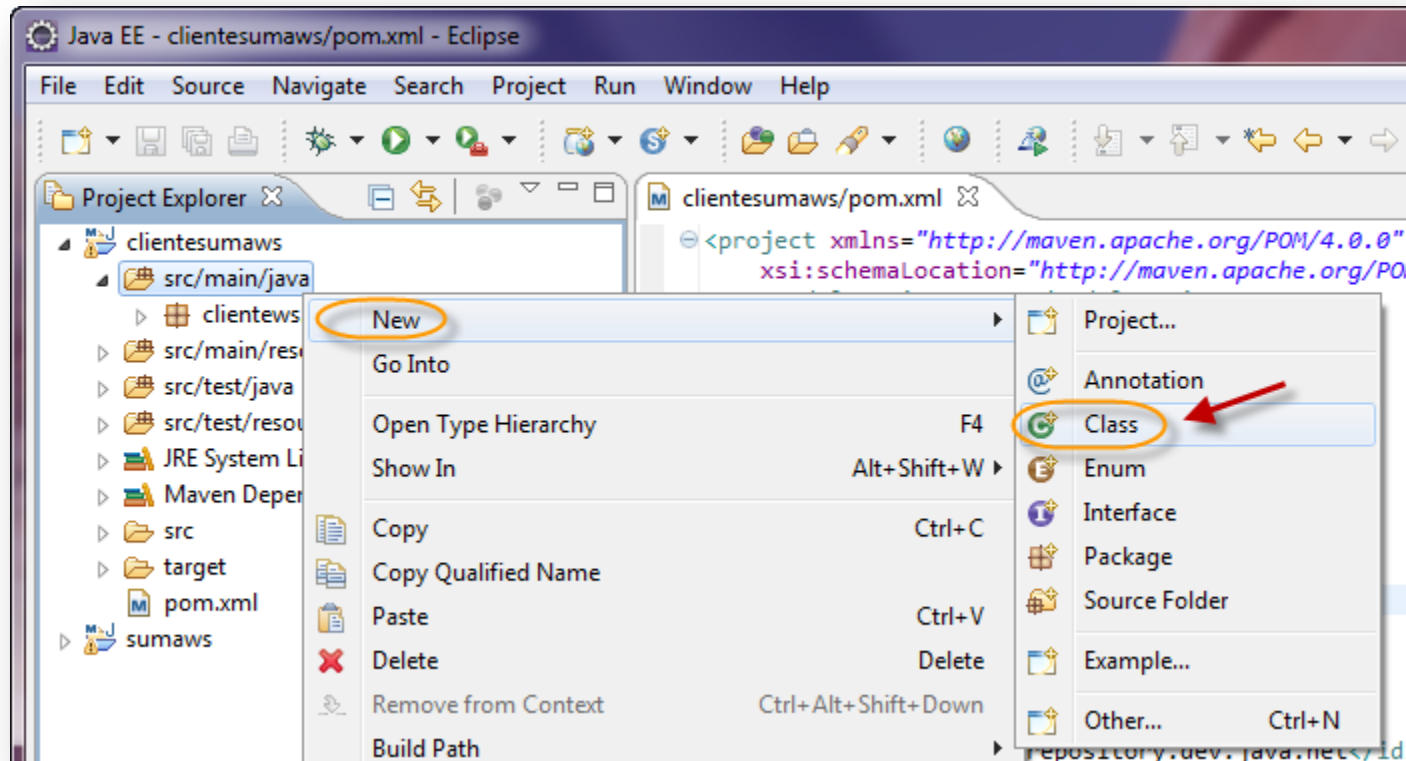
## Paso 11. Generación del Cliente SumaWS (cont)

