

**JAVA EE COURSE**

# **EXERCISE**

## **SMS SYSTEM AND SESSION EJB**

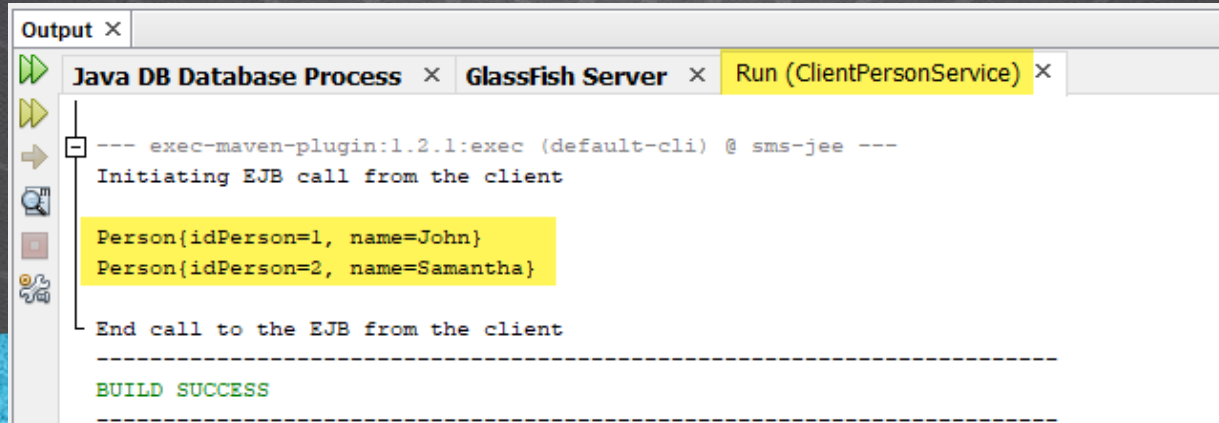


**JAVA EE COURSE**

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

# EXERCISE OBJECTIVE

- The objective of the exercise is to add a Session EJB to our SMS (Student Management System) project, which we will develop throughout the course.
- At the end we must observe the following result:



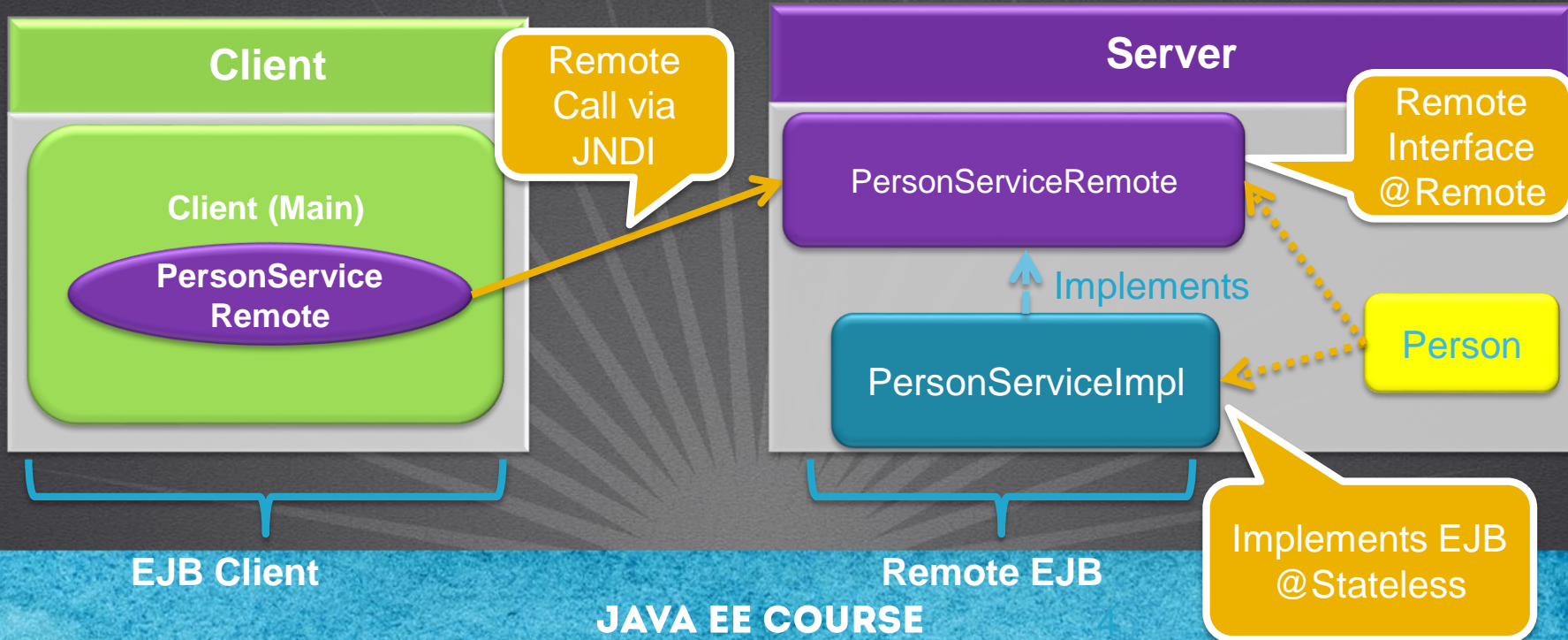
The screenshot shows an IDE output window with three tabs: "Java DB Database Process", "GlassFish Server", and "Run (ClientPersonService)". The "Run (ClientPersonService)" tab is active and displays the following output:

```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ sms-jee ---  
Initiating EJB call from the client  
  
Person{idPerson=1, name=John}  
Person{idPerson=2, name=Samantha}  
  
End call to the EJB from the client  
-----  
BUILD SUCCESS  
-----
```



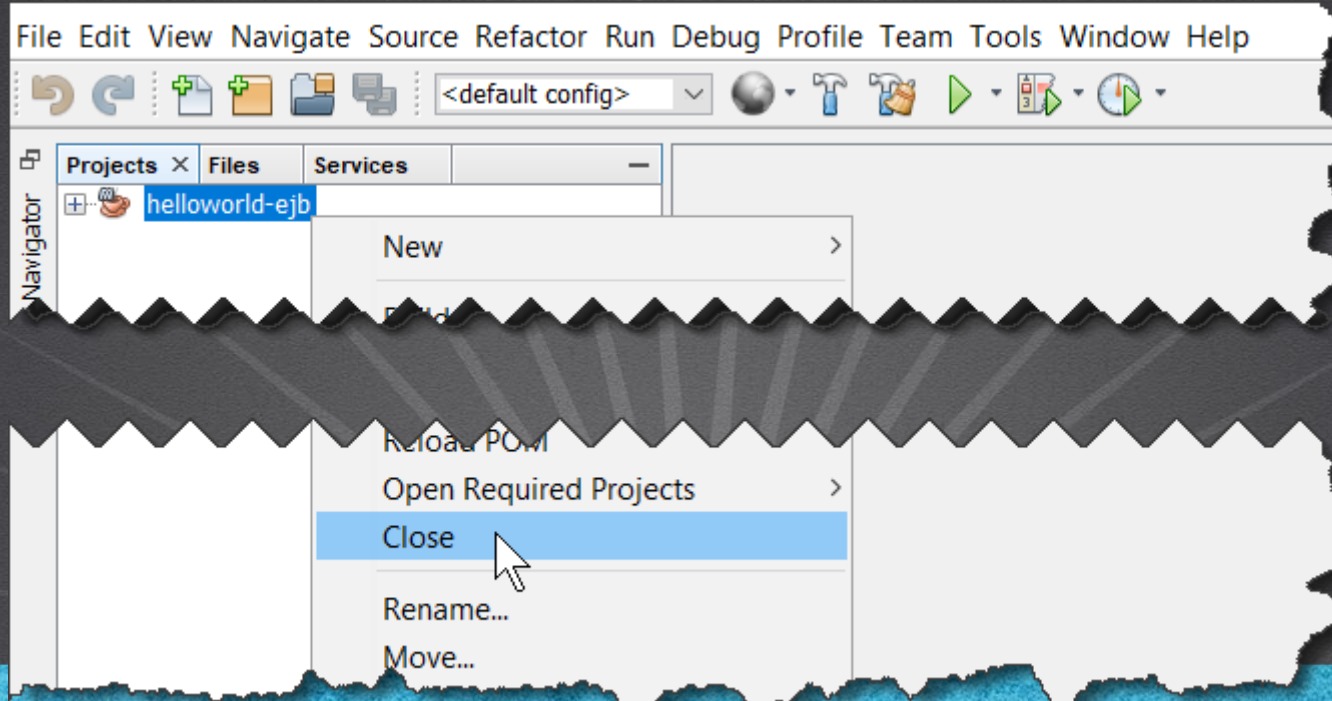
# JAVA EE ARCHITECTURE

•Throughout the course we will be adding components to our System SMS (Student Management System), which will be responsible for managing a catalog of people. This application is a Web application, but you can have remote clients and Web Services. Let's start with the following architecture:



# CLOSE ANY OTHER PROJECT THAT WE DO NOT USE

We close any other project that we no longer use:



