

**JAVA EE COURSE**

# **SMS SYSTEM AND WEB SERVICES WITH JAX-WS**



---

By the expert: Eng. Ubaldo Acosta



**JAVA EE COURSE**

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

# EXERCISE OBJECTIVE

The objective of the exercise is to expose the method of listing the People of the EJB of the SMS project as a Web Services with the help of the JAX-WS API. The result is shown below:

## listPeople Method invocation

### Method parameter(s)

Type	Value
------	-------

### Method returned

java.util.List : "[sms.service.Person@4cfdec5b, sms.service.Person@63c1fd01, sms.service.Person@27e660fa]"

### SOAP Request

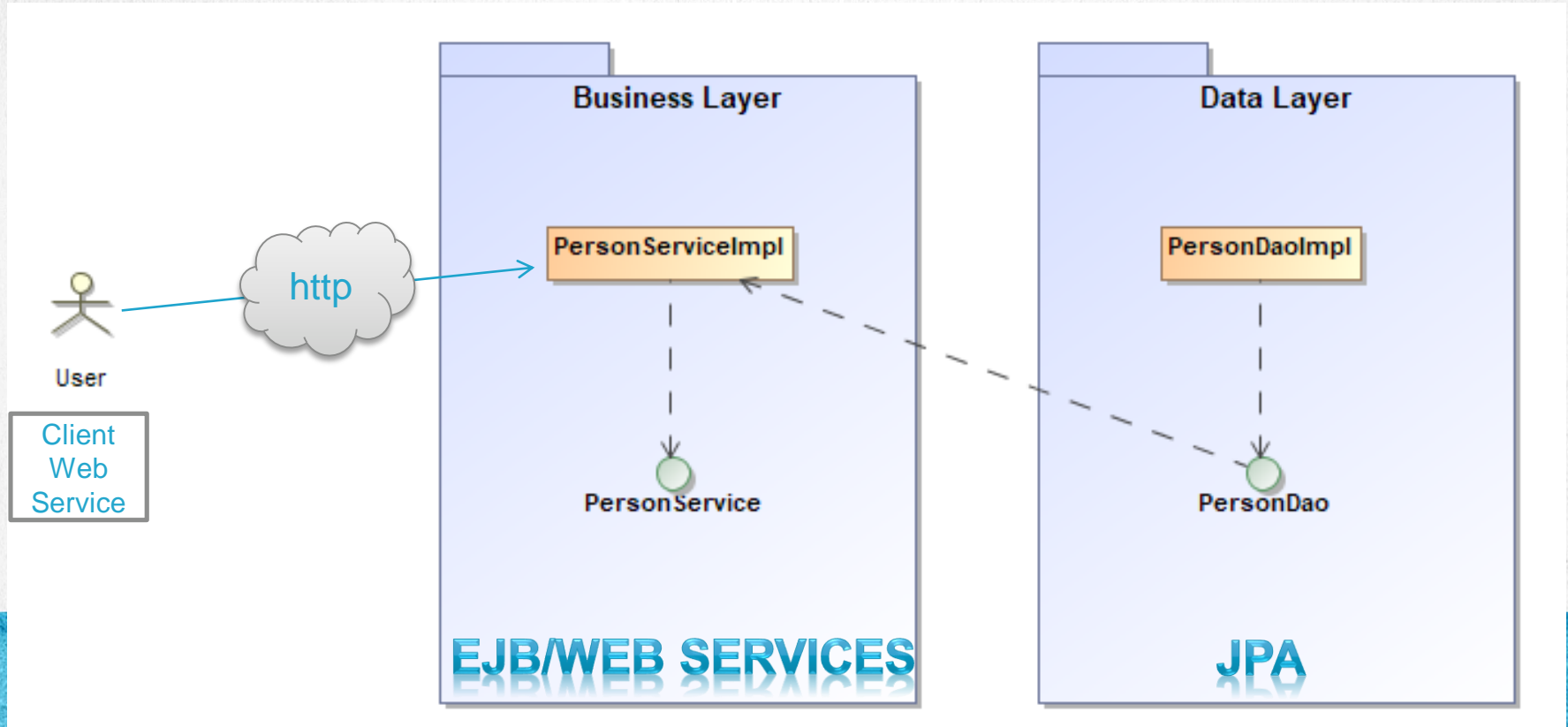
```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body xmlns:ns2="http://service.sms/">
    <ns2:listPeople/>
  </S:Body>
</S:Envelope>
```

### SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body xmlns:ns2="http://service.sms/">
    <ns2:listPeopleResponse>
      <return>
        <idPerson>1</idPerson>
        <name>John</name>
      </return>
      <return>
        <idPerson>2</idPerson>
        <name>Katty</name>
      </return>
      <return>
        <idPerson>3</idPerson>
        <name>Maria</name>
      </return>
    </ns2:listPeopleResponse>
  </S:Body>
</S:Envelope>
```

# ARCHITECTURE SMS WITH WEB SERVICES

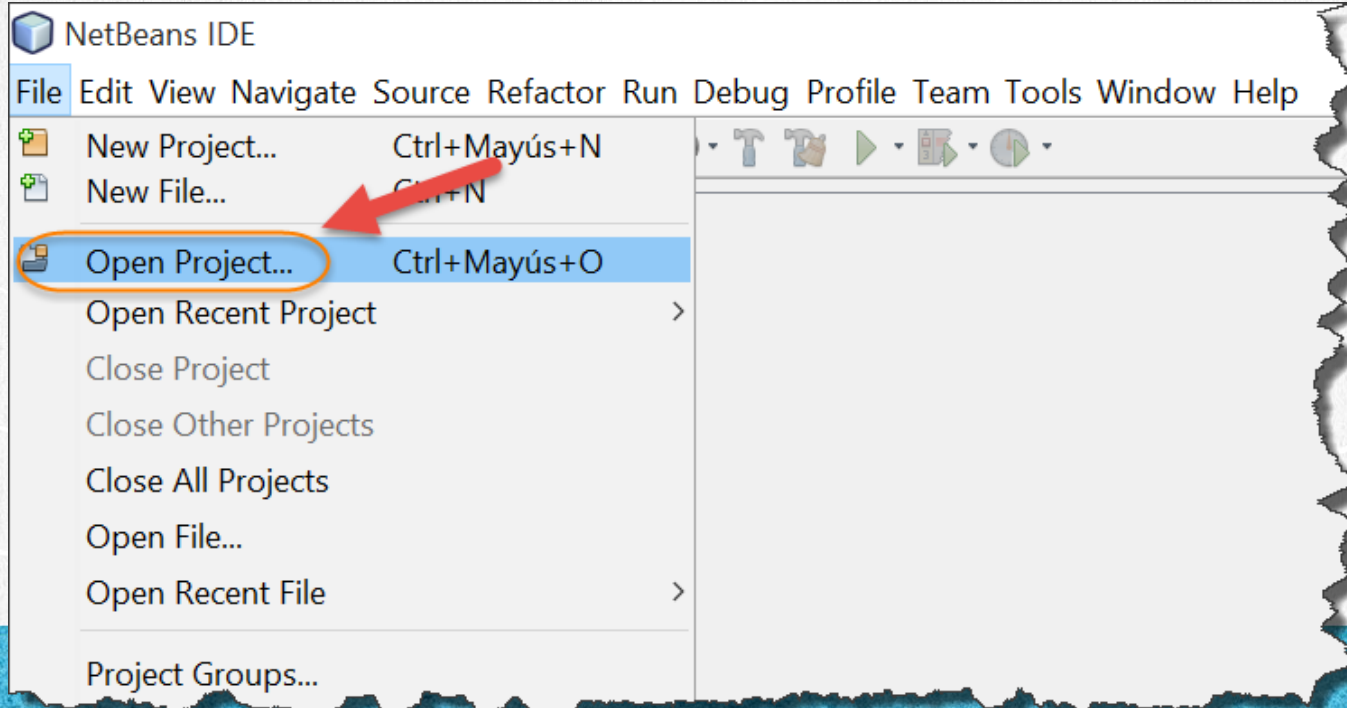
This is the Exercise Class Diagram, where you can see the Architecture of our System:





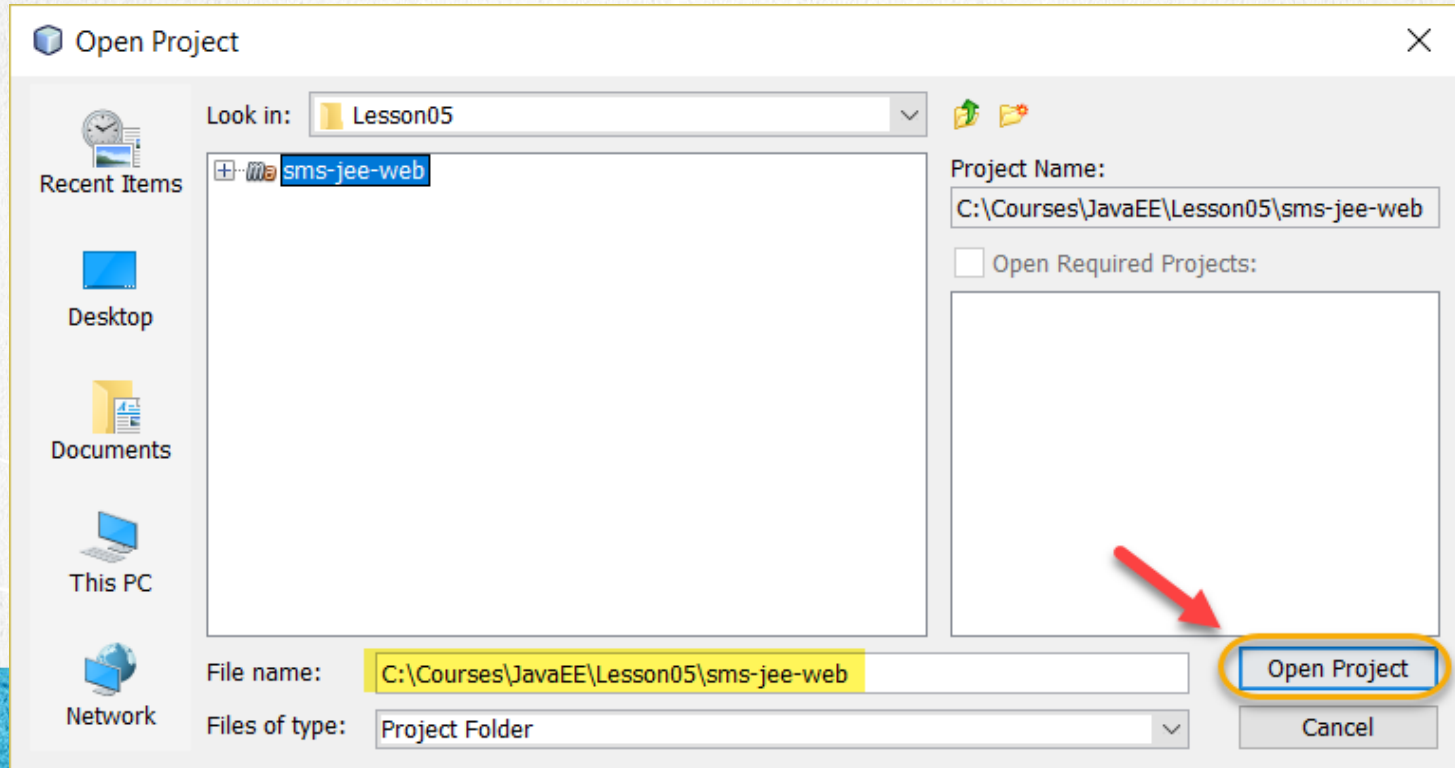
# 1. OPEN THE PROJECT

In case we do not have open the sga-jee-web project we open it:



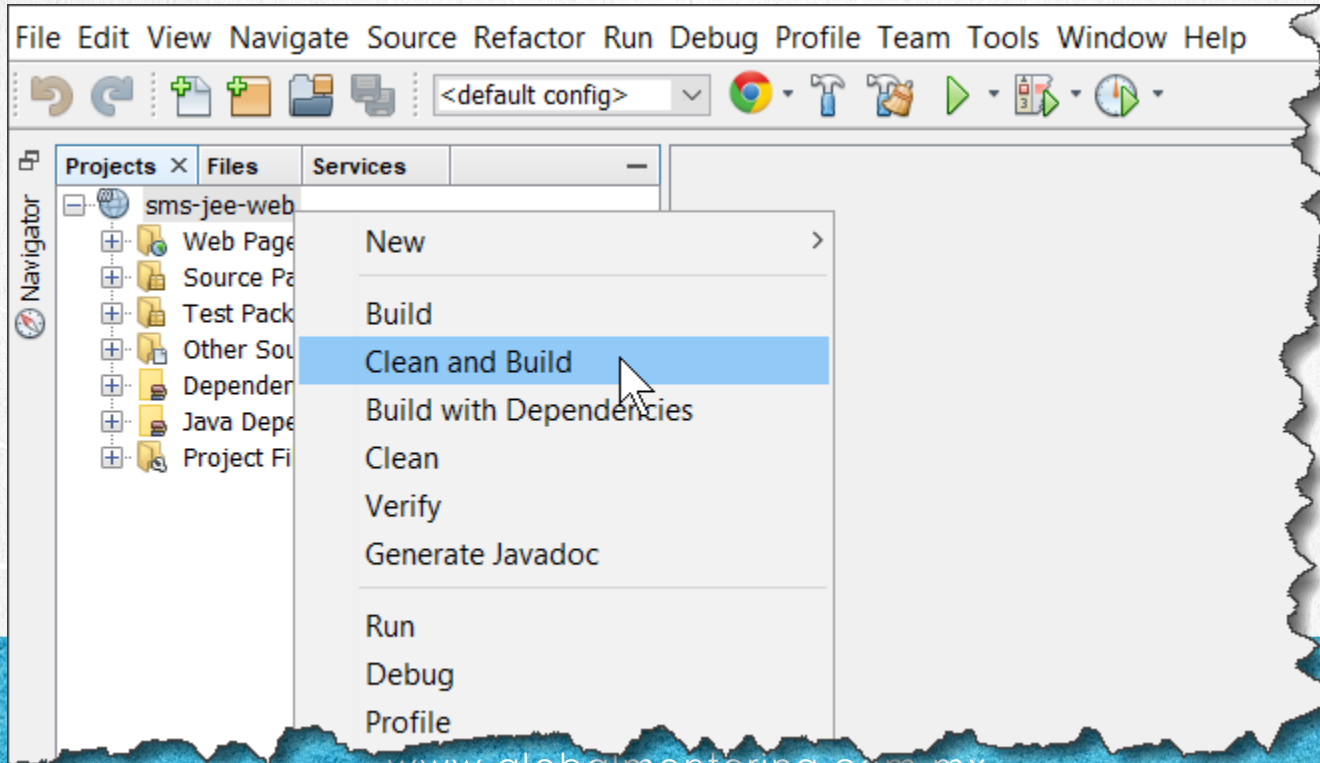
# 1. OPEN THE PROJECT

In case we do not have open the sms-jee-web project we open it:



# 1. OPEN THE PROJECT

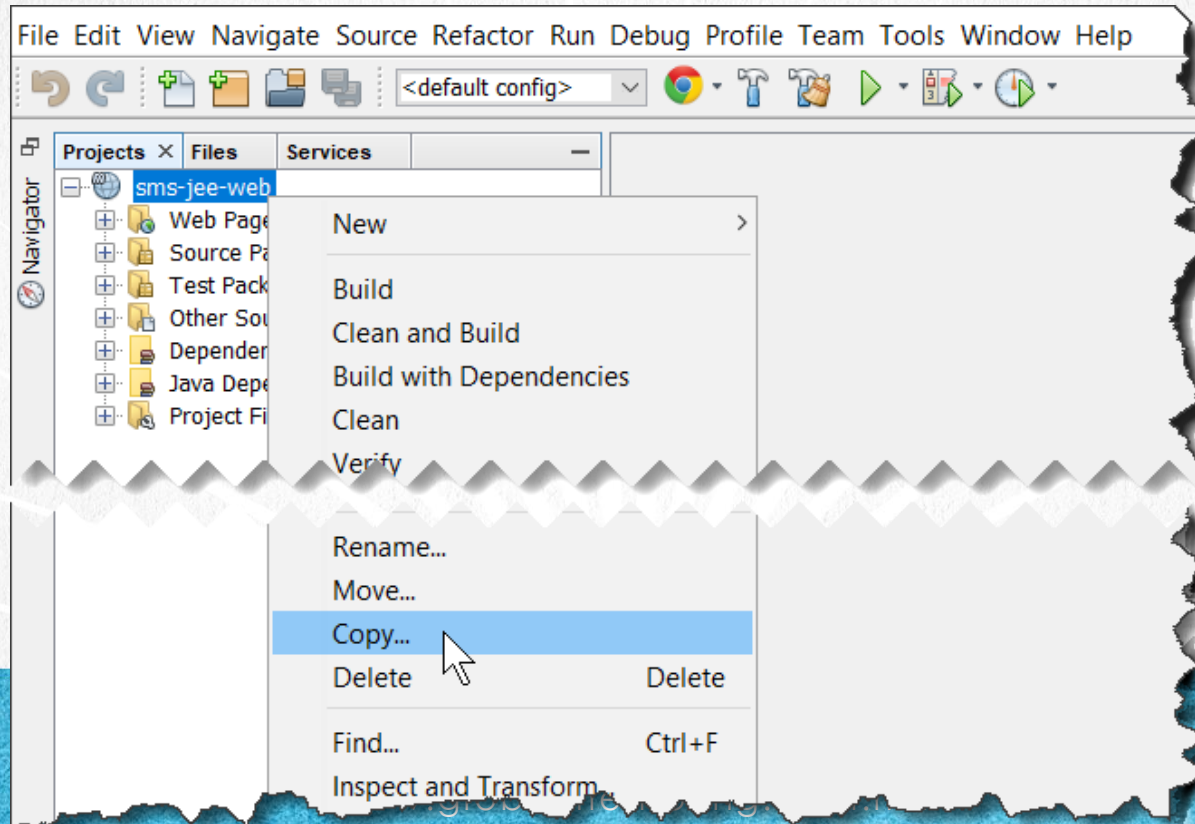
We wait for you to fully load the project. In case the project makes a mistake, we make a Clean & Build so that all the files are shown, this step is optional:





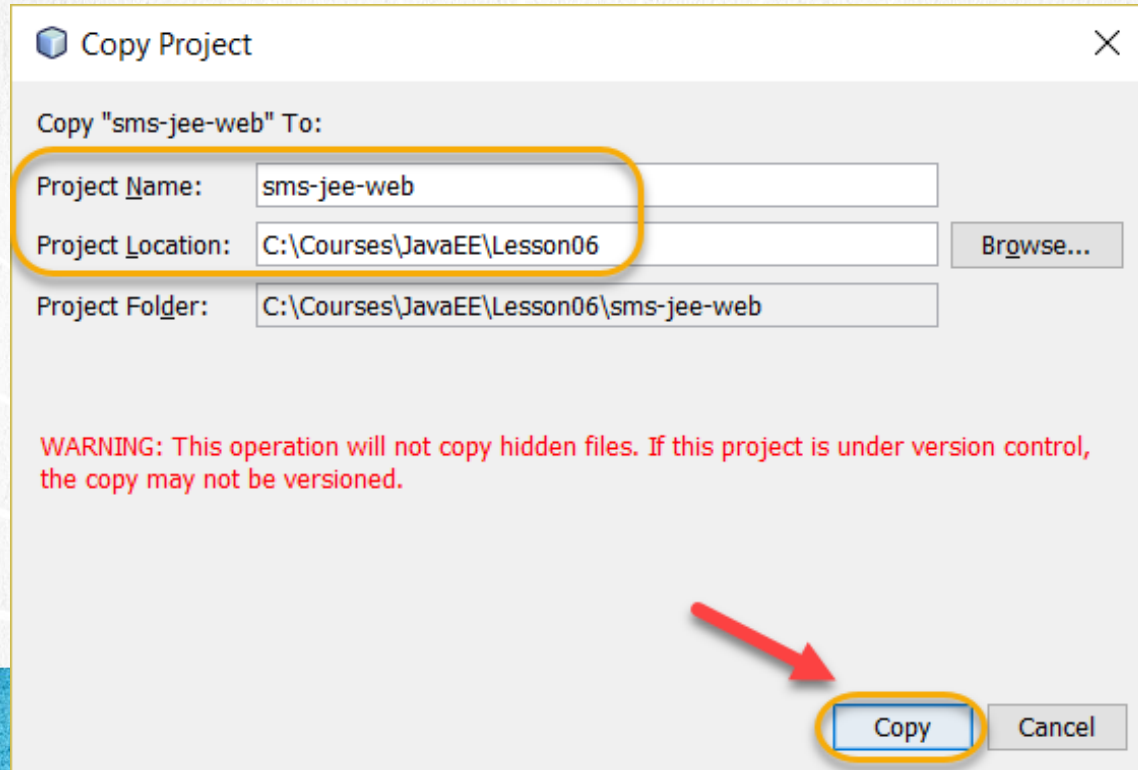
## 2. COPY THE PROJECT

We copy the project to put it in the new path:



## 2. COPY THE PROJECT

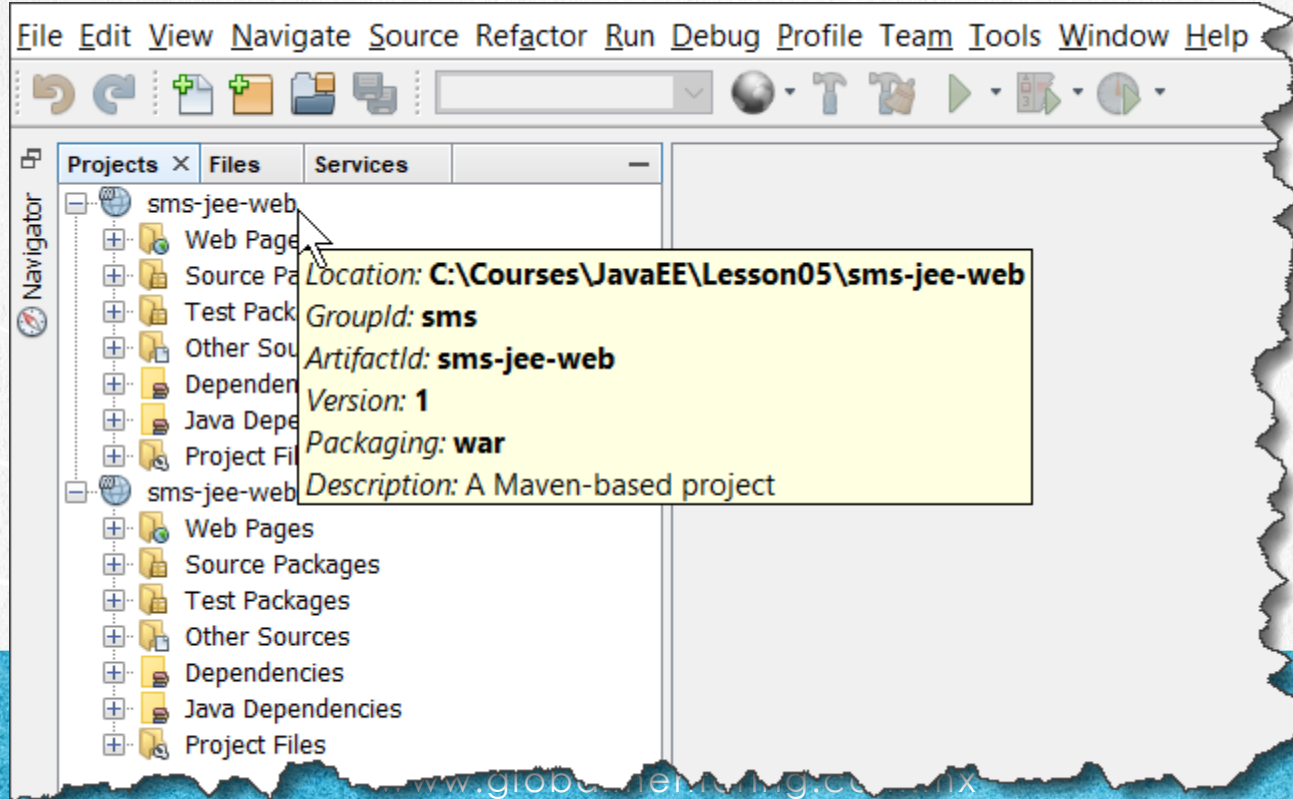
We copy the project to put it in the new path:





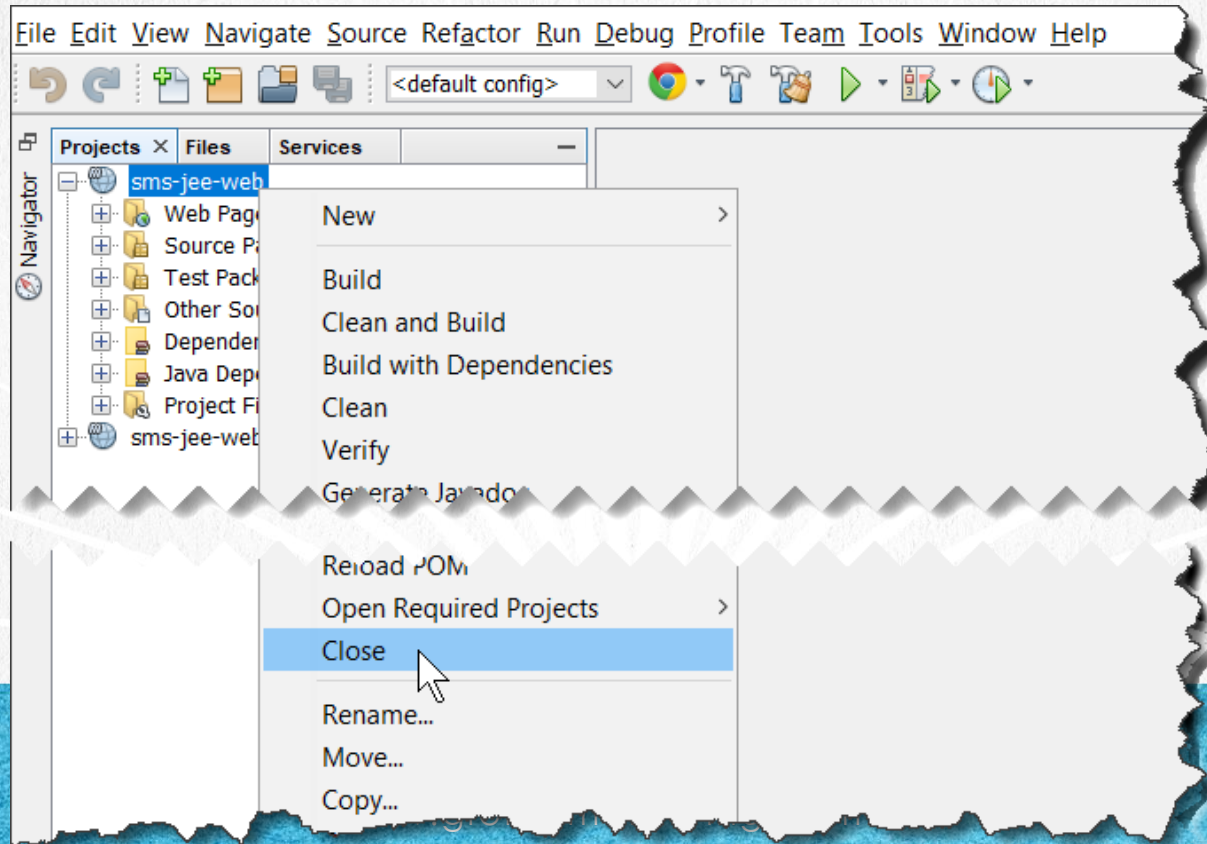
