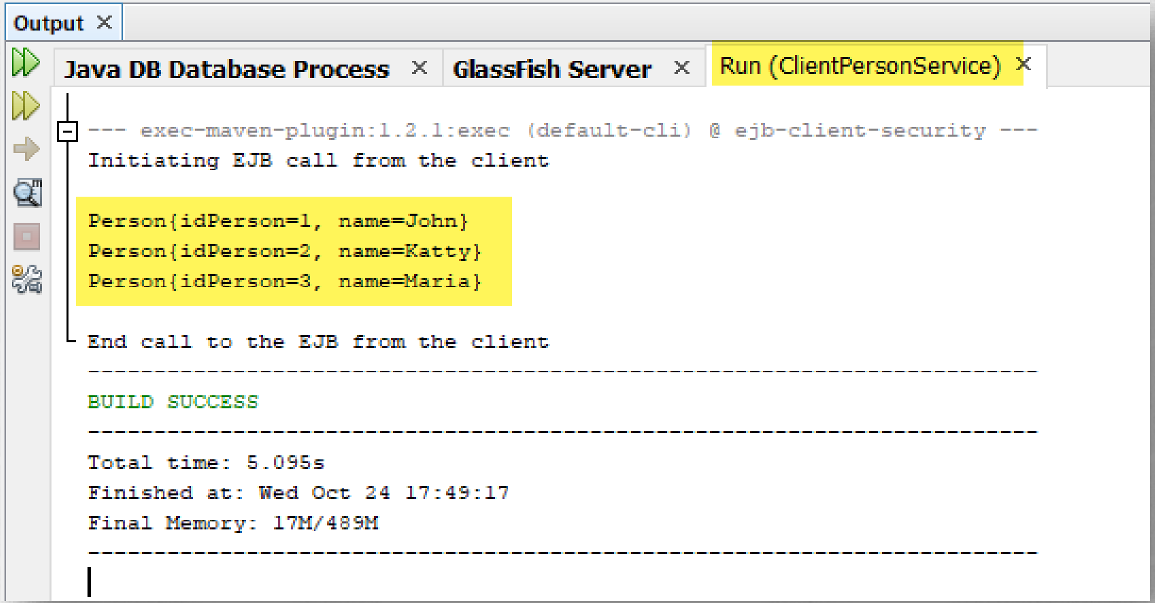
ACCESSING FROM THE EJB CLIENT

By the expert: Eng. Ubaldo Acosta

EXERCISE OBJECTIVE

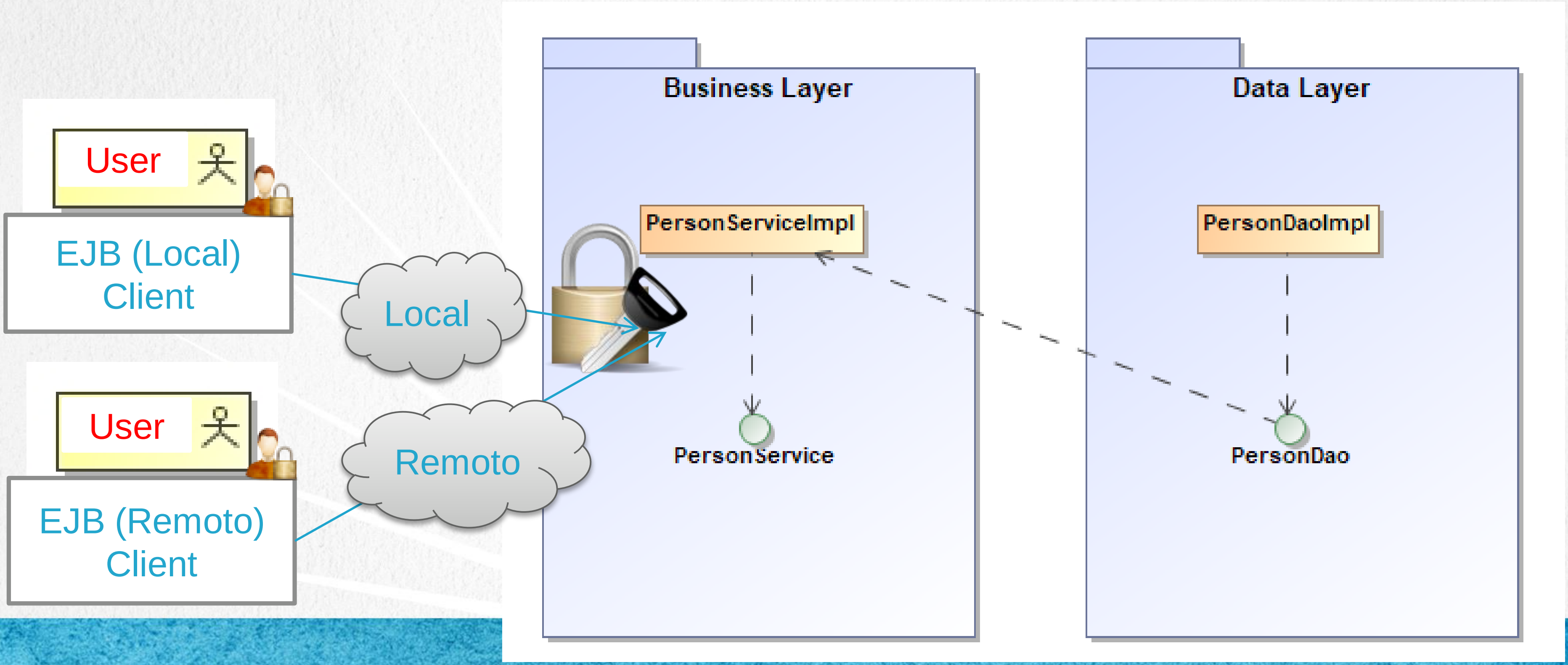
•The objective of the exercise is to access from the EJB client to the EJB of the sg-jee-web project. The result is shown below (data may vary).



```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ ejb-client-security ---
Initiating EJB call from the client

Person{idPerson=1, name=John}
Person{idPerson=2, name=Katty}
Person{idPerson=3, name=Maria}

End call to the EJB from the client
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 5.095s
Finished at: Wed Oct 24 17:49:17
Final Memory: 17M/489M
------------------------------------------------------------------------
```

