Batch Processing

Four types of Drivers

Steps to write a JDBC Program

Step 1: Loading the class

Class.forName(“oracle.jdbc.driver.OracleDriver”);

Set ojdbc5.jar to class path to load the driver successfully.

If it failed, then it throws ClassNotFoundException

Step 2:

Establish the connection

DriverManager.getConnection(“URL”,”username”,”password”)

getConnection returns Connection Object

if failed to connect with oracle, throws SQL Exception

Step 3:

Prepare the statement Object

THREE different ways to prepare statement Object

1. Statement
2. PreparedStatement
3. CallableStatement

Recommended is using PreparedStatement for executing SQL Queries

CallableStatement is used to execute procedures or functions

Step 4:

Execute the Statement.

THREE different methods to execute

1. execute() 🡪 generic method used to execute any type of SQL Query

return is Boolean

true 🡪 received ResultSet

false 🡪 No result set

1. executeUpdate() 🡪 used to execute DML statements and it returns Integer value.

Returned value shows the number of effected records in Table

1. executeQuery() 🡪 used to execute Select statement and returns Result set

Step 5: processing the Result set

Note: this step is used only for select query

Step 6: close all the connections in reverse order

What is batch processing?

Sending more than query from JDBC to Oracle for processing or executing is called Batch Processing

We can use Statement as well as PreparedStatement to implement batch processing

JDBC program to execute Multiple queries using Batch processing and statement interface

**package** com.sssit.product.controller;

**import** java.sql.Connection;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** com.sssit.product.util.JDBCConnection;

**public** **class** BatchProcessingUsingStatement {

**public** **static** **void** main(String[] args) {

Connection con = **null**;

Statement stmt = **null**;

**try**

{

con = JDBCConnection.*getConnection*();

con.setAutoCommit(**false**);

stmt = con.createStatement();

**final** String SQL\_QUERY\_1 = "insert into product values(119,'Test-19',119)";

**final** String SQL\_QUERY\_2 = "insert into product values(120,'Test-20',120)";

**final** String SQL\_QUERY\_3 = "insert into product values(121,'Test-21',121)";

stmt.addBatch(SQL\_QUERY\_1);

stmt.addBatch(SQL\_QUERY\_2);

stmt.addBatch(SQL\_QUERY\_3);

stmt.executeBatch();

con.commit();

System.***out***.println("Executed successfully....");

}

**catch**(Exception e) {

System.***out***.println("Failed to execute.....");

**try** {

con.rollback();

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

**finally** {

**try**

{

stmt.close();

con.close();

}

**catch**(SQLException e) {

System.***out***.println("Failed to close the connections...");

}

}

}

}

JDBC program to execute Multiple queries using Batch processing and Prepared statement interface

**package** com.sssit.product.controller;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** com.sssit.product.util.JDBCConnection;

**public** **class** BatchProcessingUsingPrepared {

**public** **static** **void** main(String[] args) {

Connection con = **null**;

PreparedStatement stmt = **null**;

**try**

{

con = JDBCConnection.*getConnection*();

con.setAutoCommit(**false**);

**final** String SQL\_QUERY\_1 = "insert into product values(?,?,?)";

stmt = con.prepareStatement(SQL\_QUERY\_1);

stmt.setInt(1, 122);

stmt.setString(2, "Test-22");

stmt.setDouble(3, 122);

stmt.addBatch();

stmt.setInt(1, 123);

stmt.setString(2, "Test-23");

stmt.setDouble(3, 123);

stmt.addBatch();

stmt.executeBatch();

con.commit();

System.***out***.println("Executed successfully....");

}

**catch**(Exception e) {

System.***out***.println("Failed to execute.....");

**try** {

con.rollback();

} **catch** (SQLException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

**finally** {

**try**

{

stmt.close();

con.close();

}

**catch**(SQLException e) {

System.***out***.println("Failed to close the connections...");

}

}

}

}