### 3. CLOSE THE PREVIOUS PROJECT

We closed the previous project, we identified it by positioning ourselves on the project:



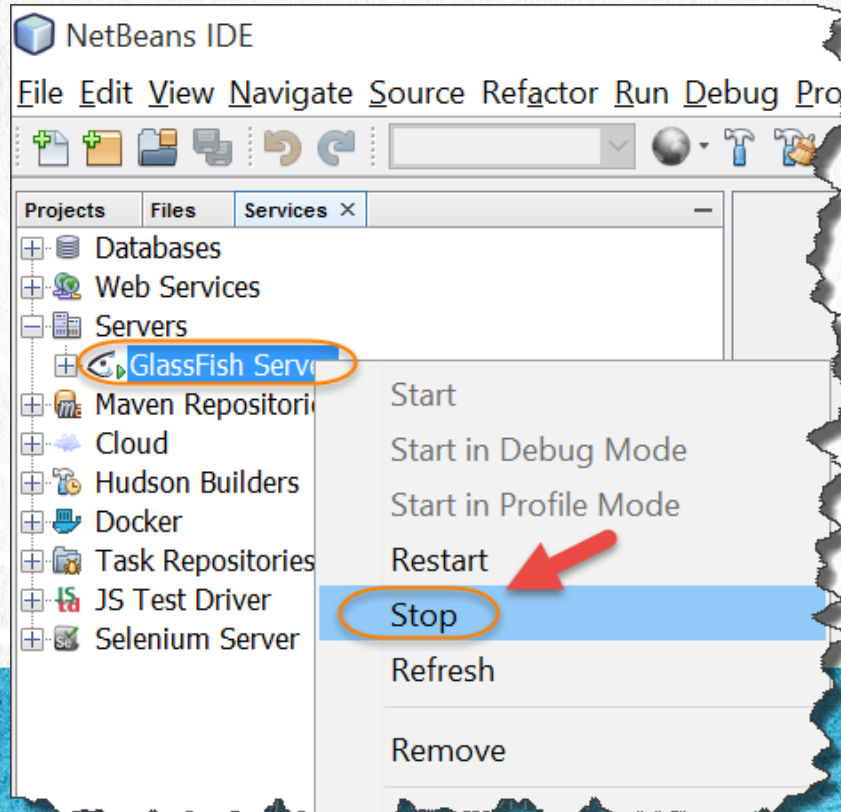
### 3. CLOSE THE PROJECT

We closed the previous project and left only the new one:



## 4. STOP GLASSFISH

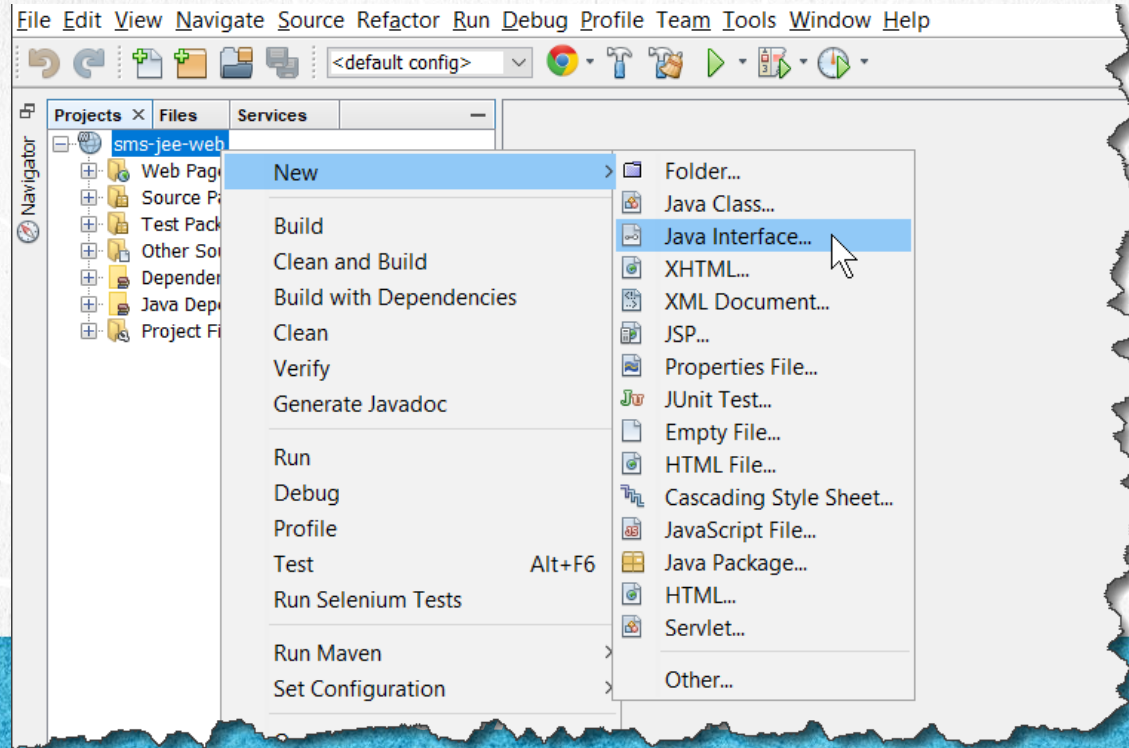
We stop the Glassfish server:





## 5. CREATE A JAVA CLASS

We create the PersonServiceWS.java interface:



## 5. CREATE A JAVA CLASS

We create the PersonServiceWS.java interface:

**New Java Interface**

**Steps**

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

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

< Back   Next >   **Finish**   Cancel   Help

## 6. MODIFY THE FILE

PersonServiceWs.java:

[Click to download](#)