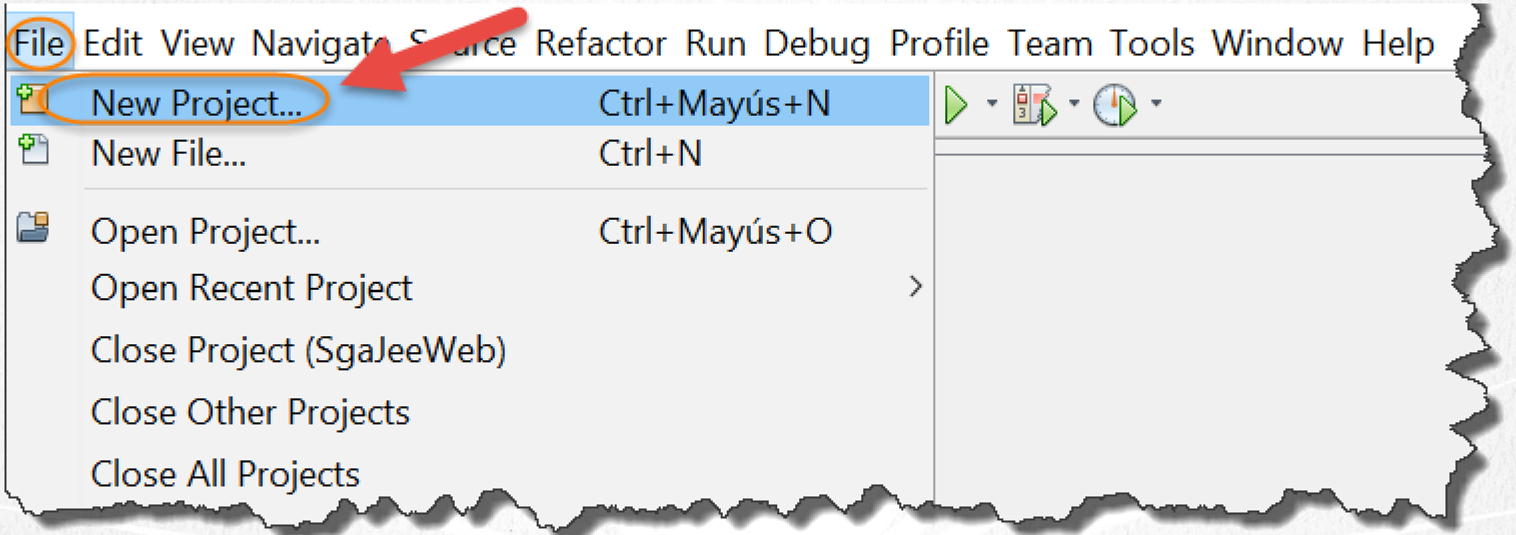
SMS ARCHITECTURE

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



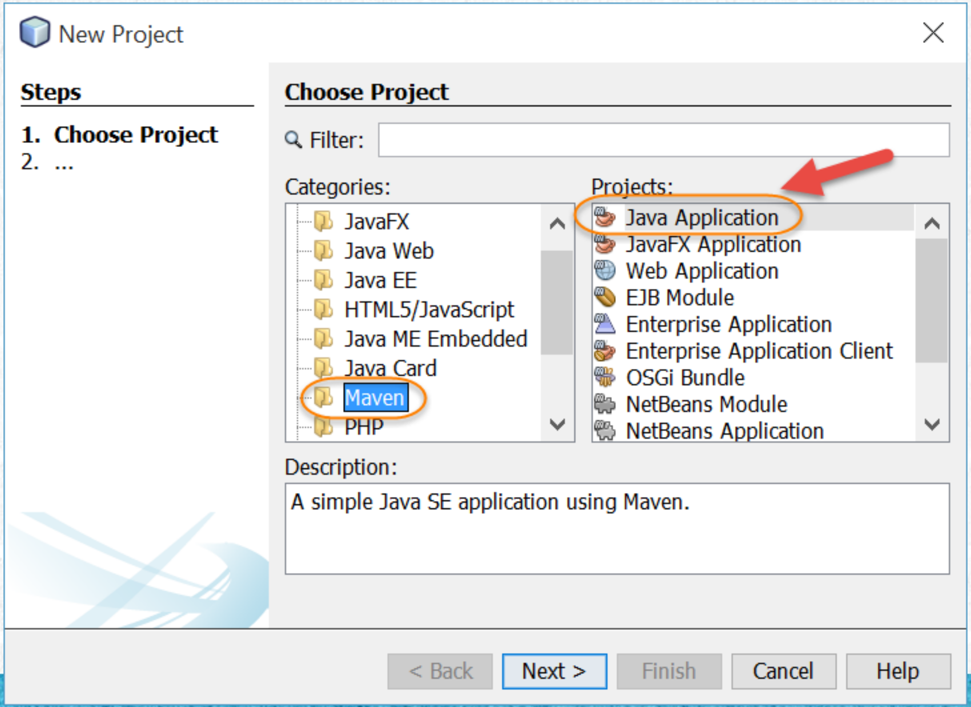
1. CREATE A NEW PROJECT

We create the following project: EjbClientSecurity :


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
New Project... Ctrl+Mayús+N
New File... Ctrl+N
Open Project... Ctrl+Mayús+O
Open Recent Project
Close Project (SgaJeeWeb)
Close Other Projects
Close All Projects

1. CREATE A NEW PROJECT

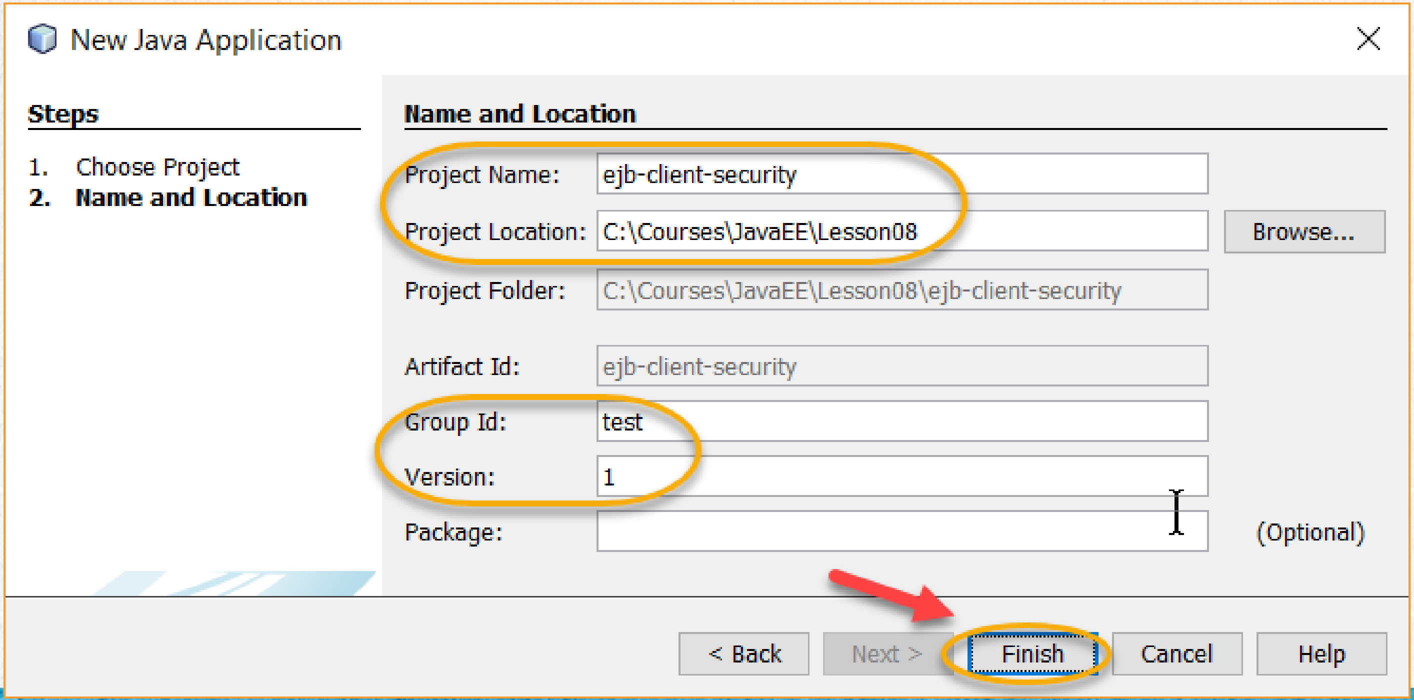
We create the following project: EjbClientSecurity :



A simple Java SE application using Maven.

1. CREATE A NEW PROJECT

We create the following project: EjbClientSecurity :


New Java Application

Steps

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

Name and Location

Project Name:	ejb-client-security
Project Location:	C:\Courses\JavaEE\Lesson08
Project Folder:	C:\Courses\JavaEE\Lesson08\ejb-client-security
Artifact Id:	ejb-client-security
Group Id:	test
Version:	1
Package:	(Optional)

2. MODIFY THE POM.XML FILE

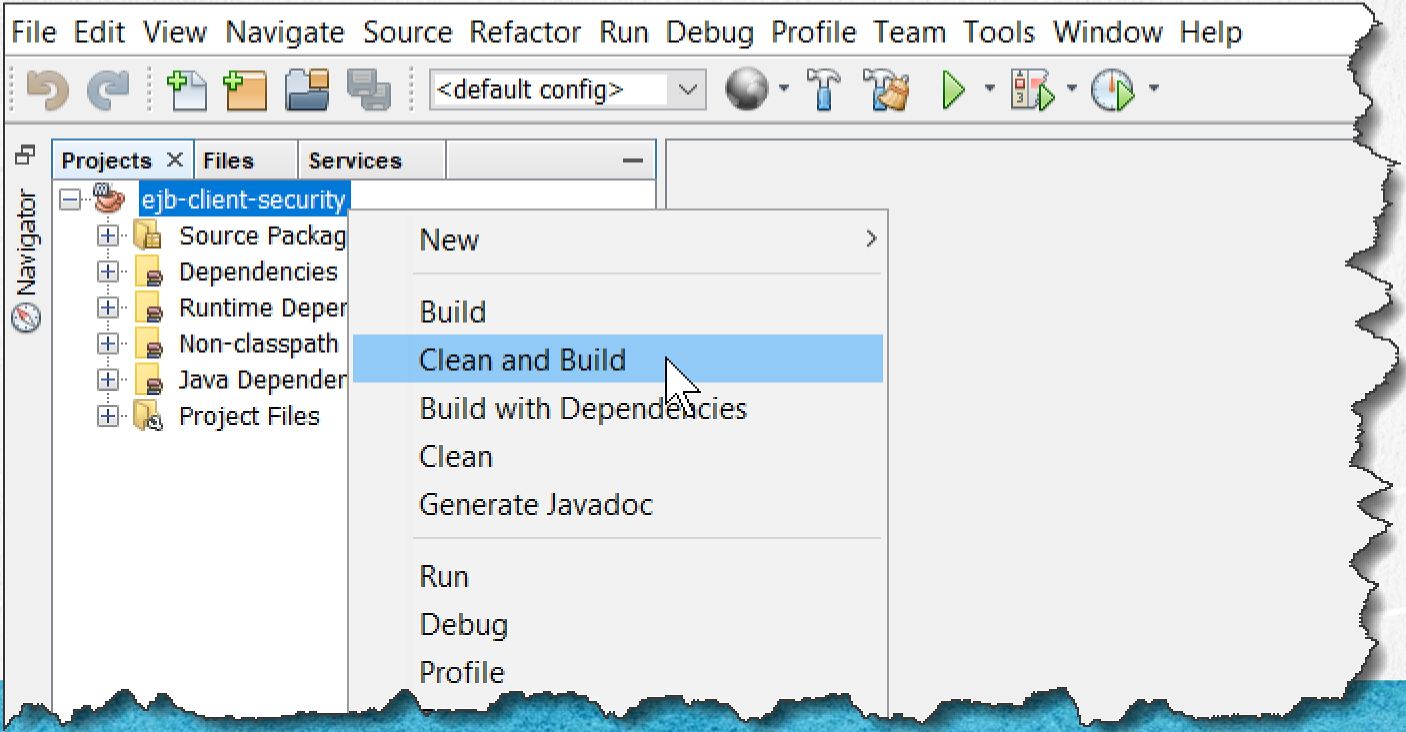
pom.xml:

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>test</groupId>
    <artifactId>ejb-client-security</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>org.glassfish.main.appclient</groupId>
            <artifactId>gf-client</artifactId>
            <version>5.0</version>
        </dependency>
    </dependencies>
</project>
```

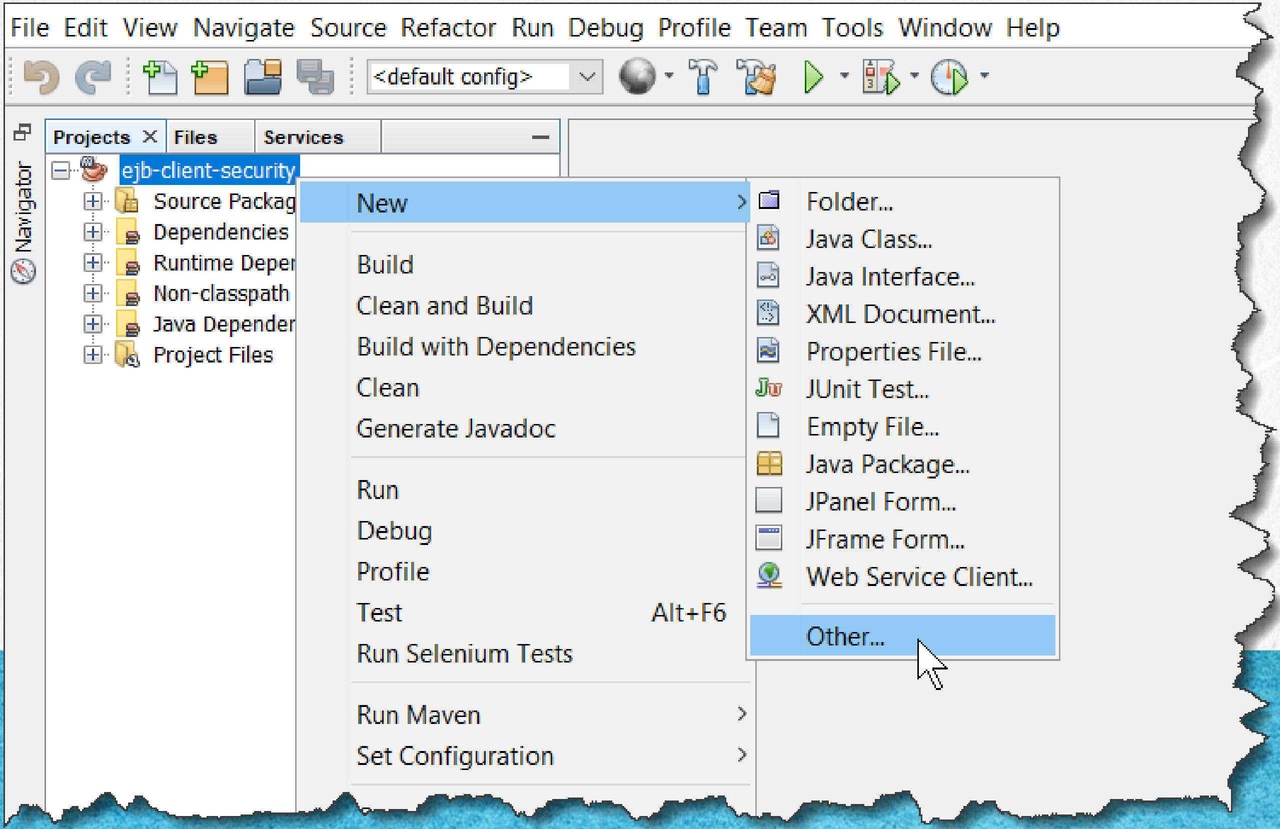
3. EXECUTE CLEAN & BUILD

We do clean & build to, if necessary, download the project libraries via Maven:



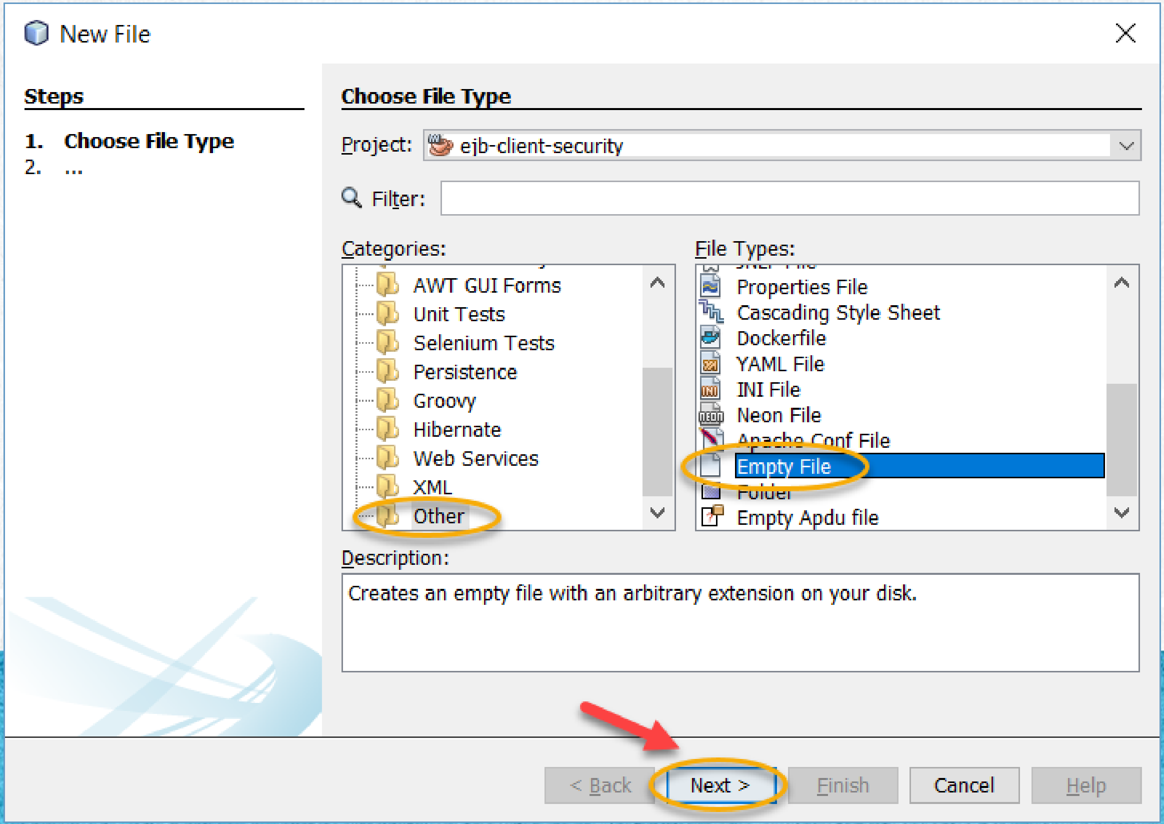
4. ADD A FILE

We add a configuration file called login.conf at the root level of the application. This is necessary to be able to login via the EJB client:



4. ADD A FILE

We add a configuration file called login.conf at the root level of the application. This is necessary to be able to login via the EJB client:



New File

Steps

1. **Choose File Type**
2. ...

Choose File Type

Project: ejb-client-security

Filter:

Categories:

- AWT GUI Forms
- Unit Tests
- Selenium Tests
- Persistence
- Groovy
- Hibernate
- Web Services
- XML
- Other

File Types:

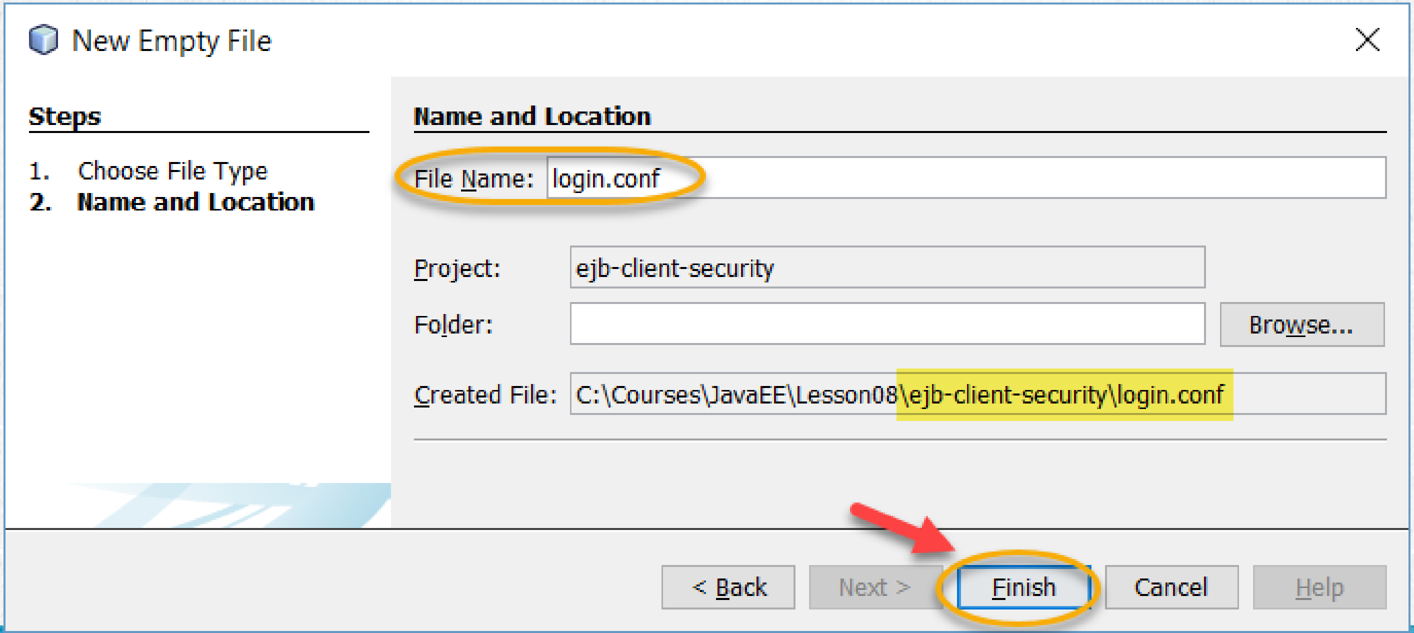
- Properties File
- Cascading Style Sheet
- Dockerfile
- YAML File
- INI File
- Neon File
- Apache Conf File
- Empty File
- Folder
- Empty Apdu file

Description:

Creates an empty file with an arbitrary extension on your disk.

4. ADD A FILE

We add a configuration file called login.conf at the root level of the application. This is necessary to be able to login via the EJB client:



File Name: login.conf

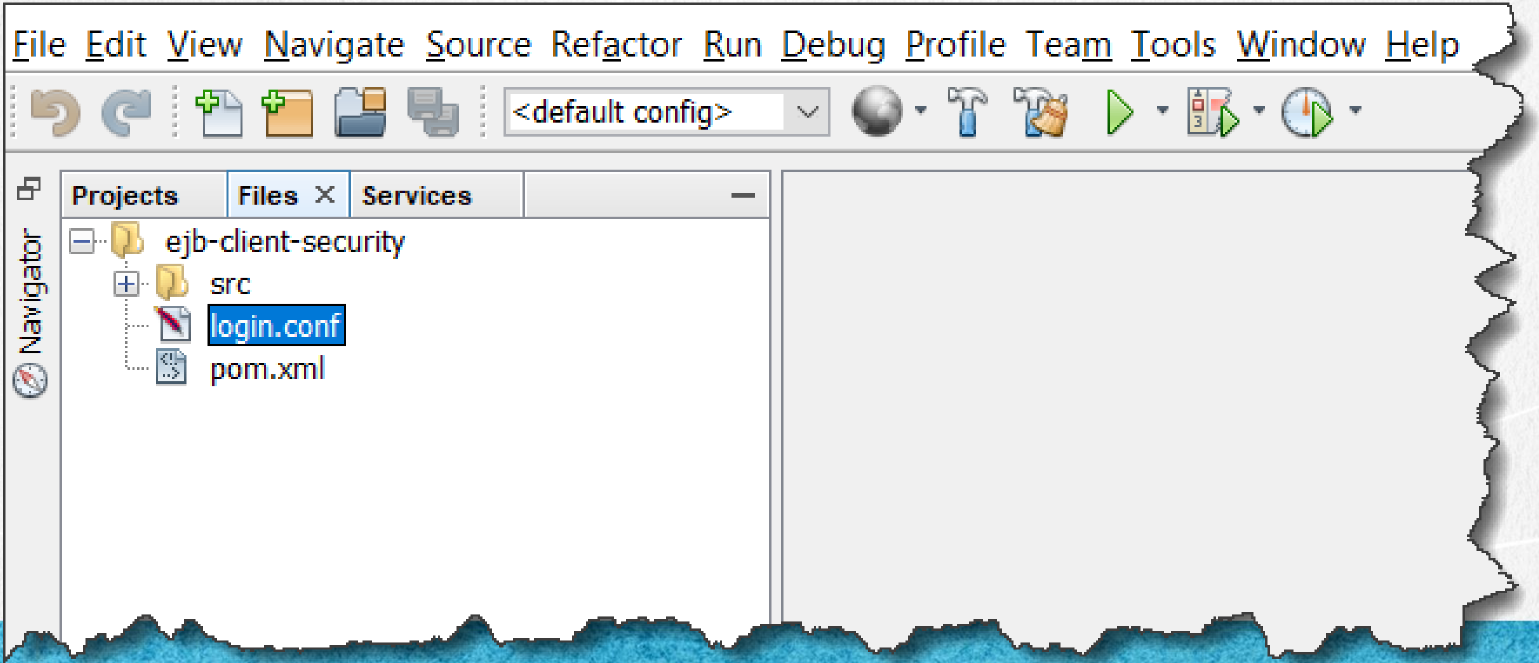
Project: ejb-client-security

Folder:

Created File: C:\Courses\JavaEE\Lesson08\ejb-client-security\login.conf

4. ADD A FILE

If we see the Files tab, we can see that the login.conf file has been created correctly. This file will not be displayed in the Projects tab, only in this Files tab:



5. MODIFY THE FILE

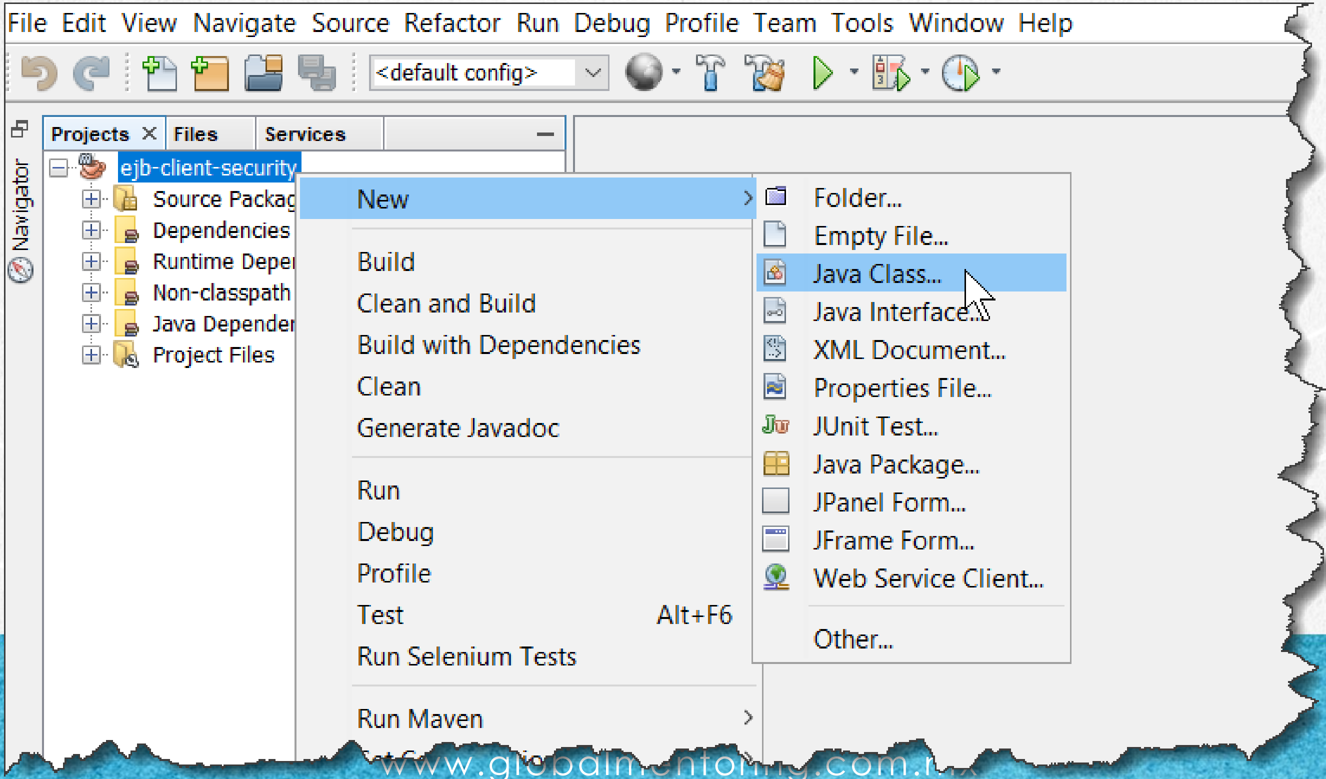
login.config:

Click to download

