ICON OF JAVA

ADV. JAVA

JDBC

6.ResultSet and ResultSet Types

Mr. Nagoor Babu M.Tech

[ICON of JAVA]
(Sun certified & Realtime Expert)
Ex. HCL Employee
Trained Lakhs of Students
for last 14 years across INDIA

6.ResultSet and ResultSet Types

1)Read only ResultSet

2)Updatable ResultSet

3)Forward only ResultSet

4)Scrollable ResultSets

1)Scroll Sensitive ResultSet

2)Scroll Insensitive ResultSet

ResultSet Types:

In jdbc applications ResultSets can be divided into two types:

-->As per the ResultSet concurrency there are two types of ResultSets.

1)Read only ResultSet

It is a ResultSet object,it will allow the users to read the data only.

To represent this ResultSet object ResultSet interface has provided the following constant

public static final int CONCUR_READ_ONLY

2)Updatable ResultSet:

It is a ResultSet object,it will allow the users to perform updations on its content.

To represent this resultset object ResultSet interface has provided the following constant.

public static final int CONCUR_UPDATABLE

-->As per the ResultSet cursor movement there are two types of ResultSets

1)Forward only ResultSet:

It is ResultSet object,it will allow the users to iterate the data in forward directiononly.

-->To represent this ResultSet,ResultSet interface has provided the following constant

public static final int TYEP_FORWARD_ONLY

2)Scrollable ResultSet:

These are the Resultset objects,which will allow the users to iterate the data in both

forward and backward directions.

There are two types of Scrollable Resultsets:
1)Scroll sensitive ResultSet
2)Scroll Insensitive ResultSet

Q)What are the differences betwwn Scroll Sensitive ResultSet and Scroll Insensitive ResultSet?

Scroll sensitive ResultSet is a Scrollable resultset object,which will allow the later Database updations.

-->To refer this ResultSet,ResultSet interface has provided the following constant

public static final int TYPE_SCROLL_SENSITIVE

-->Scroll Insensitive ResultSet is scrollable Resultset object,which will not allow the later database updations after creation.

-->To represent this ResultSet,ResultInterface has provided the following constant

public static final int TYPE_SCROLL_INSENDITIVE

-->The default ResultSet type in Jdbc applications is Forward only and Read only.

-->In Jdbc applications if we want to specify a particular type to the ResultSet object then we have to provide the above specified ResultSet constants at the time of creating Statement object,for this we have to use the following method

public Statement createStatement(int forwardonly /scrollsensitive/scrollinsensitive,int Readonly/updatable)

Ex: Statement
st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);

-->In jdbc applications by using scrollable ResultSet we are able to retrieve the data in both

forward direction and backward direction.

-->To retrieve the data in forward direction for each and every record we have to check whether the next record is available or not,if it is available we have to move resultset cursor to the next record position.when we refer a particular record then we have to retrieve the data from the respective columns.To achieve this we have to use the following piece of code:

```
while(rs.next())
{
System.out.println(rs.getInt(1));
System.out.println(rs.getString(2));

}
```

-->If we want to retrieve the data in Backward direction then for each and every record we have to check whether the previous record is available or not from the resultset cursor position,if is available then we have to move resultset cursor to the previous record.To achieve this we have to use the following methods:

public boolean previous()

-->After moving the resultset cursor to particular record then we have to retrieve the data from the corresponding columns for this we have to use the following methods

public xxx getxxx(int field_NUM)
public xxx getxxx(String field_Name)

where xxx may be byte,short,int.....

```
Ex: while(rs.previous())
{
System.out.println(rs.getInt(1));
System.out.println(rs.getString(2));
System.out.println(rs.getFloat(3));

}
```

JdbcApp19:

```
package com.durgasoft;
    import java.sql.Connection;
    import java.sql.DriverManager;
    import java.sql.ResultSet;
    import java.sql.Statement;

public class JdbcApp19 {

     public static void main(String[] args)throws Exception {
            Class.forName("oracle.jdbc.OracleDriver");
            Connection con=DriverManager.getConnection
            ("jdbc:oracle:oci8:@xe", "system", "durga");
            Statement st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
            ResultSet.CONCUR_UPDATABLE );
            ResultSet rs=st.executeQuery("select * from emp1");
            System.out.println("Data In Forward Direction");
            System.out.println("ENO\tENAME\tESAL\tEADDR\t");
            System.out.println("----------------------------------");
            while(rs.next()){
            System.out.println(rs.getInt(1)+"\t"+rs.getString(2)+"\t"
            +rs.getFloat(3)+"\t"+rs.getString(4));
            }
            System.out.println("Data In Backward Direction");
            System.out.println("ENO\tENAME\tESAL\tEADDR");
            System.out.println("-------------------------------");
            while(rs.previous()){

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

JdbcApp20:

```
package com.durgasoft;
 import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
```

```
import java.sql.Statement;

public class JdbcApp20 {

    public static void main(String[] args)throws Exception {
            Class.forName("oracle.jdbc.OracleDriver");
            Connection con=DriverManager.getConnection
            ("jdbc:oracle:oci8:@xe", "system", "durga");
            Statement st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
            ResultSet.CONCUR_UPDATABLE );
            ResultSet rs=st.executeQuery("select * from emp1");
            rs.afterLast();
            rs.previous();
            System.out.println(rs.getInt(1));
            rs.beforeFirst();
            rs.next();
            System.out.println(rs.getInt(1));
            rs.last();
            System.out.println(rs.getInt(1));
            rs.first();
            System.out.println(rs.getInt(1));
            rs.absolute(3);
            System.out.println(rs.getInt(1));
            rs.absolute(-3);
            System.out.println(rs.getInt(1));
            rs.first();
            rs.relative(2);
            System.out.println(rs.getInt(1));
            rs.last();
            rs.relative(-2);
            System.out.println(rs.getInt(1));
    }

}
```

JdbcApp21:

```
package com.durgasoft;

import java.awt.Button;
import java.awt.Color;
import java.awt.Font;
```

```
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class PlayerFrame extends Frame implements ActionListener {
    Button b1,b2,b3,b4;
    String label;
    EmployeeTo eto;
    EmployeeService es;
    public PlayerFrame() {
        this.setVisible(true);
        this.setSize(500, 500);
        this.setTitle("Player Frame");
        this.setBackground(Color.green);
        this.setLayout(null);
        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we){
                System.exit(0);
            }
        });

        b1=new Button("First");
        b2=new Button("Next");
        b3=new Button("Previous");
        b4=new Button("Last");

        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        b4.addActionListener(this);

        Font f=new Font("arial",Font.BOLD,20);
        b1.setFont(f);
        b2.setFont(f);
        b3.setFont(f);
        b4.setFont(f);

        b1.setBounds(50, 400, 100, 30);
        b2.setBounds(160, 400, 100, 30);
        b3.setBounds(270, 400, 100, 30);
        b4.setBounds(380, 400, 100, 30);
        this.add(b1);
        this.add(b2);
```

```
                this.add(b3);
                this.add(b4);
                es=new EmployeeService();
        }
        public void actionPerformed(ActionEvent ae){
                label=ae.getActionCommand();
                eto=es.getEmployee(label);
                repaint();
        }
        public void paint(Graphics g){
                Font f=new Font("arial",Font.BOLD,30);
                g.setFont(f);
                String msg=es.getMsg();
                if(msg.equals("")){
                        g.drawString("Employee Number   :"+eto.getEno(), 50,100);
                        g.drawString("Employee Name     :"+eto.getEname(), 50,150);
                        g.drawString("Employee Salary   :"+eto.getEsal(), 50,200);
                        g.drawString("Employee Address  :"+eto.getEaddr(), 50,250);
                }else{
                        g.drawString(msg, 50,300);
                }
        }
}



package com.durgasoft;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class EmployeeService {
    Connection con;
    Statement st;
    ResultSet rs;
    EmployeeTo eto;
    String msg="";
    boolean b=false;


    public EmployeeService() {
            try {
                    Class.forName("oracle.jdbc.OracleDriver");
                    con=DriverManager.getConnection
```

```
                ("jdbc:oracle:thin:@localhost:1521:xe", "system","durga");
                st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                ResultSet.CONCUR_UPDATABLE);
                rs=st.executeQuery("select * from emp1");
        } catch (Exception e) {
                e.printStackTrace();
        }
}
public EmployeeTo getEmployee(String label){
        try {
                if(label.equals("First")){
                        rs.first();
                        msg="";
                }
                if(label.equals("Next")){
                        b=rs.next();
                        if(b==false){
                                msg="No More Records In Forward Direction";
                        }else{
                                msg="";
                        }
                }
                if(label.equals("Previous")){
                        b=rs.previous();
                        if(b==false){
                                msg="No More Records In Backward Direction";
                        }else{
                                msg="";
                        }
                }
                if(label.equals("Last")){
                        rs.last();
                        msg="";
                }
                eto=new EmployeeTo();
                eto.setEno(rs.getInt(1));
                eto.setEname(rs.getString(2));
                eto.setEsal(rs.getFloat(3));
                eto.setEaddr(rs.getString(4));
        } catch (Exception e) {
                e.printStackTrace();
        }
        return eto;
}
public String getMsg(){
        return msg;
```