```
package sms.service;

import java.util.List;
import javax.jws.WebMethod;
import javax.jws.WebService;
import sms.domain.Person;

@WebService
public interface PersonServiceWS {

    @WebMethod
    public List<Person> listPeople();
}
```

**CURSO DE JAVA EE**

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



## 7. MODIFY A JAVA CLASS

Modify the PersonServiceImpl class, adding the following changes:

1) Add the WebService entry, specifying which interface to use:

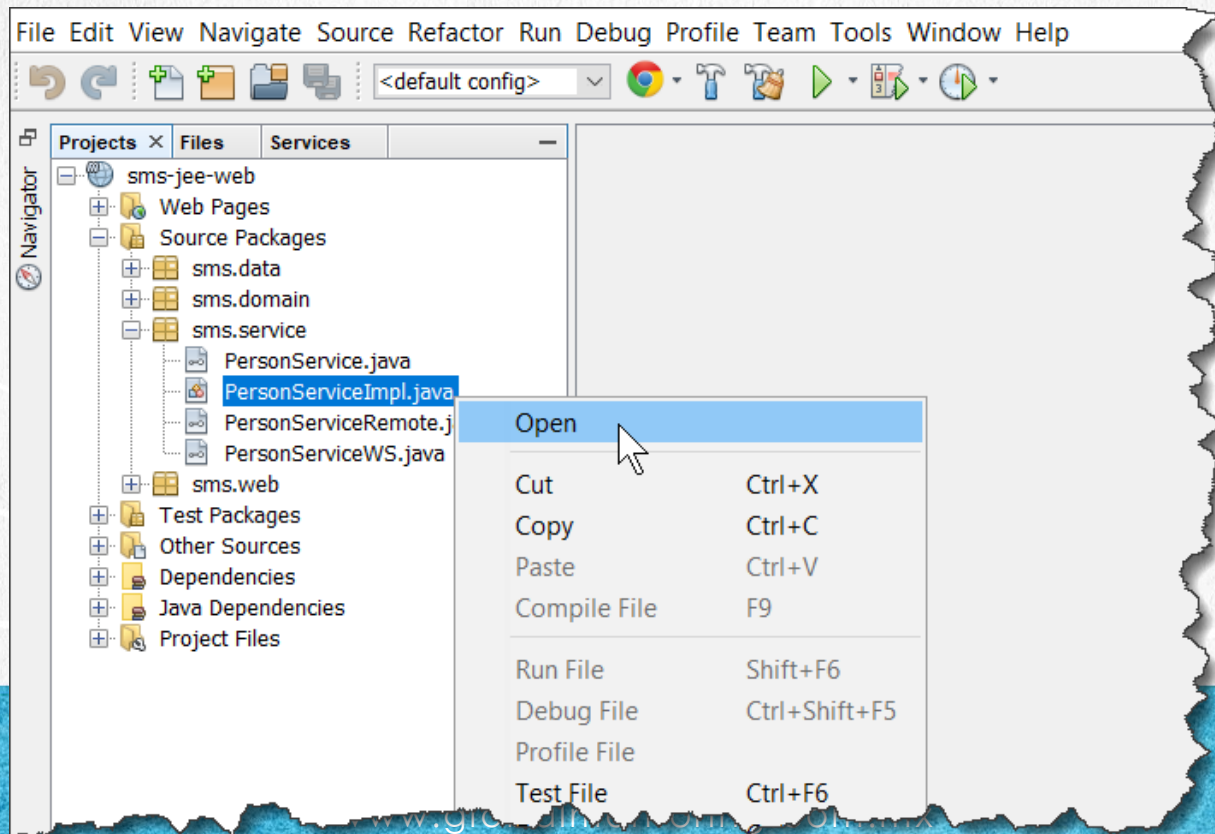
```
@WebService(endpointInterface = "sms.service.PersonServiceWS")
```

2) Extend the PersonServiceWS interface, so the definition of the class is as follows:

```
public class PersonServiceImpl  
    implements PersonServiceRemote, PersonService, PersonServiceWS
```

# 7. MODIFY A JAVA CLASS

Modify the PersonServiceImpl class:



## 7. MODIFY THE FILE

PersonServiceImpl.java:

[Click to download](#)

```
package sms.service;

import java.util.List;
import javax.annotation.Resource;
import javax.ejb.SessionContext;
import javax.ejb.Stateless;
import javax.inject.Inject;
import javax.jws.WebService;
import sms.data.PersonDao;
import sms.domain.Person;

@Stateless
@WebService(endpointInterface = "sms.service.PersonServiceWS")
public class PersonServiceImpl implements PersonServiceRemote, PersonService, PersonServiceWS {

    @Resource
    private SessionContext context;

    @Inject
    private PersonDao personDao;
```



## 7. MODIFY THE FILE

PersonServiceImpl.java:

[Click to download](#)

```
@Override
public List<Person> listPeople() {
    return personDao.findAllPeople();
}

@Override
public Person findPerson(Person person) {
    return personDao.findPerson(person);
}

@Override
public void addPerson(Person person) {
    personDao.insertPerson(person);
}

@Override
public void modifyPerson(Person person) {
    try {
        personDao.updatePerson(person);
    } catch (Throwable t) {
        context.setRollbackOnly();
        t.printStackTrace(System.out);
    }
}
```

## 7. MODIFY THE FILE

[PersonServiceImpl.java:](#)

Click to download