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



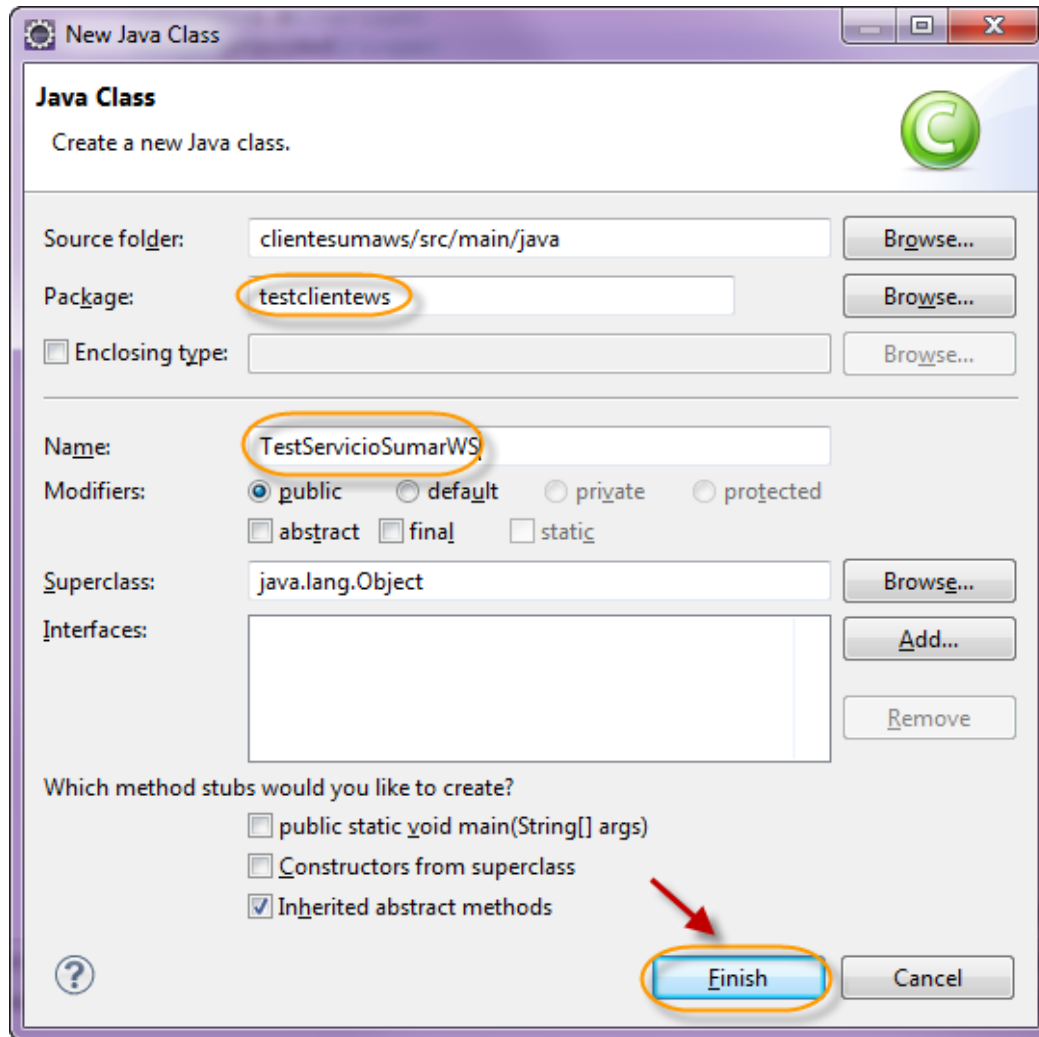
## Paso 12. Creación de la clase TestServicioSumarWS

Creamos la clase TestServicioSumarWS.java:



## Paso 12. Creación de la clase TestServicioSumarWS (cont)

Creamos la clase TestServicioSumarWS.java:





## Paso 12. Creación de la clase TestServicioSumarWS (cont)

Agregamos el siguiente código a la clase TestServicioSumarWS.java:

```
package testclientews;

import clientews.servicio.ServicioSumarImplService;
import clientews.servicio.ServicioSumarWS;

public class TestServicioSumarWS {

    public static void main(String[] args) {
        ServicioSumarWS servicioSumar = new ServicioSumarImplService().getServicioSumarImplPort();

        System.out.println("Ejecutando Servicio Sumar WS");

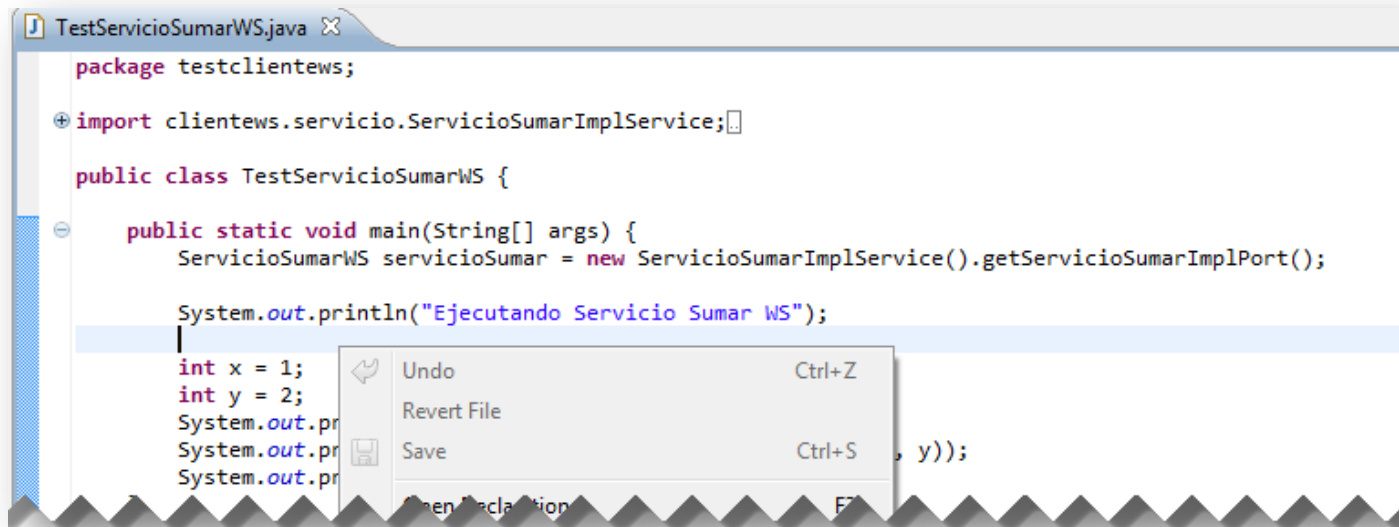
        int x = 1;
        int y = 2;

        System.out.println("Sumar: " + "x: " + x + " y: " + y);
        System.out.println("Resultado: " + servicioSumar.sumar(x, y));
        System.out.println("Fin Servicio Sumar WS");

    }
}
```