```
	}
}
```

```
package com.durgasoft;

public class EmployeeTo {
	private int eno;
	private String ename;
	private float esal;
	private String eaddr;
	public int getEno() {
		return eno;
	}
	public void setEno(int eno) {
		this.eno = eno;
	}
	public String getEname() {
		return ename;
	}
	public void setEname(String ename) {
		this.ename = ename;
	}
	public float getEsal() {
		return esal;
	}
	public void setEsal(float esal) {
		this.esal = esal;
	}
	public String getEaddr() {
		return eaddr;
	}
	public void setEaddr(String eaddr) {
		this.eaddr = eaddr;
	}
}
```

```
package com.durgasoft;


public class JdbcApp21 {

	public static void main(String[] args) {
		PlayerFrame pf=new PlayerFrame();
	}

}
```

Scroll Sensitive ResultSet:

```
JdbcApp22
package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class JdbcApp22 {
   public static void main(String[] args)throws Exception {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      Connection con=DriverManager.getConnection("jdbc:odbc:nag","system","durga");
       Statement st=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
      ResultSet.CONCUR_UPDATABLE);
      ResultSet rs=st.executeQuery("select * from emp1");
      System.out.println("Data Before Updations");
      System.out.println("ENO    ENAME    ESAL    EADDR");
      System.out.println("---------------------------------------");
      while(rs.next()){
            System.out.println(rs.getInt(1)+"    "+rs.getString(2)+"    "+rs.getFloat(3)+"
      "+rs.getString(4));
      }
      System.out.println("Application is in Pausing state, please update database");
      System.in.read();
      rs.beforeFirst();
      System.out.println("Data After Updations");
      System.out.println("ENO    ENAME    ESAL    EADDR");
```

```
        System.out.println("---------------------------------------");
        while(rs.next()){
           rs.refreshRow();
                System.out.println(rs.getInt(1)+"    "+rs.getString(2)+"    "+rs.getFloat(3)+"
       "+rs.getString(4));
        }
        con.close();
     }

  }
```

-->To move ResultSet cursor to before first record we have to use the following method from ResultSet

public void beforeFirst()

-->To move ResultSet cursor to After the last record we have to use the following method from ResultSet

public void afterLast()

-->To move ResultSet cursor to a particular record we have to use the following method from ResultSet

public void absolute(int rec_position)

-->In case of scroll sensitive ResultSet objects to reflect later database updations into the ResultSet object we have to refresh each and every record for this we have to use the following method from ResultSet

public void refreshRow()

-->where refreshRow() method can be used to refresh only one record.

-->If we use Type4 Driver provided by oracle in the above application then JVM will raise an exception like java.sql.SQLException:unsupportedfeature:refreshrow

-->In jdbc applications scroll sensitive ResultSet object should be supported by Type1 Driver provided by Sun MicroSystems,which could not be supported by Type4 Driver provided by oracle.

-->In Jdbc applications scroll Insensitive ResultSet object could not be supported by both Type1 Driver provided by Sun MicroSystems and Type4 Driver provided by oracle.

-->In jdbc applications the main purpose of UpdatableResultSet object is to perform updations on its content in order to perform the manipulations with the data available at Database

-->In jdbc applications Updatable ResultSet objects can be used to insert records on database table to achieve the above requirement we have to use the following steps:

Step1:get Updatable ResultSet object

```
Statement
st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);

ResultSet rs=st.executeQuery("select * from emp1");
```

Step2: After creating ResultSet Object we have to move ResultSet cursor to end of the ResultSet object
where we have to take a buffer to insert new Record data temporarily to achieve this we have to use the following method from ResultSet

public void moveToInsertRow()

Ex:rs.moveToInsertRow();

Step3:Insert record data on resultset object temporarily to do this we have to use the following method

public void updatexxx(int field_Num,xxx value)

```
Ex:rs.updateInt(1,555);
rs.updateString(2,'xys');
rs.updateFloat(3,9000);
```

Step4:Make the temporarily insertion as permanent insertion in resultset object as well as on database table to achieve this we have to use the following method

public void insertRow()

ex:rs.insertRow();

NOTE:The main advantage of this updatable ResultSet object is to perform updations on database table without using SQL Queries.

JdbcApp23:
The following example demonstrates how to insert no.of records into database table through a Jdbc application