```
@Override
public void deletePerson(Person person) {
    personDao.deletePerson(person);
}
}
```

## 8. MODIFY A JAVA CLASS

Modify the Person domain class, adding the following changes:

Add the @XmlAccessorType annotation at the class level, specifying which interface to use:  
@XmlAccessorType(XmlAccessType.FIELD)

*This annotation of the JAXB API means that each of the attributes of the class will be used in the Web Service, and will be validated with the XSD schema, which is part of the Web Service message.*



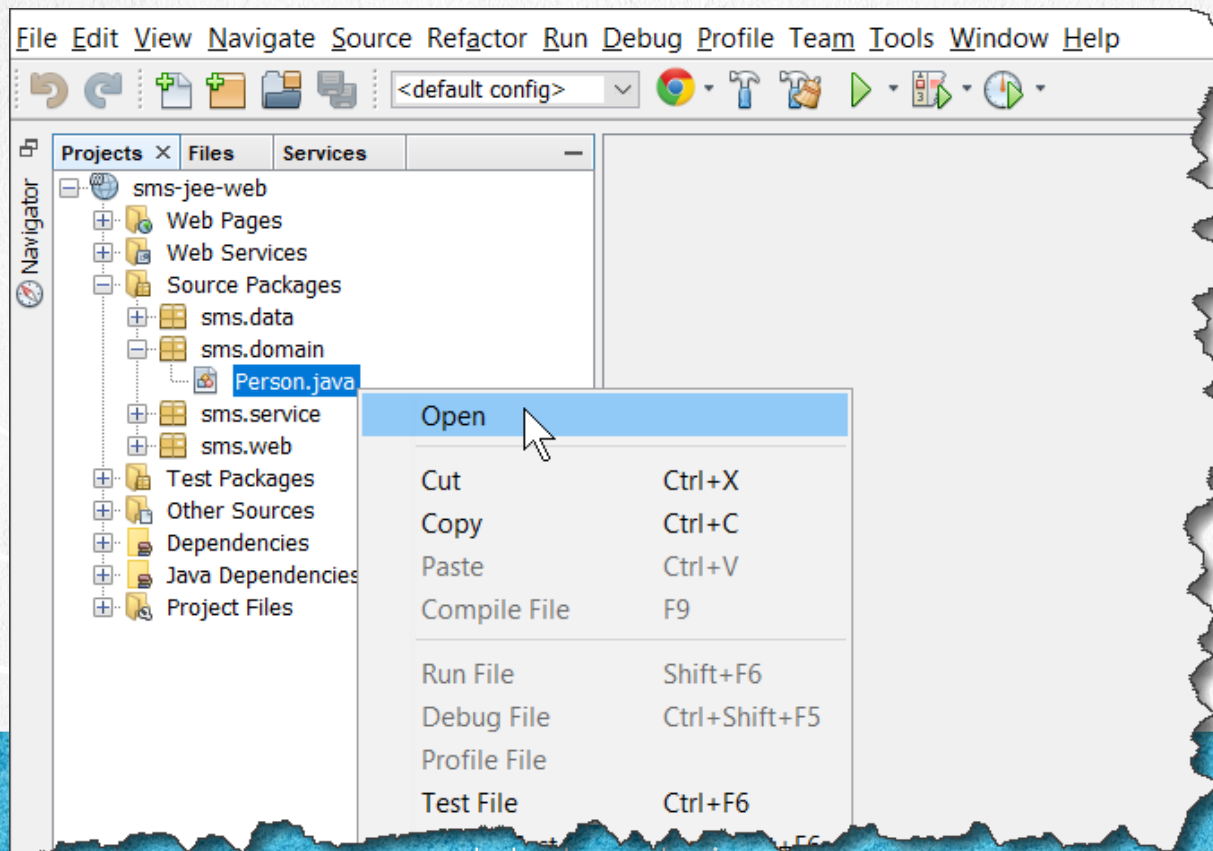
**JAVA EE COURSE**

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



## 8. MODIFY A JAVA FILE

We modify the Person domain class:



## 8. MODIFY THE FILE

Person.java:

Click to download

```
package sms.domain;

import java.io.Serializable;
import javax.persistence.*;
import javax.xml.bind.annotation.*;

@Entity
@NamedQueries({
    @NamedQuery(name = "Person.findAll", query = "SELECT p FROM Person p ORDER BY p.idPerson") })
@Table(name = "person")
@XmlAccessorType(XmlAccessType.FIELD)
public class Person implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_person")
    private int idPerson;

    private String name;

    public Person() {
    }
}
```

## 8. MODIFY THE FILE

Person.java:

Click to download

```
public Person(int idPersona, String name) {
    this.idPerson = idPersona;
    this.name = name;
}

public int getIdPerson() {
    return idPerson;
}

public void setIdPerson(int idPerson) {
    this.idPerson = idPerson;
}

public String getName() {
    return name;
}

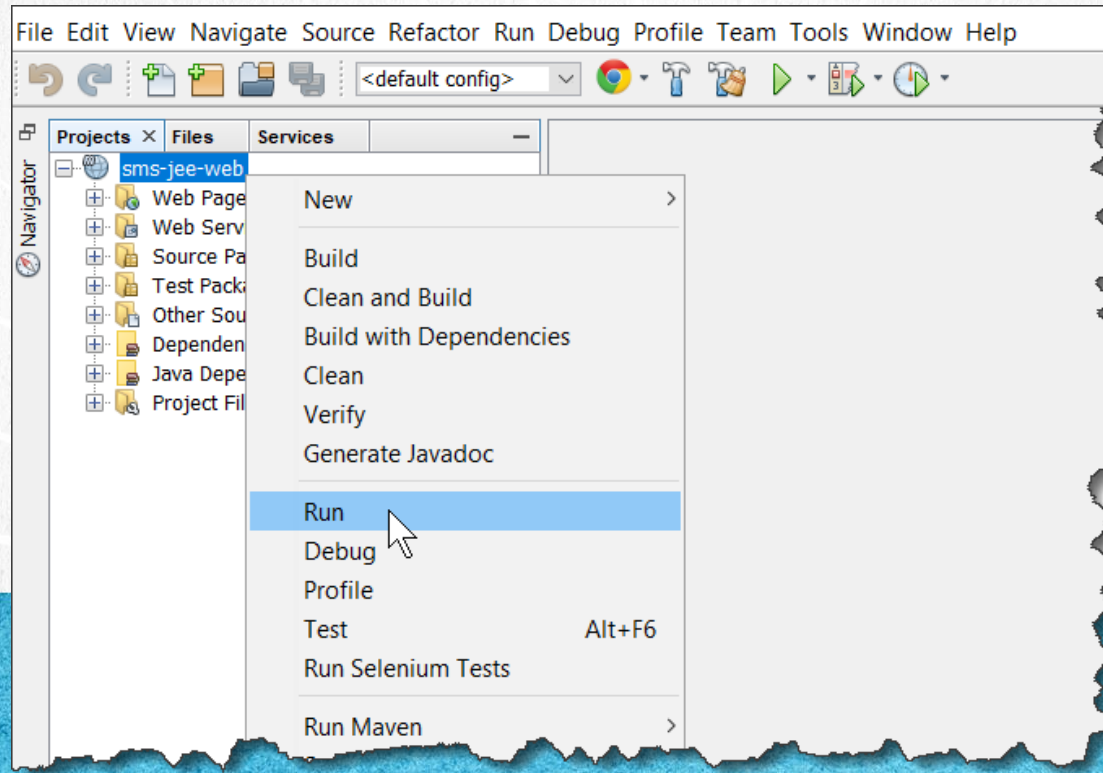
public void setName(String name) {
    this.name = name;
}

