JAVA EE COURSE

EXERCISE

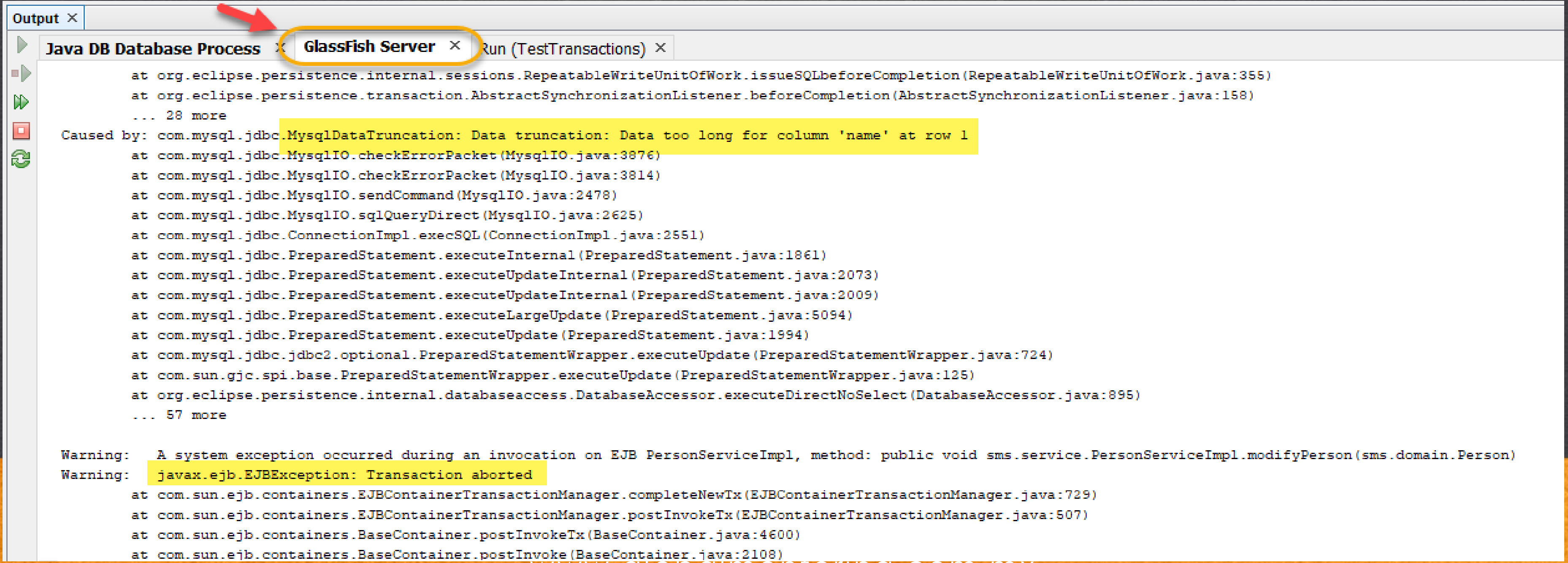
HANDLING TRANSACTIONS WITH JAVA EE

JAVA EE COURSE

www.globalmentoring.com.mx

EXERCISE OBJECTIVE

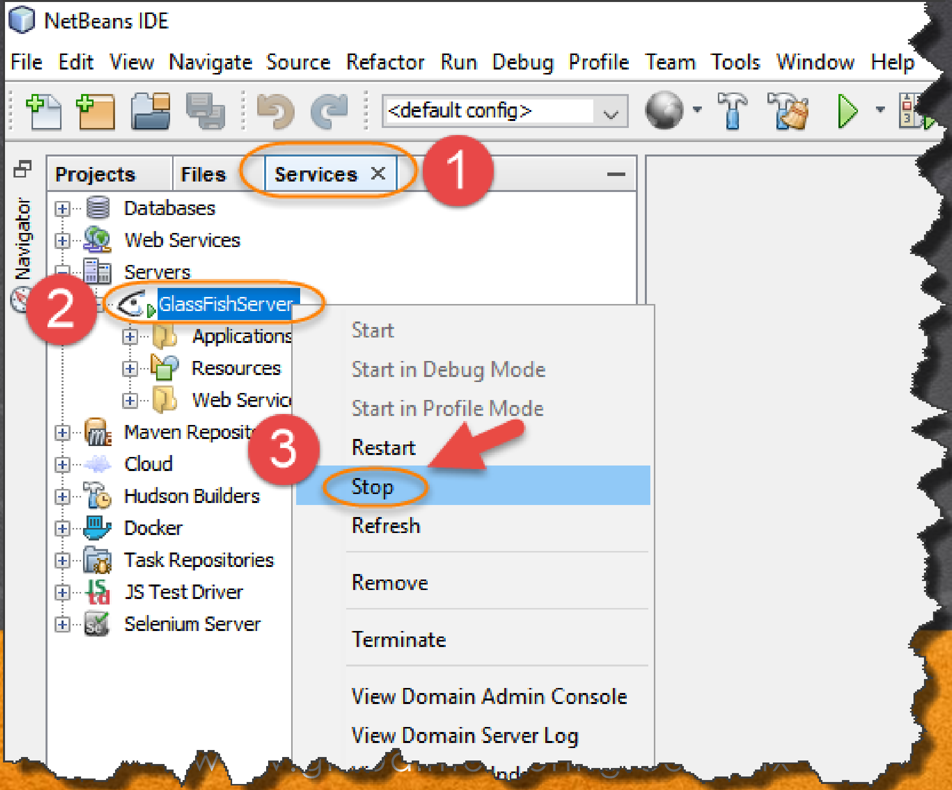
Create a project to implement the handling of transactions in Java EE. At the end we should observe the following:



```
        at org.eclipse.persistence.internal.sessions.RepeatableWriteUnitOfWork.issueSQLbeforeCompletion(RepeatableWriteUnitOfWork.java:355)
        at org.eclipse.persistence.transaction.AbstractSynchronizationListener.beforeCompletion(AbstractSynchronizationListener.java:158)
        ... 28 more
Caused by: com.mysql.jdbc.MysqlDataTruncation: Data truncation: Data too long for column 'name' at row 1
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3876)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3814)
        at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:2478)
        at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2625)
        at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2551)
        at com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:1861)
        at com.mysql.jdbc.PreparedStatement.executeUpdateInternal(PreparedStatement.java:2073)
        at com.mysql.jdbc.PreparedStatement.executeUpdateInternal(PreparedStatement.java:2009)
        at com.mysql.jdbc.PreparedStatement.executeLargeUpdate(PreparedStatement.java:5094)
        at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:1994)
        at com.mysql.jdbc.jdbc2.optional.PreparedStatementWrapper.executeUpdate(PreparedStatementWrapper.java:724)
        at com.sun.gjc.spi.base.PreparedStatementWrapper.executeUpdate(PreparedStatementWrapper.java:125)
        at org.eclipse.persistence.internal.databaseaccess.DatabaseAccessor.executeDirectNoSelect(DatabaseAccessor.java:895)
        ... 57 more

Warning:   A system exception occurred during an invocation on EJB PersonServiceImpl, method: public void sms.service.PersonServiceImpl.modifyPerson(sms.domain.Person)
Warning:   javax.ejb.EJBException: Transaction aborted
        at com.sun.ejb.containers.EJBContainerTransactionManager.completeNewTx(EJBContainerTransactionManager.java:729)
        at com.sun.ejb.containers.EJBContainerTransactionManager.postInvokeTx(EJBContainerTransactionManager.java:507)
        at com.sun.ejb.containers.BaseContainer.postInvokeTx(BaseContainer.java:4600)
        at com.sun.ejb.containers.BaseContainer.postInvoke(BaseContainer.java:2108)
```