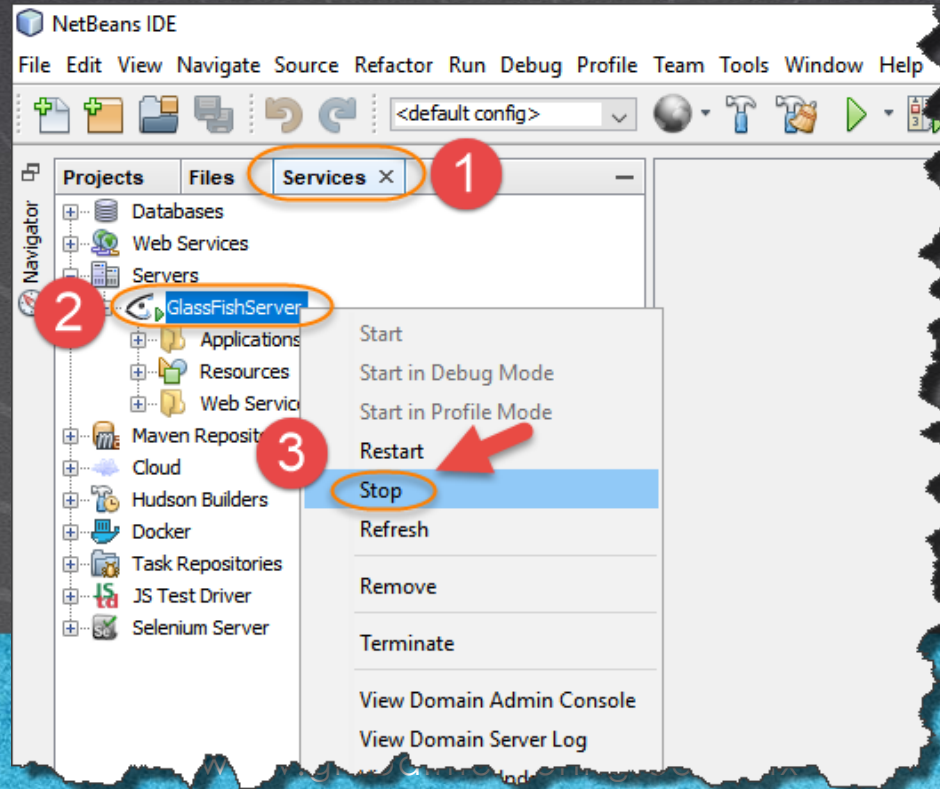
**JAVA EE COURSE**

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



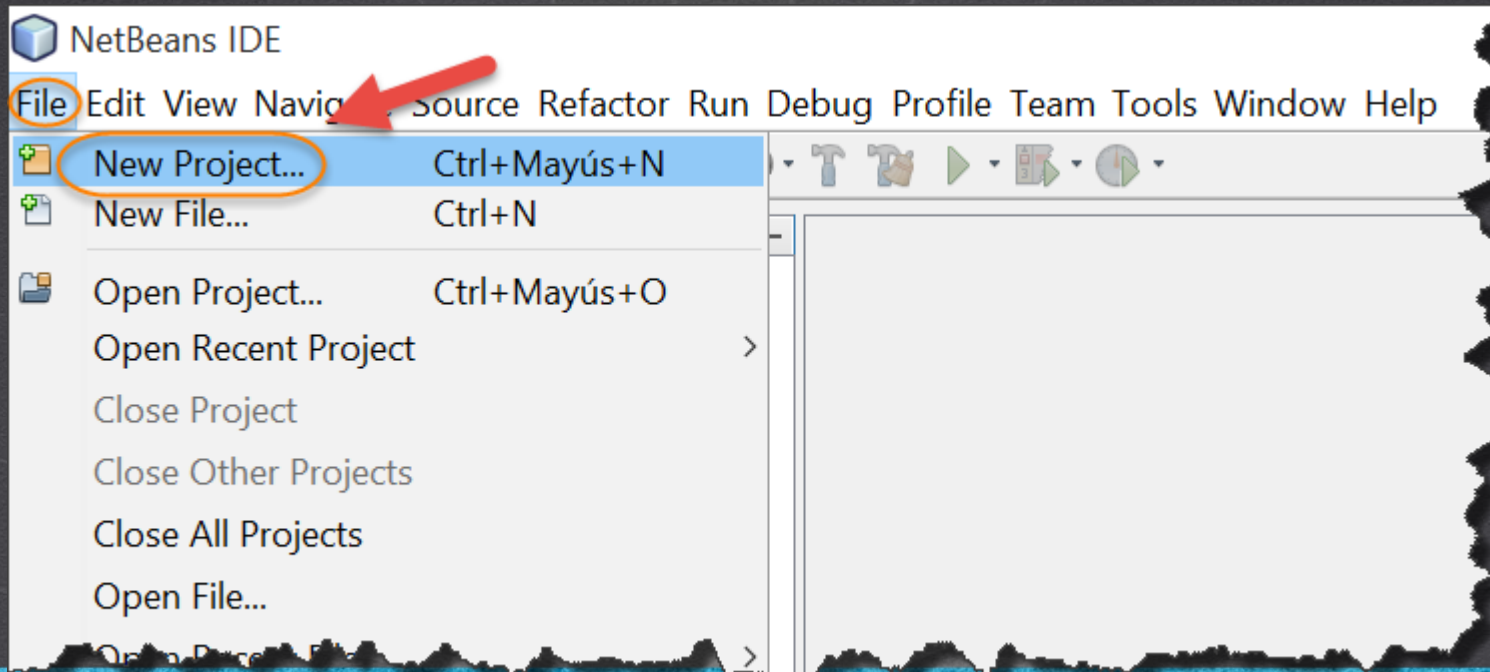
# STOP GLASSFISH IF IT IS ACTIVE

Stop the Glassfish server if it was started:



# 1. CREATE A NEW PROJECT

We create the sms-je project:

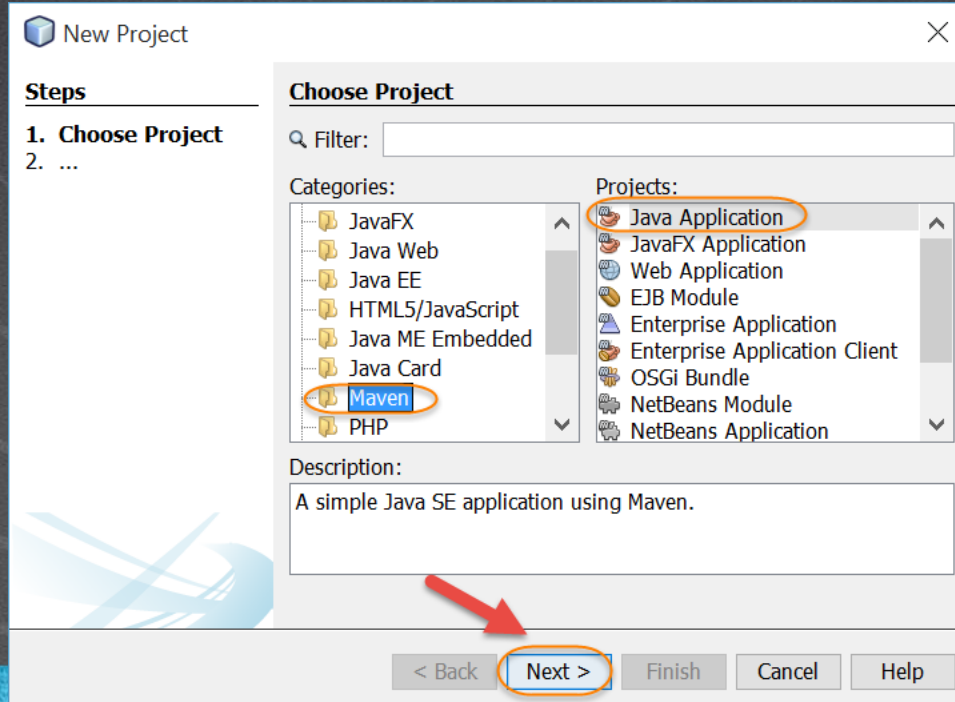


**JAVA EE COURSE**

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

# 1. CREATE A NEW PROJECT

We create the sms-jee project:



**JAVA EE COURSE**

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



# 1. CREATE A NEW PROJECT

We create the sms-jee project:

**New Java Application**

**Steps**

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

**Name and Location**

Project Name: sms-jee

Project Location: C:\Courses\JavaEE\Lesson01 Browse...

Project Folder: C:\Courses\JavaEE\Lesson01\sms-jee

Artifact Id: sms-jee

Group Id: sms

Version: 1

Package: (Optional)

< Back Next > **Finish** Cancel Help

**JAVA EE COURSE**

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



# CREATION OF THE PROJECT STRUCTURE

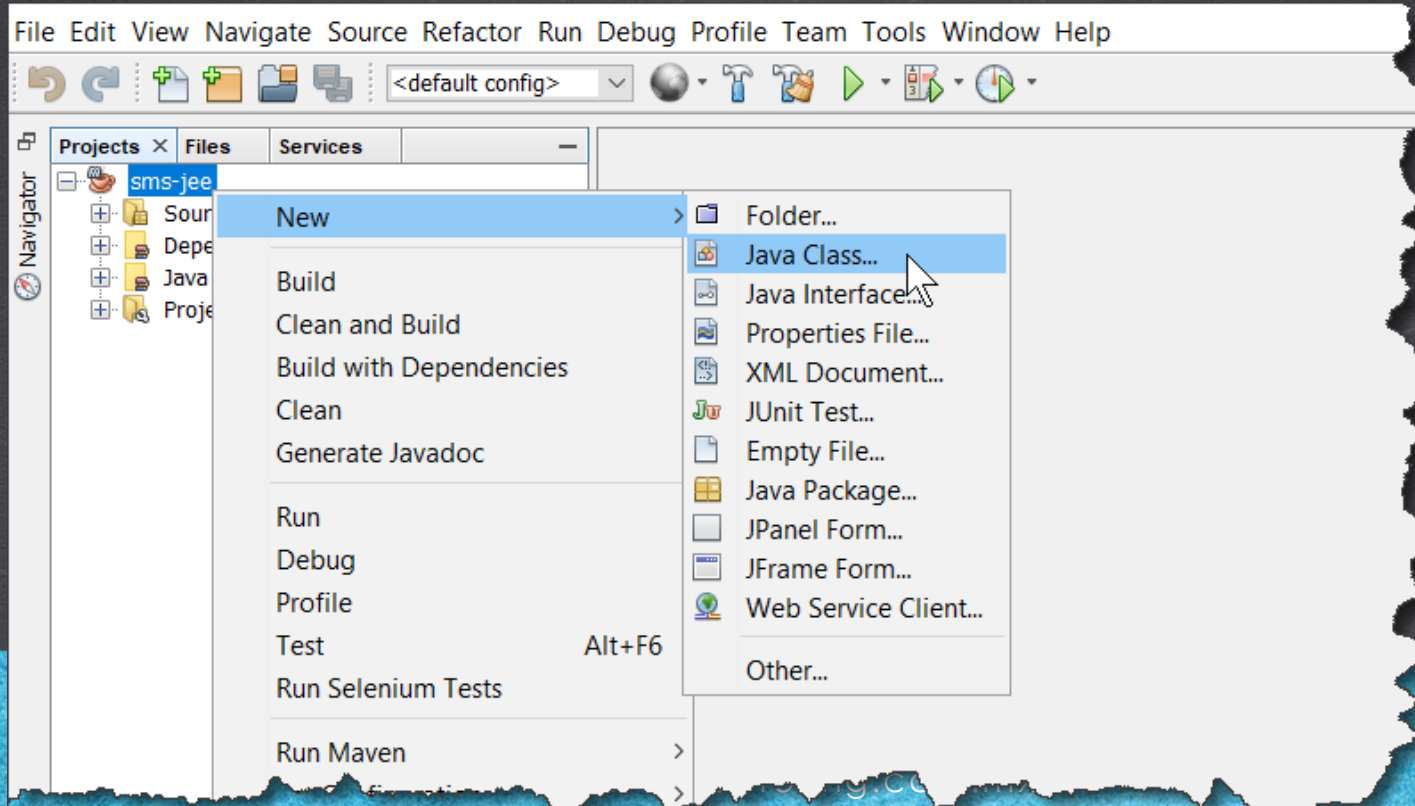
First we will create the entire structure of the project, and then we will begin to modify the code of the same.

The Java files that we are going to create are:

- sms.domain.Person.java
- sms.service.PersonServiceRemote.java
- sms.service.PersonServiceImpl.java
- sms.client.ClientPersonService.java
- sms.client.ClientPersonServiceWithIP.java

## 2. CREATE A NEW CLASS

We create the Person.java class:



## 2. CREATE A NEW CLASS

We create the Person.java class:

**New Java Class**

**Steps**

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

**Name and Location**

Class Name: Person

Project: sms-jee

Location: Source Packages

Package: sms.domain

Created File: C:\Courses\JavaEE\Lesson01\sms-jee\src\main\java\sms\domain\Person.java

< Back Next > **Finish** Cancel Help

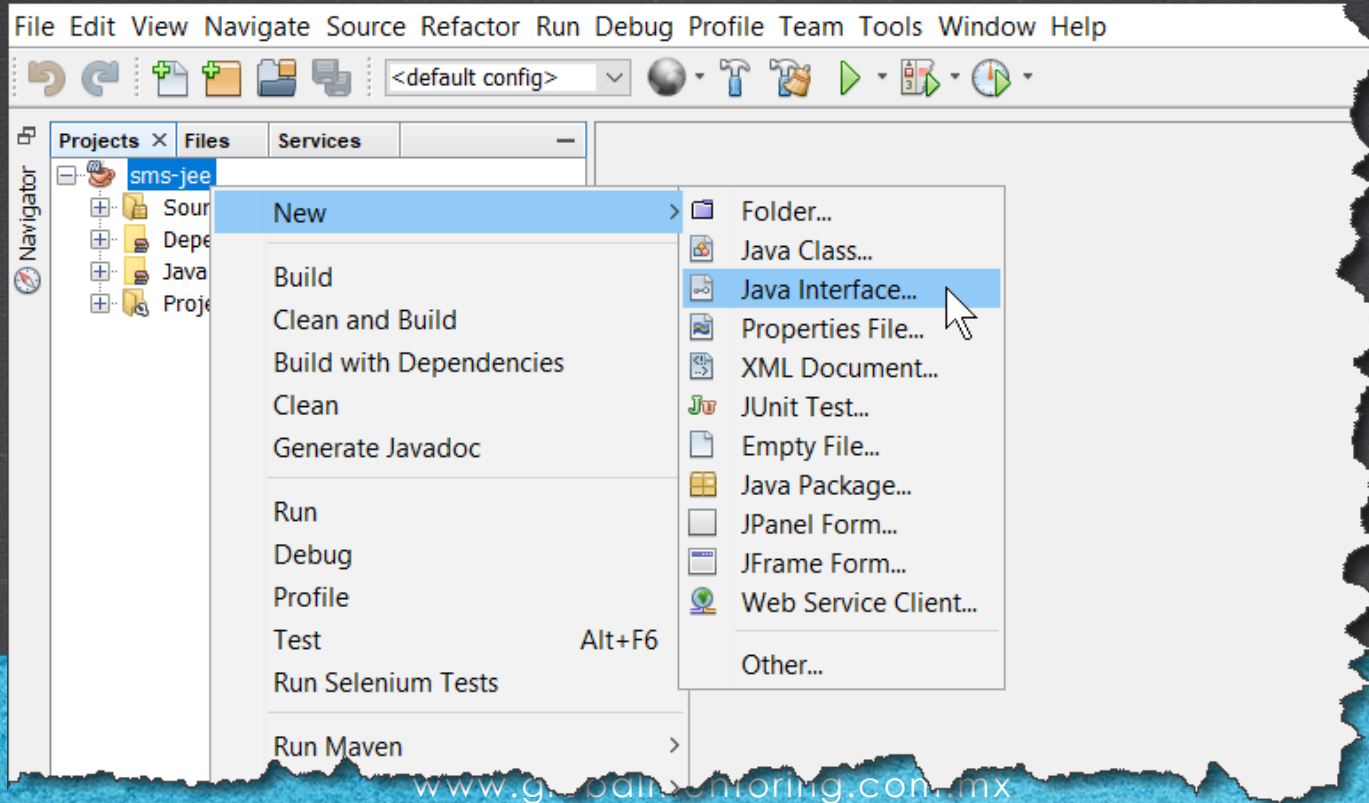
**JAVA EE COURSE**

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



### 3. CREATE AN INTERFACE

We created the PersonServiceRemote.java interface:



### 3. CREATE AN INTERFACE

We created the PersonServiceRemote.java interface:

**New Java Interface**

**Steps**

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

**Name and Location**

Class Name: PersonServiceRemote

Project: sms-jee

Location: Source Packages

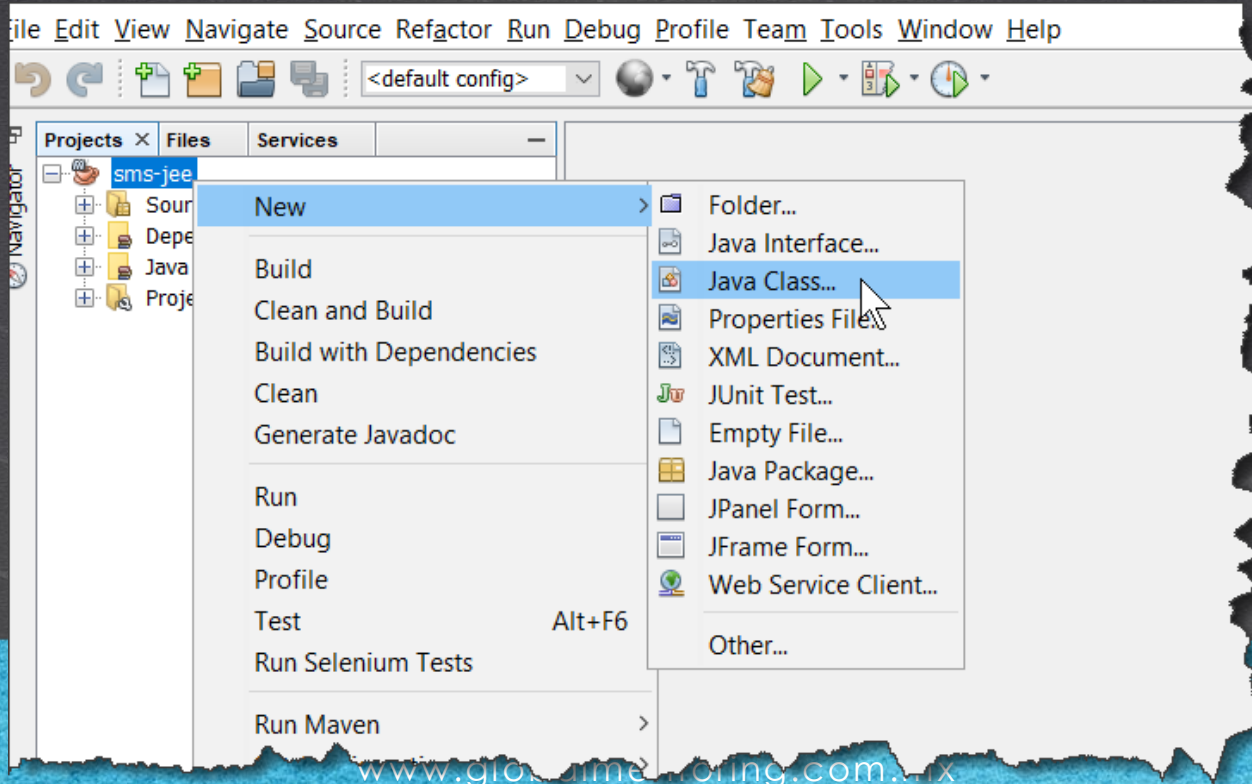
Package: sms.service

Created File: C:\Courses\JavaEE\Lesson01\sms-jee\src\main\java\sms\service\PersonServiceRemote.java

< Back   Next >   **Finish**   Cancel   Help

## 4. CREATE A NEW CLASS

We create the PersonServiceImpl.java class:





## 4. CREATE A NEW CLASS

We create the PersonServiceImpl.java class:

**New Java Class**

**Steps**

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

**Name and Location**

Class Name: PersonServiceImpl

Project: sms-jee

Location: Source Packages

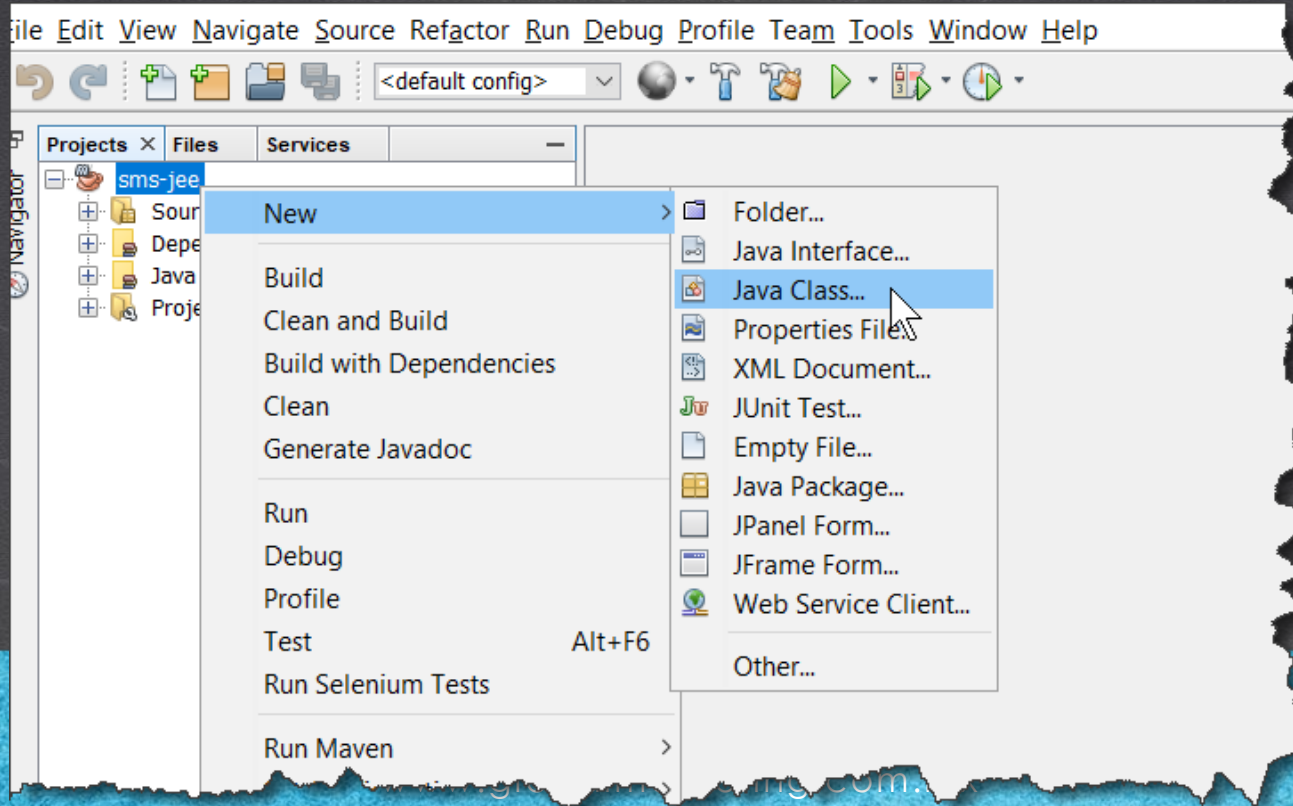
Package: sms.service

Created File: C:\Courses\JavaEE\Lesson01\sms-jee\src\main\java\sms\service\PersonServiceImpl.java

< Back   Next >   **Finish**   Cancel   Help

# 5. CREATE A NEW CLASS

We create the ClientPersonService.java class:



# 5. CREATE A NEW CLASS

We create the ClientPersonService.java class:

**New Java Class**

**Steps**

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

**Name and Location**

Class Name: ClientPersonService

Project: sms-jee

Location: Source Packages

Package: sms.client

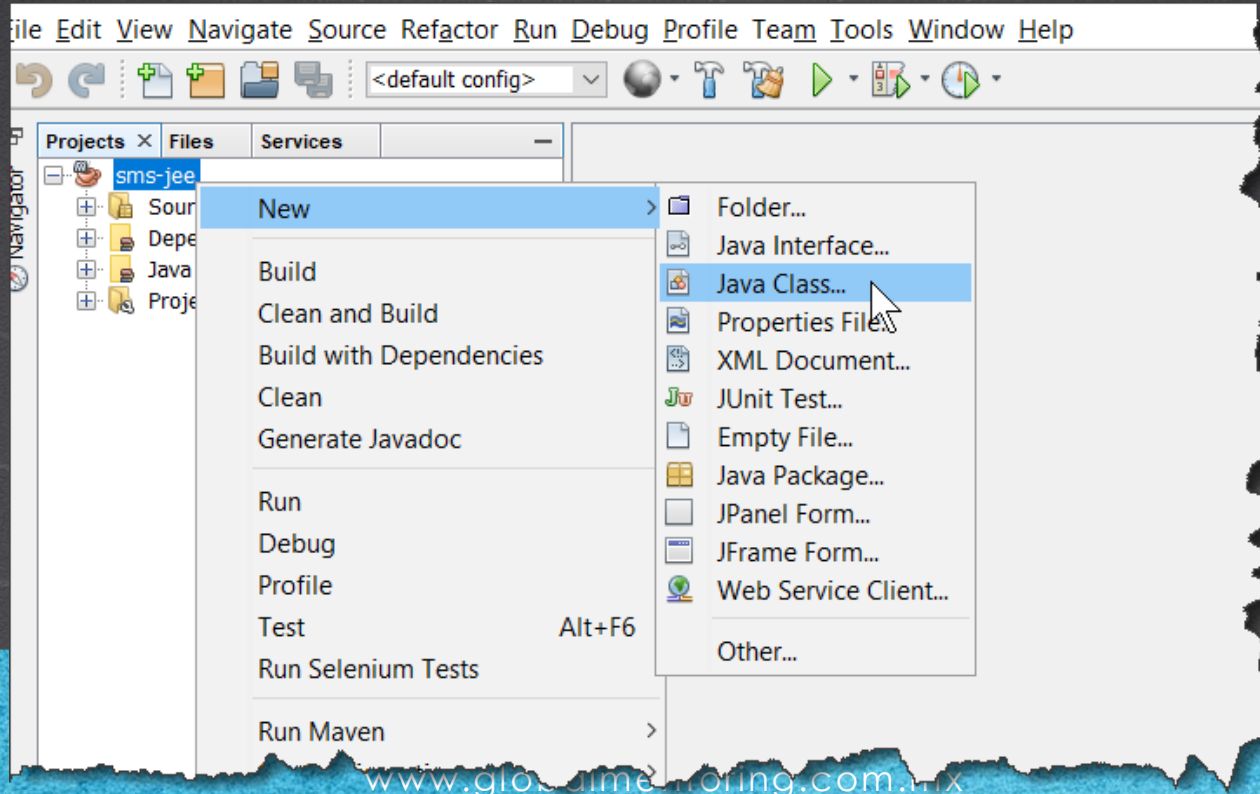
Created File: C:\Courses\JavaEE\Lesson01\sms-jee\src\main\java\sms\client\ClientPersonService.java

< Back Next > **Finish** Cancel Help



## 6. CREATE A NEW CLASS

We create the ClientPersonServiceWithIP.java class:



## 6. CREATE A NEW CLASS

We create the ClientPersonServiceWithIP.java class:

**New Java Class**

**Steps**

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

**Name and Location**

Class Name: ClientPersonServiceWithIP

Project: sms-jee

Location: Source Packages

Package: sms.client

Created File: C:\Courses\JavaEE\Lesson01\sms-jee\src\main\java\sms\client\ClientPersonServiceWithIP.java