@Override
public String toString() {
    return "Person{" + "idPerson=" + idPerson + ", name=" + name + '}';
}
}
```



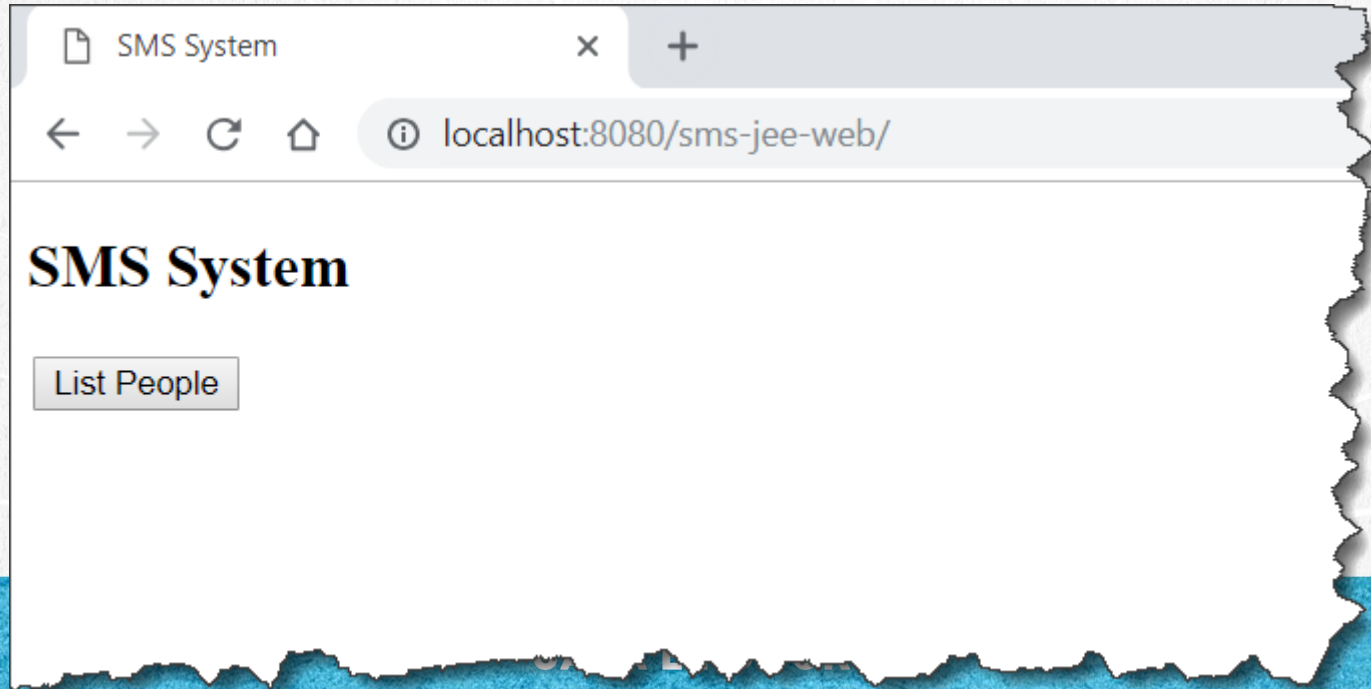
## 9. DEPLOY ON GLASSFISH

We run the application, and this will automatically deploy the application, including the Web Service:



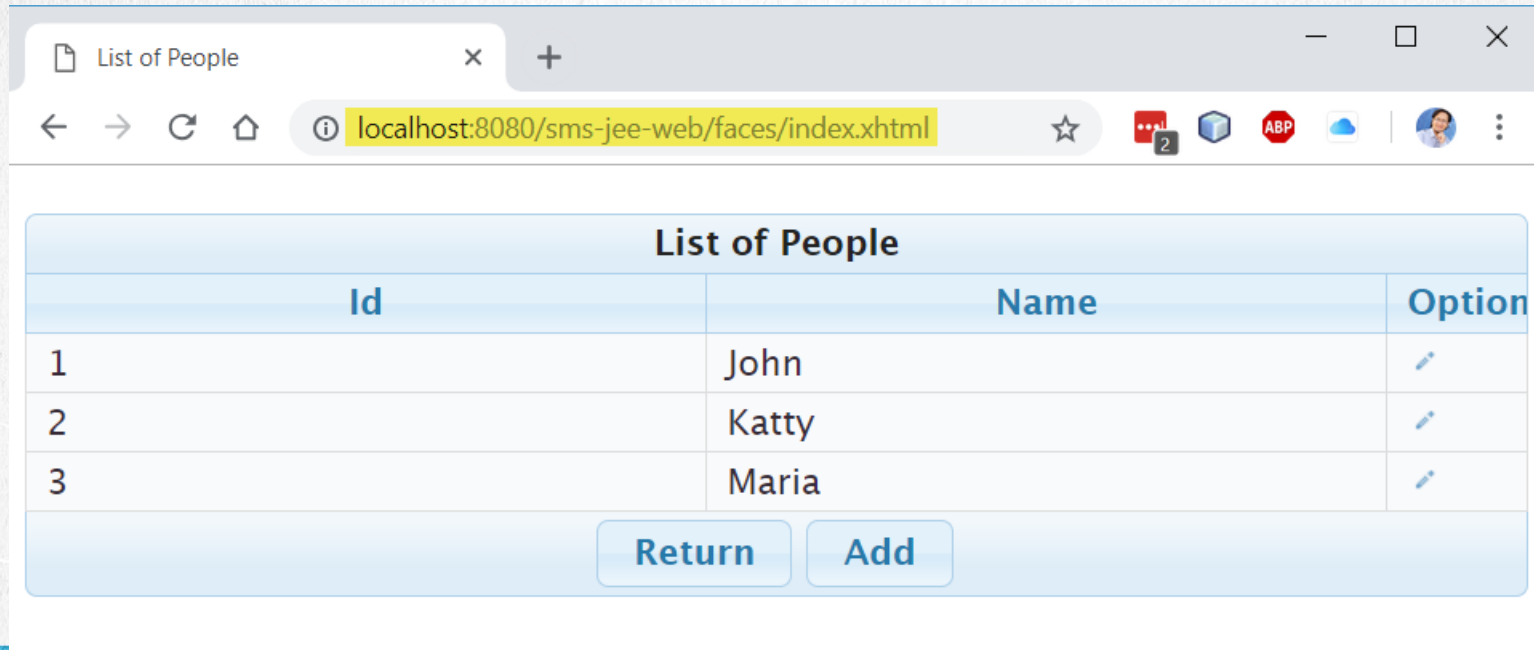
## 9. DEPLOY ON GLASSFISH

We execute the application. We verify the list of people only to know the data that we should obtain in the web service client:






## 9. DEPLOY ON GLASSFISH

We execute the application. We verify the list of people :



The screenshot shows a web browser window with the title 'List of People'. The address bar displays 'localhost:8080/sms-jee-web/faces/index.xhtml'. The main content area features a table titled 'List of People' with the following data:

Id	Name	Option
1	John	
2	Katty	
3	Maria	

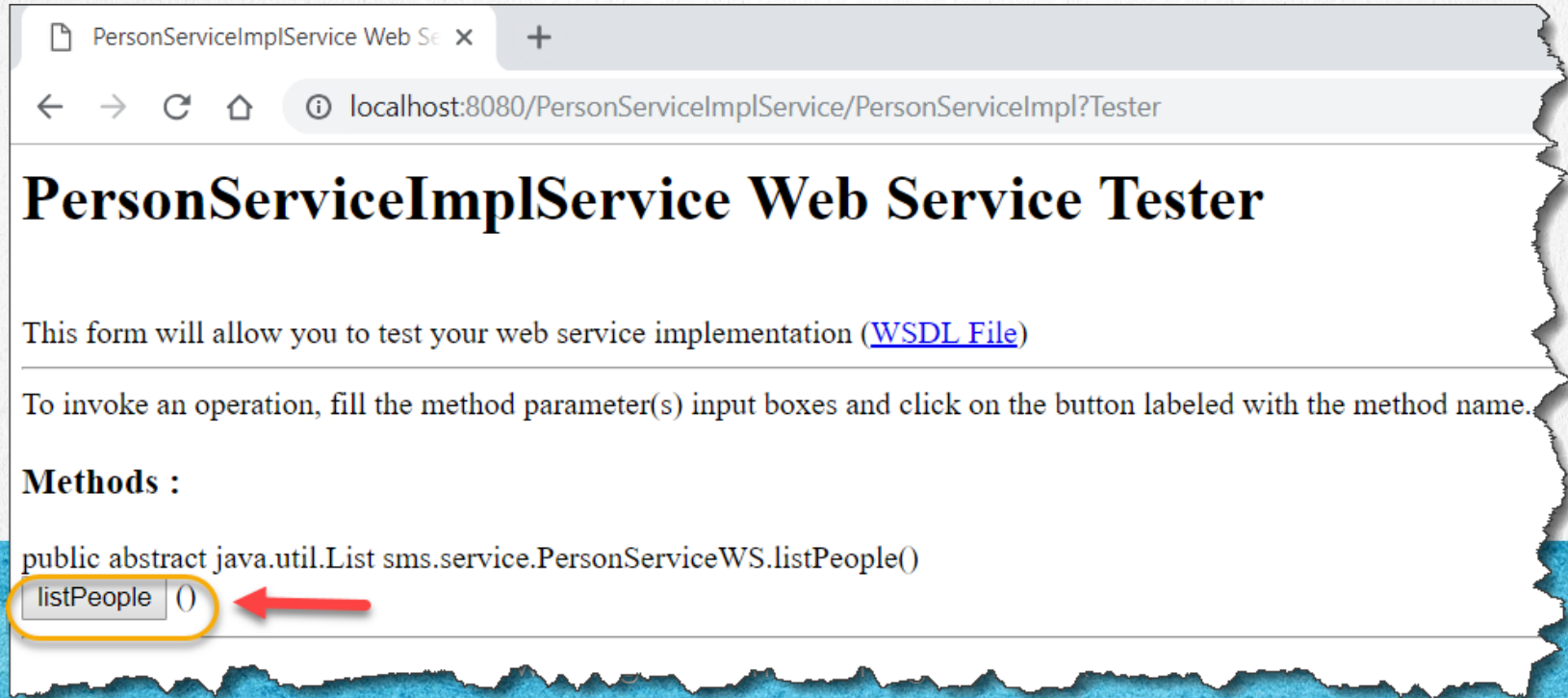
Below the table, there are two buttons: 'Return' and 'Add'.



# 10. TESTING THE WEB SERVICE

Once the application is deployed in Glassfish, we verify the web service of posted People. With the following URLs you can run the Web Services Test. Test URL:

<http://localhost:8080/PersonServiceImplService/PersonServiceImpl?Tester>



# 10. TESTING THE WEB SERVICE

The result should be similar to the following (values may vary):

## listPeople Method invocation

### Method parameter(s)

Type	Value
------	-------

### Method returned

`java.util.List : "[sms.service.Person@4cfdec5b, sms.service.Person@63c1fd01, sms.service.Person@27e660fa]"`

### SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body xmlns:ns2="http://service.sms/">
    <ns2:listPeople/>
  </S:Body>
</S:Envelope>
```