```
default {
com.sun.enterprise.security.auth.login.ClientPasswordLoginModule required;
};
```

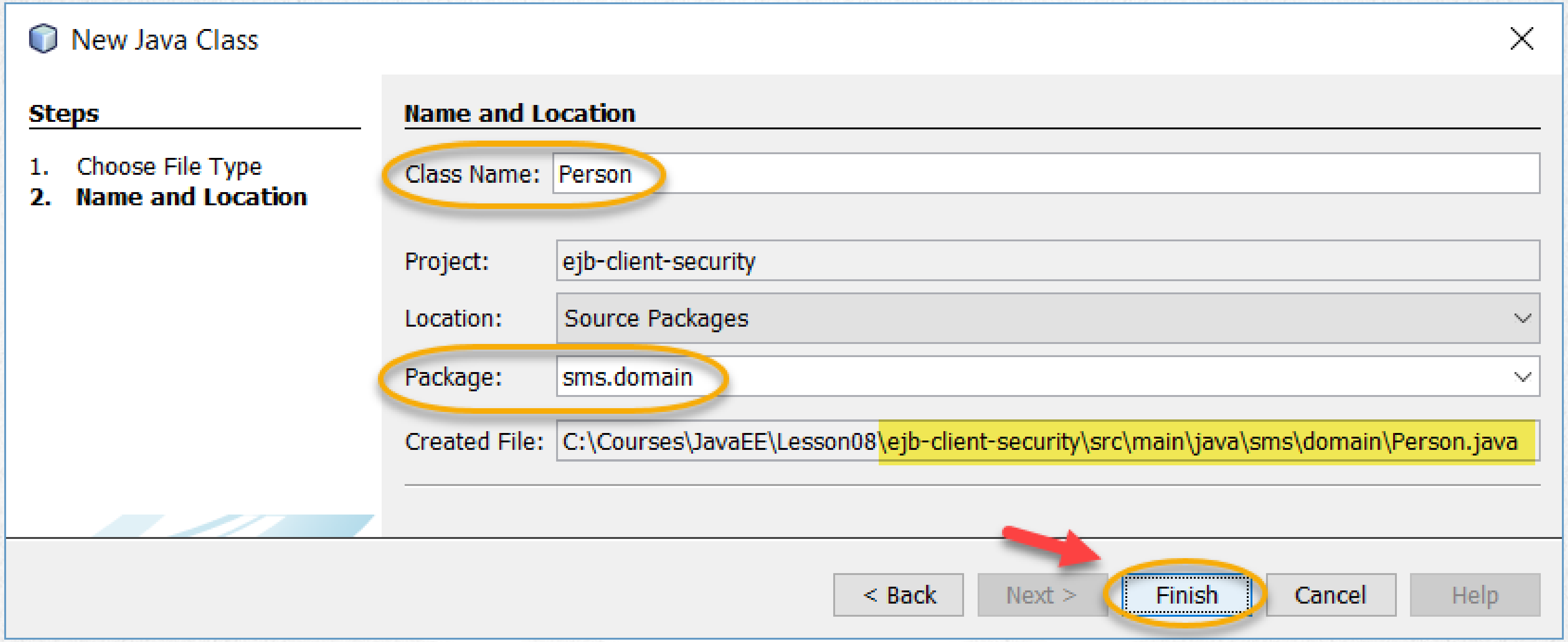
6. CREATE A JAVA CLASS

We add the Java classes that we will use in the EJB client. We only add the essentials so that the types or Java classes that we will use are recognized: We create the class Persona.java:



6. CREATE A JAVA 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: ejb-client-security

Location: Source Packages

Package: sms.domain

Created File: C:\Courses\JavaEE\Lesson08\ejb-client-security\src\main\java\sms\domain\Person.java

7. MODIFY THE FILE

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 idPerson) {
        this.idPerson = idPerson;
    }

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

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

7. MODIFY THE FILE

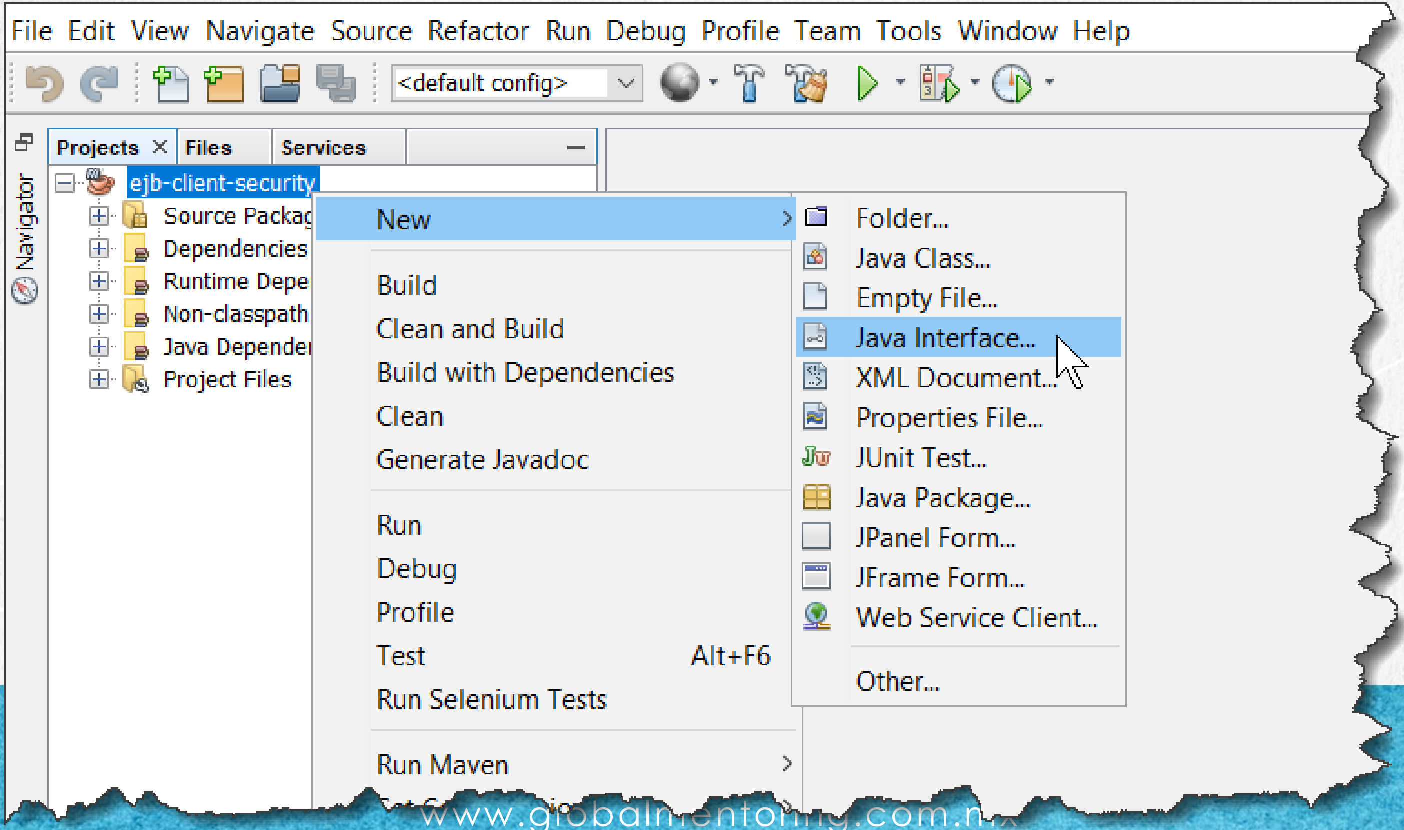
Person.java:

Click to download