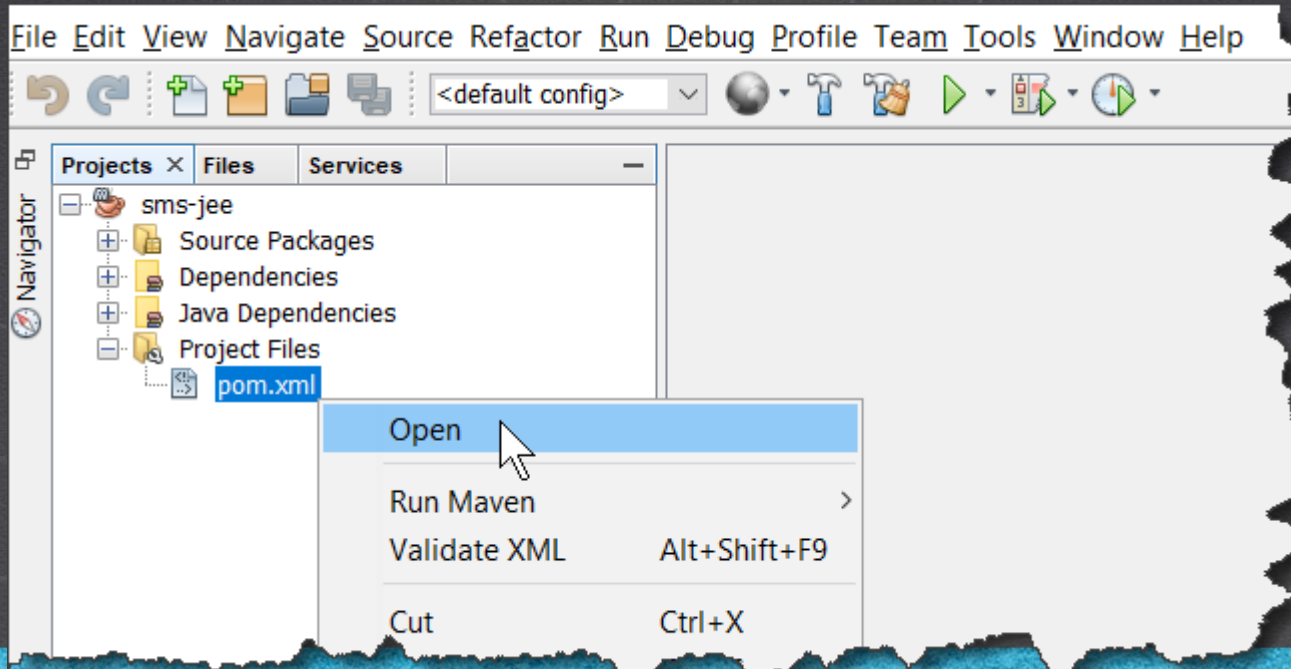
< Back Next > **Finish** Cancel Help

**JAVA EE COURSE**

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

# 7. MODIFY THE POM.XML FILE

Modify the Maven file called pom.xml:



**JAVA EE COURSE**

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



# 7. MODIFY THE CODE

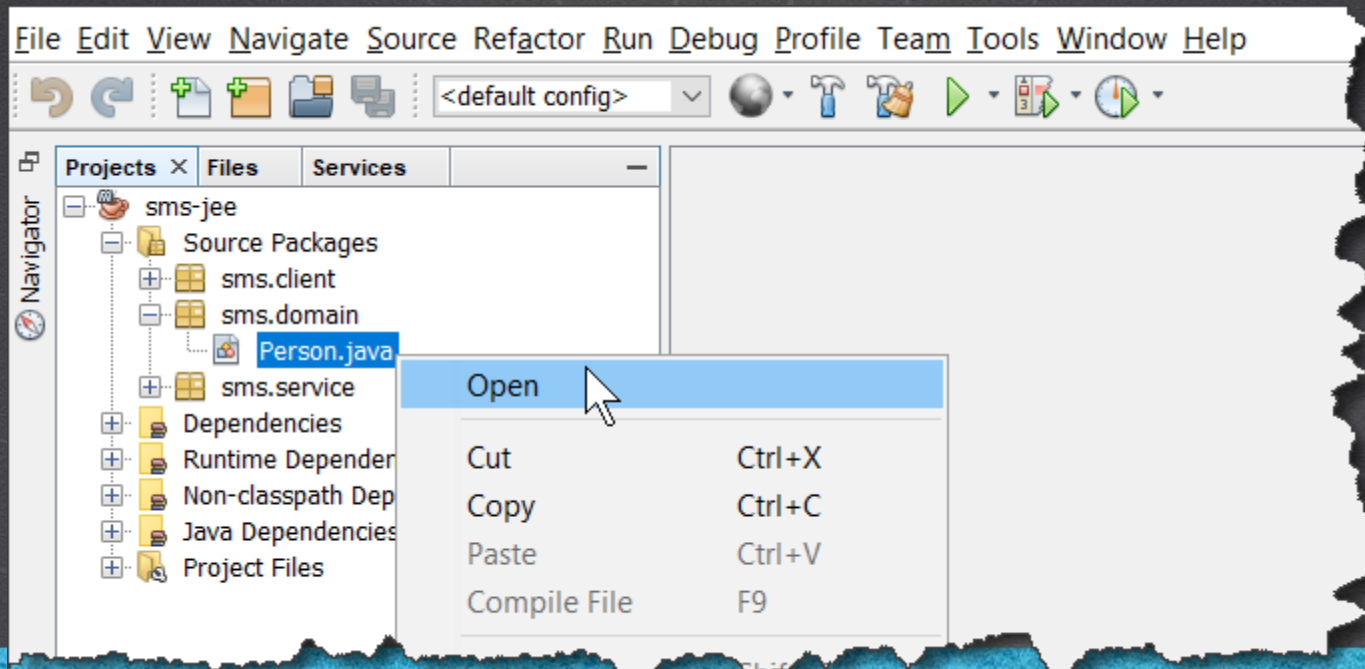
[pom.xml:](#)

Click to download

```
<?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>sms</groupId>
  <artifactId>sms-jee</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>
    <!--Library to run the EJB client-->
    <dependency>
      <groupId>org.glassfish.main.appclient</groupId>
      <artifactId>gf-client</artifactId>
      <version>5.0</version>
    </dependency>
  </dependencies>
</project>
```

# 8. MODIFY A JAVA CLASS

We modified the file Persona.java:



**JAVA EE COURSE**

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

## 8. MODIFY THE CODE

Person.java:

Click to download

```
package sms.domain;

import java.io.Serializable;

public class Person implements Serializable {

    private static final long serialVersionUID = 1L;
    private int idPerson;
    private String name;

    public Person() {
    }

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

**JAVA EE COURSE**

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



## 8. MODIFY THE CODE

Person.java:

Click to download

```
public int getIdPerson() {
    return idPerson;
}

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

public String getName() {
    return name;
}

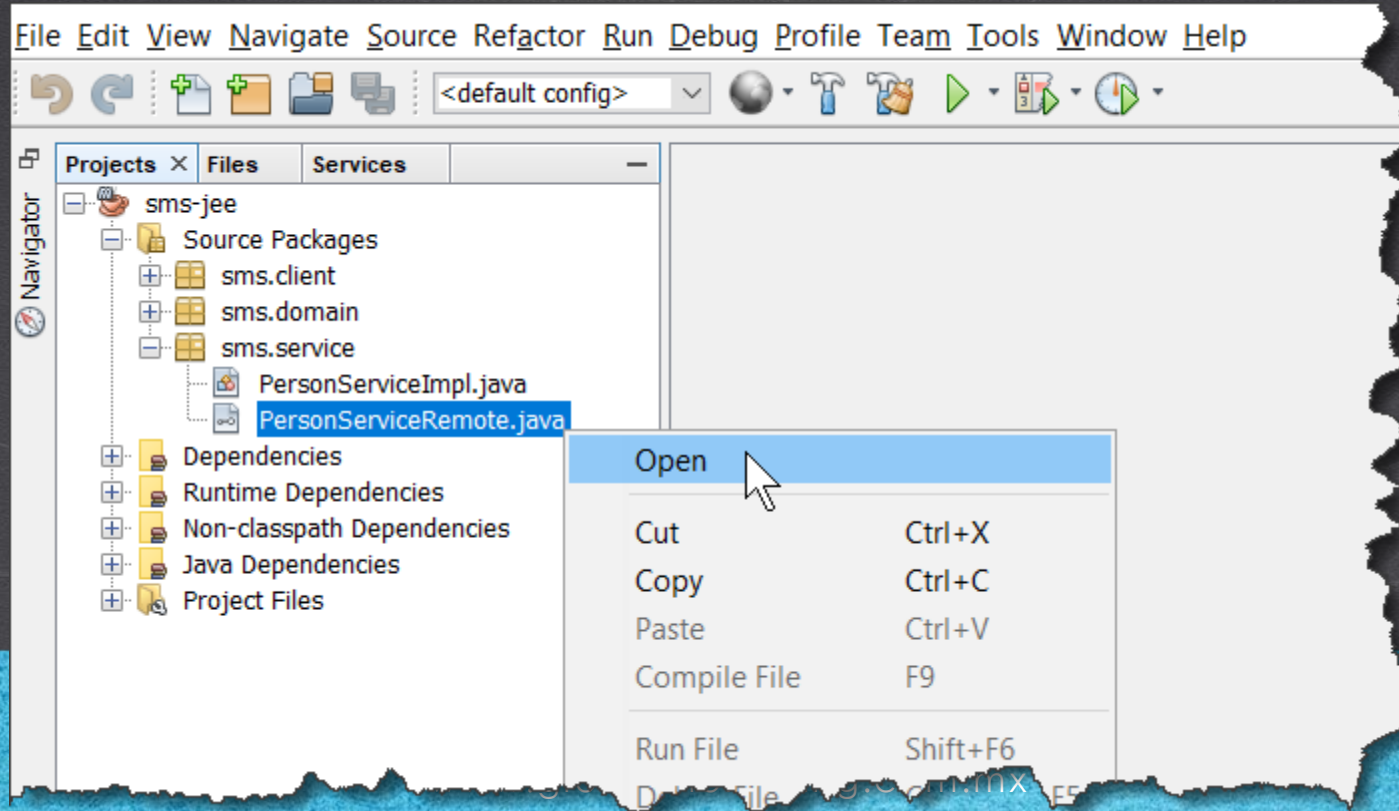
public void setName(String name) {
    this.name = name;
}

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

}
```

# 9. MODIFY THE JAVA CLASS

We modified the PersonServiceRemote.java file:



## 9. MODIFY THE CODE

### PersonServiceRemote.java:

Click to download

```
package sms.service;

import java.util.List;
import javax.ejb.Remote;
import sms.domain.Person;

@Remote
public interface PersonServiceRemote {

    public List<Person> listPeople();

    public Person findPerson(Person person);

    public void addPerson(Person person);

    public void modifyPerson(Person person);

    public void deletePerson(Person person);

}
```

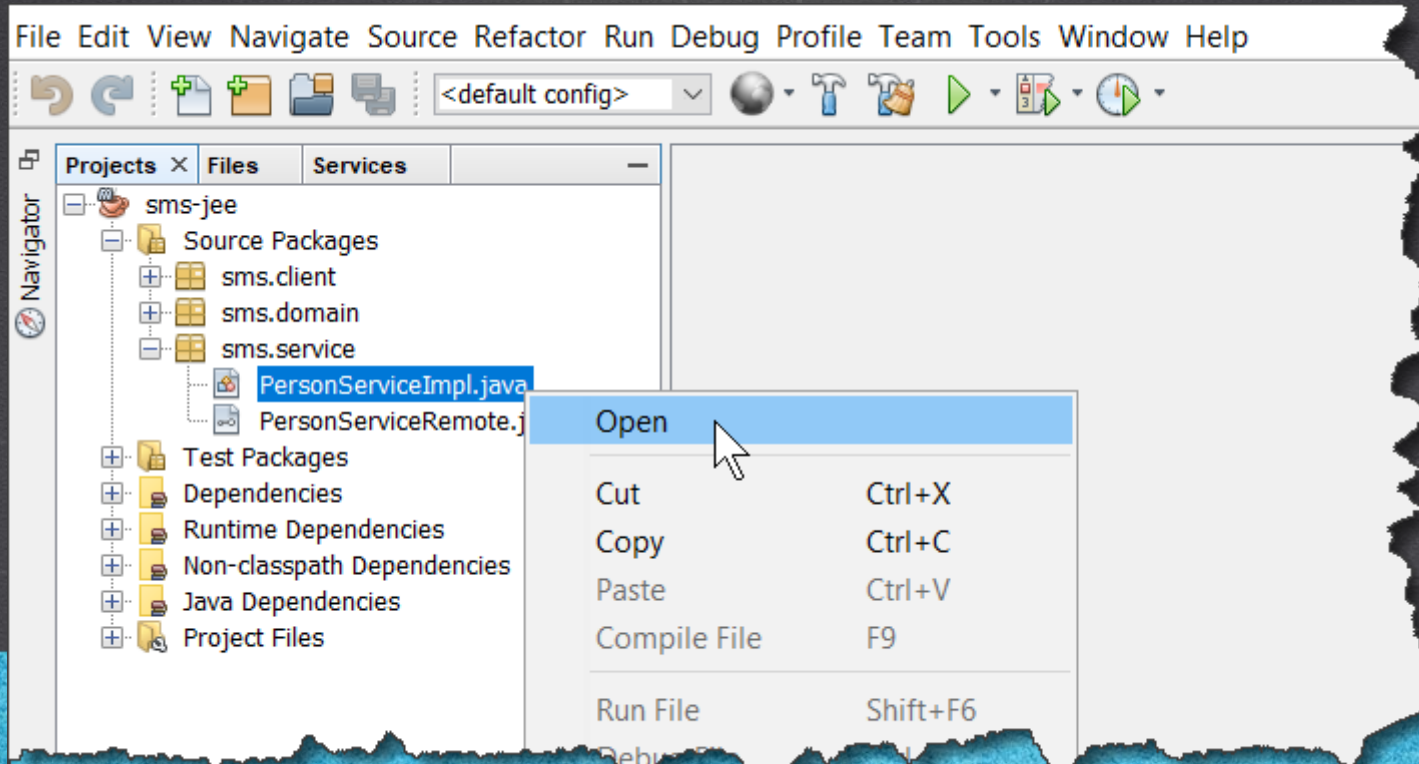
**JAVA EE COURSE**

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



# 10. MODIFY A JAVA FILE

Modify the PersonServiceImpl.java file:



# 10. MODIFY THE FILE

## PersonServiceImpl.java:

Click to download