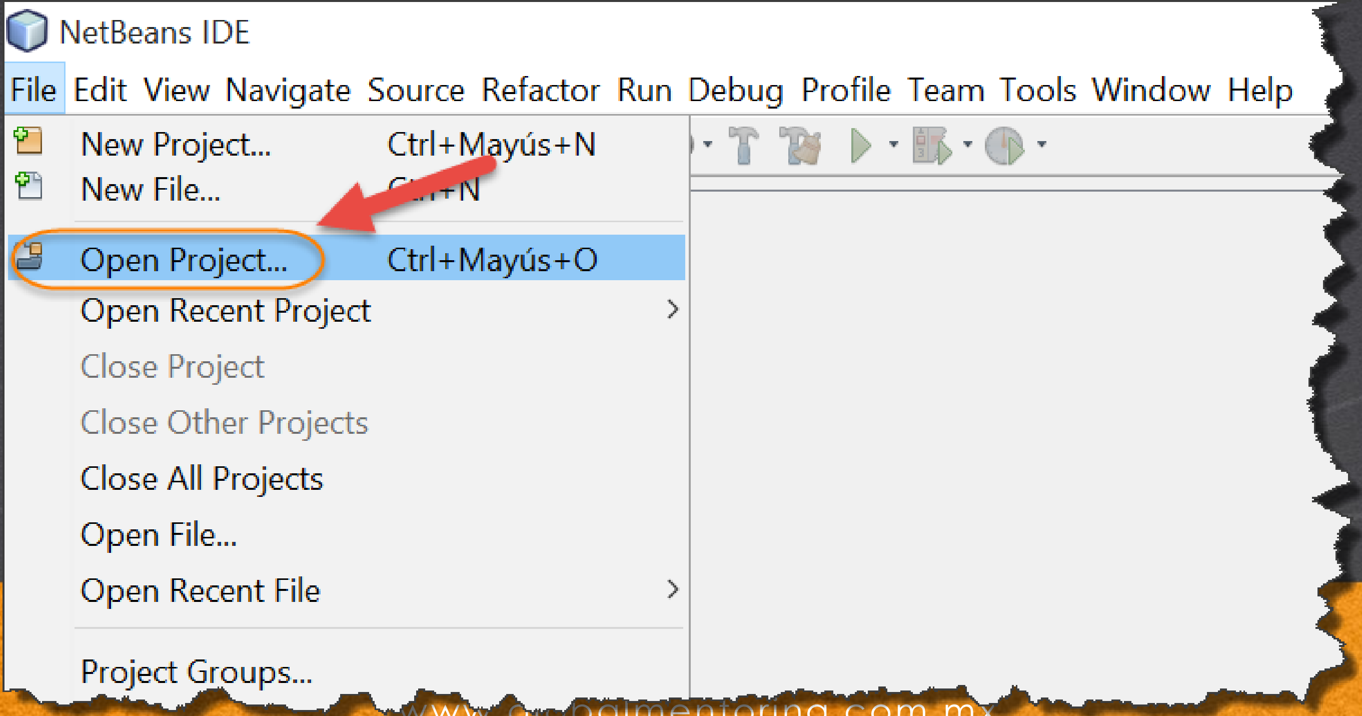
STOP GLASSFISH

In case it is running, we stop the Glassfish server:



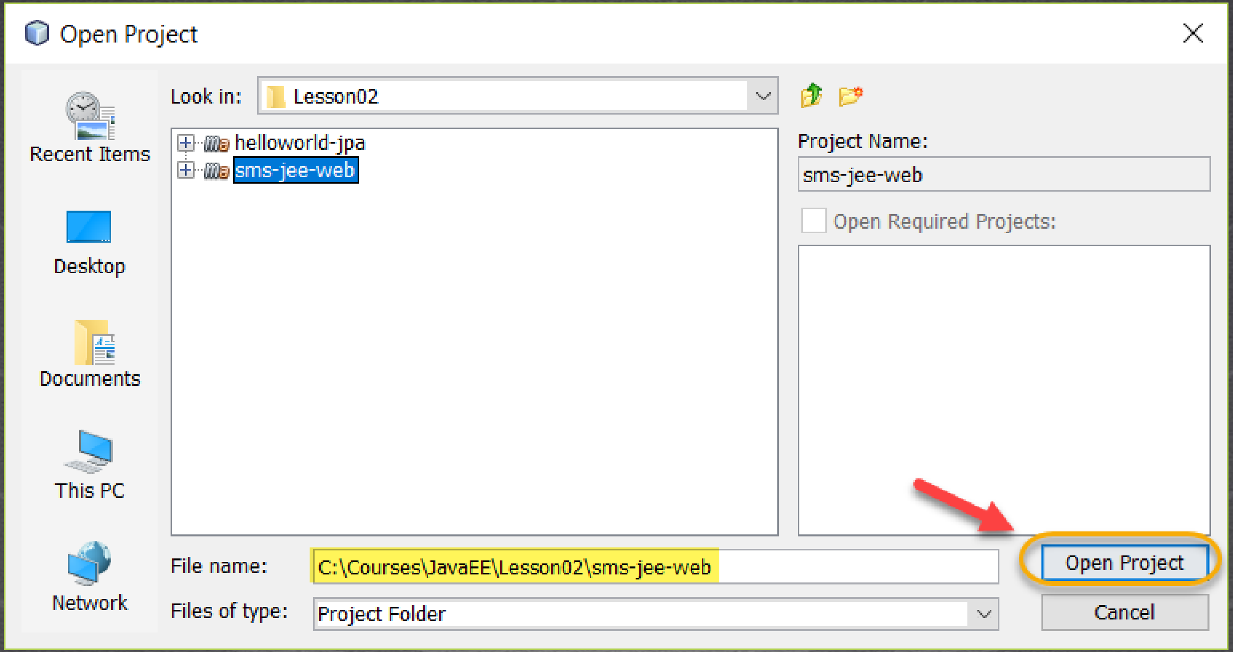
1. OPEN THE PROJECT

In case we do not have open the project sms-jee-web we open it (the latest version):



1. OPEN THE PROJECT

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



Look in: Lesson02

helloworld-jpa
sms-jee-web

Project Name: sms-jee-web

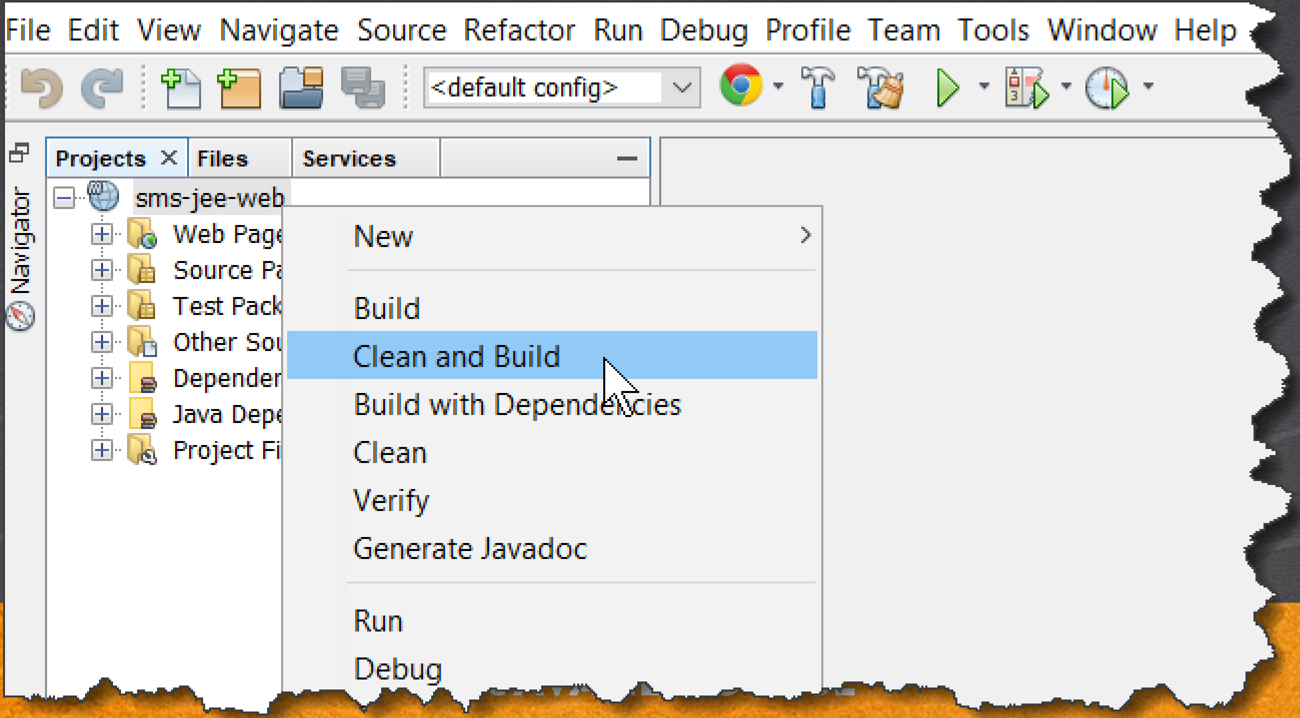
Open Required Projects:

File name: C:\Courses\JavaEE\Lesson02\sms-jee-web

Files of type: Project Folder

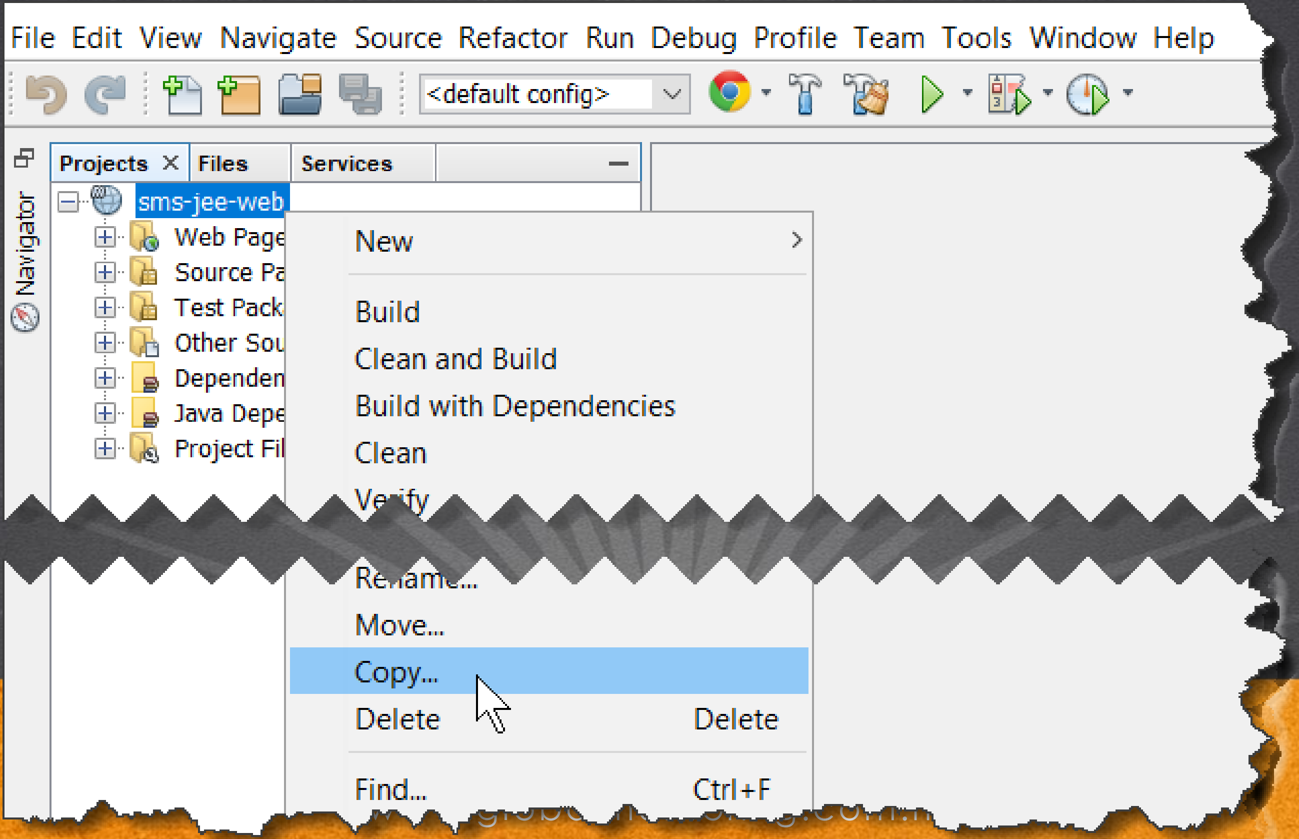
1. OPEN THE PROJECT

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:



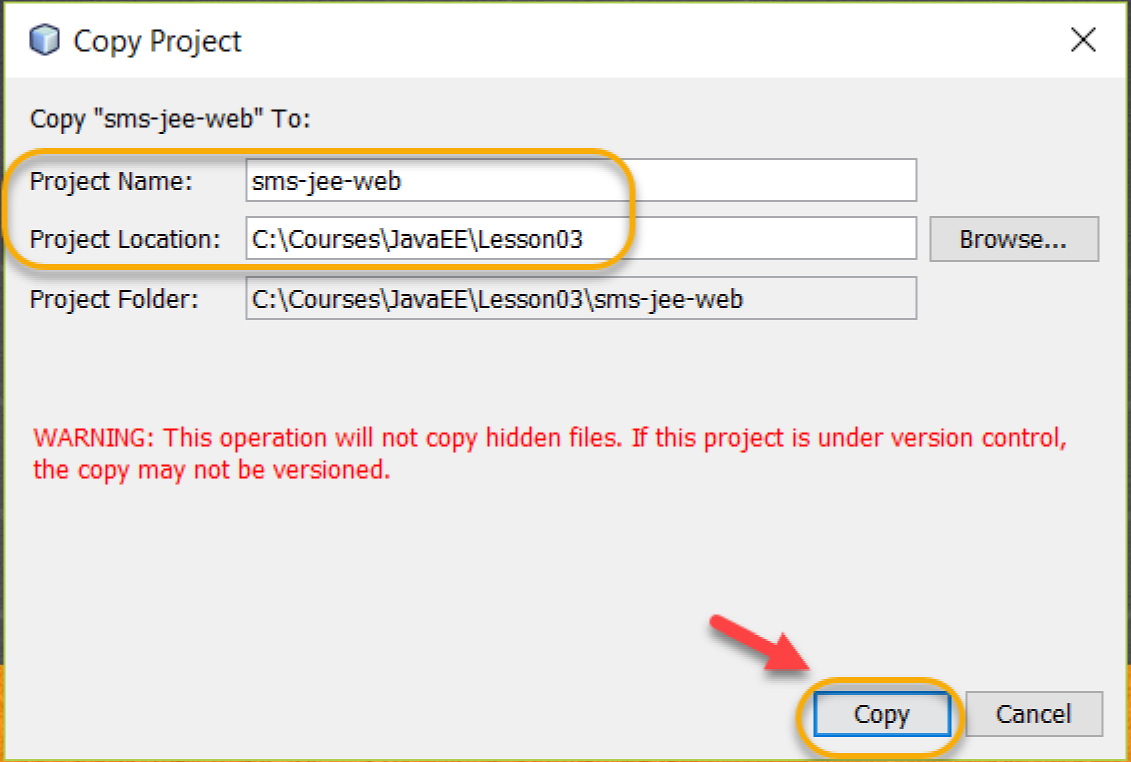
2. COPY THE PROJECT

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:



Copy Project

Copy "sms-jee-web" To:

Project Name: sms-jee-web

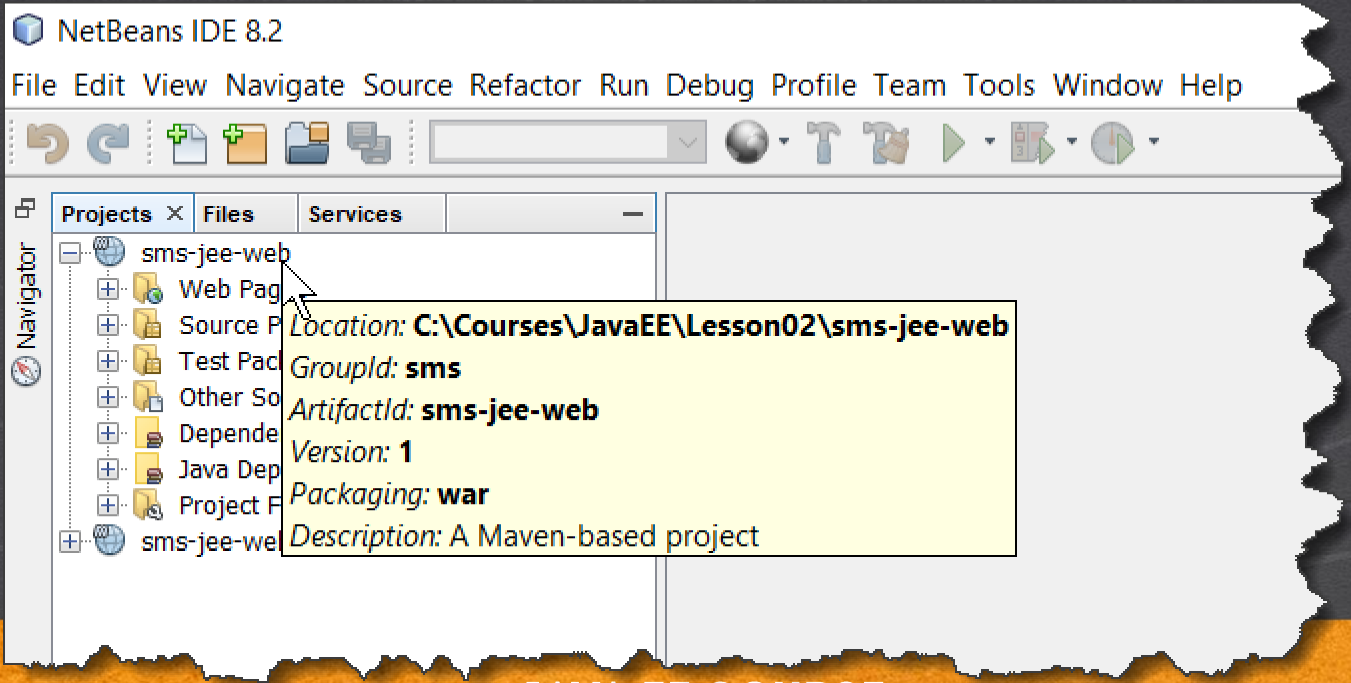
Project Location: C:\Courses\JavaEE\Lesson03