```
    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 + '}';
    }
}
```

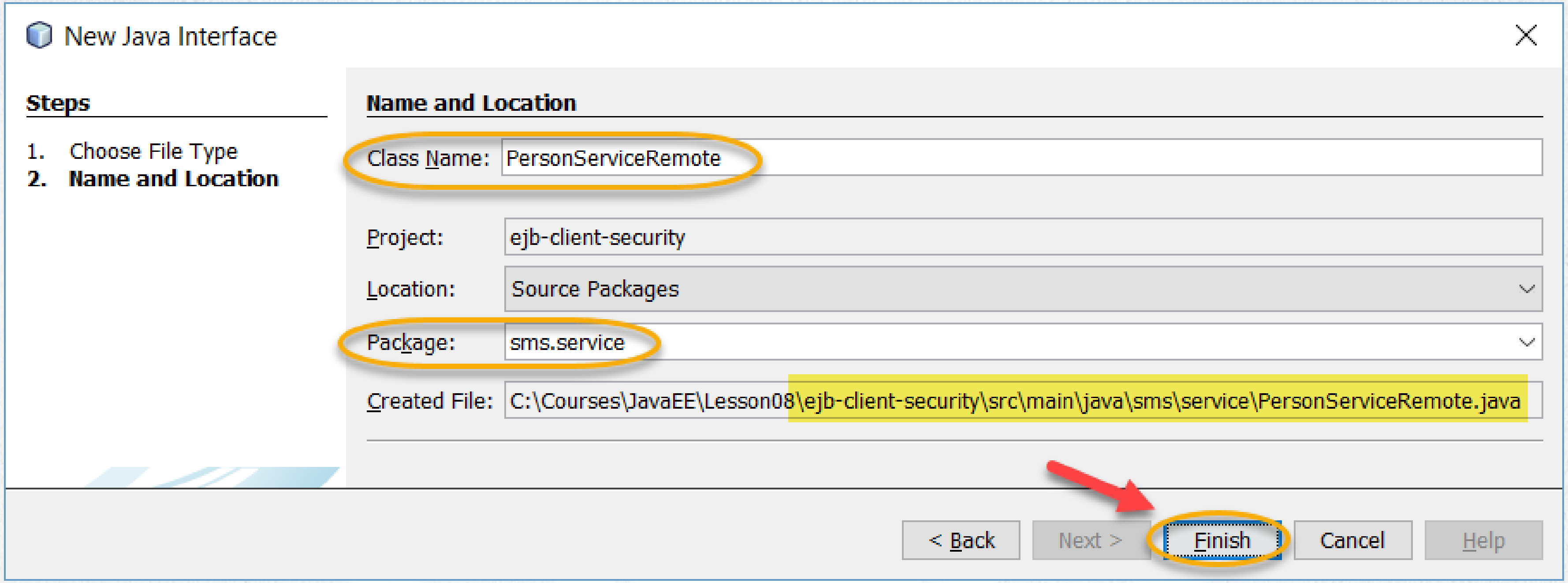
8. Create a Java Interface

We create the PersonServiceRemote.java interface:



8. CREATE A JAVA INTERFACE

We create the PersonServiceRemote.java interface:



New Java Interface

Steps

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

Name and Location

Class Name: PersonServiceRemote

Project: ejb-client-security

Location: Source Packages

Package: sms.service

Created File: C:\Courses\JavaEE\Lesson08\ejb-client-security\src\main\java\sms\service\PersonServiceRemote.java

9. MODIFY THE FILE

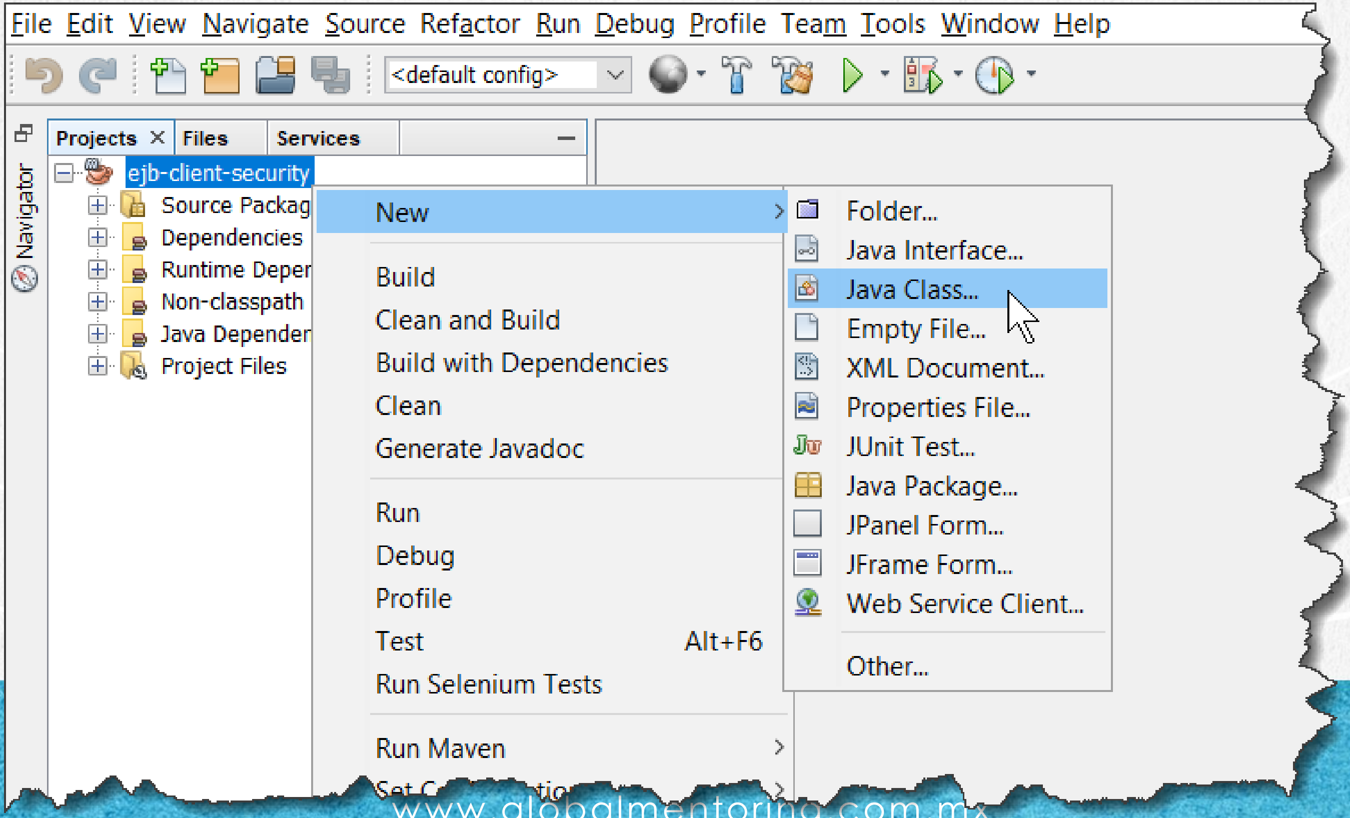
PersonServiceRemote.java:

Click to download