```
package sms.service;

import java.util.ArrayList;
import java.util.List;
import javax.ejb.Stateless;
import sms.domain.Person;

@Stateless
public class PersonServiceImpl implements PersonServiceRemote {

    @Override
    public List<Person> listPeople() {
        List<Person> people = new ArrayList<>();
        people.add(new Person(1, "John"));
        people.add(new Person(2, "Samantha"));
        return people;
    }

    @Override
    public Person findPerson(Person person) {
        return null;
    }
}
```

# 10. MODIFY THE FILE

PersonServiceImpl.java:

Click to download

```
@Override
public void addPerson(Person person) {}

@Override
public void modifyPerson(Person person) {}

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

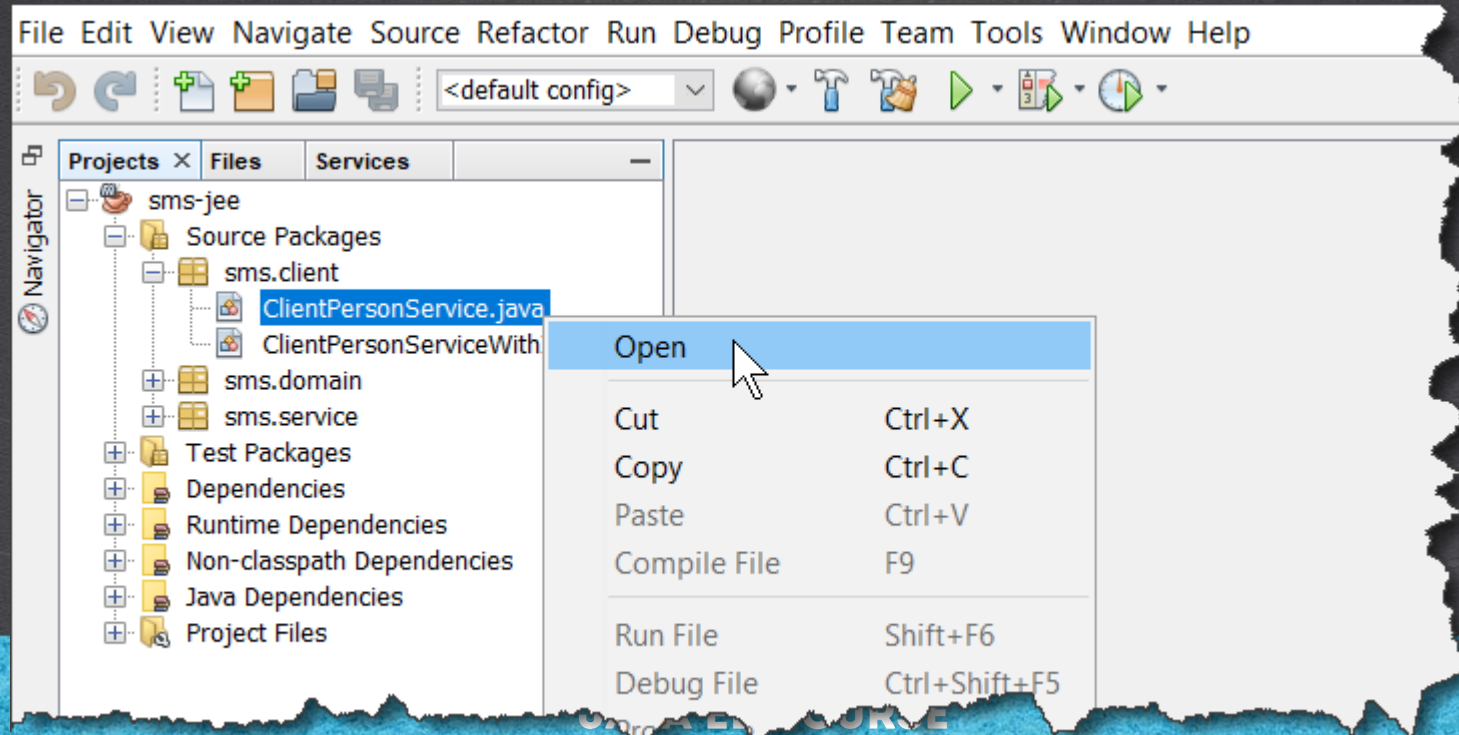
**JAVA EE COURSE**

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



# 11. MODIFY THE JAVA FILE

Modify the ClientPersonService.java file:



# 11. MODIFY THE FILE

## ClientPersonService.java:

Click to download

```
package sms.client;

import java.util.List;
import javax.naming.*;
import sms.domain.Person;
import sms.service.PersonServiceRemote;

public class ClientPersonService {

    public static void main(String[] args) {

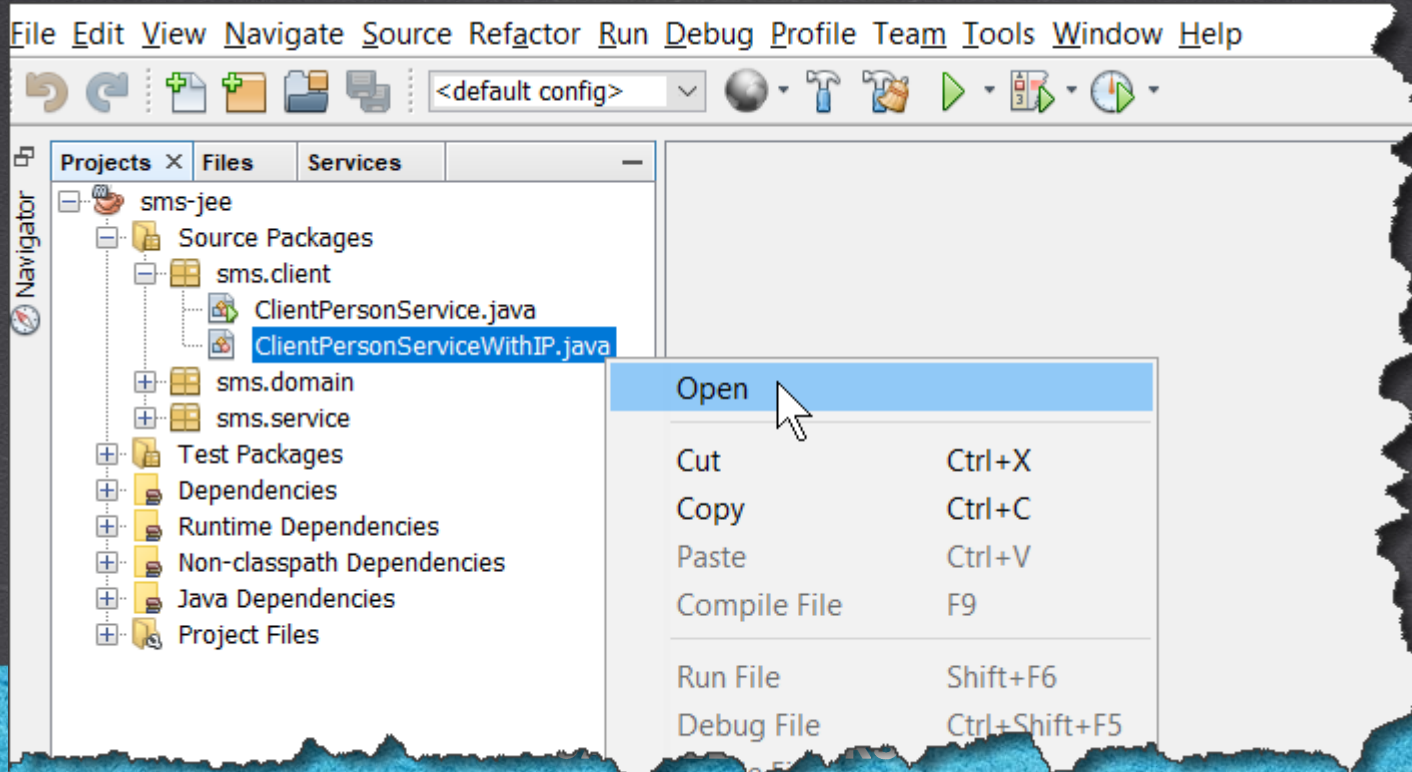
        System.out.println("Initiating EJB call from the client\n");
        try {
            Context jndi = new InitialContext();
            PersonServiceRemote personService =
                (PersonServiceRemote)
                    jndi.lookup("java:global/sms-jee/PersonServiceImpl!sms.service.PersonServiceRemote");

            List<Person> people = personService.listPeople();

            for (Person person : people) {
                System.out.println(person);
            }
            System.out.println("\nEnd call to the EJB from the client");
        } catch (NamingException e) {
            e.printStackTrace(System.out);
        }
    }
}
```

# 12. MODIFY THE JAVA FILE

Modify the ClientPersonServiceWithIP.java file:





# 12. MODIFY THE FILE

## ClientPersonServiceWithIP.java:

[Click to download](#)

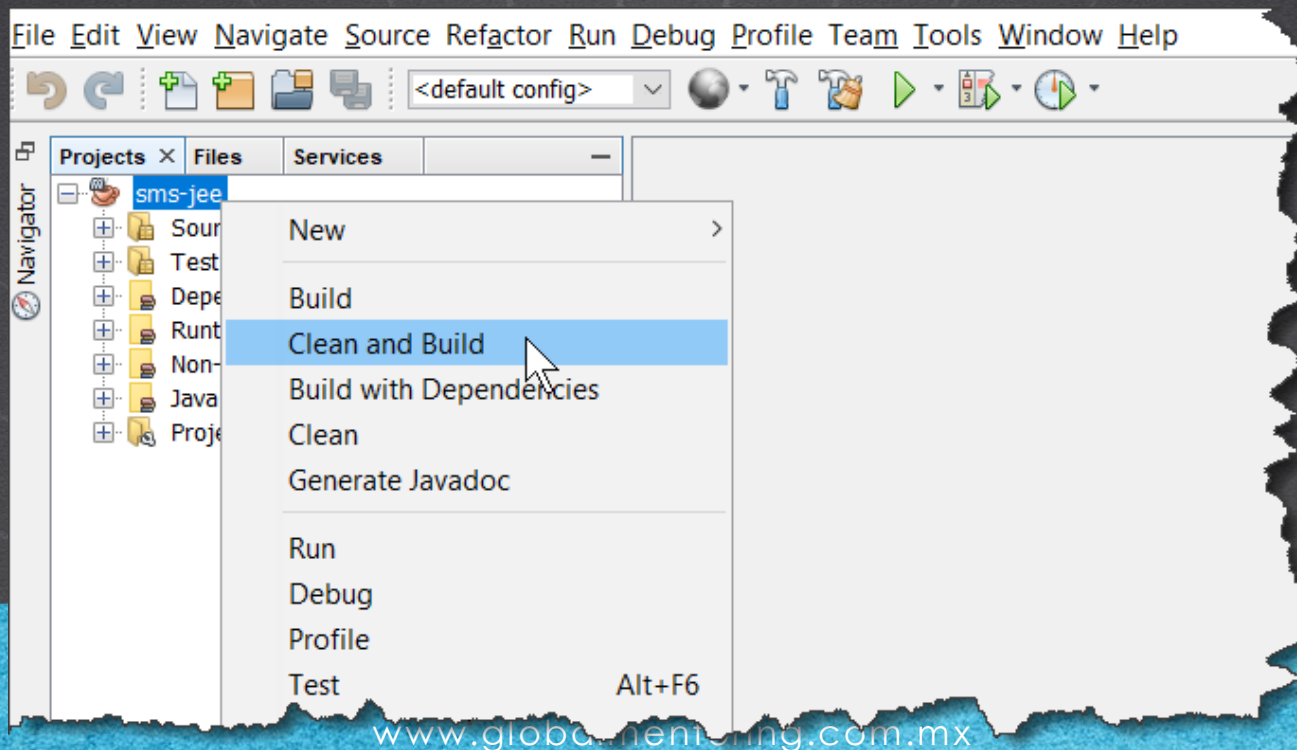
```
package sms.client;

import java.util.*;
import javax.naming.*;
import sms.domain.Person;
import sms.service.PersonServiceRemote;

public class ClientPersonServiceWithIP {
    public static void main(String[] args) {
        System.out.println("Initiating EJB call from the client with IP\n");
        try {
            Properties props = new Properties();
            props.setProperty("java.naming.factory.initial", "com.sun.enterprise.naming.SerialInitContextFactory");
            props.setProperty("java.naming.factory.url.pkgs", "com.sun.enterprise.naming");
            props.setProperty("java.naming.factory.state", "com.sun.corba.ee.impl.presentation.rmi.JNDIStateFactoryImpl");
            // optional Default localhost. Here the IP of the server where Glassfish is changed
            props.setProperty("org.omg.CORBA.ORBInitialHost", "127.0.0.1");
            // optional Port by Default 3700. You only need to change if the port is not 3700.
            //props.setProperty("org.omg.CORBA.ORBInitialPort", "3700");
            Context jndi = new InitialContext(props);
            PersonServiceRemote personService =
                (PersonServiceRemote) jndi.lookup("java:global/sms-jee/PersonServiceImpl!sms.service.PersonServiceRemote");
            List<Person> persons = personService.listPeople();
            for (Person person : persons) {
                System.out.println(person);
            }
            System.out.println("\nEnd call to the EJB from the client with IP");
        } catch (NamingException e) {
            e.printStackTrace(System.out);
        }
    }
}
```

