JAVA EE COURSE

# SMS CLIENT WEB SERVICE WITH JAX-WS

By the expert: Eng. Ubaldo Acosta

Global Mentoring

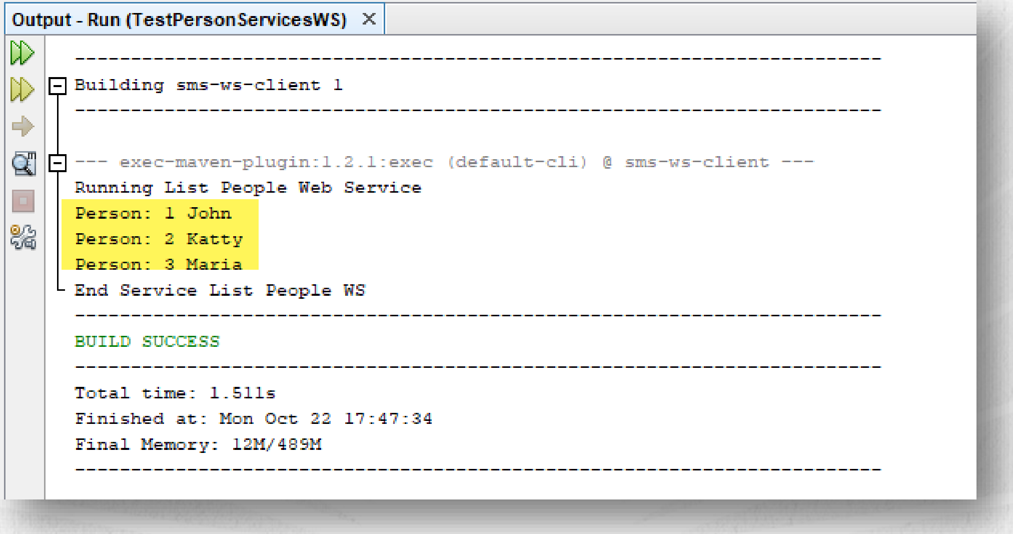
JAVA ONLINE COURSES UNIVERSITY

JAVA EE COURSE

www.globalmentoring.com.mx

# EXERCISE OBJECTIVE

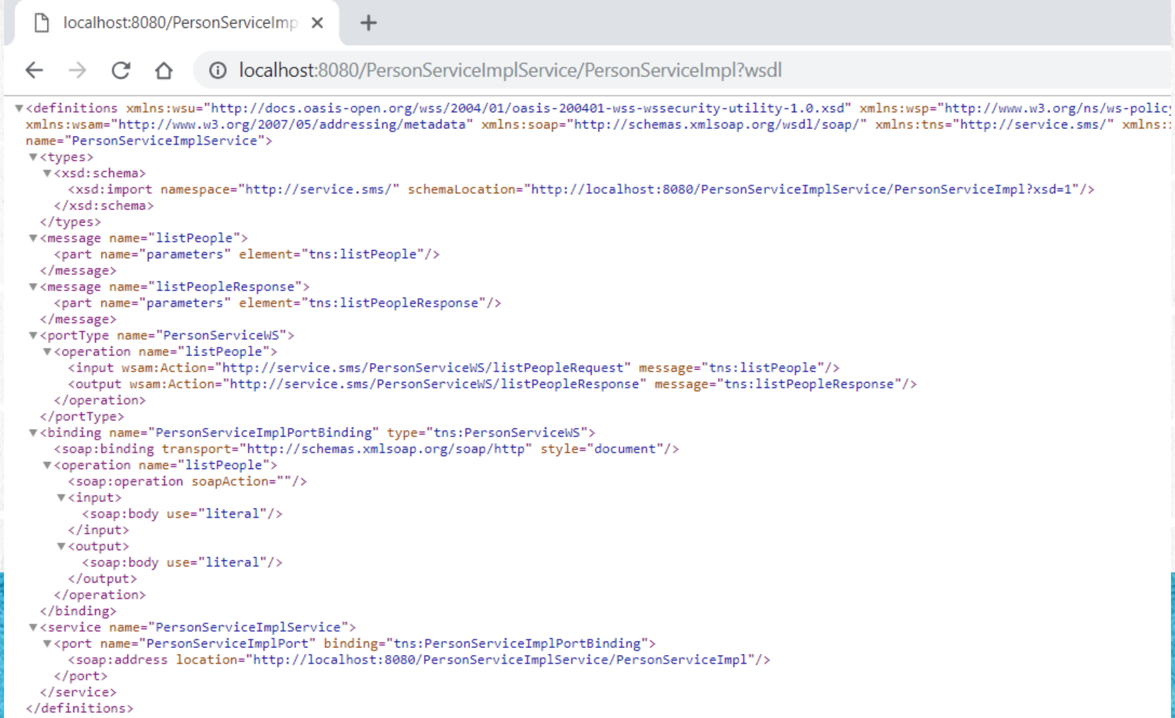
The objective of the exercise is to create a client of the sms-jee-web project Web Service of the list of people created in the previous exercise. The result is shown below:



```
------------------------------------------------------------------------
Building sms-ws-client 1
------------------------------------------------------------------------

--- exec-maven-plugin:1.2.1:exec (default-cli) @ sms-ws-client ---
Running List People Web Service
Person: 1 John
Person: 2 Katty
Person: 3 Maria
End Service List People WS
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 1.511s
Finished at: Mon Oct 22 17:47:34
Final Memory: 12M/489M
------------------------------------------------------------------------
```

