Dt : 13/5/2022

**imp*

Batch Processing in JDBC:

=>The process of collecting multiple queries as batch and executing at-a-time is known as Batch Processing.

=>The following are the methods used in Batch Processing:

(a)addBatch()

(b)executeBatch()

(c)clearBatch()

(a)addBatch():

=>This addBatch() method is used to add query to the batch.

Method Signature:

```
public abstract void addBatch(java.lang.String)
                        throws java.sql.SQLException;
```

(b)executeBatch():

=>This executeBatch() method is used to execute the queries from the batch at-a-time.

Method Signature:

```
public abstract int[] executeBatch() throws java.sql.SQLException;
```

(c)clearBatch():

=>This clearBatch() method is used to clear all the queries from

the batch and deletes the batch.

Method Signature:

public abstract void clearBatch()throws java.sql.SQLException;

Program-1 : Batch Processing using 'Statement'

DBCon13.java

```
package test;
import java.sql.*;
public class DBCon13 {
     public static void main(String[] args) {
          try {
            Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
            Statement stm = con.createStatement();

            stm.addBatch
            ("insert into Product45 values('B123','ER',123,12)");
            stm.addBatch
            ("insert into Bank45
values(456,'Alex',234,'savings')");
            stm.addBatch
            ("insert into CustDetails45
values(456,'SRN','a@..',7878)");

            int k[] = stm.executeBatch();
            for(int i=0;i<k.length;i++)
            {
                System.out.println("Data updated...");
            }//end of loop
            stm.clearBatch();
            con.close();
          }catch(Exception e) {e.printStackTrace();}
     }
}
```

Note:

=>Batch Processing using 'Statement' we can update multiple

DataBase tables at-a-time

==========

Program-2 : Batch Processing using 'PreparedStatement'

DBCon14.java

```
package test;
import java.sql.*;
public class DBCon14 {
     public static void main(String[] args) {
      try {
       Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
       PreparedStatement ps = con.prepareStatement
       ("insert into Product45 values(?,?,?,?)");

       ps.setString(1,"C111");
       ps.setString(2,"TRY");
       ps.setFloat(3,234);
       ps.setInt(4,12);
       ps.addBatch();

       ps.setString(1,"C222");
       ps.setString(2,"Catch");
       ps.setFloat(3,244);
       ps.setInt(4,11);
       ps.addBatch();

       int k[] = ps.executeBatch();
       for(int i=0;i<k.length;i++)
       {
            System.out.println("Data Updated...");
       }
       ps.clearBatch();
       con.close();
      }catch(Exception e) {e.printStackTrace();}
     }
}
```

Note:

=>Batch processing using 'PreparedStatement' we can update

multiple records into same DataBase table.

==========

faq:

wt is the Advantage of Batch processing?

=>In Batch processing the execution control is transferred to DataBase only once and executes all the queries from the batch at-a-time,in this process the execution time is saved and generate HighPerformance of an application.

==========

faq:

wt is the diff b/w

(i)Batch Processing

(ii)Procedures

=>using Batch Processing we can execute only Non-Select queries on DataBase product,which means it is Batch Update processing.

=>Using Procedures we can execute both select and Non-select queries

==========

*imp

Types of ResultSet objects:

=>Based on control over the cursor,the ResultSet objects are categorized into two types:

1.Non-Scrollable ResultSet object

2.Scrollable ResultSet object

1.Non-Scrollable ResultSet object:

=>In Non-Scrollable ResultSet Objects the cursor is moved only in one direction,which means the cursor moves from top-of-the-table-data to Bottom-of-table-data.

Ex:

above programs related to ResultSet

**imp*

2.Scrollable ResultSet object:

=>In Scrollable ResultSet object the cursor can be moved in two directions,which means down the table data and upward the table data.

Syntax for Creating Scrollable ResultSet object:

```
Statement stm = con.createStatement(type,mode);
PreparedStatement ps = con.prepareStatement("query-S",type,mode);
```

Type:

public static final int TYPE_FORWARD_ONLY=1003

public static final int TYPE_SCROLL_INSENSITIVE=1004

public static final int TYPE_SCROLL_SENSITIVE=1005

Mode:

public static final int CONCUR_READ_ONLY=1007

public static final int CONCUR_UPDATABLE=1008

Note:

'type' specifies the direction of the cursor and 'mode' specifies the

action to be performed(read or update).

The following are some Methods used to control cursor on Scrollable

ResultSet object:

afterLast() =>Moves the cursor after the Last row

beforeFirst() =>Moves the cursor before the First row

previous() =>Moves the Cursor in the BackWard Direction

next() => Moves the cursor in the ForWard Direction

first() => Moves the cursor to the First row

last() => Moves the cursor to the Last row

absolute(int)=>Moves the cursor to the specified row number

relative(int)=>Moves the cursor from the current position

in forward or backward direction by increment or decrement.

