


JAVA DANS LA BASE DE DONNEES

ORACLE

Exemples

Table des matières

 HelloWorld.....	1
 Gestion des commandes.....	4

HelloWorld

Cf. répertoire HelloWorld

Dans cet exemple, nous avons une classe publique appelée HelloWorld qui a une seule méthode nommée direHello(). Cette méthode retourne une variable de type String.

Les étapes sont les suivantes :

1. Créer la classe Java.
2. Compiler la classe Java.
3. Télécharger la classe.
4. Publier la classe.
5. Appeler la procédure stockée.
6. Déboguer les procédures si nécessaire

1- Créer la classe Java

```
public class HelloWorld{  
    // Retourne une citation d'Oscar Wilde  
    public static String direHello(){
```

```
        return "I can resist everything except temptation.";
    }
}
```

2- Compiler la classe Java

Compiler la classe avec le compilateur Java d'Oracle (\$HOME_ORACLE\jdk\bin\javac).

```
C:\> C:\oracle11\product\11.2.0\dbhome_1\jdk\bin\javac HelloWorld.java
```

Après la compilation de cette classe, vous obtenez dans le répertoire courant le fichier HelloWorld.class

Attention : vous devez compiler votre programme avec la même version de Java que celle installée avec Oracle. La version Java installée avec Oracle se trouve dans le répertoire suivant : \$ORACLE_HOME\jdk\

Ex : C:\oracle11\product\11.2.0\dbhome_1\jdk\

Si vous compilez votre programme avec une version différente que celle du SGBD Oracle, vous pouvez obtenir l'erreur suivante :

```
ORA-29516: Aurora assertion failure: Assertion failure at eox.c:332
Uncaught exception System error: java/lang/UnsupportedClassVersionError
```

3- Télécharger la classe

Vous pouvez télécharger la classe Java dans la base de données avec l'outil loadjava qui est présent dans le répertoire bin de l'installation d'Oracle.

```
C:\> loadjava -user papyrus/afpa123@orcl HelloWorld.class
```

Note : Avec SQL Developer, vous pouvez utiliser la procédure PL/SQL dbms_java.loadjava().

Quand la méthode direHello() est appelée, le serveur utilise le « resolver » pour résoudre les classes utilisées dans la méthode comme la classe String.

Il cherche la classe dans le schéma courant. S'il ne trouve pas la classe, il cherche dans le schéma SYS où il y a toutes les classes de la JVM Oracle.

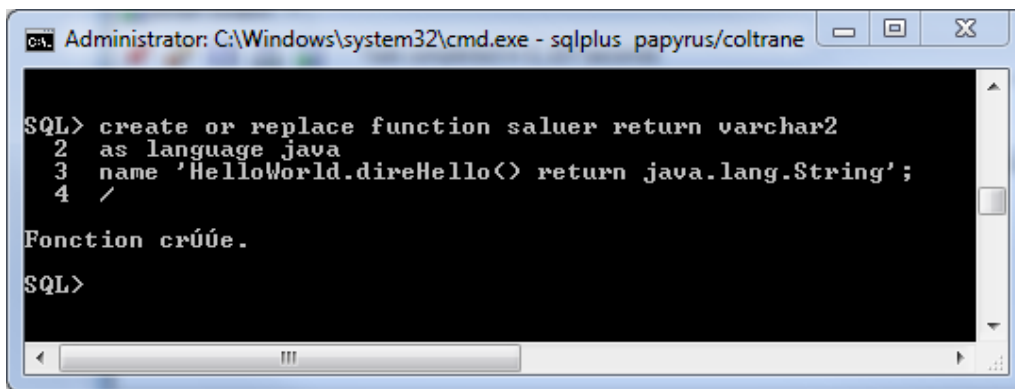
Vous pouvez spécifier un autre resolver que celui par défaut si nécessaire (Cf. documentation de loadjava).

4- Publier la classe

Pour chaque méthode que vous comptez appeler, vous devez écrire une spécification d'appel qui expose le point d'entrée de haut niveau de la méthode dans la base de données.