# START FROM THE PREVIOUS EXERCISE

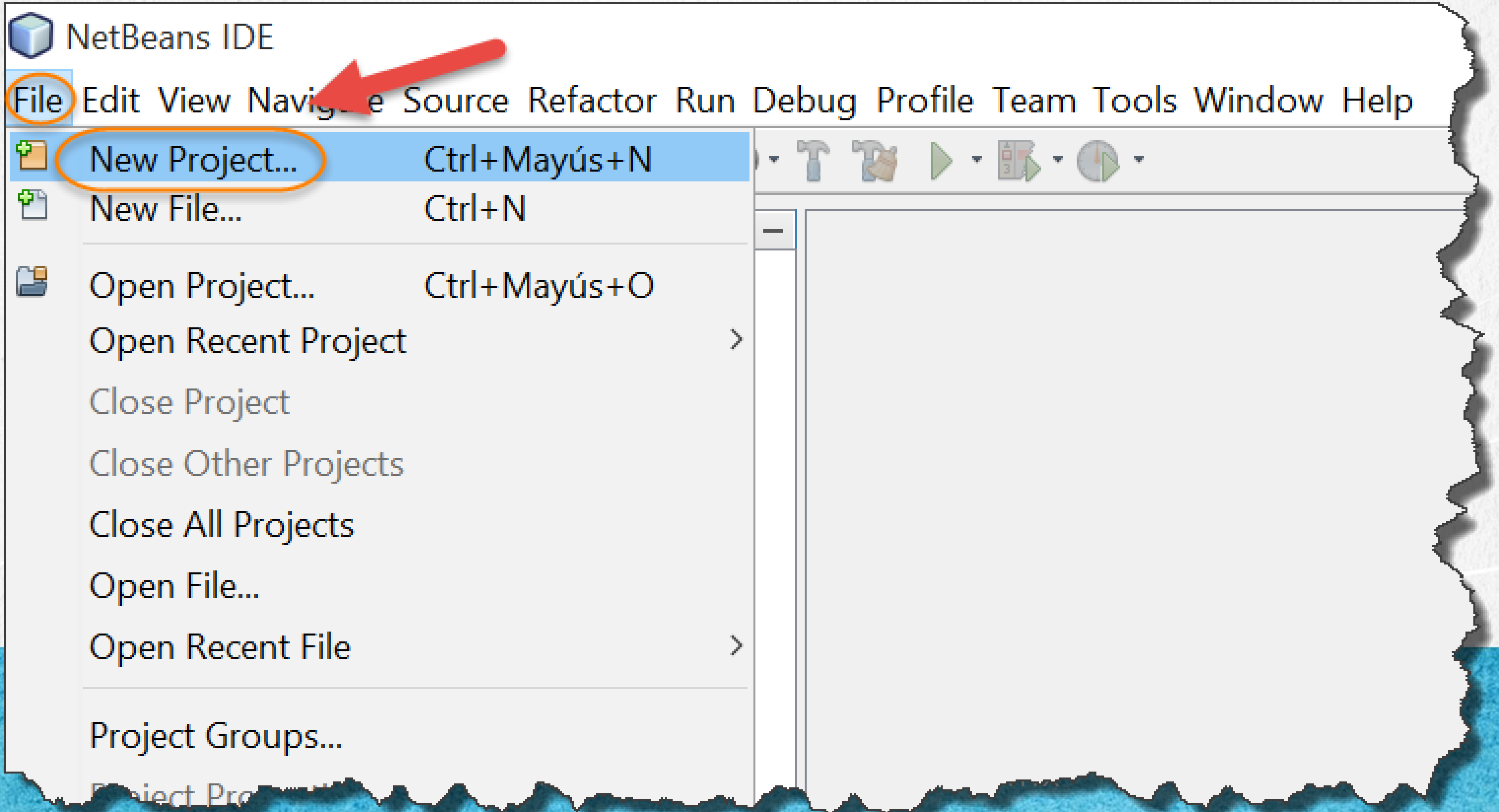
We will start from the previous exercise. It should already be deployed in Glassfish and working. We must be able to visualize the following url: http://localhost:8080/PersonServiceImplService/PersonServiceImpl?wsdl


localhost:8080/PersonServiceImp ×
+
localhost:8080/PersonServiceImplService/PersonServiceImpl?wsdl

```
▼<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-polic
 xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://service.sms/" xmlns:
 name="PersonServiceImplService">
  ▼<types>
    ▼<xsd:schema>
       <xsd:import namespace="http://service.sms/" schemaLocation="http://localhost:8080/PersonServiceImplService/PersonServiceImpl?xsd=1"/>
     </xsd:schema>
   </types>
  ▼<message name="listPeople">
     <part name="parameters" element="tns:listPeople"/>
   </message>
  ▼<message name="listPeopleResponse">
     <part name="parameters" element="tns:listPeopleResponse"/>
   </message>
  ▼<portType name="PersonServiceWS">
    ▼<operation name="listPeople">
       <input wsam:Action="http://service.sms/PersonServiceWS/listPeopleRequest" message="tns:listPeople"/>
       <output wsam:Action="http://service.sms/PersonServiceWS/listPeopleResponse" message="tns:listPeopleResponse"/>
     </operation>
   </portType>
  ▼<binding name="PersonServiceImplPortBinding" type="tns:PersonServiceWS">
     <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    ▼<operation name="listPeople">
       <soap:operation soapAction=""/>
      ▼<input>
         <soap:body use="literal"/>
       </input>
      ▼<output>
         <soap:body use="literal"/>
       </output>
     </operation>
   </binding>
  ▼<service name="PersonServiceImplService">
    ▼<port name="PersonServiceImplPort" binding="tns:PersonServiceImplPortBinding">
       <soap:address location="http://localhost:8080/PersonServiceImplService/PersonServiceImpl"/>
     </port>
   </service>
 </definitions>
```