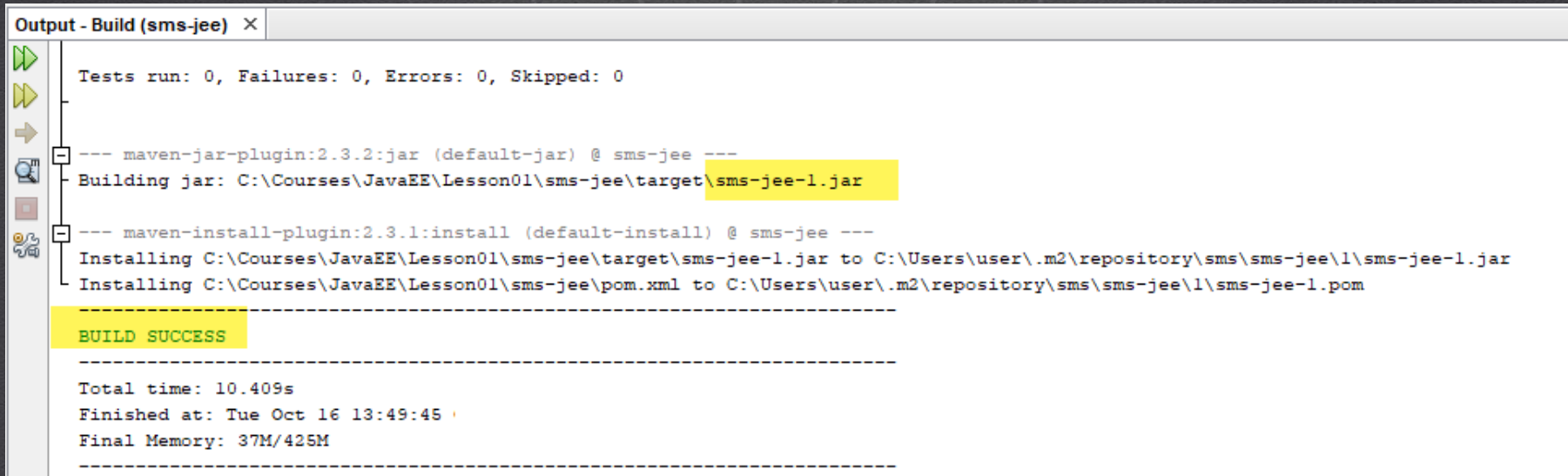
# 13. GENERATE THE .JAR

We are going to deploy the EJB in Glassfish. For this we generate the .jar of the project, we make clean & build:



# 13. GENERATE THE .JAR

We see that the file sms-jee.jar was generated.



```
Output - Build (sms-jee) X
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
--- maven-jar-plugin:2.3.2:jar (default-jar) @ sms-jee ---
Building jar: C:\Courses\JavaEE\Lesson01\sms-jee\target\sms-jee-1.jar
--- maven-install-plugin:2.3.1:install (default-install) @ sms-jee ---
Installing C:\Courses\JavaEE\Lesson01\sms-jee\target\sms-jee-1.jar to C:\Users\user\.m2\repository\sms\sms-jee\1\sms-jee-1.jar
Installing C:\Courses\JavaEE\Lesson01\sms-jee\pom.xml to C:\Users\user\.m2\repository\sms\sms-jee\1\sms-jee-1.pom
-----
BUILD SUCCESS
-----
Total time: 10.409s
Finished at: Tue Oct 16 13:49:45
Final Memory: 37M/425M
-----
```

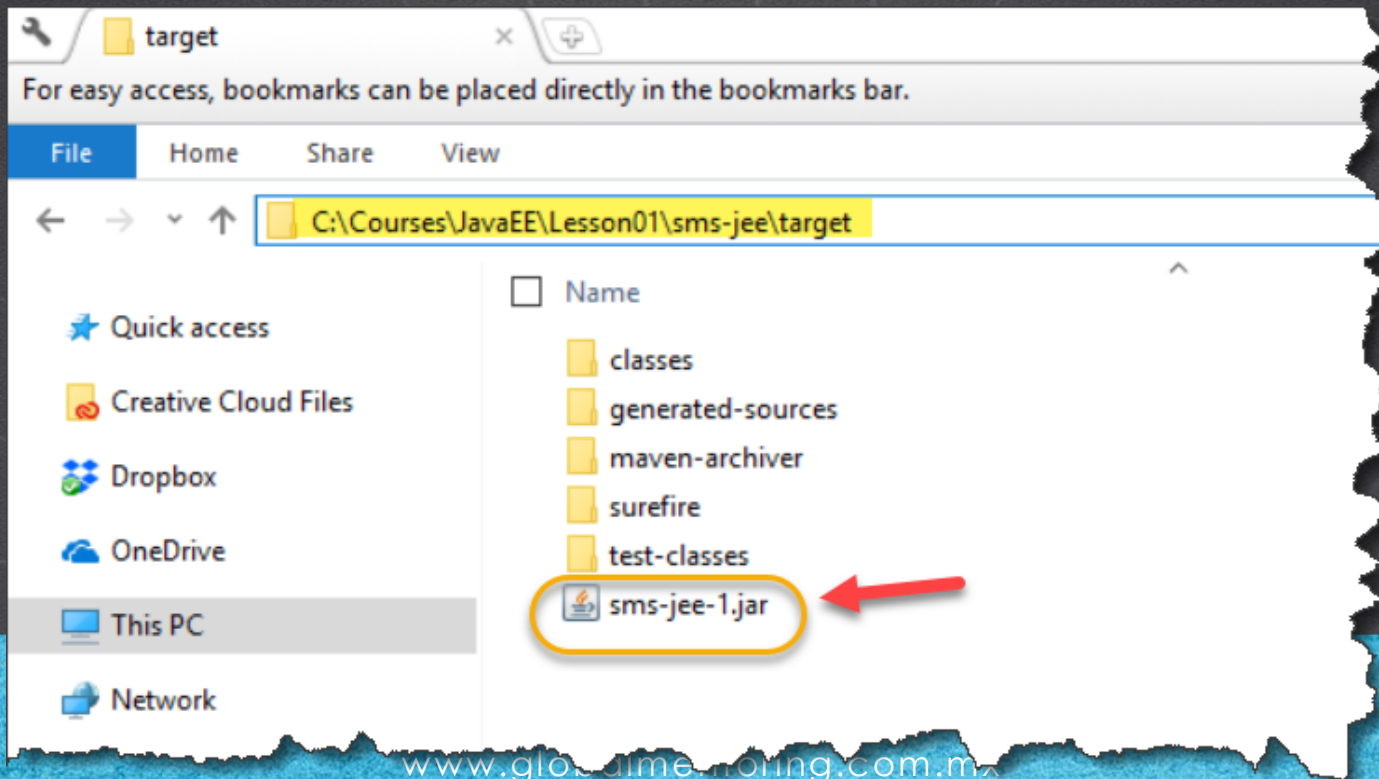
**JAVA EE COURSE**

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



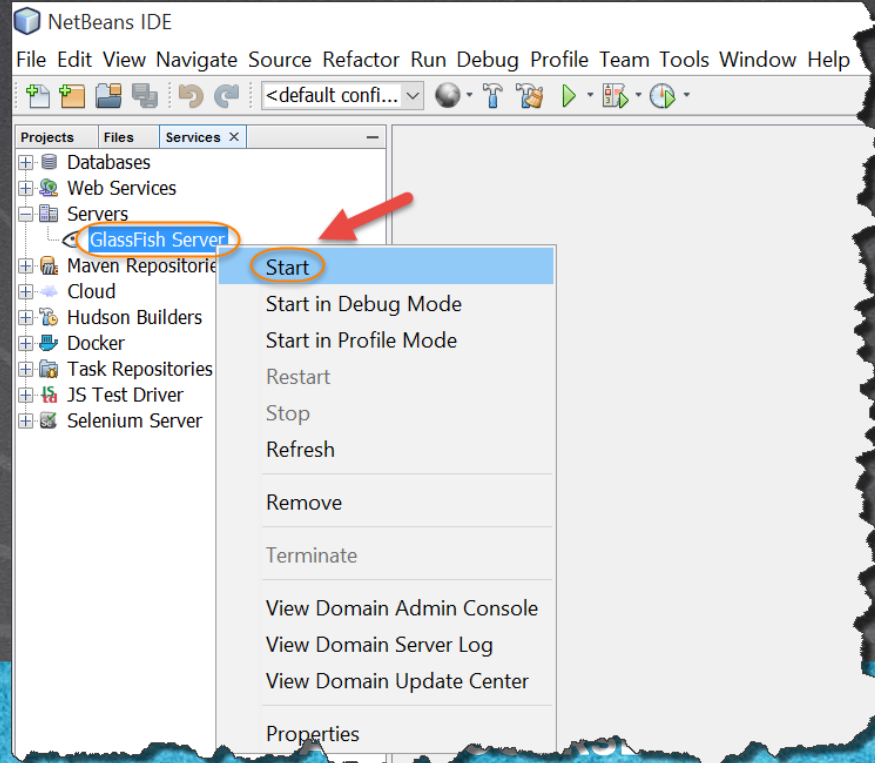
# 13. GENERATE THE .JAR

We see that the .jar was generated, we take the path and use it later:



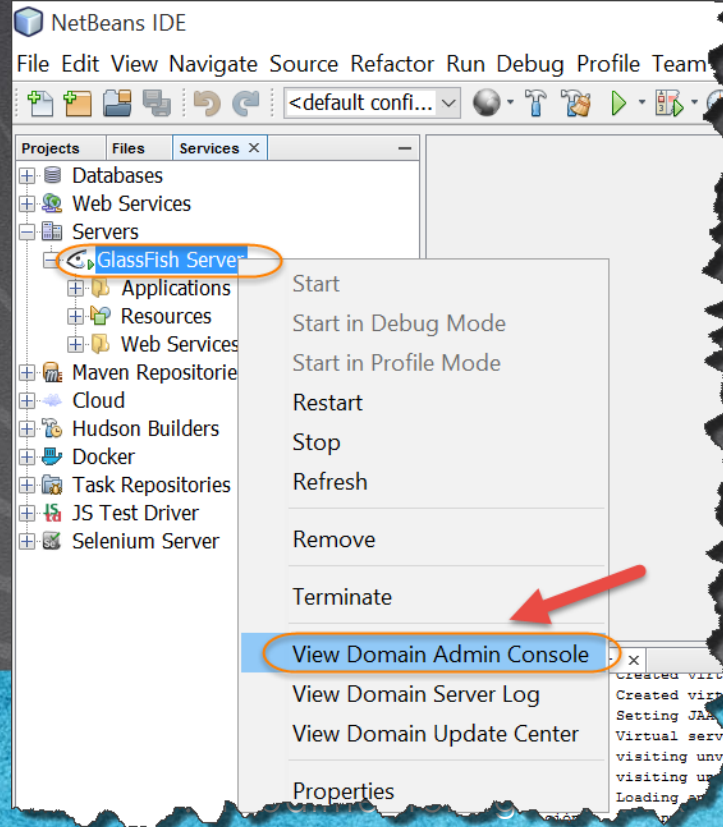
# 14. START UP GLASSFISH

Start Glassfish to deploy the EJB:



# 14. START UP GLASSFISH

We enter the Glassfish administration console:





## We deploy the .jar application in Glassfish:



# 15. DEPLOY THE APPLICATION

We deploy the .jar application in Glassfish:

The screenshot shows the GlassFish Server Open Source Edition web console. The top navigation bar includes 'Home' and 'About...' buttons. Below this, the user information is displayed: 'User: admin | Domain: domain1 | Server: localhost'. The main title is 'GlassFish™ Server Open Source Edition'.