Avec SQL*plus, vous pouvez définir la spécification d'appel avec la commande suivante :

```
SQL> CREATE OR REPLACE FUNCTION saluer RETURN VARCHAR2
2 AS LANGUAGE JAVA
3 NAME 'HelloWorld.direHello() return java.lang.String';
4 /
```



```
C:\Windows\system32\cmd.exe - sqlplus papyrus/coltrane
SQL> create or replace function saluer return varchar2
2 as language java
3 name 'HelloWorld.direHello() return java.lang.String';
4 /
Fonction créée.
SQL>
```

5- Appeler la procédure stockée

Appel de la procédure à partir de SQL*Plus.

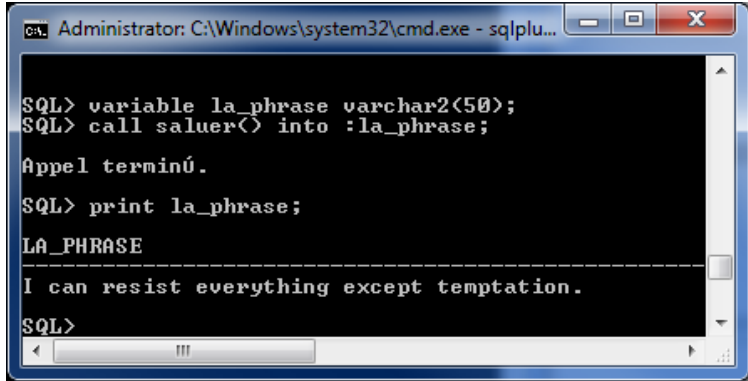
Définir la variable la_phrase pour mémoriser le retour de la procédure
SQL> VARIABLE la_phrase VARCHAR2(50);

Appeler la procédure
SQL> CALL saluer() INTO :la_phrase ;

Afficher le résultat
SQL> PRINT la_phrase;

LA_PHRASE

I can resist everything except temptation.



6- D boguer les proc dures si n cessaire

Le d bogage  tant impossible sur le serveur, il doit  tre effectu  sur le client. Apr s les modifications, vous devez t l charger   nouveau la classe avec loadjava.



Gestion des commandes

Cf. r pertoire GestionCommande

Dans cet exemple, nous avons une classe publique appel e GestionCommande qui permet de cr er des produits et des clients, de cr er/supprimer des commandes et de calculer le total des commandes. Cette classe est constitu e de 7 m thodes.

Les  tapes sont les suivantes :

1. Cr er le sch ma et les tables.
2. Cr er la classe Java nomm e GestionCommande.
3. Compiler la classe Java.
4. T l charger la classe.
5. D finir les sp cifications d'appels
6. Publier la classe.
7. Tester les proc dures stock es.

1- Cr er le sch ma et les tables

Cr ation de l'utilisateur user1

```
-- USER
CREATE USER user1 IDENTIFIED BY afpa123
DEFAULT TABLESPACE "USERS"
```

```
TEMPORARY TABLESPACE "TEMP";
```

Affectation des rôles.

```
-- ROLES
GRANT "RESOURCE" TO user1;
GRANT "CONNECT" TO user1;
GRANT "JAVA_DEPLOY" TO user1;
```