## Paso 13. Ejecución clase TestServicioSumarWS

Ejecutamos la clase TestServicioSumarWS.java:



```
TestServicioSumarWS.java
package testclientews;

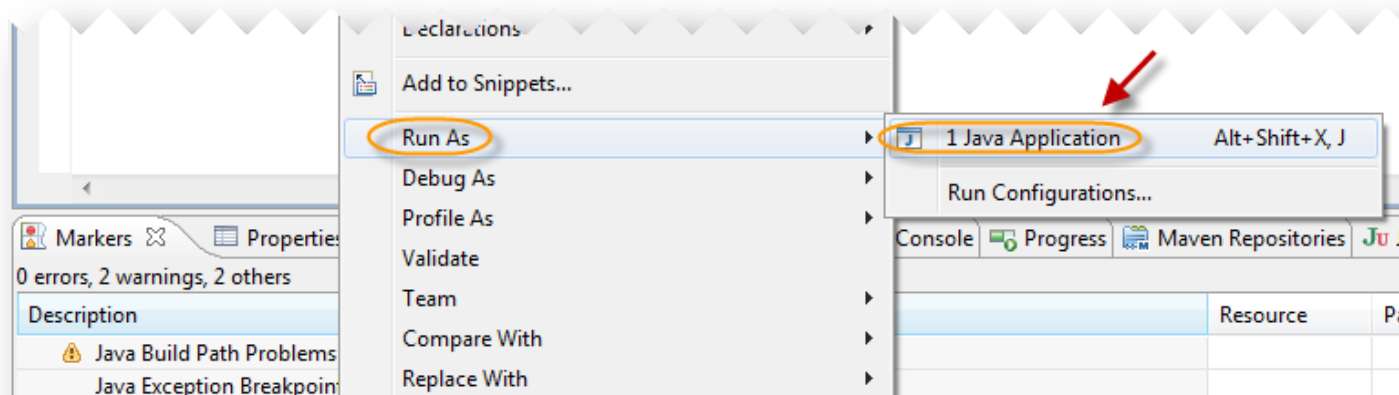
import clientews.servicio.ServicioSumarImplService;

public class TestServicioSumarWS {

    public static void main(String[] args) {
        ServicioSumarWS servicioSumar = new ServicioSumarImplService().getServicioSumarImplPort();

        System.out.println("Ejecutando Servicio Sumar WS");

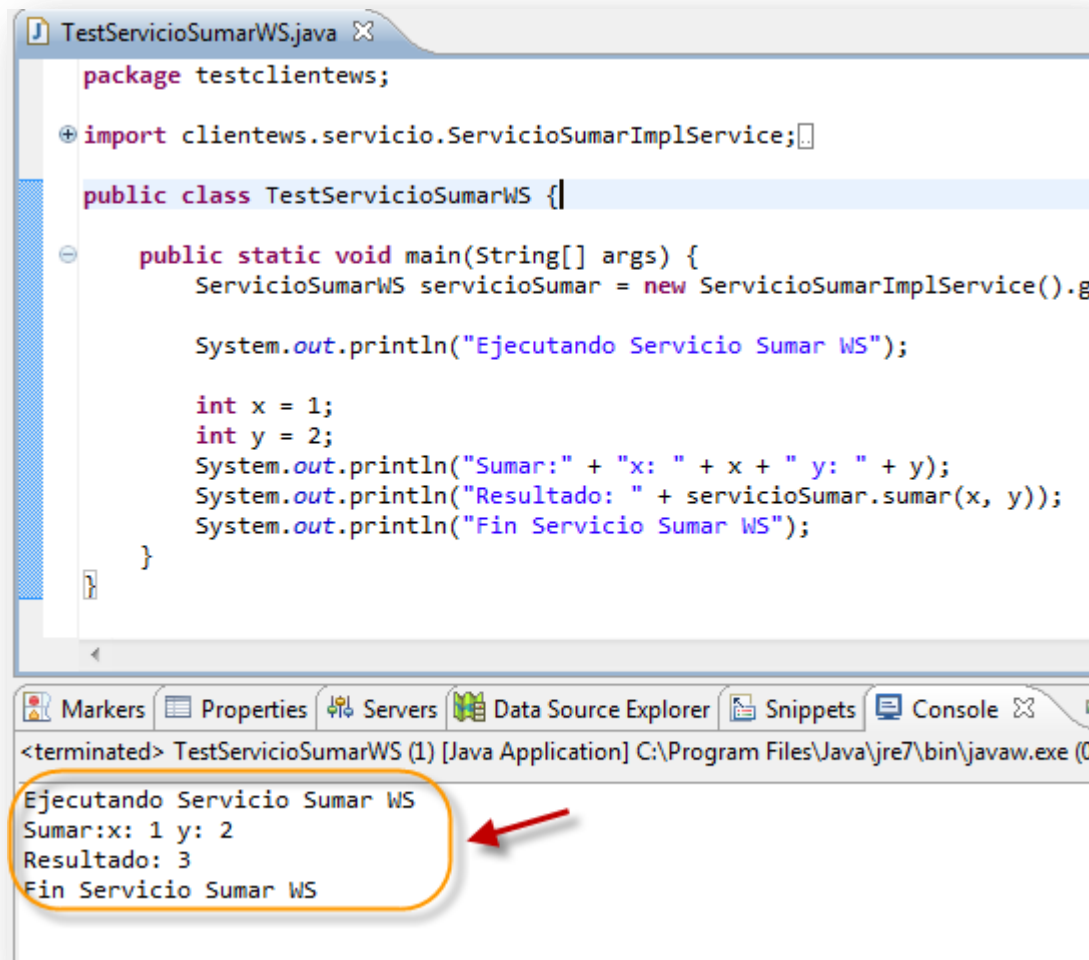
        int x = 1;
        int y = 2;
        System.out.pr
        System.out.pr
        System.out.pr
    }
}
```





## Paso 13. Ejecución clase TestServicioSumarWS (cont)

Ejecutamos la clase TestServicioSumarWS.java:



```
TestServicioSumarWS.java
package testclientews;

import clientews.servicio.ServicioSumarImplService;

public class TestServicioSumarWS {

    public static void main(String[] args) {
        ServicioSumarWS servicioSumar = new ServicioSumarImplService().g

        System.out.println("Ejecutando Servicio Sumar WS");

        int x = 1;
        int y = 2;
        System.out.println("Sumar:" + "x: " + x + " y: " + y);
        System.out.println("Resultado: " + servicioSumar.sumar(x, y));
        System.out.println("Fin Servicio Sumar WS");
    }
}
```

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> TestServicioSumarWS (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (C

Ejecutando Servicio Sumar WS  
Sumar:x: 1 y: 2  
Resultado: 3  
Fin Servicio Sumar WS



## Conclusión

- ✓ Con este ejercicio pudimos observar cómo exponer los métodos de los EJB's 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.
- ✓ En el siguiente ejercicio vamos a exponer el listado de Personas de nuestro sistema SGA utilizando SOAP Web Services con el API de JAX-WS.



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

*Pasión por la tecnología Java*

Experiencia y Conocimiento para tu vida