Project Folder: C:\Courses\JavaEE\Lesson03\sms-jee-web

WARNING: This operation will not copy hidden files. If this project is under version control, the copy may not be versioned.

3. CLOSE THE PROJECT

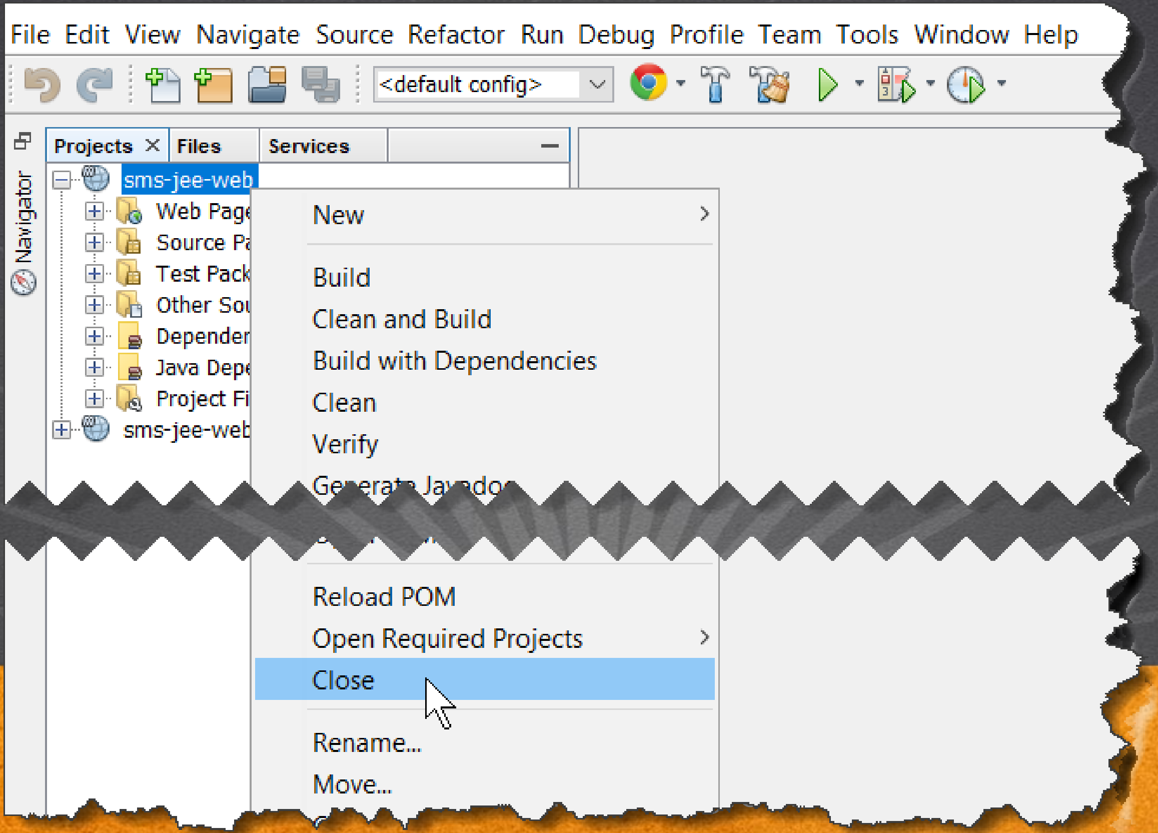
We closed the previous project and left only the new one. We identify the project to be closed:



Location: **C:\Courses\JavaEE\Lesson02\sms-jee-web**
GroupId: **sms**
ArtifactId: **sms-jee-web**
Version: **1**
Packaging: **war**
Description: A Maven-based project

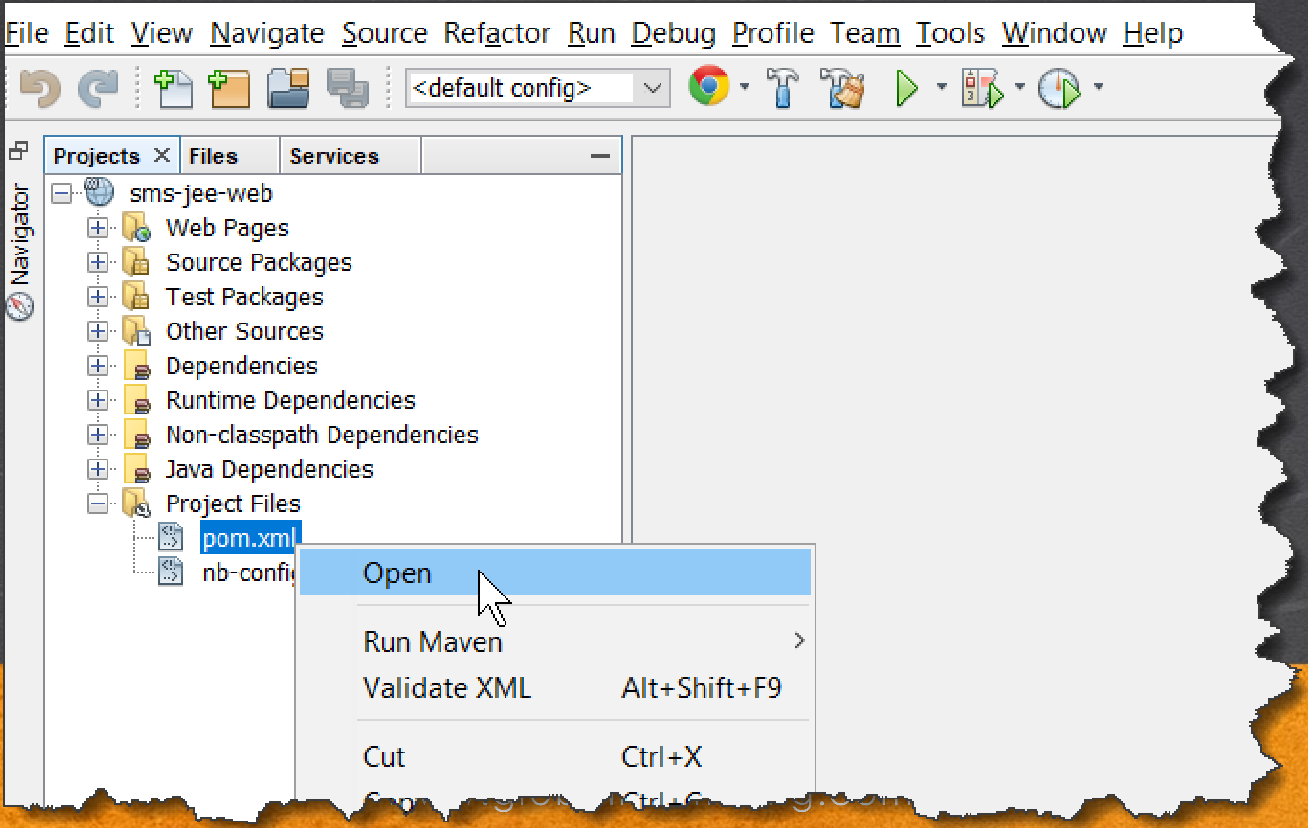
3.CLOSE THE PROJECT

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



4. MODIFY THE POM.XML

Modify the pom.xml to add the glassfish client library:



4. 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-web</artifactId>
    <version>1</version>
    <packaging>war</packaging>