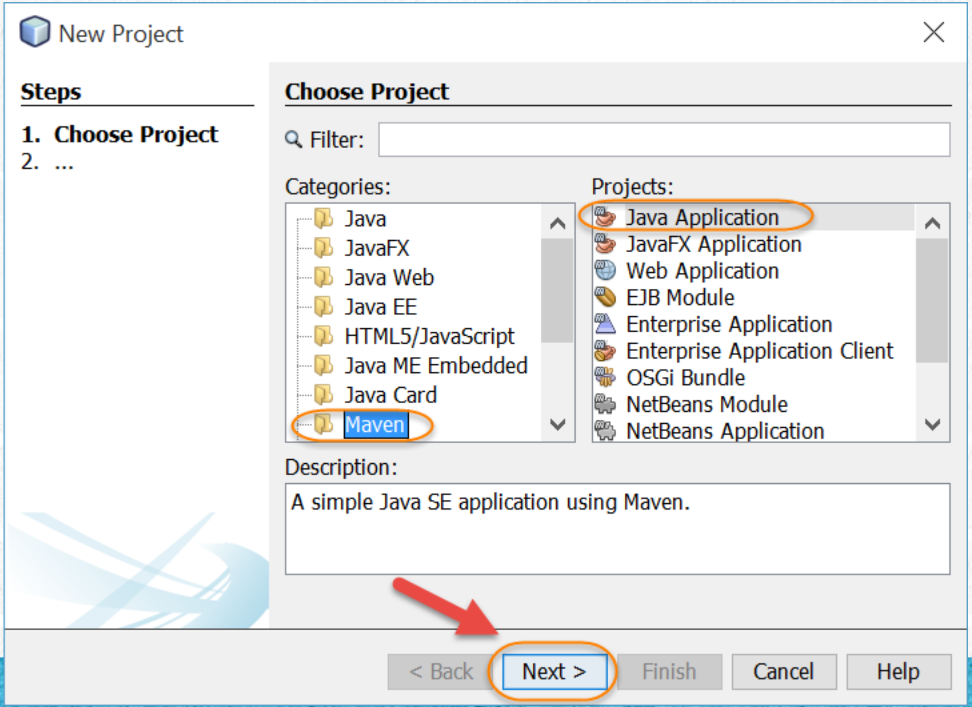
# 1. CREATE A NEW PROJECT

We create the sms-ws-client project:



# 1. CREATE A NEW PROJECT

We create the sms-ws-client project:



New Project

Steps

1. Choose Project
2. ...

Choose Project

Filter:

Categories:

- Java
- JavaFX
- Java Web
- Java EE
- HTML5/JavaScript
- Java ME Embedded
- Java Card
- Maven

Projects:

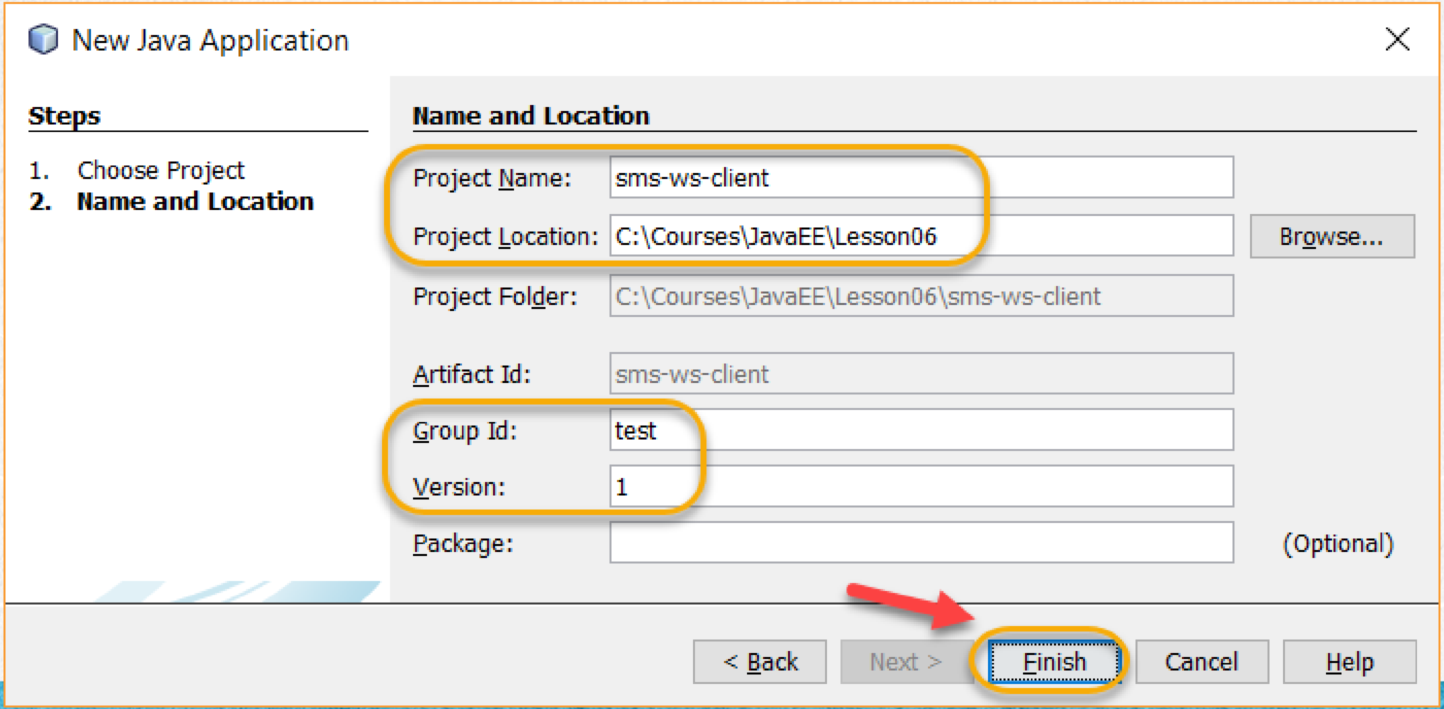
- Java Application
- JavaFX Application
- Web Application
- EJB Module
- Enterprise Application
- Enterprise Application Client
- OSGi Bundle
- NetBeans Module
- NetBeans Application

Description:

A simple Java SE application using Maven.