```java
package sms.service;

import java.util.List;
import sms.domain.Person;

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);
}
```

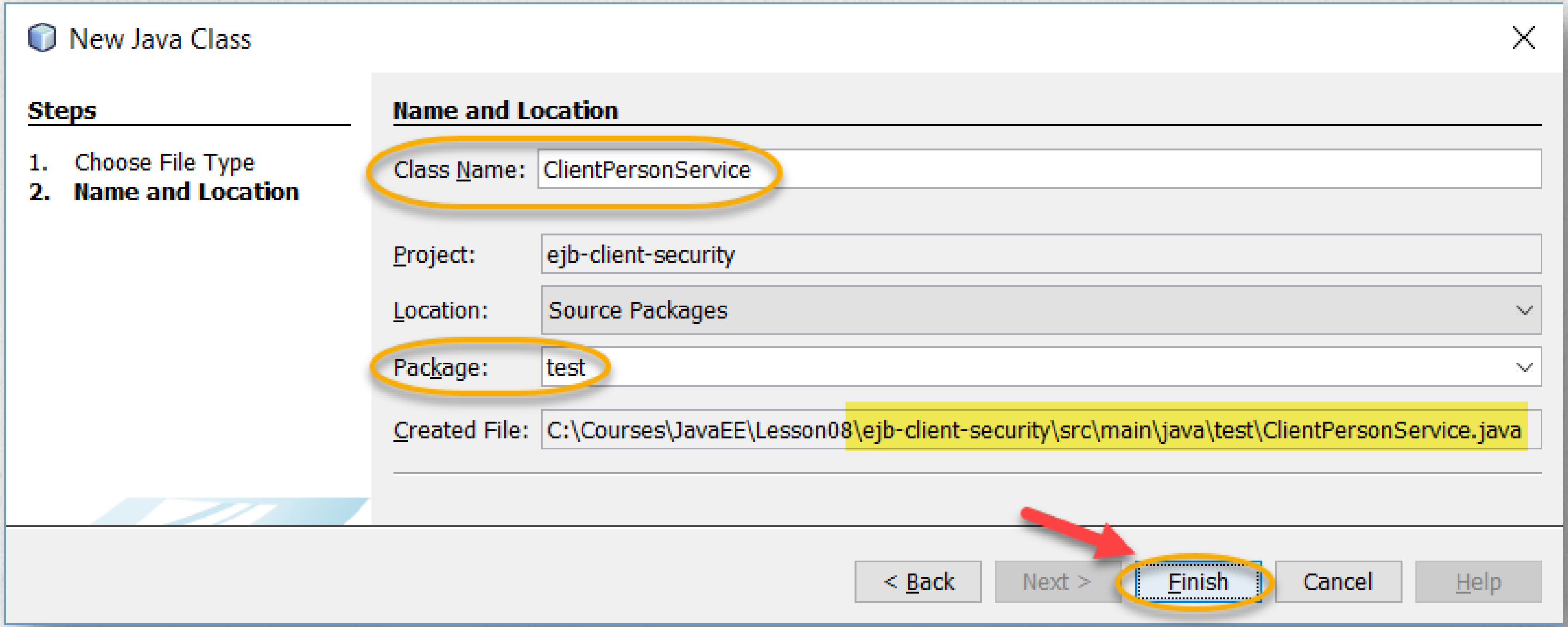
10. CREATE THE CLIENT

We create the ClientPersonService.java class:



10. CREATE THE CLIENT

We create the ClientPersonService.java class:



Class Name: ClientPersonService

Project: ejb-client-security

Location: Source Packages

Package: test

Created File: C:\Courses\JavaEE\Lesson08\ejb-client-security\src\main\java\test\ClientPersonService.java

11. MODIFY THE FILE

ClientPersonService.java:

Click to download

```java
package test;

import com.sun.enterprise.security.ee.auth.login.ProgrammaticLogin;
import java.util.List;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
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");

        String authFile = "login.conf";
        System.setProperty("java.security.auth.login.config", authFile);
        ProgrammaticLogin programmaticLogin = new ProgrammaticLogin();
        programmaticLogin.login("admin", "admin".toCharArray());
```

11. MODIFY THE FILE

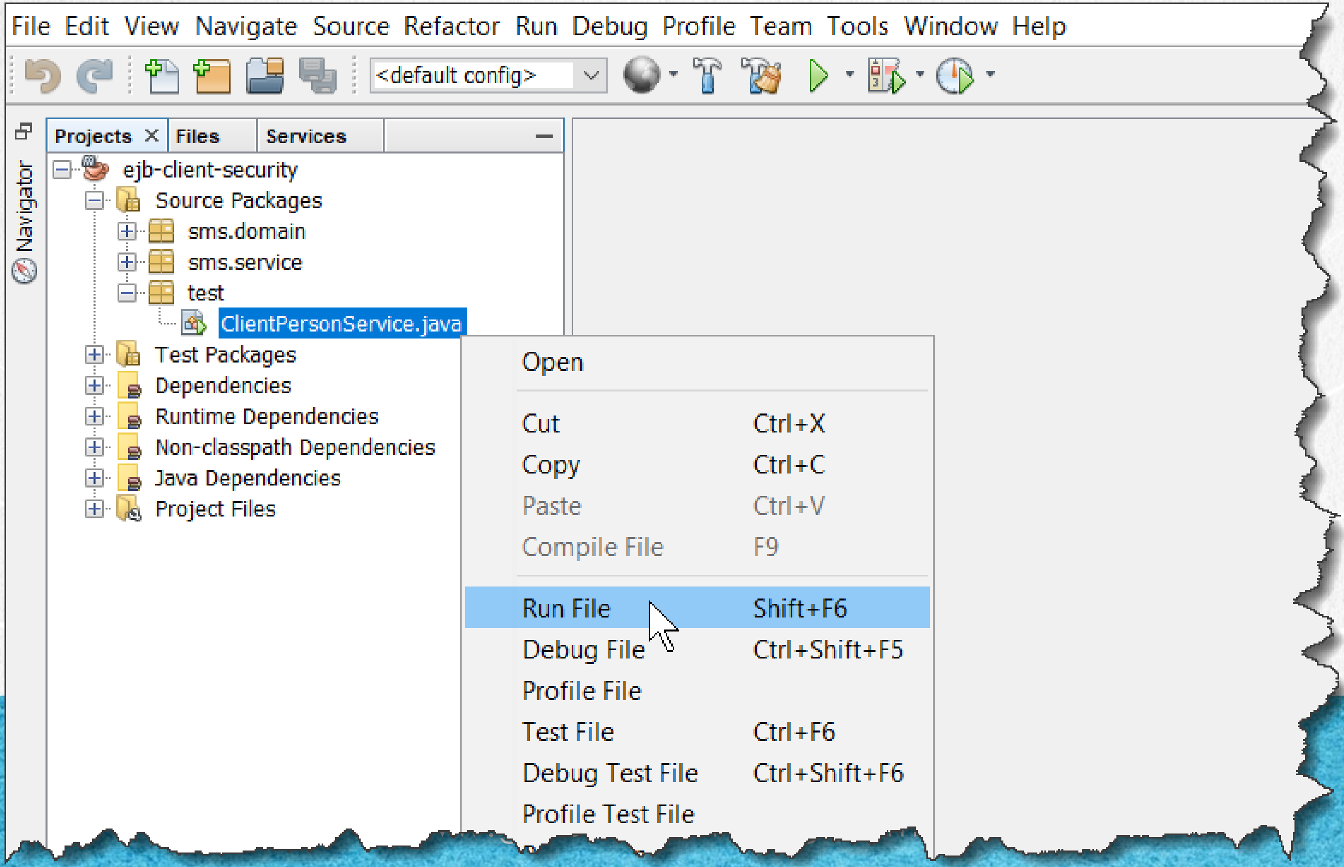
ClientPersonService.java:

Click to download