    <name>sms-jee-web</name>

   <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>javax</groupId>
            <artifactId>javaee-api</artifactId>
            <version>8.0</version>
            <scope>provided</scope>
        </dependency>
         <dependency>
            <groupId>org.glassfish.main.appclient</groupId>
            <artifactId>gf-client</artifactId>
            <version>5.0</version>
        </dependency>
    </dependencies>
```

4. MODIFY THE CODE

pom.xml:

Click to download

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>2.6</version>
                <configuration>
                    <failOnMissingWebXml>false</failOnMissingWebXml>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

TRANSACTIONAL MANAGEMENT

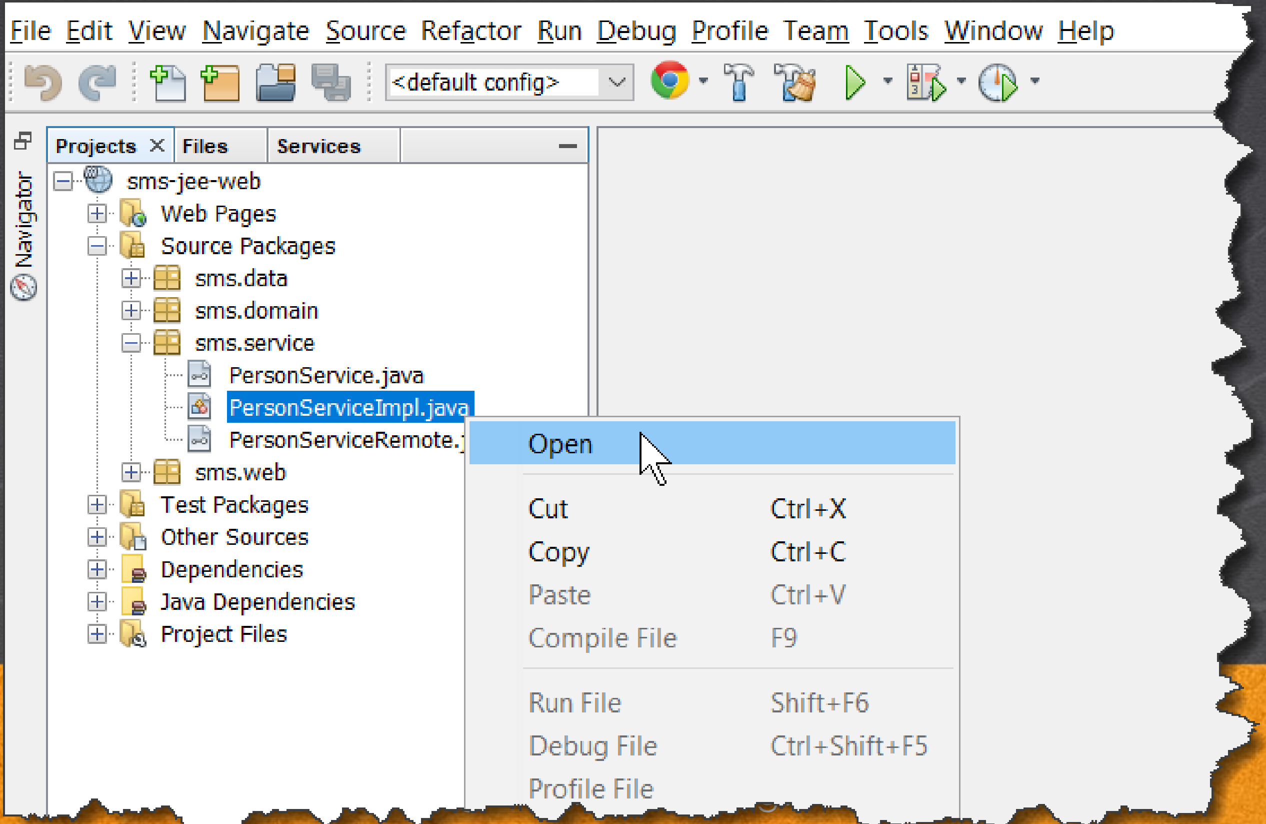
In automatic the EJB handle the transactional concept when performing operations on the EJB methods.

We are going to modify the update method of the EJB PersonaServiceImpl.java to process the transaction and in case of error, rollback and also print the error occurred in the console.

Let's see how our class is:

5. MODIFY A JAVA CLASS

Modify the PersonServiceImpl.java class:



5. MODIFY THE CODE

PersonServiceImpl.java:

Click to download

```java
package sms.service;

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

@Stateless
public class PersonServiceImpl implements PersonServiceRemote, PersonService {

    @Resource
    private SessionContext context;

    @Inject
    private PersonDao personDao;

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

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

5. MODIFY THE CODE

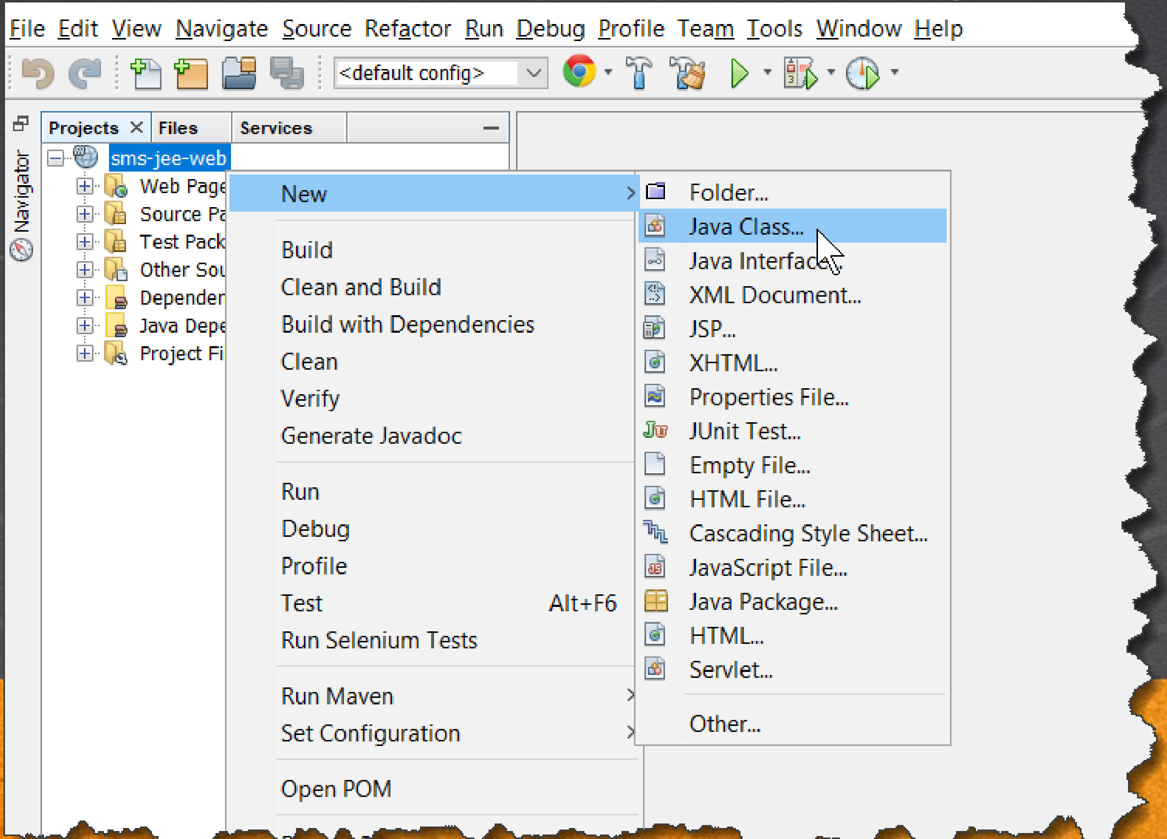
PersonServiceImpl.java:

Click to download

```
    @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);
        }
    }

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

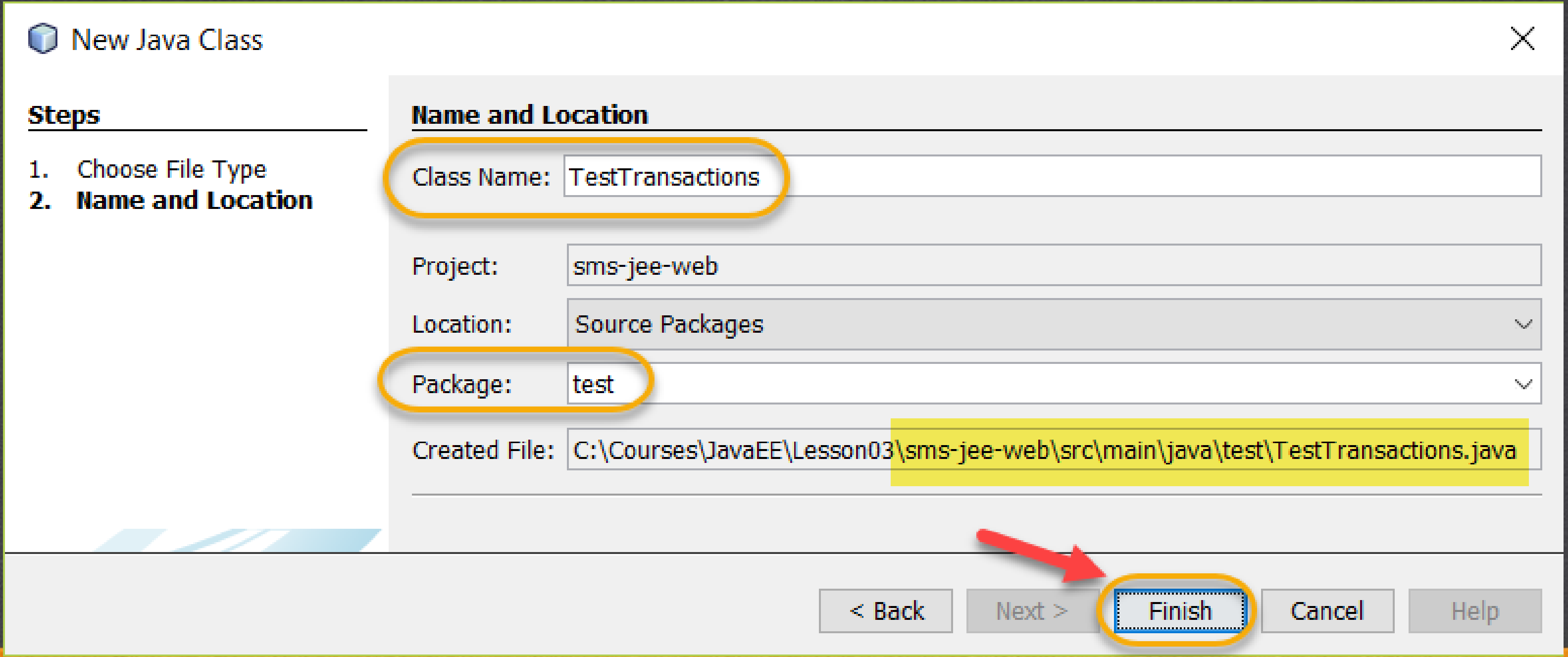
6. CREATE A TEST CLASS

We create a class to test transactional management:



6. CREATE A TEST CLASS

We create a class to test transactional management:



New Java Class

Steps

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

Name and Location

Class Name: TestTransactions

Project: sms-jee-web

Location: Source Packages

Package: test

Created File: C:\Courses\JavaEE\Lesson03\sms-jee-web\src\main\java\test\TestTransactions.java

7. MODIFY THE FILE

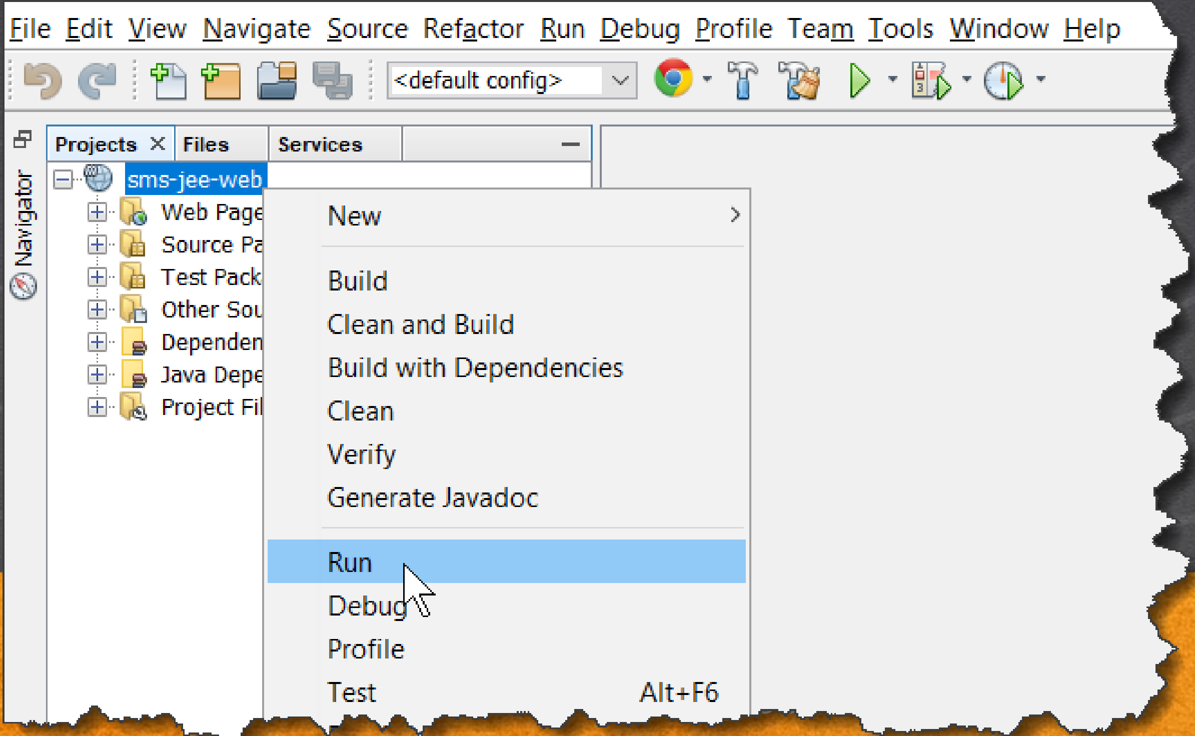
TestTransactions.java:

Click to download

```
package test;

import javax.naming.Context;
import javax.naming.InitialContext;
import sms.domain.Person;
import sms.service.PersonServiceRemote;

public class TestTransactions {

    public static void main(String[] args) throws Exception {
        Context jndi = new InitialContext();
        PersonServiceRemote personService =
             (PersonServiceRemote) jndi.lookup("java:global/sms-jee-web/PersonServiceImpl!sms.service.PersonServiceRemote");

        System.out.println("Starting a Transactional PersonService test");

        //We are looking for a person object
        Person person = new Person();
        person.setIdPerson(1);
        person = personService.findPerson(person);

        //We change the person
        person.setName("Change with error.........................................................");
        //person.setName("John2");