# 1. CREATE A NEW PROJECT

We create the sms-ws-client project:



New Java Application

**Steps**

1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name: sms-ws-client

Project Location: C:\Courses\JavaEE\Lesson06

Project Folder: C:\Courses\JavaEE\Lesson06\sms-ws-client

Artifact Id: sms-ws-client

Group Id: test

Version: 1

Package: (Optional)

# 2. MODIFY THE CODE

pom.xml:

Click to download

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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>test</groupId>
    <artifactId>sms-ws-client</artifactId>
    <version>1</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-api</artifactId>
            <version>8.0</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
```

# 2. MODIFY THE CODE

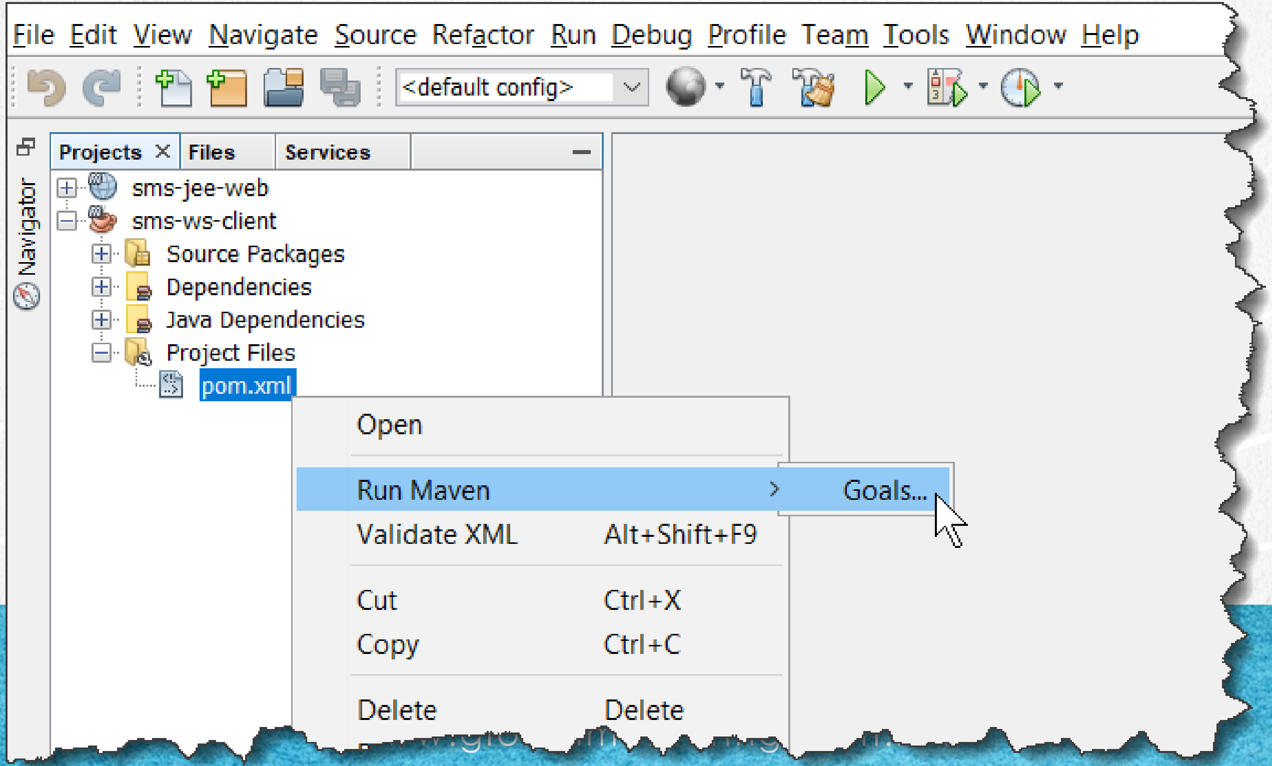
pom.xml:

Click to download

```
     <build>
        <plugins>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>jaxws-maven-plugin</artifactId>
                <version>2.5</version>
                <configuration>
                    <vmArgs>
                        <vmArg>-Djavax.xml.accessExternalSchema=all</vmArg>
                    </vmArgs>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>wsimport</goal>
                        </goals>
                        <configuration>
                            <wsdlUrls>
                                <wsdlUrl>http://localhost:8080/PersonServiceImplService/PersonServiceImpl?wsdl</wsdlUrl>
                            </wsdlUrls>
                            <packageName>sms.ws</packageName>
                            <sourceDestDir>${basedir}/src/main/java</sourceDestDir>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

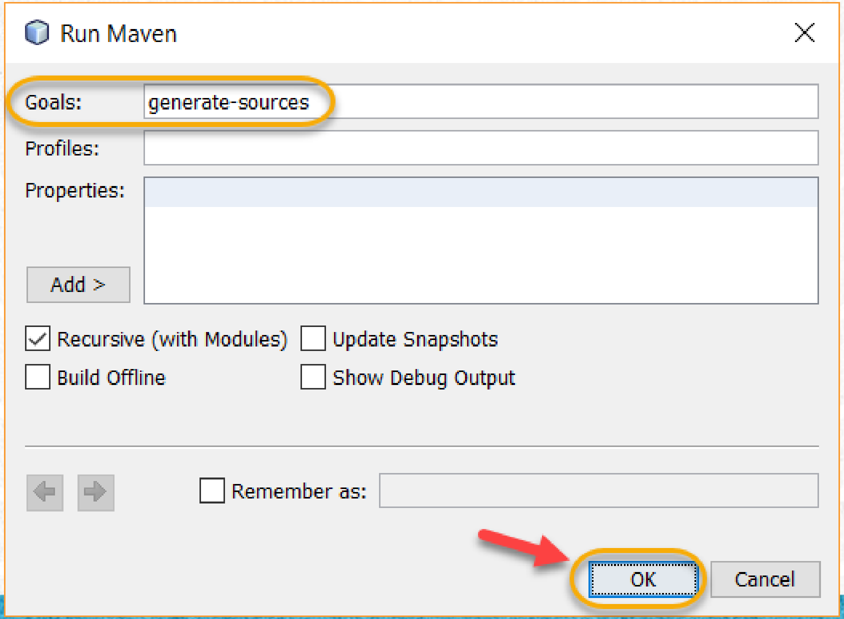
# 3. GENERATE THE WS CLIENT CODE

We execute the client to generate the Web Service code by means of the wsimport command added to the pom.xml file:



# 3. GENERATE THE WS CLIENT CODE

We execute the client to generate the Web Service code. We write generate-sources in the field of Goals to generate the client code:



Run Maven

Goals: generate-sources

Profiles:

Properties:

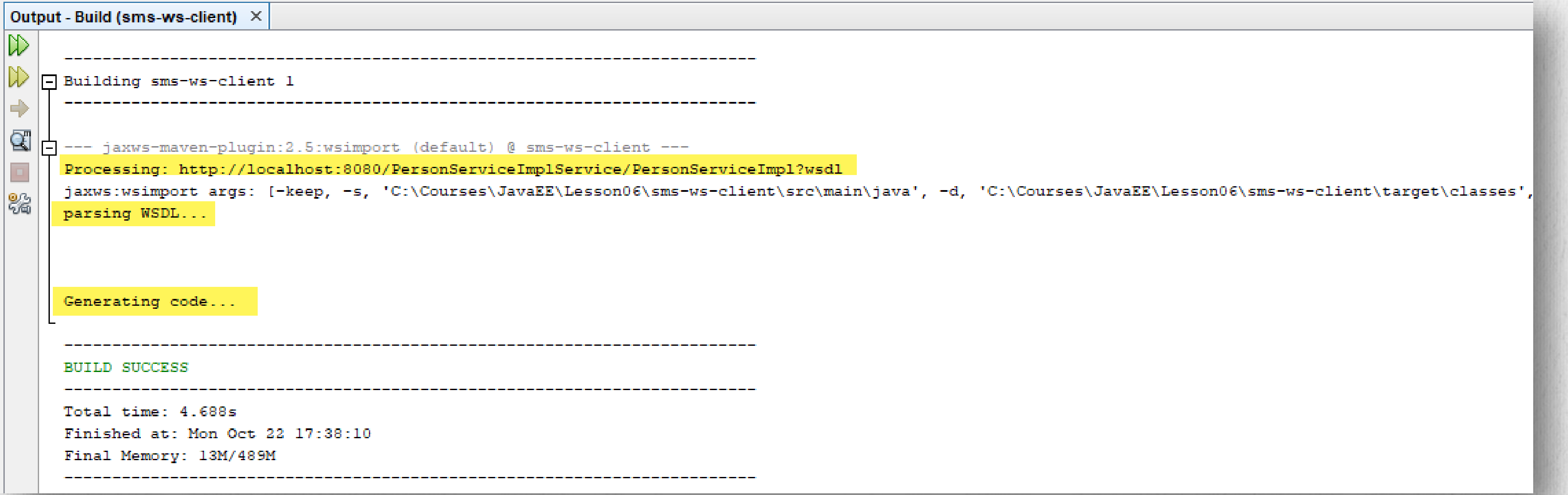
☑ Recursive (with Modules) □ Update Snapshots

□ Build Offline □ Show Debug Output

□ Remember as:

# 3. GENERATE THE WS CLIENT CODE

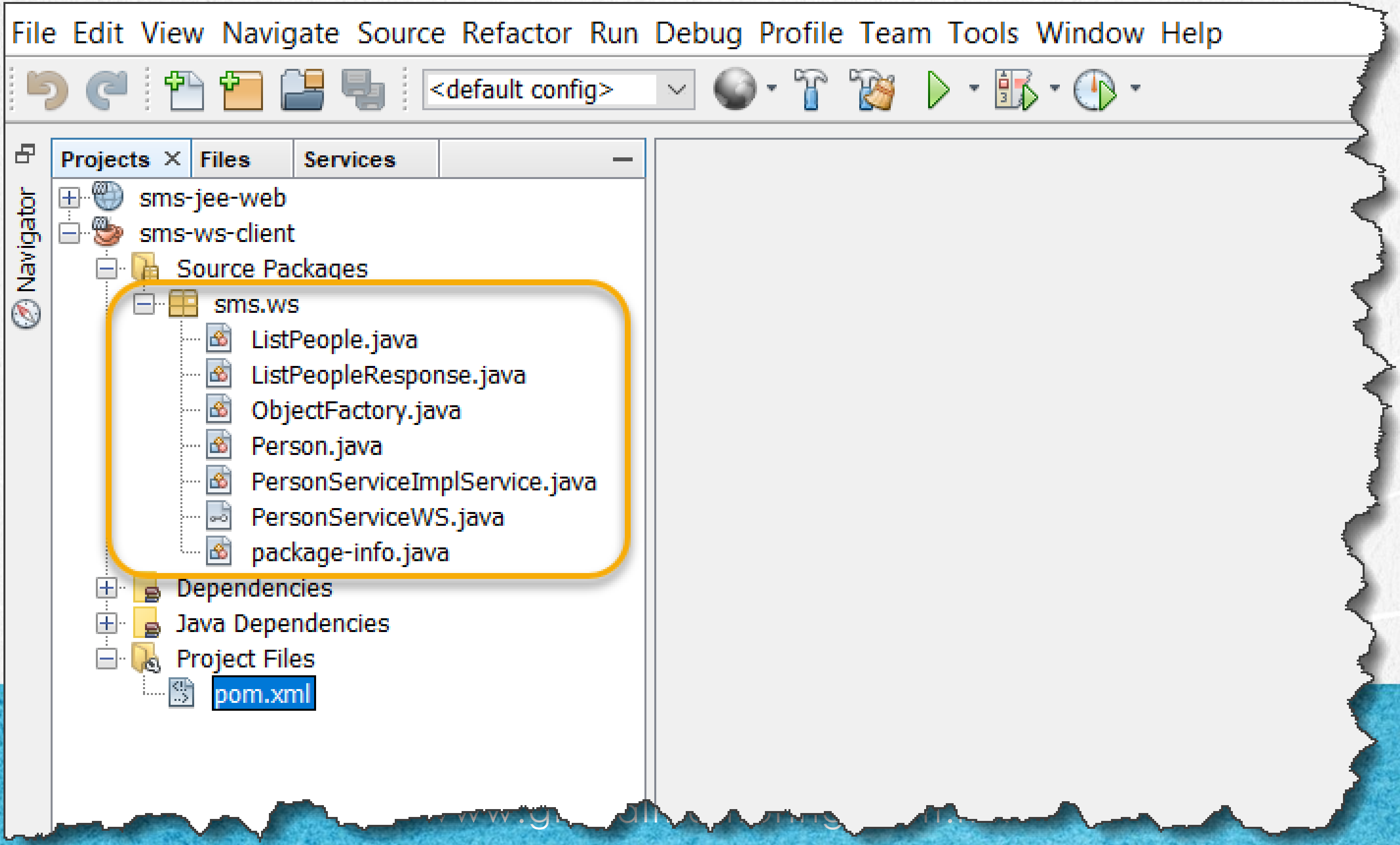
We should see an exit similar to this one:



```
------------------------------------------------------------------------
Building sms-ws-client 1
------------------------------------------------------------------------

--- jaxws-maven-plugin:2.5:wsimport (default) @ sms-ws-client ---
Processing: http://localhost:8080/PersonServiceImplService/PersonServiceImpl?wsdl
jaxws:wsimport args: [-keep, -s, 'C:\Courses\JavaEE\Lesson06\sms-ws-client\src\main\java', -d, 'C:\Courses\JavaEE\Lesson06\sms-ws-client\target\classes',
parsing WSDL...



Generating code...

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 4.688s
Finished at: Mon Oct 22 17:38:10
Final Memory: 13M/489M
------------------------------------------------------------------------
```

# 3. GENERATE THE WS CLIENT CODE

We must see the code generated in the project:

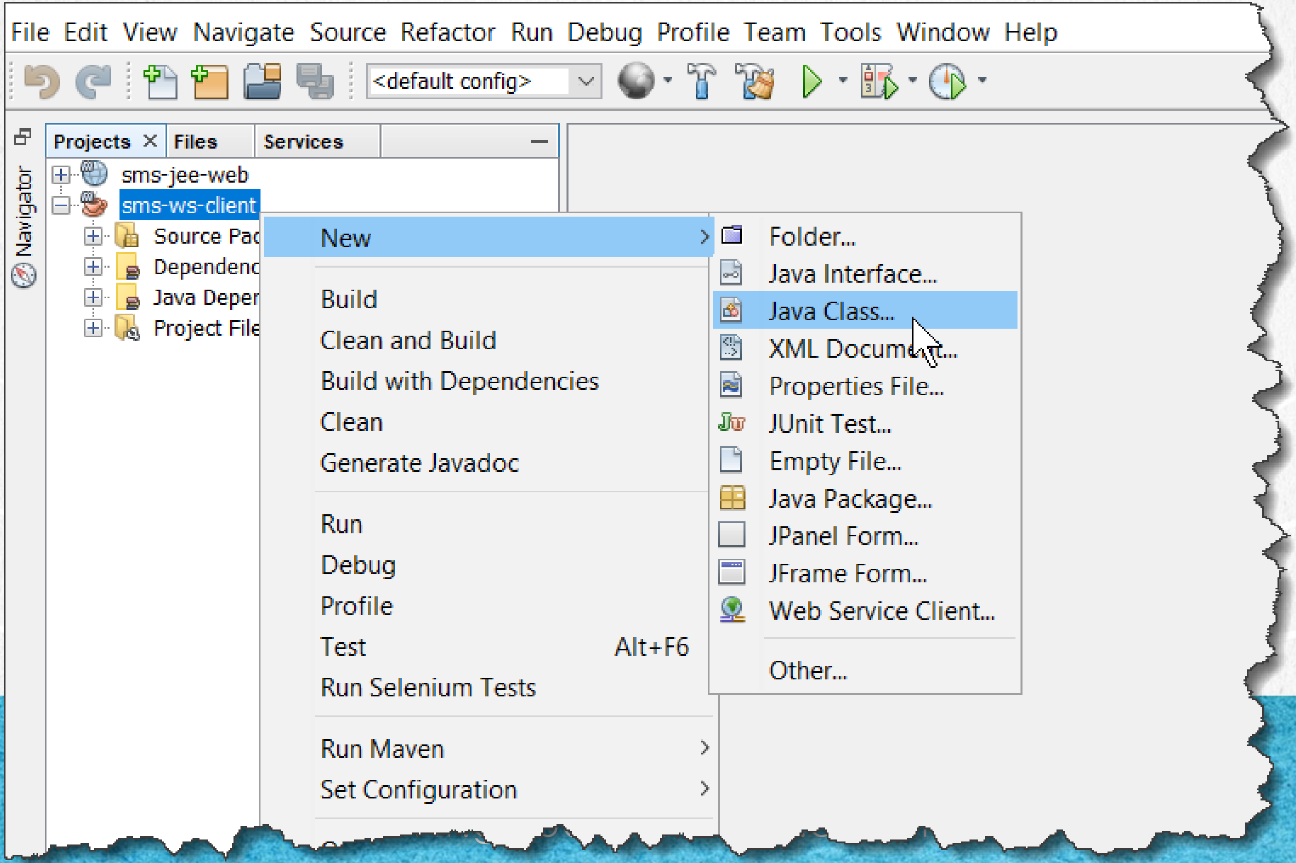


```
sms-jee-web
sms-ws-client
    Source Packages
        sms.ws
            ListPeople.java
            ListPeopleResponse.java
            ObjectFactory.java
            Person.java
            PersonServiceImplService.java
            PersonServiceWS.java
            package-info.java
    Dependencies
    Java Dependencies
    Project Files
        pom.xml
```