Création des tables

```
DROP TABLE CLIENT CASCADE CONSTRAINTS;
DROP TABLE COMMANDE CASCADE CONSTRAINTS;
DROP TABLE PRODUIT CASCADE CONSTRAINTS;
DROP TABLE LIGNE_COMMANDE CASCADE CONSTRAINTS;
```

```
CREATE TABLE CLIENT(
C_ID NUMBER(3) NOT NULL,
NOM VARCHAR2(30) NOT NULL,
PRENOM VARCHAR2(20) NOT NULL,
N_ADR VARCHAR2(4) NOT NULL,
RUE CHAR(20) NOT NULL,
VILLE VARCHAR2(10) NOT NULL,
PAYS VARCHAR2(20) NOT NULL,
TEL VARCHAR2(20),
PRIMARY KEY (C_ID)
);
```

```
CREATE TABLE COMMANDE (
CO_ID NUMBER(5),
C_ID NUMBER(3) REFERENCES CLIENT,
DATE_COM DATE,
DATE_LIV DATE,
PRIMARY KEY (CO_ID)
);
```

```
CREATE TABLE PRODUIT (
PRO_ID NUMBER(4),
DESCRIPTION VARCHAR2(20),
PRIX NUMBER(6,2),
```

```
PRIMARY KEY (PRO_ID)
);

CREATE TABLE LIGNE_COMMANDE (
LIGNE_ID NUMBER(2),
CO_ID NUMBER(5) REFERENCES COMMANDE,
PRO_ID NUMBER(4) REFERENCES PRODUIT,
QNT NUMBER(2),
REMISE NUMBER(4,2),
PRIMARY KEY (LIGNE_ID, CO_ID)
);
```

2- Créer la classe de gestion des commandes

Après la création du schéma, on écrit la classe Java qui va accéder aux données localement. Cette classe contient les méthodes suivantes :