```
        try {
            Context jndi = new InitialContext();
            PersonServiceRemote personService = (PersonServiceRemote)
                jndi.lookup("java:global/sms-jee-web/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);
        }
    }
}
```

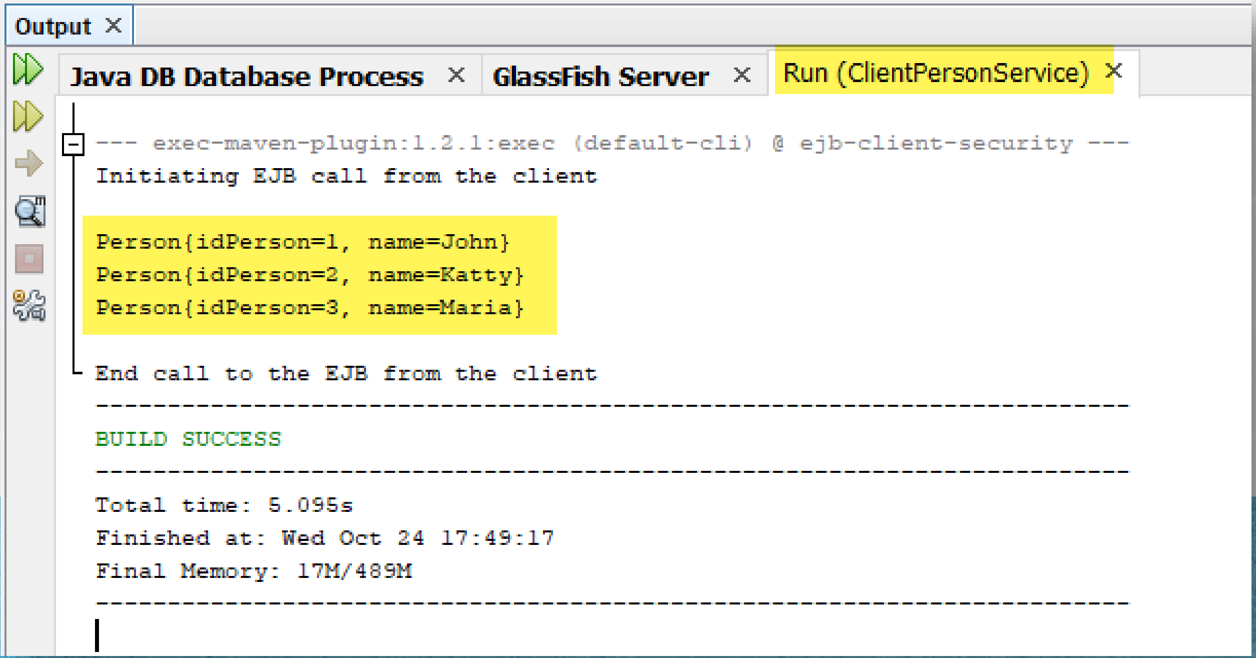
12. EXECUTE THE PROJECT

We execute the ClientPersonService.java class:



12. EXECUTE THE PROJECT

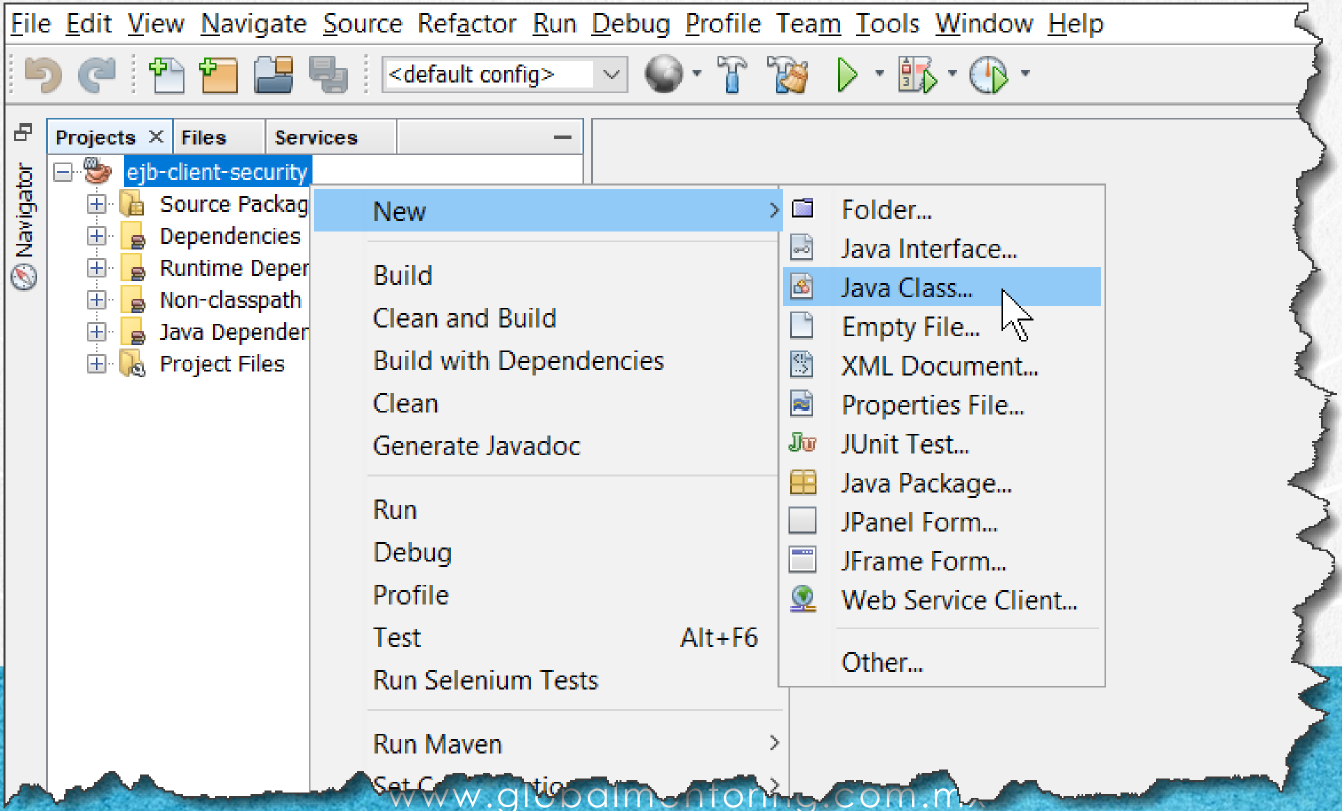
We can observe the result (the data may vary) of executing the list of people of the EJB, as we have seen the code has already added the credentials of a valid user. Note: The sms-jee-web application must already be deployed and running in Glassfish, otherwise the call to the server can not be executed:



```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ ejb-client-security ---
Initiating EJB call from the client

Person{idPerson=1, name=John}
Person{idPerson=2, name=Katty}
Person{idPerson=3, name=Maria}