www.globalmentoring.com.mx

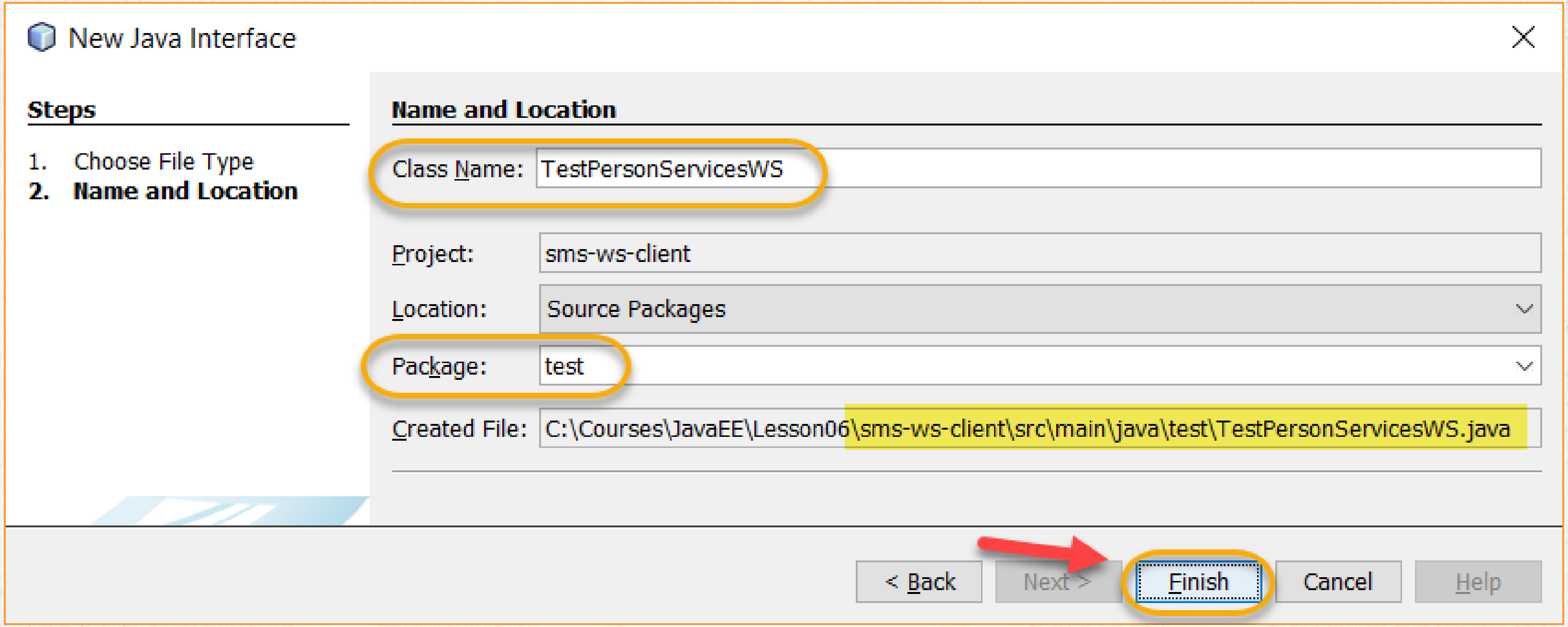
# 4. CREATE A JAVA CLASS

We create the class TestPersonServiceWS.java:



# 4. CREATE A JAVA CLASS

We create the class TestPersonServiceWS.java:



New Java Interface

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name: TestPersonServicesWS

Project: sms-ws-client

Location: Source Packages

Package: test

Created File: C:\Courses\JavaEE\Lesson06\sms-ws-client\src\main\java\test\TestPersonServicesWS.java