- ajouterClient() : ajouter un client dans la table CLIENT.
- ajouterProduit() : ajouter un produit dans la table PRODUIT.
- entrerCommande() : ajouter une commande dans la table COMMANDE.
- ajouterLigne() : ajouter une ligne de commande dans la table LIGNE_COMMANDE.
- calculerTotal() : calculer le total des commandes.
- supprimerCommande() : supprimer une commande dans la table COMMANDE.
- printResults() : afficher les résultats. Méthode utilisée par calculerTotal().

```
import java.sql.*;

public class GestionCommande {
    public static void ajouterClient (int cid, String nom, String prenom,
    String numAdr, String rue, String ville, String pays,String tel) throws SQLException
    {
        String sql = "INSERT INTO CLIENT VALUES (?,?,,?,,?,?)";
        try
        {
            Connection conn = DriverManager.getConnection("jdbc:default:connection:");
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setInt(1, cid);
            pstmt.setString(2, nom);
            pstmt.setString(3, prenom);
            pstmt.setString(4, numAdr);
            pstmt.setString(5, rue);
```

```
pstmt.setString(6, ville);
pstmt.setString(7, pays);
pstmt.setString(8, tel);
pstmt.executeUpdate();
pstmt.close();
}
catch (SQLException e)
{
    System.err.println(e.getMessage());
}
}

public static void ajouterProduit (int proid, String desc, float prix)
                                throws SQLException
{
    String sql = "INSERT INTO PRODUIT VALUES (?, ?, ?)";
    try
    {
        Connection conn = DriverManager.getConnection("jdbc:default:connection:");
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, proid);
        pstmt.setString(2, desc);
        pstmt.setFloat(3, prix);
        pstmt.executeUpdate();
        pstmt.close();
    }
    catch (SQLException e)
    {
        System.err.println(e.getMessage());
    }
}

public static void entrerCommande (int coid, int cid, String dateCommande,
    String dateLivraison) throws SQLException
{
    String sql = "INSERT INTO COMMANDE VALUES (?, ?, ?, ?)";
    try
    {
        Connection conn = DriverManager.getConnection("jdbc:default:connection:");
        PreparedStatement pstmt = conn.prepareStatement(sql);
```

```
pstmt.setInt(1, coid);
pstmt.setInt(2, cid);
pstmt.setDate(3, java.sql.Date.valueOf(dateCommande));
pstmt.setDate(4, java.sql.Date.valueOf(dateLivraison));
pstmt.executeUpdate();
pstmt.close();
}
catch (SQLException e)
{
    System.err.println(e.getMessage());
}
}

public static void ajouterLigne (int ligneid, int coid, int proid,
                                int qnt, float remise) throws SQLException
{
    String sql = "INSERT INTO LIGNE_COMMANDE VALUES (?, ?, ?, ?, ?)";
    try
    {
        Connection conn = DriverManager.getConnection("jdbc:default:connection:");
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, ligneid);
        pstmt.setInt(2, coid);
        pstmt.setInt(3, proid);
        pstmt.setInt(4, qnt);
        pstmt.setFloat(5, remise);
        pstmt.executeUpdate();
        pstmt.close();
    }
    catch (SQLException e)
    {
        System.err.println(e.getMessage());
    }
}

public static void calculerTotal () throws SQLException
{
    String sql = "SELECT C.CO_ID, ROUND(SUM(P.PRIX * L.QNT)) AS TOTAL " +
        "FROM COMMANDE C, LIGNE_COMMANDE L, PRODUIT P " +
        "WHERE L. CO_ID = C.CO_ID AND L.PRO_ID = P. PRO_ID "
```



```
+ "GROUP BY C.CO_ID";
try
{
    Connection conn = DriverManager.getConnection("jdbc:default:connection:");
    PreparedStatement pstmt = conn.prepareStatement(sql);
    ResultSet rset = pstmt.executeQuery();
    printResults(rset);
    rset.close();
    pstmt.close();
}
catch (SQLException e)
{
    System.err.println(e.getMessage());
}
}
```

```
public static void supprimerCommande (int coid) throws SQLException
{
    String sql = "DELETE FROM LIGNE_COMMANDE WHERE CO_ID = ?";
    try
    {
        Connection conn = DriverManager.getConnection("jdbc:default:connection:");
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, coid);
        pstmt.executeUpdate();
        sql = "DELETE FROM COMMANDE WHERE CO_ID = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, coid);
        pstmt.executeUpdate();
        pstmt.close();
    }
    catch (SQLException e)
    {
        System.err.println(e.getMessage());
    }
}
```

```
static void printResults (ResultSet rset) throws SQLException
{
```

```
String buffer = "";
try
{
    ResultSetMetaData meta = rset.getMetaData();
    int cols = meta.getColumnCount(), rows = 0;
    for (int i = 1; i <= cols; i++)
    {
        int size = meta.getPrecision(i);
        String label = meta.getColumnLabel(i);
        if (label.length() > size)
            size = label.length();
        while (label.length() < size)
            label += " ";
        buffer = buffer + label + " ";
    }
    buffer = buffer + "\n";
    while (rset.next())
    {
        rows++;
        for (int i = 1; i <= cols; i++)
        {
            int size = meta.getPrecision(i);
            String label = meta.getColumnLabel(i);
            String value = rset.getString(i);
            if (label.length() > size)
                size = label.length();
            while (value.length() < size)
                value += " ";
            buffer = buffer + value + " ";
        }
        buffer = buffer + "\n";
    }
    if (rows == 0)
        buffer = "No data found!\n";
    System.out.println(buffer);
}
catch (SQLException e)
{
    System.err.println(e.getMessage());
}
```

```
}  
  
}
```

3- Compiler la classe

Compiler la classe avec le compilateur Java d'Oracle (\$HOME_ORACLE\jdk\bin\javac).