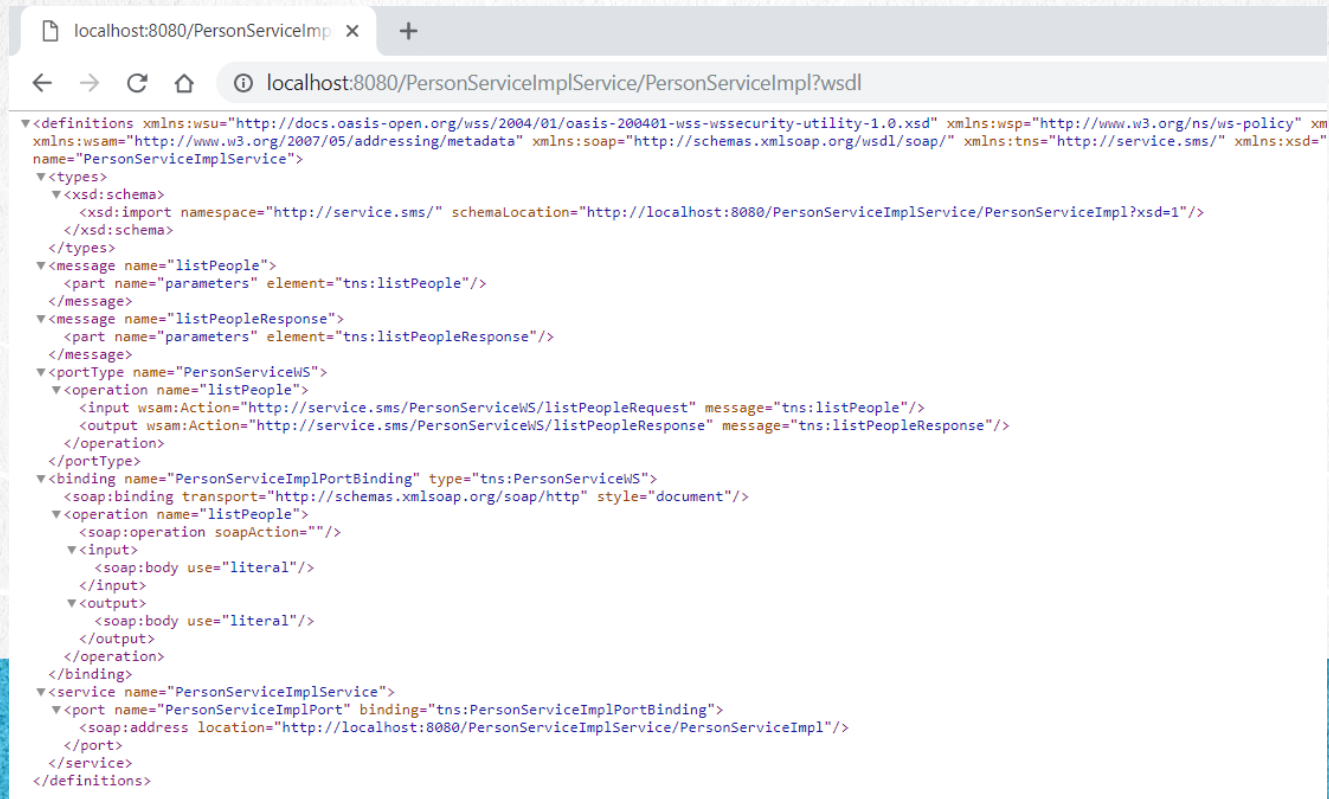
### SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body xmlns:ns2="http://service.sms/">
    <ns2:listPeopleResponse>
      <return>
        <idPerson>1</idPerson>
        <name>John</name>
      </return>
      <return>
        <idPerson>2</idPerson>
        <name>Katty</name>
      </return>
      <return>
        <idPerson>3</idPerson>
        <name>Maria</name>
      </return>
    </ns2:listPeopleResponse>
  </S:Body>
</S:Envelope>
```

# 11. CHECKING THE WSDL

Review of the wsdl document:

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



The screenshot shows a web browser window with the address bar displaying `localhost:8080/PersonaServiceImplService/PersonaServiceImpl?wsdl`. The browser's developer tools are open, showing the XML content of the WSDL document. The XML is a WSDL 1.1 document with the following structure:

```
<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-policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://service.sms/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="PersonaServiceImplService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://service.sms/" schemaLocation="http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl?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="PersonaServiceImplPortBinding" 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>
  </binding>
  <service name="PersonaServiceImplService">
    <port name="PersonaServiceImplPort" binding="tns:PersonaServiceImplPortBinding">
      <soap:address location="http://localhost:8080/PersonaServiceImplService/PersonaServiceImpl"/>
    </port>
  </service>
</definitions>
```



## 12. CHECKING THE XSD FILE

Review of the xsd document:

<http://localhost:8080/PersonServiceImplService/PersonServiceImpl?xsd=1>



```
<?xml version='1.0' encoding='UTF-8'>
<xs:schema xmlns:tns="http://service.sms/" xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0" targetNamespace="http://service.sms/">
  <xs:element name="listPeople" type="tns:listPeople"/>
  <xs:element name="listPeopleResponse" type="tns:listPeopleResponse"/>
  <xs:complexType name="listPeople">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="listPeopleResponse">
    <xs:sequence>
      <xs:element name="return" type="tns:person" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="person">
    <xs:sequence>
      <xs:element name="idPerson" type="xs:int"/>
      <xs:element name="name" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

**JAVA EE COURSE**

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

# EXERCISE CONCLUSION

With this exercise we have modified our sms-jee-web project to add the web service of the list of people.

We check that the WSDL, XSD is available and we did a test of the people list web service.

In the following exercise we will create the client to consume this web service of list of people.



**JAVA EE COURSE**

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



# ONLINE COURSE

# JAVA EE

# JAKARTA EE

---

By: Eng. Ubaldo Acosta



**JAVA EE COURSE**  
[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)