# 5. MODIFY THE CODE

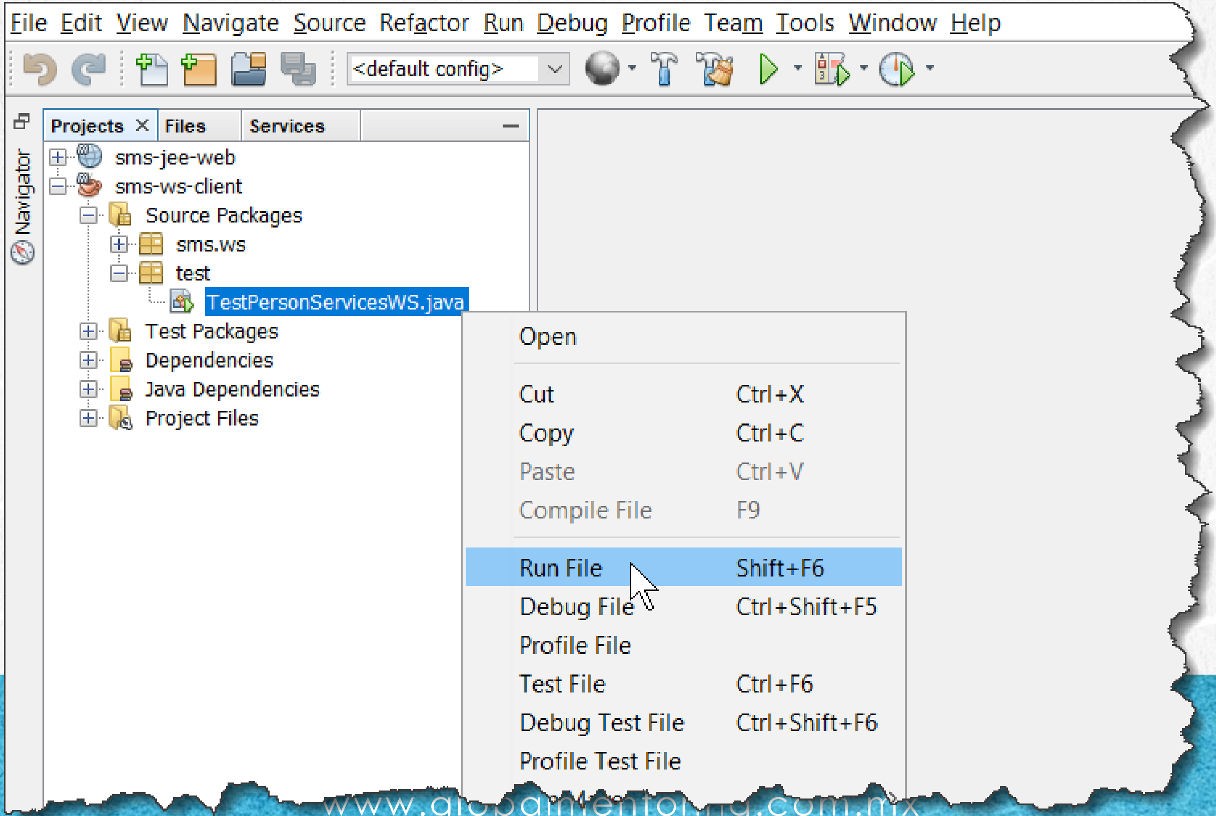
## TestPersonServiceWS.java:

Click to download

```
package test;

import java.util.List;
import sms.ws.*;

public class TestPersonServicesWS {

    public static void main(String[] args) {
        PersonServiceWS personService = new PersonServiceImplService().getPersonServiceImplPort();

        System.out.println("Running List People Web Service");
        List<Person> people = personService.listPeople();
        for (Person person : people) {
            System.out.println("Person: " + person.getIdPerson() + " " + person.getName()) ;
        }
        System.out.println("End Service List People WS");
    }
}
```

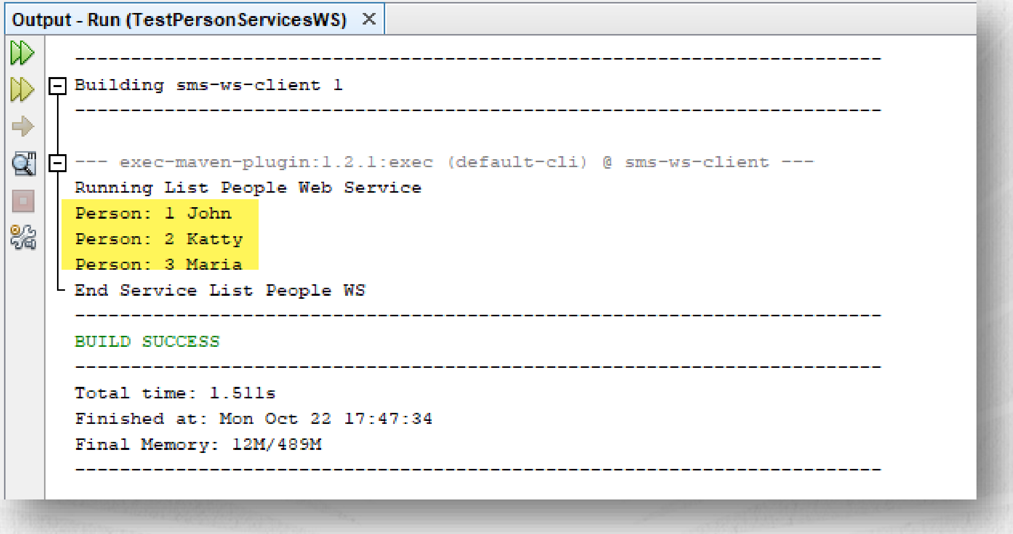
# 6. EXECUTE THE APPLICATION

Execute the class:



# 6. EXECUTE THE APPLICATION

We see the result of consulting the Web service. As we have already mentioned, the Web Service of the previous exercise must be deployed, since this client consumes the web service of listing people, and the response of the call to the requested Web Service is received:



```
------------------------------------------------------------------------
Building sms-ws-client 1
------------------------------------------------------------------------

--- exec-maven-plugin:1.2.1:exec (default-cli) @ sms-ws-client ---
Running List People Web Service
Person: 1 John
Person: 2 Katty
Person: 3 Maria
End Service List People WS
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 1.511s
Finished at: Mon Oct 22 17:47:34
Final Memory: 12M/489M
------------------------------------------------------------------------
```

# EXERCISE CONCLUSION

- With this exercise we were able to observe how to expose the EJB's methods of PersonService using Web Services with the JAX-WS API.
- We observe how to test the Web Service once it is deployed on the GlassFish server.
- In addition, we review how to create the Java classes of the Web Service Client from the WSDL document, all this with the help of the wsimport command, which is part of the Java JDK.
- With this we have seen the complete process of how to create a SOAP Web Service and the respective client.
- In addition, we saw how to exclude some attributes to be sent using JAXB in our Person domain class.
- Lastly, and most importantly, we used a 3-layer architecture and we saw how to expose the business logic as a Web service to any customer interested in SMS system information.

ONLINE COURSE

# JAVA EE
# JAKARTA EE

By: Eng. Ubaldo Acosta

JAVA EE COURSE

www.globalmentoring.com.mx