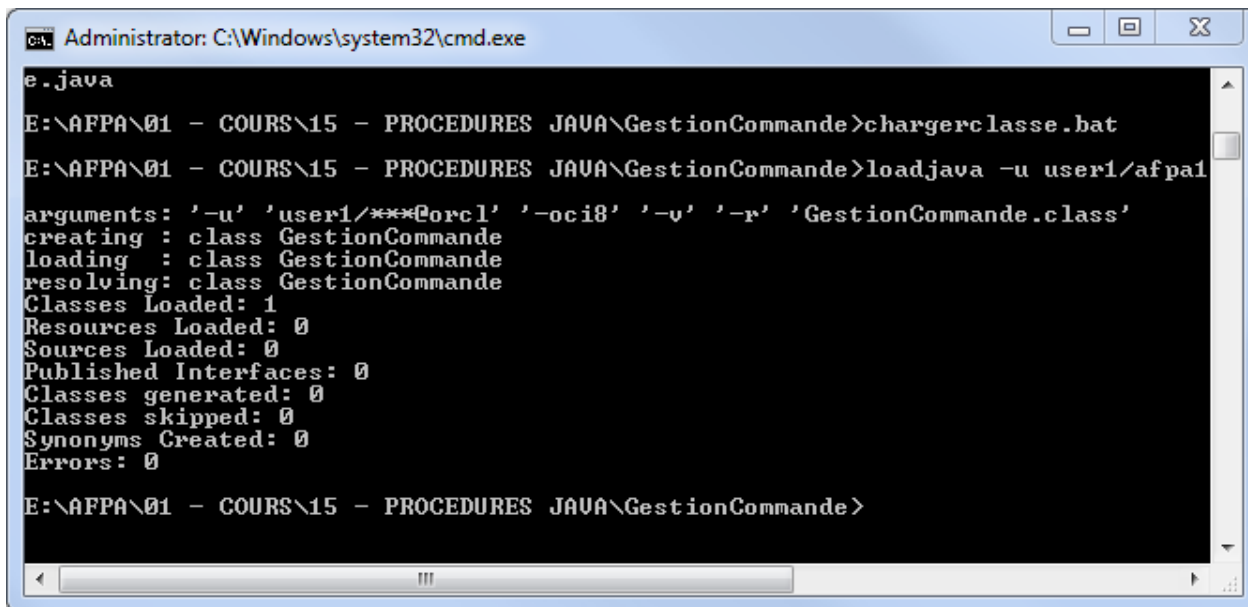
```
C:\> C:\oracle11\product\11.2.0\dbhome_1\jdk\bin\javac GestionCommande.java
```

Après la compilation de cette classe, vous obtenez le fichier GestionCommande.class.

4- Charger la classe

Charger la classe à l'aide de l'outil loadjava.

```
C:\> loadjava -u user1/afpa123@orcl -oci8 -v -r GestionCommande.class
```



```
C:\Windows\system32\cmd.exe
e.java
E:\AFPA\01 - COURS\15 - PROCEDURES JAVA\GestionCommande>chargerclasse.bat
E:\AFPA\01 - COURS\15 - PROCEDURES JAVA\GestionCommande>loadjava -u user1/afpa1
arguments: '-u' 'user1/***@orcl' '-oci8' '-v' '-r' 'GestionCommande.class'
creating : class GestionCommande
loading : class GestionCommande
resolving: class GestionCommande
Classes Loaded: 1
Resources Loaded: 0
Sources Loaded: 0
Published Interfaces: 0
Classes generated: 0
Classes skipped: 0
Synonyms Created: 0
Errors: 0
E:\AFPA\01 - COURS\15 - PROCEDURES JAVA\GestionCommande>
```

5- Définir les spécifications d'appels

Les spécifications d'appel sont définies dans le fichier nommé publierclasse.sql. Elles sont créées dans un package nommé gest_com.

```
CREATE OR REPLACE PACKAGE gest_com AS
```

```
PROCEDURE ajouter_client (num_cli NUMBER, nom VARCHAR2,prenom VARCHAR2,
nim_adr VARCHAR2, rue CHAR, ville VARCHAR2,pays VARCHAR2,num_tel VARCHAR2)
AS LANGUAGE JAVA
NAME 'GestionCommande.ajouterClient (int, java.lang.String,java.lang.String,
java.lang.String,java.lang.String,java.lang.String, java.lang.String,java.lang.String)';

PROCEDURE ajouter_produit (num_pro NUMBER, description VARCHAR2,prix NUMBER)
AS LANGUAGE JAVA
NAME 'GestionCommande.ajouterProduit(int, java.lang.String, float)';

PROCEDURE entrer_commande (num_com NUMBER, nim_cli NUMBER,date_com
VARCHAR2, date_liv VARCHAR2)
AS LANGUAGE JAVA
NAME 'GestionCommande.entrerCommande (int,int,java.lang.String,java.lang.String)';

PROCEDURE ajouter_ligne (num_ligne NUMBER, num_com NUMBER,num_pro NUMBER,
quantite NUMBER, remise NUMBER)
AS LANGUAGE JAVA
NAME 'GestionCommande.ajouterLigne (int,int,int,int,float)';

PROCEDURE calculer_total
AS LANGUAGE JAVA
NAME 'GestionCommande.calculerTotal()';

PROCEDURE delete_com(num_com NUMBER)
AS LANGUAGE JAVA
NAME 'GestionCommande.supprimerCommande(int)';

END;
```