End call to the EJB from the client
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 5.095s
Finished at: Wed Oct 24 17:49:17
Final Memory: 17M/489M
------------------------------------------------------------------------
```

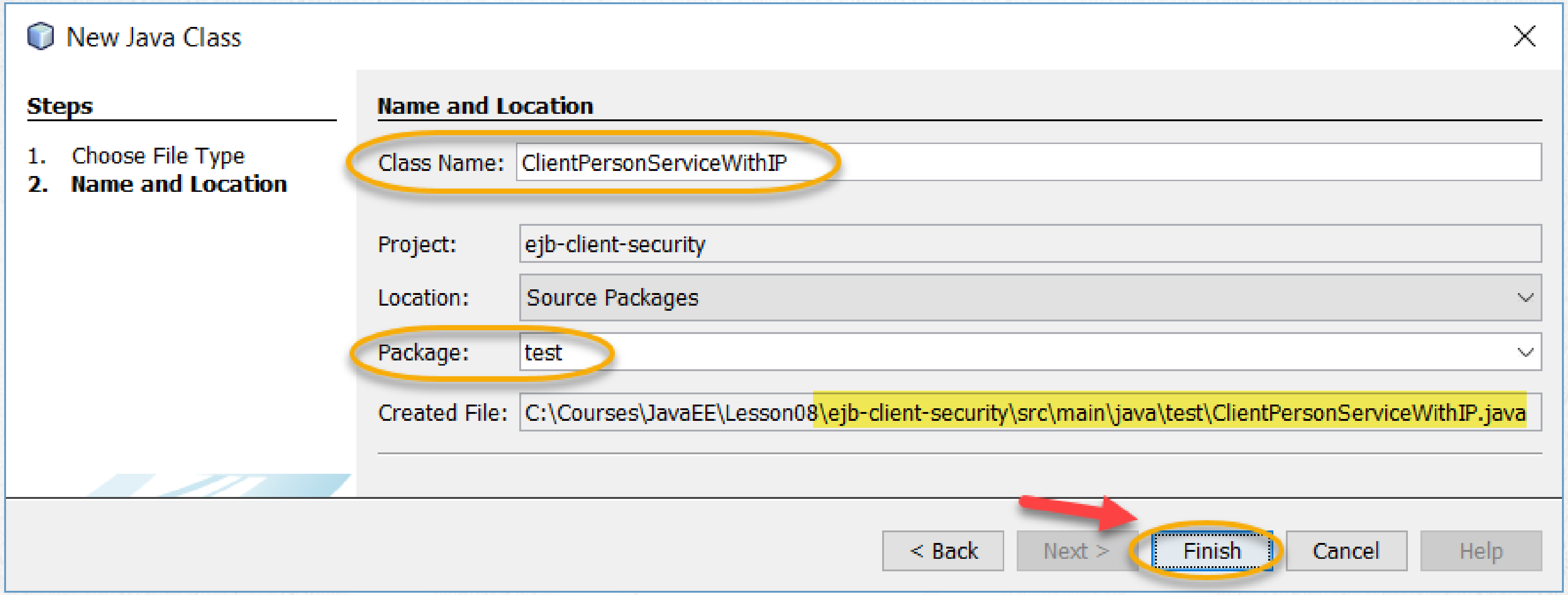
13. CREATE A TEST

We create the ClientPersonServiceWithIP.java class:



13. CREATE A TEST

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: ejb-client-security

Location: Source Packages

Package: test

Created File: C:\Courses\JavaEE\Lesson08\ejb-client-security\src\main\java\test\ClientPersonServiceWithIP.java

14. MODIFY THE FILE

ClientPersonServiceWithIP.java:

Click to download

```
package test;

import com.sun.enterprise.security.ee.auth.login.ProgrammaticLogin;
import java.util.List;
import java.util.Properties;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
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\n");

        String authFile = "login.conf";
        System.setProperty("java.security.auth.login.config", authFile);
        ProgrammaticLogin programmaticLogin = new ProgrammaticLogin();
        programmaticLogin.login("admin", "admin".toCharArray());

        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");
```

14. MODIFY THE FILE

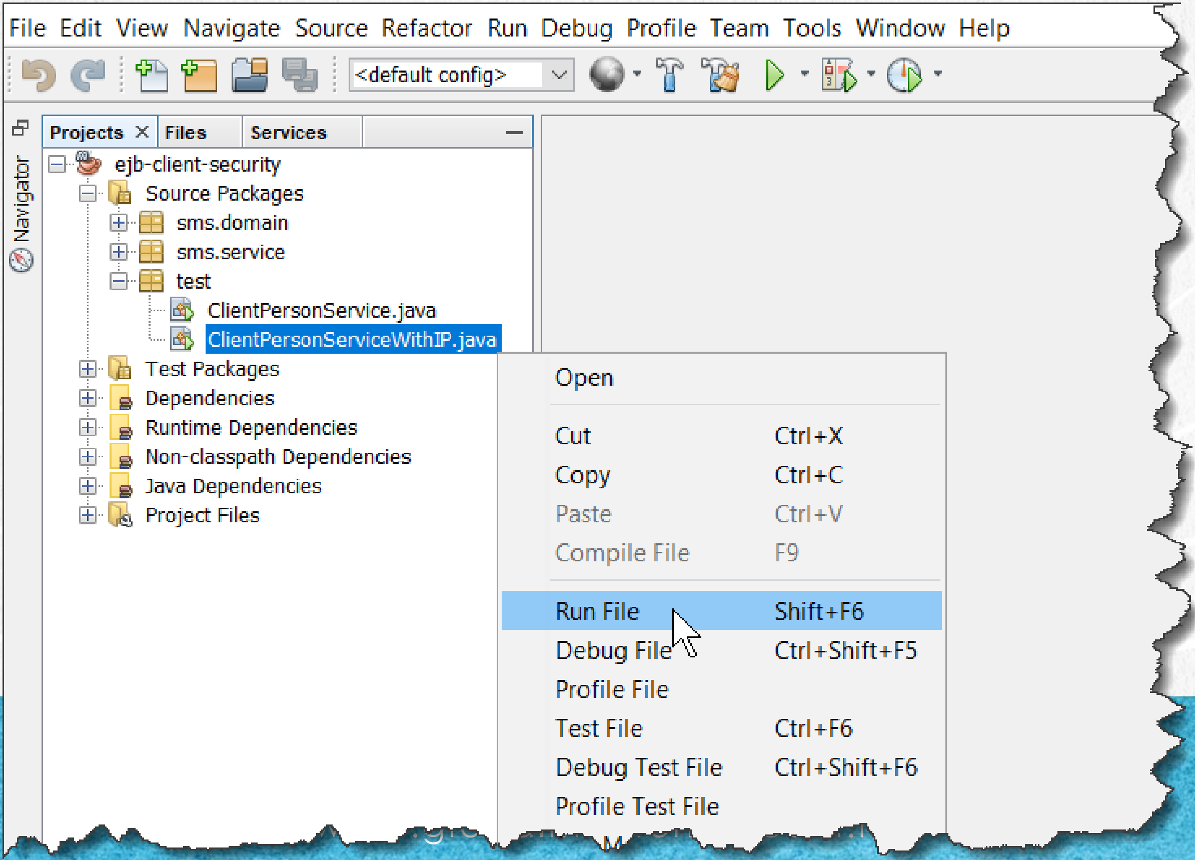
ClientPersonServiceWithIP.java:

Click to download

```
            // 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-web/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);
        }
    }

}
```

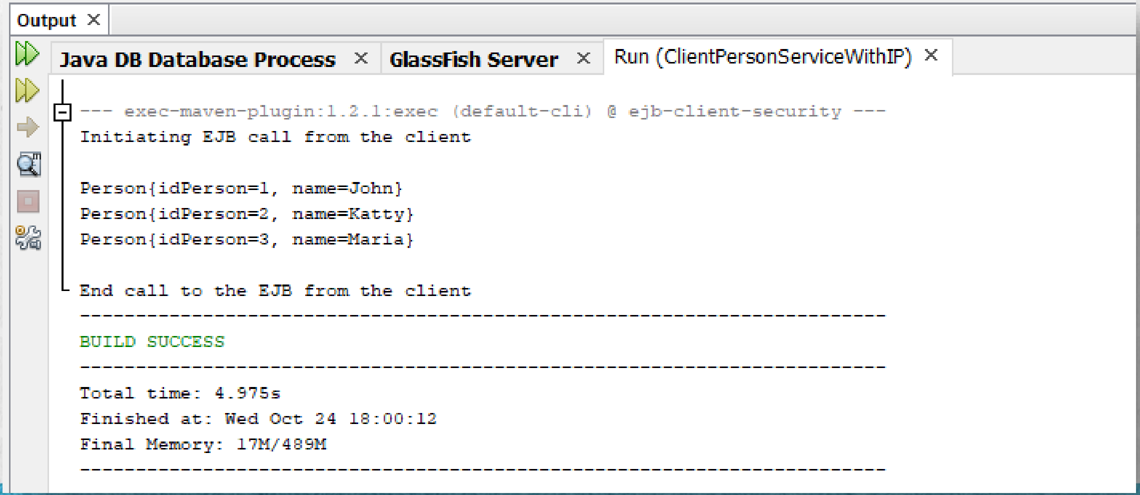
15. EXECUTE THE PROJECT

We execute the ClientPersonServiceWithIP.java class:



15. EXECUTE THE PROJECT

We can observe the result (the data may vary) of executing the EJB Guest list from a client with code to access remotely (from another IP), as we have seen the code has already added the credentials of a valid user. Note: The sms-jee-web application must already be deployed and running in Glassfish, otherwise the call to the server can not be executed:



```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ ejb-client-security ---
Initiating EJB call from the client

Person{idPerson=1, name=John}
Person{idPerson=2, name=Katty}
Person{idPerson=3, name=Maria}

End call to the EJB from the client
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 4.975s
Finished at: Wed Oct 24 18:00:12
Final Memory: 17M/489M
------------------------------------------------------------------------
```

EXERCISE CONCLUSION

With this exercise we have created the client to add the user and password and thus be able to access the EJB that already has security on the server side.

We create a project with a client that runs locally or we also create a client that can be run remotely from another IP.

ONLINE COURSE

JAVA EE JAKARTA EE

By: Eng. Ubaldo Acosta

JAVA EE COURSE

www.globalmentoring.com.mx