On the left side, there is a 'Common Tasks' sidebar with a tree view containing: Domain, server (Admin Server), Clusters, Standalone Instances, Nodes, Applications (highlighted), Lifecycle Modules, Monitoring Data, Resources, and Concurrent Resources.

The main content area is titled 'Deploy Applications or Modules'. It contains the following text: 'Specify the location of the application or module to deploy. An application can be in a packaged file or specified as a directory.'

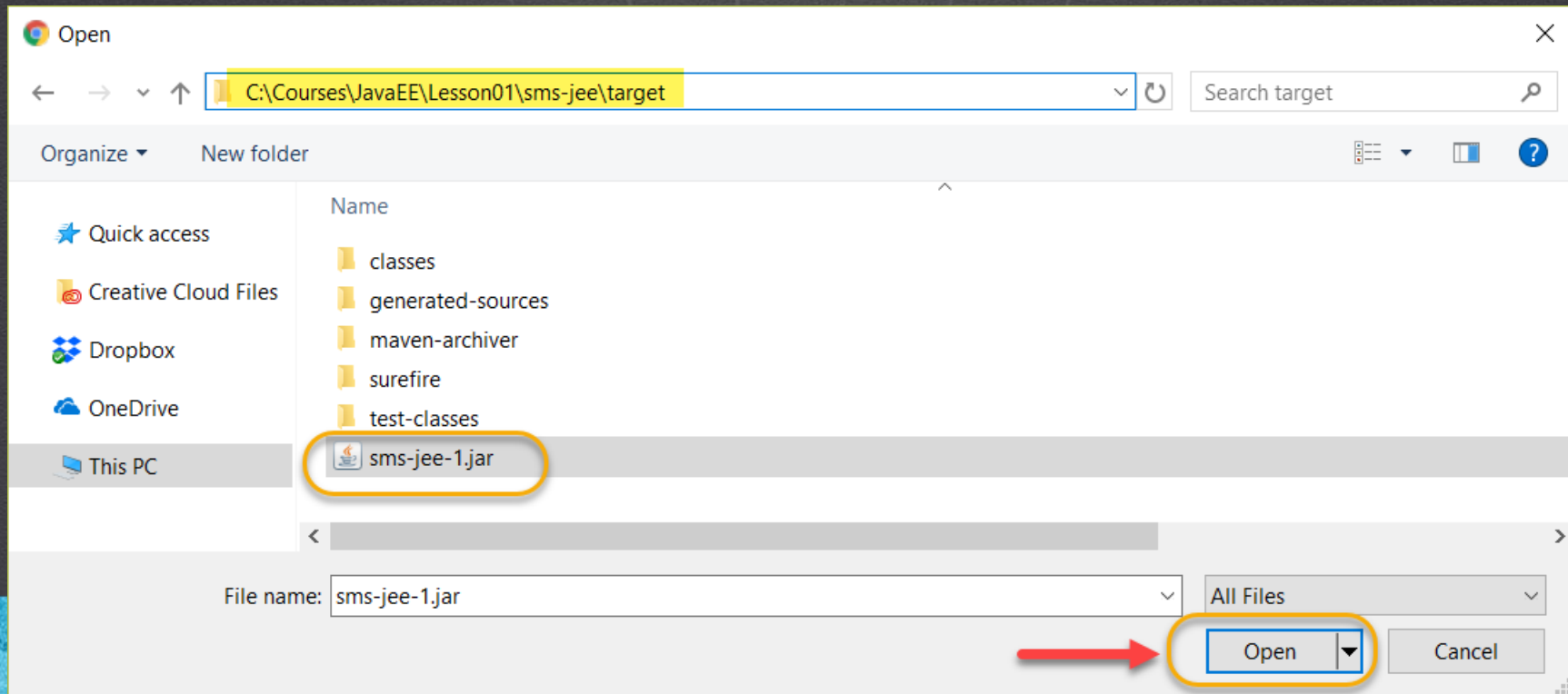
There are two radio button options for the 'Location':

- ☒ **Packaged File to Be Uploaded to the Server**  
This option is selected. Below it, there is a 'Choose File' button (circled in yellow) and the text 'No file chosen'. A red arrow points to the 'Choose File' button.
- ☐ **Local Packaged File or Directory That Is Accessible from GlassFish Server**  
Below this option, there is a text input field, a 'Browse Files...' button, and a 'Browse Folders...' button.

At the bottom, there is a 'Type: \*' label followed by a dropdown menu.

# 15. DEPLOY THE APPLICATION

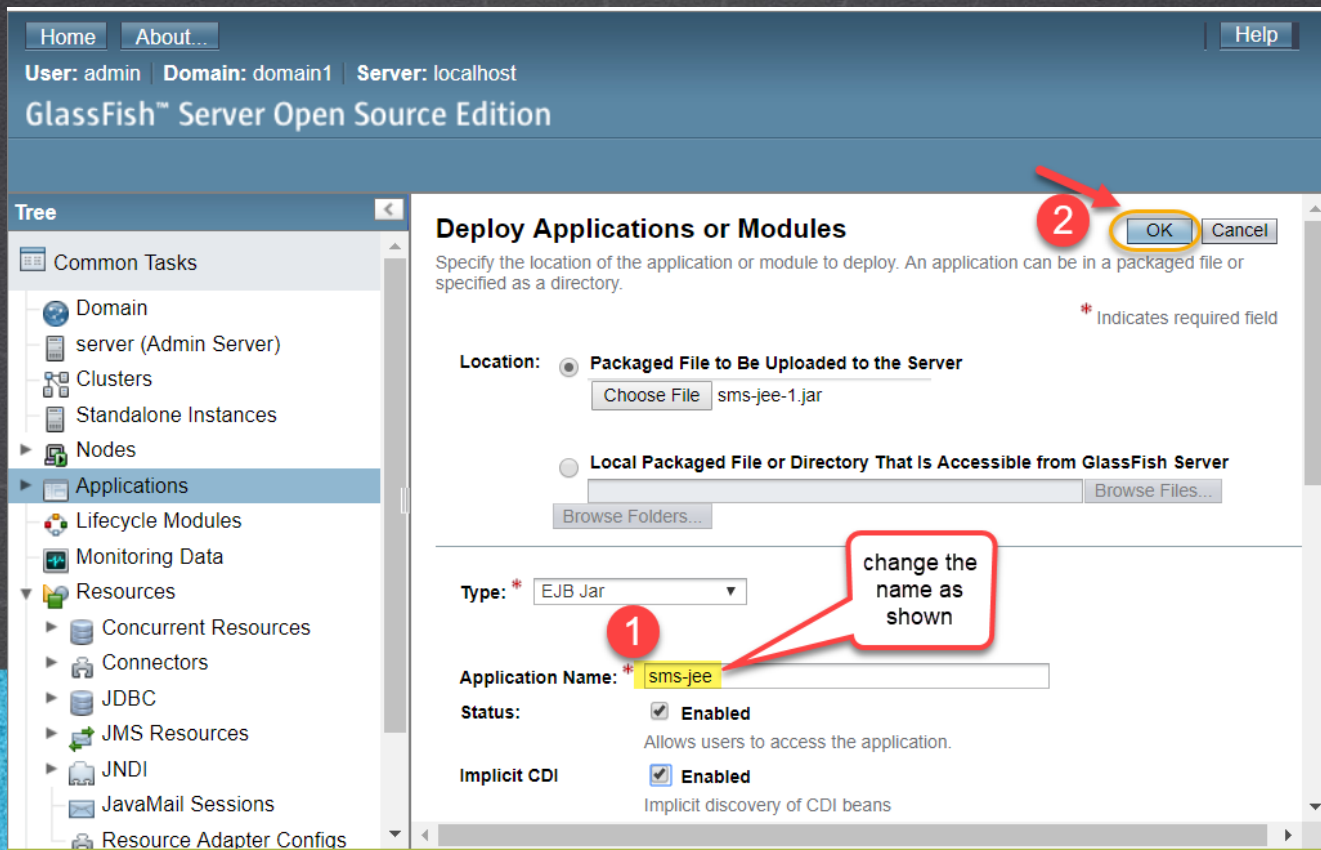
We deployed the .jar application in Glassfish:





# 15. DEPLOY THE APPLICATION

We deployed the .jar application in Glassfish. Rename the app:



# 15. DEPLOY THE APPLICATION

We deployed the .jar application in Glassfish:

The screenshot displays the GlassFish Server Open Source Edition administration console. The top navigation bar includes links for 'Home', 'About...', and 'Help'. Below this, the user information is shown: 'User: admin | Domain: domain1 | Server: localhost'. The main title is 'GlassFish™ Server Open Source Edition'.

On the left, a 'Tree' view shows the navigation structure. The 'Applications' item is selected and highlighted. The tree includes 'Common Tasks', 'Domain' (with 'server (Admin Server)'), 'Clusters', 'Standalone Instances', 'Nodes', 'Applications', 'Lifecycle Modules', 'Monitoring Data', and 'Resources' (with sub-items 'Concurrent Resources', 'Connectors', and 'JDBC').

The main content area is titled 'Applications'. It contains a descriptive paragraph: 'Applications can be enterprise or web applications, or various kinds of modules. Restart an application or module by clicking on the reload link, this action will apply only to the targets that the application or module is enabled on.'

Below the text is a section titled 'Deployed Applications (2)'. It features a toolbar with icons for selection, a 'Deploy...' button, and buttons for 'Undeploy', 'Enable', and 'Disable'. There is also a 'Filter:' dropdown menu.

The core of this section is a table with the following columns: 'Select', 'Name', 'Deployment Order', 'Enabled', 'Engines', and 'Action'. Two applications are listed:

Select	Name	Deployment Order	Enabled	Engines	Action
<input type="checkbox"/>	helloworld-ejb	100	✓	ejb	Redeploy   Reload
<input type="checkbox"/>	sms-jee	100	✓	ejb	Redeploy   Reload

The second row, corresponding to the 'sms-jee' application, is highlighted with a yellow border.

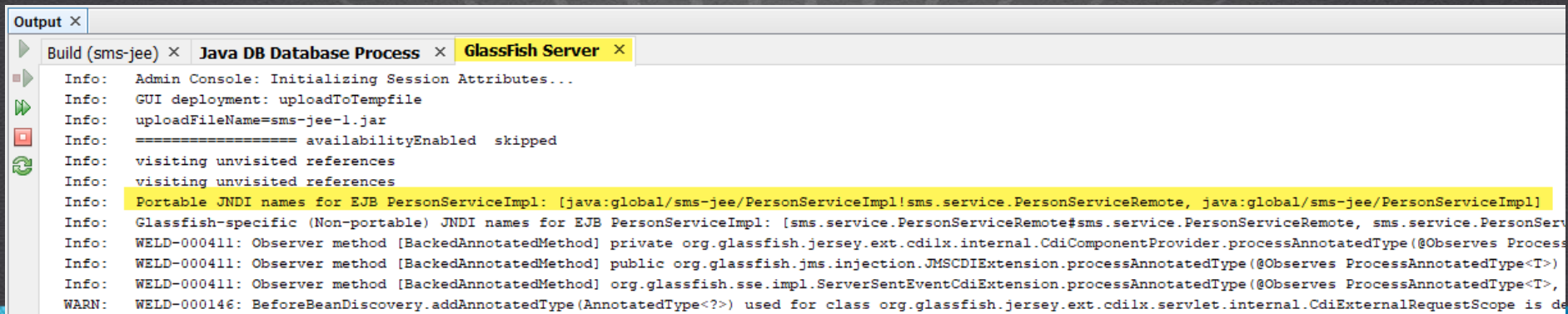
# 15. DEPLOY THE APPLICATION

We reviewed the Glassfish log from the IDE, and we observed the names with which we can call via JNDI to the EJB:

Portable JNDI names for EJB PersonServiceImpl:

java:global/sms-jee/PersonServiceImpl!sms.service.PersonServiceRemote

java:global/sms-jee/PersonServiceImpl]



```
Output x
Build (sms-jee) x Java DB Database Process x GlassFish Server x
Info: Admin Console: Initializing Session Attributes...
Info: GUI deployment: uploadToTempfile
Info: uploadFileName=sms-jee-1.jar
Info: ===== availabilityEnabled skipped
Info: visiting unvisited references
Info: visiting unvisited references
Info: Portable JNDI names for EJB PersonServiceImpl: [java:global/sms-jee/PersonServiceImpl!sms.service.PersonServiceRemote, java:global/sms-jee/PersonServiceImpl]
Info: Glassfish-specific (Non-portable) JNDI names for EJB PersonServiceImpl: [sms.service.PersonServiceRemote#sms.service.PersonServiceRemote, sms.service.PersonServiceRemote]
Info: WELD-000411: Observer method [BackedAnnotatedMethod] private org.glassfish.jersey.ext.cdilx.internal.CdiComponentProvider.processAnnotatedType(@Observes ProcessAnnotatedType<T>) void
Info: WELD-000411: Observer method [BackedAnnotatedMethod] public org.glassfish.jms.injection.JMSCDIExtension.processAnnotatedType(@Observes ProcessAnnotatedType<T>) void
Info: WELD-000411: Observer method [BackedAnnotatedMethod] org.glassfish.sse.impl.ServerSentEventCdiExtension.processAnnotatedType(@Observes ProcessAnnotatedType<T>) void
WARN: WELD-000146: BeforeBeanDiscovery.addAnnotatedType(AnnotatedType<?>) used for class org.glassfish.jersey.ext.cdilx.servlet.internal.CdiExternalRequestScope is deprecated
```

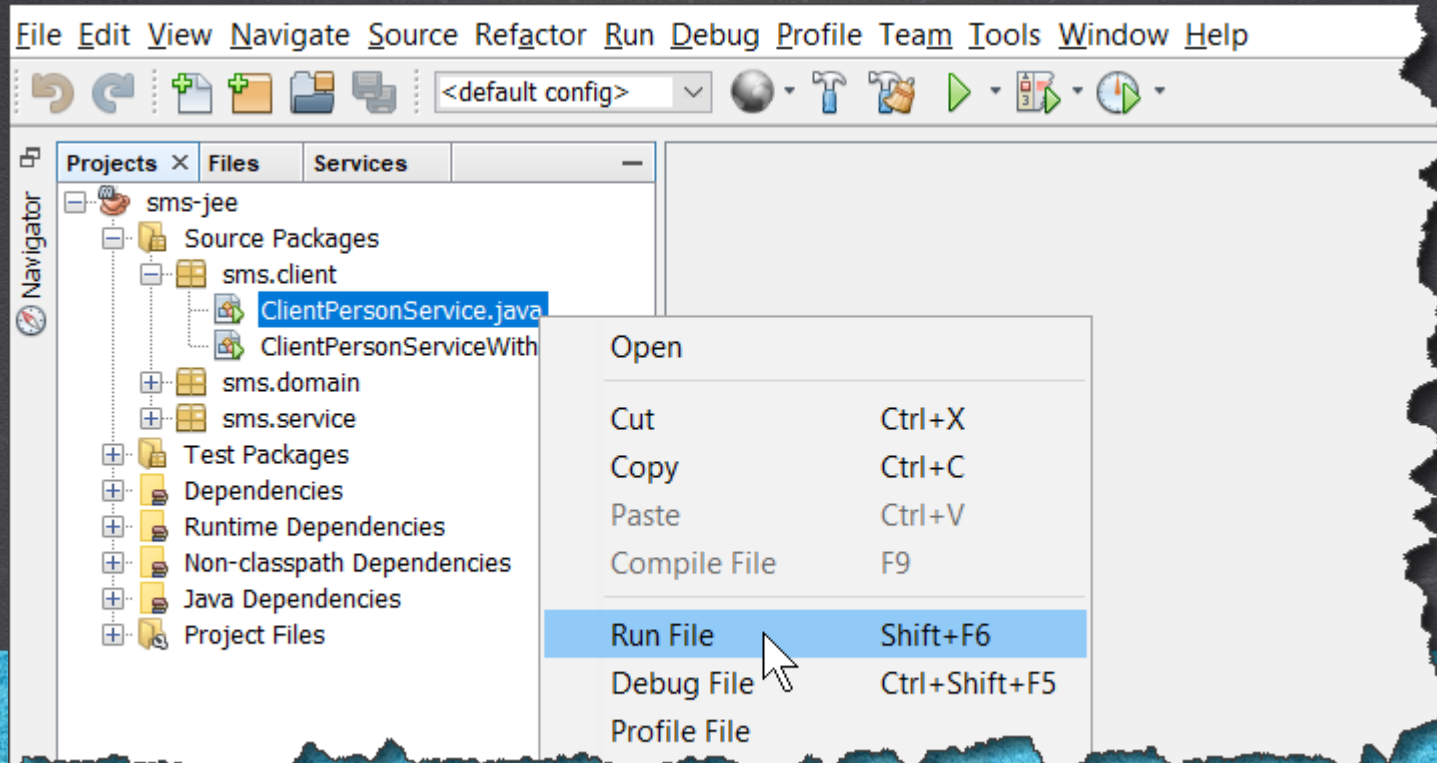
**JAVA EE COURSE**

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



# 16. EXECUTE THE CLASS

We execute the ClientPersonService.java class:



# 16. EXECUTE THE CLASS

We execute the class. We see the result of executing the call to the EJB:



```
Output X
Java DB Database Process x GlassFish Server x Run (ClientPersonService) x
--- exec-maven-plugin:1.2.1:exec (default-cli) @ sms-jee ---
Initiating EJB call from the client

Person{idPerson=1, name=John}
Person{idPerson=2, name=Samantha}

End call to the EJB from the client

-----
BUILD SUCCESS
-----

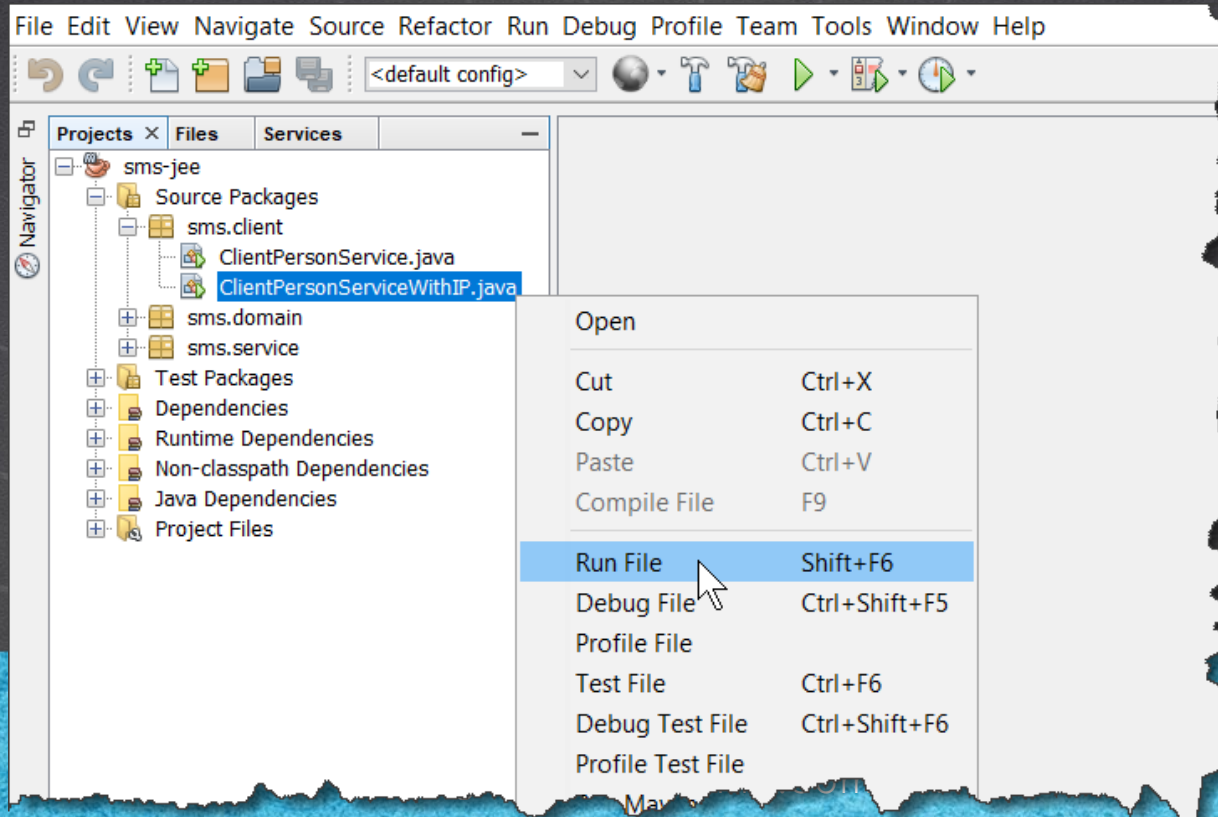
Total time: 5.197s
Finished at: Tue Oct 16 14:31:35
Final Memory: 17M/489M
-----
```

**JAVA EE COURSE**

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

# 17. EXECUTE THE CLASS

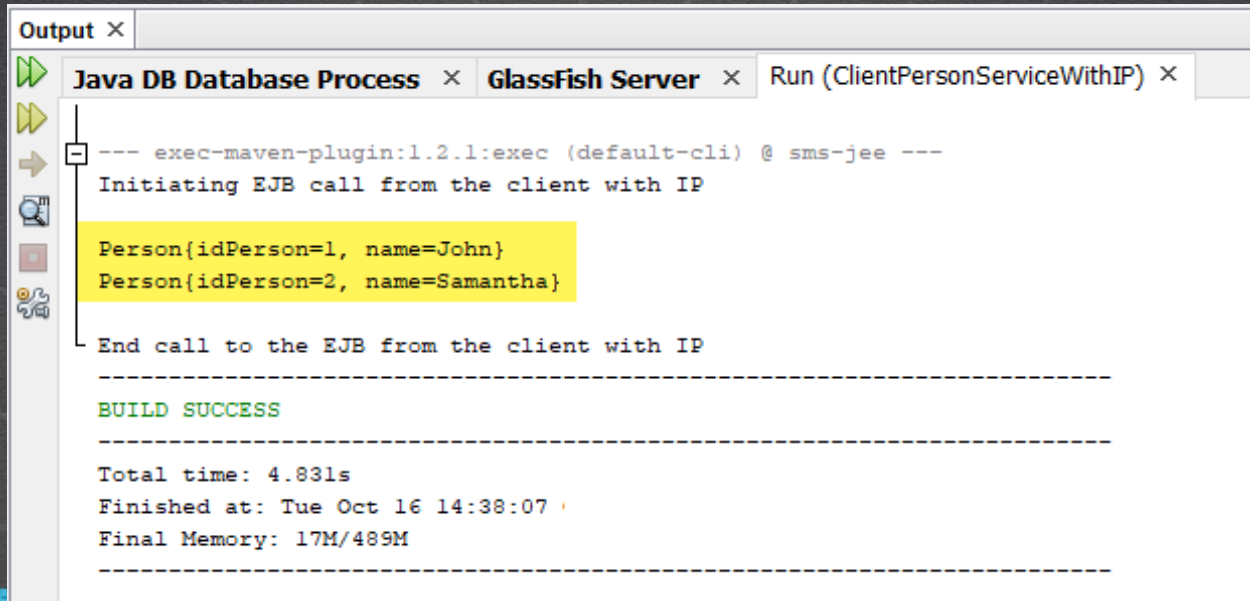
We execute the ClientPersonServiceWithIP.java class:





# 17. EXECUTE THE CLASS

We execute the class. We observe the result of executing the call to the EJB remotely:



```
Output x
Java DB Database Process x GlassFish Server x Run (ClientPersonServiceWithIP) x

--- exec-maven-plugin:1.2.1:exec (default-cli) @ sms-jee ---
Initiating EJB call from the client with IP

Person{idPerson=1, name=John}
Person{idPerson=2, name=Samantha}

End call to the EJB from the client with IP

-----
BUILD SUCCESS
-----

Total time: 4.831s
Finished at: Tue Oct 16 14:38:07
Final Memory: 17M/489M
-----
```

**JAVA EE COURSE**

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

# OBSERVATIONS IN CASE OF PROBLEMS

If for some reason the execution of the project does not work, we recommend you to carry out the following actions :

- 1) The Glassfish server must be running in order to access the remote EJB.
- 2) Remember that any change in the EJB or the interface has to redeploy the .jar file in Glassfish server, since the changes are not published automatically (Select the application and undeploy and redo deploy in Glassfish)
- 3) Check that in step 15 the application was renamed at the time of deploying it in Glassfish (sga-jee).
- 4) If none of this works, try loading the resolved project, which is 100% functional.



# EXERCISE CONCLUSION

- With this exercise we have implemented the call to a remote EJB from a Java client.
- We had to deploy the application to the Glassfish server, and once the application was deployed we were able to execute the client, which via JNDI (Java Naming & Directory Interface) was able to make the call to the EJB remotely, using both a local client and a client which can be outside the server and specifying the IP of the Glassfish server.



**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)