6- Publier la classe

Publier la classe en exécutant le fichier publierclasse.sql sous SQL*Plus.

```
C:\> sqlplus user1/afpa123
SQL> @publierclasse.sql
2 /
```

7- Tester les procédures

Syntaxe : nom_package.nom_procedure (paramètres).

-- Insertion des produits

BEGIN

```
gest_com.ajouter_produit(2010, 'camshaft', 245.00);
gest_com.ajouter_produit(2011, 'connecting rod', 122.50);
gest_com.ajouter_produit(2012, 'crankshaft', 388.25);
gest_com.ajouter_produit(2013, 'cylinder head', 201.75);
gest_com.ajouter_produit(2014, 'cylinder sleeve', 73.50);
gest_com.ajouter_produit(2015, 'engine bearing', 43.85);
gest_com.ajouter_produit(2016, 'flywheel', 155.00);
gest_com.ajouter_produit(2017, 'freeze plug', 17.95);
gest_com.ajouter_produit(2018, 'head gasket', 36.75);
gest_com.ajouter_produit(2019, 'lifter', 96.25);
gest_com.ajouter_produit(2020, 'oil pump', 207.95);
gest_com.ajouter_produit(2021, 'piston', 137.75);
gest_com.ajouter_produit(2022, 'piston ring', 21.35);
gest_com.ajouter_produit(2023, 'pushrod', 110.00);
gest_com.ajouter_produit(2024, 'rocker arm', 186.50);
gest_com.ajouter_produit(2025, 'valve', 68.50);
gest_com.ajouter_produit(2026, 'valve spring', 13.25);
gest_com.ajouter_produit(2027, 'water pump', 144.50);
```

END;

-- Insertion des clients

BEGIN

```
gest_com.ajouter_client(101, 'Martin', 'Jacques', '112', 'Metz', 'Toulouse',
'France', '0651875653');
gest_com.ajouter_client(102, 'Marley', 'Bob', '12', 'Metz', 'Toulouse', 'France', '0651875653');
gest_com.ajouter_client(103, 'Mc Ferrin', 'Boby', '2', 'Metz', 'Paris', 'France', '0651875653');
gest_com.ajouter_client(104, 'No one', 'No Body', '100', 'noland', 'nowhere',
'somewhere', '0000000000');
```

END;

-- Insertion des commandes et des lignes de commande

BEGIN

```
gest_com.entrer_commande(30501, 103, '2012-09-14', '2012-09-21');
gest_com.ajouter_ligne(01, 30501, 2011, 5, 0.02);
gest_com.ajouter_ligne(02, 30501, 2018, 25, 0.10);
gest_com.ajouter_ligne(03, 30501, 2026, 10, 0.05);
```