        personService.modifyPerson(person);
        System.out.println("Modified Object:" + person);
        System.out.println("End EJB PersonService test");
    }
}
```

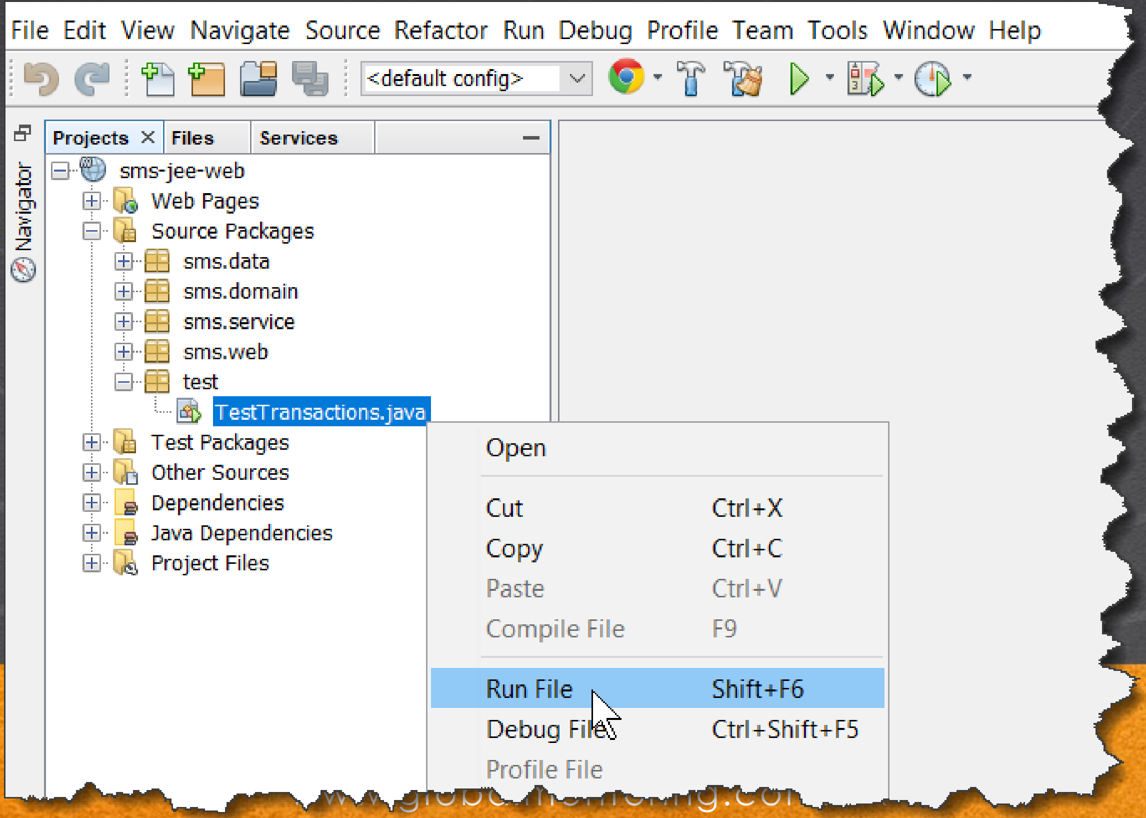
8. EXECUTE THE PROJECT

We execute the project to deploy the changes in Glassfish, this will include the changes made to the EJB called PersonaServiceImpl.java and wait a few seconds to wait for the project to finish deploying on Glassfish:



9. EXECUTE THE TEST CLASS

We execute the class TestTransactions.java:



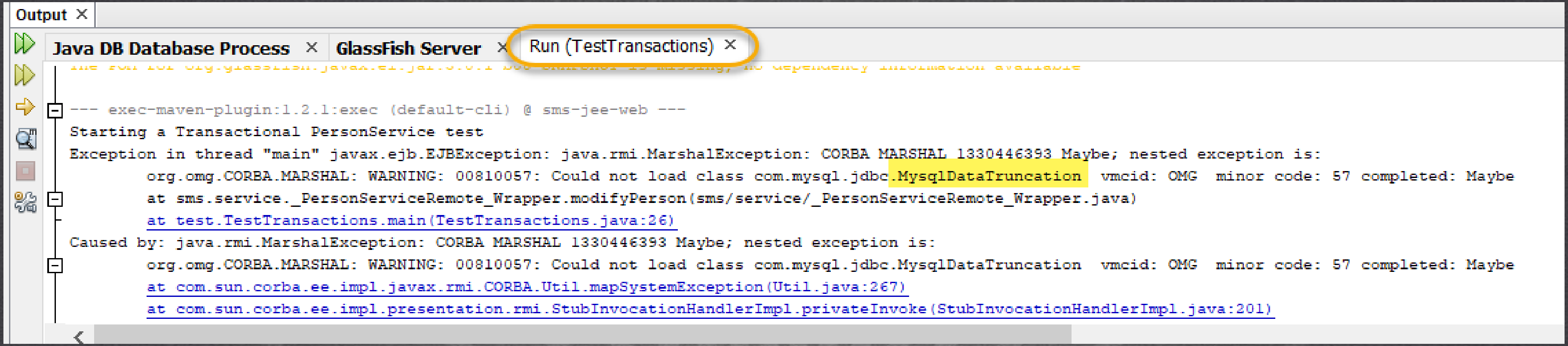
9. EXECUTE THE CLASS

The error is observed in the Glassfish log, but also in the execution console of the class TestManageTransactions.java, so both outputs must be observed to have all the detail of the error, both the error that throws the EJB in Glassfish, as well as the error received in the EJB client.

Let's see how is the execution of our program when we have an error in the EJB and the transaction is aborted:

9. EXECUTE THE CLASS

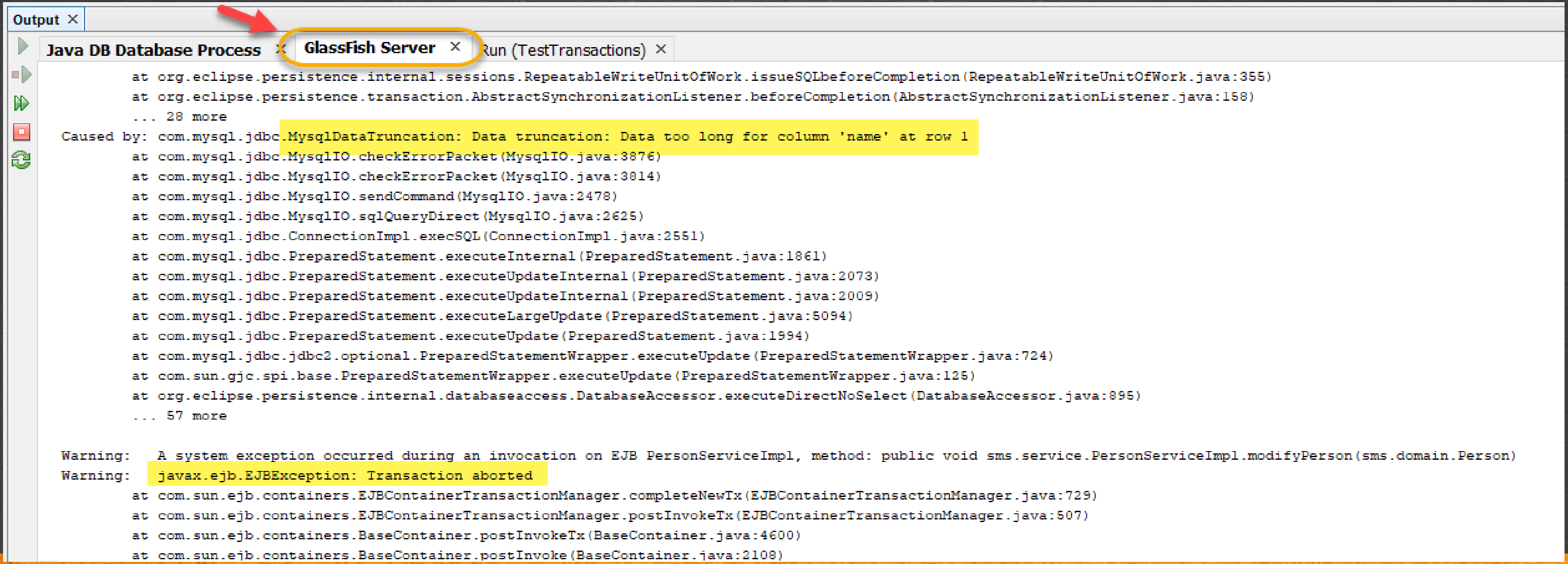
This is the result of the rollback (aborted transaction) in the Client's class:



```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ sms-jee-web ---
Starting a Transactional PersonService test
Exception in thread "main" javax.ejb.EJBException: java.rmi.MarshalException: CORBA MARSHAL 1330446393 Maybe; nested exception is:
        org.omg.CORBA.MARSHAL: WARNING: 00810057: Could not load class com.mysql.jdbc.MysqlDataTruncation  vmcid: OMG  minor code: 57 completed: Maybe
        at sms.service._PersonServiceRemote_Wrapper.modifyPerson(sms/service/_PersonServiceRemote_Wrapper.java)
        at test.TestTransactions.main(TestTransactions.java:26)
Caused by: java.rmi.MarshalException: CORBA MARSHAL 1330446393 Maybe; nested exception is:
        org.omg.CORBA.MARSHAL: WARNING: 00810057: Could not load class com.mysql.jdbc.MysqlDataTruncation  vmcid: OMG  minor code: 57 completed: Maybe
        at com.sun.corba.ee.impl.javax.rmi.CORBA.Util.mapSystemException(Util.java:267)
        at com.sun.corba.ee.impl.presentation.rmi.StubInvocationHandlerImpl.privateInvoke(StubInvocationHandlerImpl.java:201)
```

9. EXECUTE THE CLASS

This is the result of the rollback in Glassfish:



```
        at org.eclipse.persistence.internal.sessions.RepeatableWriteUnitOfWork.issueSQLbeforeCompletion(RepeatableWriteUnitOfWork.java:355)
        at org.eclipse.persistence.transaction.AbstractSynchronizationListener.beforeCompletion(AbstractSynchronizationListener.java:158)
        ... 28 more
Caused by: com.mysql.jdbc.MysqlDataTruncation: Data truncation: Data too long for column 'name' at row 1
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3876)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3814)
        at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:2478)
        at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2625)
        at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2551)
        at com.mysql.jdbc.PreparedStatement.executeInternal(PreparedStatement.java:1861)
        at com.mysql.jdbc.PreparedStatement.executeUpdateInternal(PreparedStatement.java:2073)
        at com.mysql.jdbc.PreparedStatement.executeUpdateInternal(PreparedStatement.java:2009)
        at com.mysql.jdbc.PreparedStatement.executeLargeUpdate(PreparedStatement.java:5094)
        at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:1994)
        at com.mysql.jdbc.jdbc2.optional.PreparedStatementWrapper.executeUpdate(PreparedStatementWrapper.java:724)
        at com.sun.gjc.spi.base.PreparedStatementWrapper.executeUpdate(PreparedStatementWrapper.java:125)
        at org.eclipse.persistence.internal.databaseaccess.DatabaseAccessor.executeDirectNoSelect(DatabaseAccessor.java:895)
        ... 57 more

Warning:   A system exception occurred during an invocation on EJB PersonServiceImpl, method: public void sms.service.PersonServiceImpl.modifyPerson(sms.domain.Person)
Warning:   javax.ejb.EJBException: Transaction aborted
        at com.sun.ejb.containers.EJBContainerTransactionManager.completeNewTx(EJBContainerTransactionManager.java:729)
        at com.sun.ejb.containers.EJBContainerTransactionManager.postInvokeTx(EJBContainerTransactionManager.java:507)
        at com.sun.ejb.containers.BaseContainer.postInvokeTx(BaseContainer.java:4600)
        at com.sun.ejb.containers.BaseContainer.postInvoke(BaseContainer.java:2108)
```

10. MODIFY THE TEST CLASS

We modify the class TestTransactions.java so that now we execute a test that does not throw error. This way we can observe the result when our class is correctly executed.

Let's see how the test class is:

10. MODIFY THE FILE

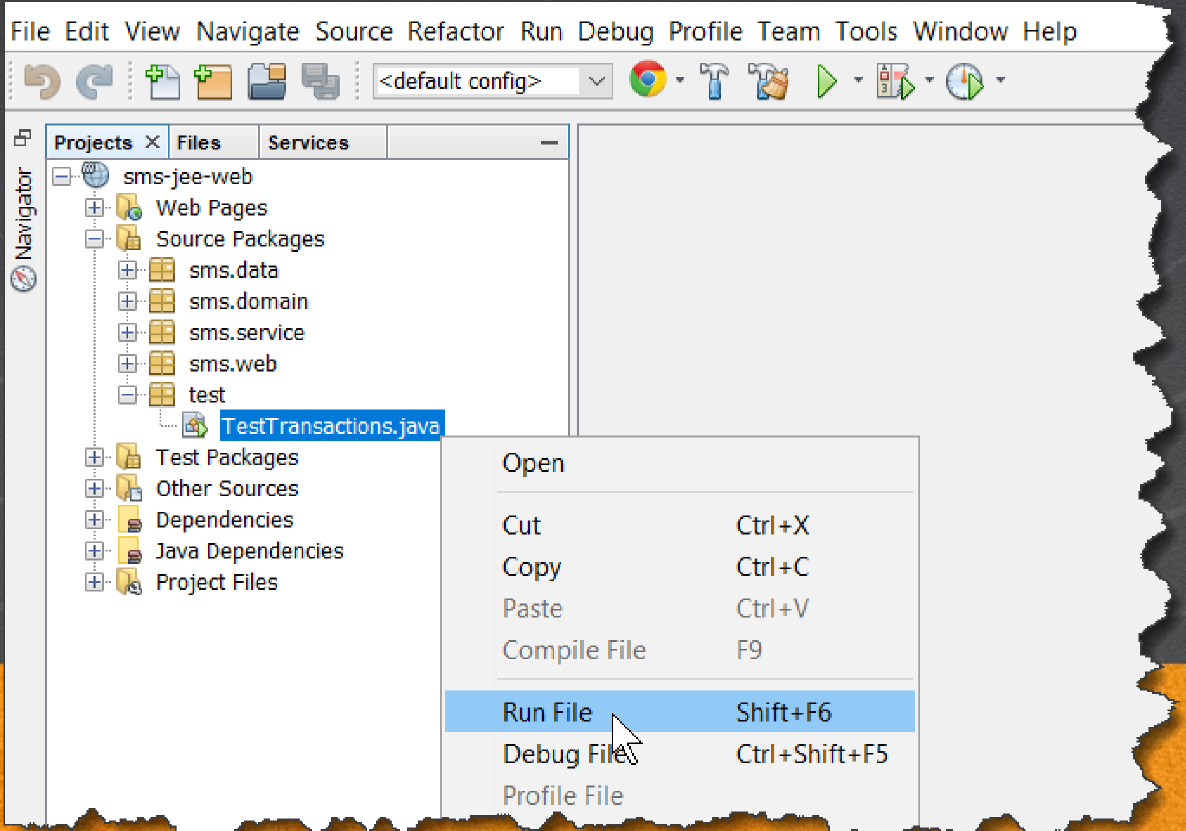
TestTransacciones.java:

Click to download

```java
package test;

import javax.naming.Context;
import javax.naming.InitialContext;
import sms.domain.Person;
import sms.service.PersonServiceRemote;

public class TestTransactions {

    public static void main(String[] args) throws Exception {
        Context jndi = new InitialContext();
        PersonServiceRemote personService =
             (PersonServiceRemote) jndi.lookup("java:global/sms-jee-web/PersonServiceImpl!sms.service.PersonServiceRemote");

        System.out.println("Starting a Transactional PersonService test");

        //We are looking for a person object
        Person person = new Person();
        person.setIdPerson(1);
        person = personService.findPerson(person);

        //We change the person
        //person.setName("Change with error...................................................................");
        person.setName("John2");

        personService.modifyPerson(person);
        System.out.println("Modified Object:" + person);
        System.out.println("End EJB PersonService test");
    }
}
```

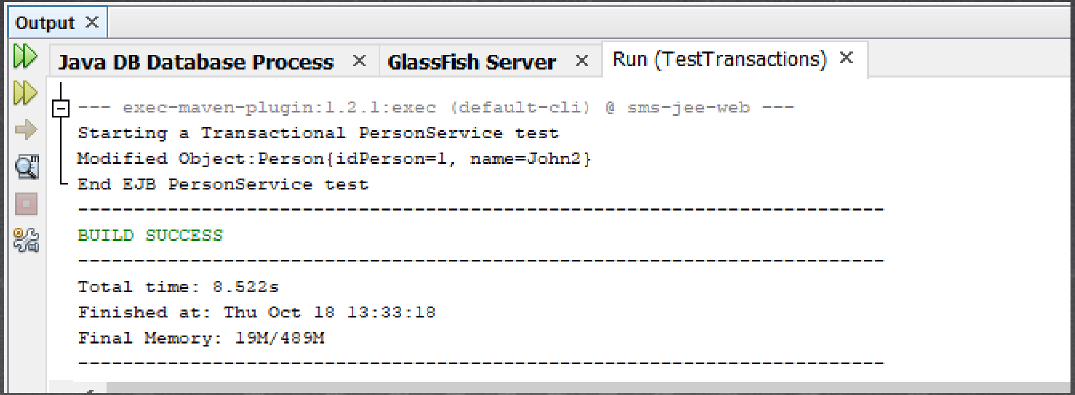
11. EXECUTE THE CLASS

Execute the application:



11. EXECUTE THE CLASS

This is the result of the transaction without error:



```
--- exec-maven-plugin:1.2.1:exec (default-cli) @ sms-jee-web ---
Starting a Transactional PersonService test
Modified Object:Person{idPerson=1, name=John2}
End EJB PersonService test
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 8.522s
Finished at: Thu Oct 18 13:33:18
Final Memory: 19M/489M
------------------------------------------------------------------------
```

EXERCISE CONCLUSION

- With this exercise we have put into practice the concept of Transactional Management.

- We implemented both an error transaction, to see how the rollback of the transaction is executed, and we also tried executing a query where there is no error in the transaction and therefore it executes and ends correctly.

ONLINE COURSE

JAVA EE JAKARTA EE

By: Eng. Ubaldo Acosta

JAVA EE COURSE

www.globalmentoring.com.mx