Dt : 16/5/2022

Ex_Program:(Demonstrating Scrollable ResultSet object)

Program : DBCon15.java

```
package test;
import java.sql.*;;
public class DBCon15 {
     public static void main(String[] args) {
         try {
           Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
           Statement stm = con.createStatement
                     (ResultSet.TYPE_SCROLL_INSENSITIVE,
                                ResultSet.CONCUR_READ_ONLY);
           ResultSet rs = stm.executeQuery("select * from
Product45");
           System.out.println("====Display Product in
reverse====");
           rs.afterLast();//Cursor pointing after last row
           while(rs.previous()) {

     System.out.println(rs.getString(1)+"\t"+rs.getString(2)+
                          "\t"+rs.getFloat(3)+"\t"+rs.getInt(4));
           }//end of loop
           con.close();;
         }catch(Exception e) {e.printStackTrace();}
     }
}
```

o/p:

====Display Product in reverse====

C222 Catch 244.0 11

```
C111   TRY    234.0  12

A545   ER     123.0  12

A100   CDR    1200.0 45

A104   FDD    700.0  3

A105   CDR    1300.0 13

A222   KBB    1100.0 10

B123   ER     123.0  12
```

Program : DBCon16.java

```
package test;
import java.sql.*;
public class DBCon16 {
     public static void main(String[] args) {
        try {
          Connection con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","manager");
          PreparedStatement ps = con.prepareStatement
                    ("select * from Product45",
                              ResultSet.TYPE_SCROLL_INSENSITIVE,
                              ResultSet.CONCUR_READ_ONLY);
          ResultSet rs = ps.executeQuery();
          System.out.println("====Display Product in
reverse====");
          rs.afterLast();//Cursor pointing after last row
          while(rs.previous()) {

     System.out.println(rs.getString(1)+"\t"+rs.getString(2)+
                         "\t"+rs.getFloat(3)+"\t"+rs.getInt(4));
          }//end of loop
          System.out.println("====Display Last row====");
          rs.last();//Cursor pointing to last row

     System.out.println(rs.getString(1)+"\t"+rs.getString(2)+
                    "\t"+rs.getFloat(3)+"\t"+rs.getInt(4));
          System.out.println("====Display First row====");
          rs.first();//Cursor pointing to first row

     System.out.println(rs.getString(1)+"\t"+rs.getString(2)+
```

```
                    "\t"+rs.getFloat(3)+"\t"+rs.getInt(4));
            System.out.println("====Display row number 4====");
            rs.absolute(4);//Cursor pointing to 4th row

        System.out.println(rs.getString(1)+"\t"+rs.getString(2)+
                    "\t"+rs.getFloat(3)+"\t"+rs.getInt(4));
            System.out.println("====Display relative(+1)====");
            rs.relative(+1);//Cursor  moved forward by one row

        System.out.println(rs.getString(1)+"\t"+rs.getString(2)+
                    "\t"+rs.getFloat(3)+"\t"+rs.getInt(4));
            con.close();
          }catch(Exception e) {e.printStackTrace();}
        }
}
```

o/p:

====Display Product in reverse====

C222 Catch 244.0 11

C111 TRY 234.0 12

A545 ER 123.0 12

A100 CDR 1200.0 45

A104 FDD 700.0 3

A105 CDR 1300.0 13

A222 KBB 1100.0 10

B123 ER 123.0 12

====Display Last row====

C222 Catch 244.0 11

====Display First row====

B123 ER 123.0 12

====Display row number 4====

A104 FDD 700.0 3

====Display relative(+1)====

A100 CDR 1200.0 45

==========

**imp*

define 'RowSet'?

=>RowSet object will encapsulate the rows generated from ResultSets or any other data sources.

=>RowSet is an interface from javax.sql package and which is extended from 'java.sql.ResultSet' interface.

=>The following are the interfaces extended from RowSet:

(a)JDBCRowSet

(b)CachedRowSet

=>WebRowSet

(i)FilteredRowSet

(ii)JoinRowSet

Hierarchy of RowSet:

---

faq:

wt is the diff b/w

(i)JdbcRowSet

(ii)CachedRowSet

(i)JdbcRowSet:

=>JdbcRowSet will hold ResultSet and connection to DataBase is active.

(ii)CachedRowSet:

=>cachedRowSet will hold ResultSet,but connection to DataBase is Dis-Connected automatically.

Note:

=>WebRowSet is used to transfer the data from one layer to another layer in Application architectures.

=>FilteredRowSet will hold the data retrieved based on condition.

=>JoinRowSet will hold the data joined from more than one ResultSet

==========

**imp*

define RowSetFactory?

=>RowSetFactory is an interface from 'javax.sql.rowset' package and which provide the following methods to create the implementations of RowSet:

(a)createJdbcRowSet()

(b)createCachedRowSet()

(c)createWebRowSet()

(d)createFilteredRowSet()

(e)createJoinRowSet()

(a)createJdbcRowSet():

=>This method is used to create the implementation object of 'JdbcRowSet'.

Method Signature:

```
public abstract javax.sql.rowset.JdbcRowSet createJdbcRowSet()
                        throws java.sql.SQLException;
```

(b)createCachedRowSet():

=>This method is used to create the implementation object of 'CachedRowSet'.

Method Signature:

```
public abstract javax.sql.rowset.CachedRowSet createCachedRowSet()
                    throws java.sql.SQLException;
```

(c)createWebRowSet():

=>This method is used to create the implementation object of 'WebRowSet'.

Method Signature:

```
public abstract javax.sql.rowset.WebRowSet createWebRowSet()
                    throws java.sql.SQLException;
```

(d)createFilteredRowSet():

=>This method is used to create the implementation object of 'FilteredRowSet'.

Method Signature:

```
public abstract javax.sql.rowset.FilteredRowSet createFilteredRowSet()
        throws java.sql.SQLException;
```

(e)createJoinRowSet():

=>This method is used to create the implementation object of 'JoinRowSet'.

Method Signature:

```
public abstract javax.sql.rowset.JoinRowSet createJoinRowSet()
        throws java.sql.SQLException;
```

Note:

=>we use the following methods from 'javax.sql.rowset.RowSetProvider' class to create the implementation object of 'RowSetFactory' interface.

```
public static javax.sql.rowset.RowSetFactory newFactory()
    throws java.sql.SQLException;
public static javax.sql.rowset.RowSetFactory newFactory
    (java.lang.String, java.lang.ClassLoader)
    throws java.sql.SQLException;
```

==========