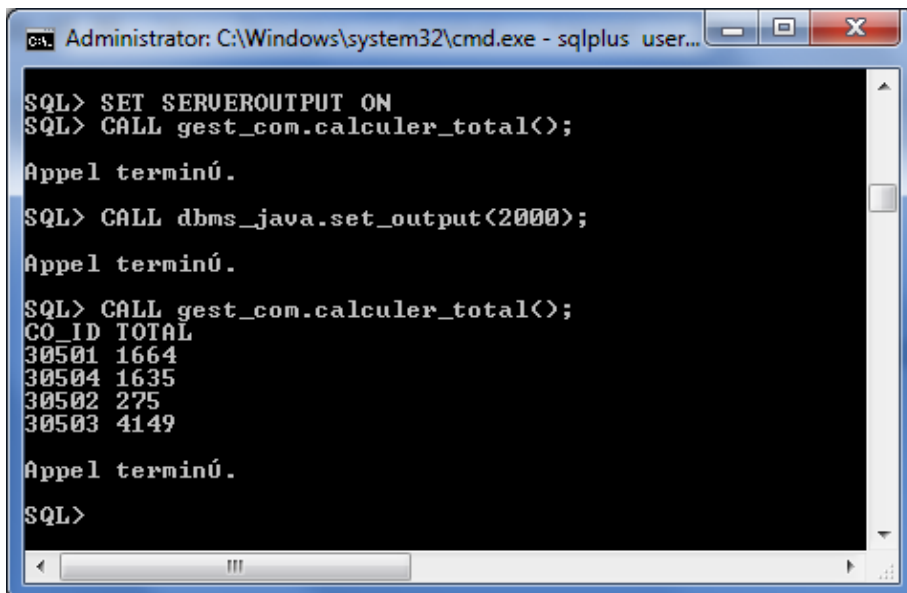
```
gest_com.entrer_commande(30502, 102, '2012-09-15', '2012-09-22');
gest_com.ajouter_ligne(01, 30502, 2013, 1, 0.00);
gest_com.ajouter_ligne(02, 30502, 2014, 1, 0.00);

gest_com.entrer_commande(30503, 104, '2012-09-15', '2012-09-23');
gest_com.ajouter_ligne(01, 30503, 2020, 5, 0.02);
gest_com.ajouter_ligne(02, 30503, 2027, 5, 0.02);
gest_com.ajouter_ligne(03, 30503, 2021, 15, 0.05);
gest_com.ajouter_ligne(04, 30503, 2022, 15, 0.05);

gest_com.entrer_commande(30504, 101, '2012-09-16', '2012-09-23');
gest_com.ajouter_ligne(01, 30504, 2025, 20, 0.10);
gest_com.ajouter_ligne(02, 30504, 2026, 20, 0.10);
END;
```

Appel de la méthode `calculer_total()` à partir de SQL*Plus.

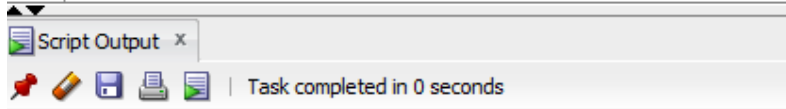
```
C:\> sqlplus user1/afpa123
SQL> SET SERVEROUTPUT ON
SQL> CALL dbms_java.set_output(2000);
SQL> CALL gest_com.calculer_total();
```

A screenshot of a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe - sqlplus user...". The window shows the execution of SQL*Plus commands. The output includes "Appel termin ." (Call terminated.) after the first two commands, and a table of results after the third command. The table has two columns: "CO_ID" and "TOTAL". The data rows are: 30501 1664, 30504 1635, 30502 275, and 30503 4149. The prompt returns to "SQL>" after the final command.

```
SQL> SET SERVEROUTPUT ON
SQL> CALL gest_com.calculer_total();
Appel termin .
SQL> CALL dbms_java.set_output(2000);
Appel termin .
SQL> CALL gest_com.calculer_total();
CO_ID TOTAL
30501 1664
30504 1635
30502 275
30503 4149
Appel termin .
SQL>
```

Appel de la méthode `calculer_total()` à partir de SQL Developer.

```
23  
24 SET SERVEROUTPUT ON  
25 CALL dbms_java.set_output(2000);  
26 CALL gest_com.calculer_total();  
27  
28
```



```
dbms_java.set_output 2000) succeeded.  
gest_com.calculer_total ) succeeded.  
CO_ID TOTAL  
30501 1664  
30504 1635  
30502 275  
30503 4149
```