```
package com.durgasoft;
    import java.io.BufferedReader;
    import java.io.InputStreamReader;
    import java.sql.Connection;
    import java.sql.DriverManager;
    import java.sql.ResultSet;
    import java.sql.Statement;

    public class Jdbcapp23 {

       public static void main(String[] args)throws Exception{
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          Connection con=DriverManager.getConnection("jdbc:odbc:nag","system","durga");
           Statement
          st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE
          );

     ResultSet rs=st.executeQuery("select * from emp1");
          rs.moveToInsertRow();
          BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

     while(true){
             System.out.print("Employee Number   :");
             int eno=Integer.parseInt(br.readLine());
             System.out.print("Employee Name     :");
             String ename=br.readLine();
             System.out.print("Employee Salary    :");
             float esal=Float.parseFloat(br.readLine());
             System.out.print("Employee Address  :");
             String eaddr=br.readLine();
             rs.updateInt(1, eno);
             rs.updateString(2, ename);
             rs.updateFloat(3, esal);
             rs.updateString(4, eaddr);
             rs.insertRow();

    System.out.println("Employee inserted Successfully");
             System.out.print("Onemore Employee[Yes/no]  :");
             String option=br.readLine();
             if(option.equals("no")){
                break;
```

```
            }
          }
          con.close();
       }

    }
```

NOTE:In jdbc applications updatable ResultSets could not be supported by Type 4 Driver provided by oracle

-->In jdbc applications by using updatable ResultSet object it is possible to update database

-->To perform this we have to use the following steps

Step1:get updatable ResultSet object

```
Statement
st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_UPDATABLE);

ResultSet rs=st.executeQuery("select * from emp1");
```

Step2:update Resultset Object Temporarily

To achieve this we have to use the following method

public void updatexxx(int field_Name,xxx value)

ex: rs.updateFloat(3,7000.0f);

Step3:Make temporary updation as permanent updation on resultset obejct as well as on database to achieve this we have to use the following method

public void updateRow()

Ex:rs.updateRow();

```
JdbcApp24:
package com.durgasoft;
import java.sql.Connection;
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
import java.sql.Statement;

public class Jdbcapp24 {

   public static void main(String[] args)throws Exception {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      Connection con=DriverManager.getConnection("jdbc:odbc:nag", "system","durga");
       Statement st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
      ResultSet.CONCUR_UPDATABLE);


      ResultSet rs=st.executeQuery("select * from emp1");
      while(rs.next()){
         float esal=rs.getFloat(3);
         if(esal<10000){
            float new_Sal=esal+500;
            rs.updateFloat(3, new_Sal);
            rs.updateRow();
         }
      }
      con.close();
   }

}
```

-->In Jdbc applications by using updatable ResultSet object it is possible to delete records on database table to achieve this we have to use the following method

public void deleteRow()

Ex:rs.deleteRow();

```
JdbcApp25:
package com.durgasoft;
 import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class Jdbcapp25: {
   public static void main(String[] args)throws Exception {
      Class.forName("oracle.jdbc.OracleDriver");
       Connection
      con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","durga"
```

```
        );
         Statement st=con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
        ResultSet.CONCUR_UPDATABLE);
         ResultSet rs=st.executeQuery("select * from emp1");
         rs.last();
         rs.deleteRow();
         st.close();
         con.close();
      }

   }
```

-->To move ResultSet cursor to a particular Record position we have to use the following method from resultset

public void absolute(int position)

-->To move ResultSet cursor over some no.of records we have to use the following method from Resultset

public void relative(int no.of.records)

-->If we have bulk of records in database table,where if we are trying to retrieve all the

records at a time into resultset object automatically Jdbc application performance

will be reduced.

-->In the above context to improve the performance of Jdbc application we have to fetch the limited

no. of records in multiple attempts.

-->To specify the no.of records which we want to fetch at an attempt then we have to use

the following method

public void setFetchSize(int size)

-->To get the specified fetch size value from resultset we have to use the following method

public int getFetchSize

What is the difference between Statement and PreparedStatement?
Ans:
In Jdbc applications,when we have a requirement to execute the SQL queries independently,we have to use Statement.
In Jdbc applications,when we have a requirement to execute same SQL query in the next sequence where to improve the performance of Jdbc applications,we have to use PreparedStatement.
To achieve the above requirement,if we use Statement,then for every time of executing the same SQL query,DB engine has to perform Query Tokenization,Query Parsing,Query Optimization and Query Execution without having any validation from one time to another time.
This approach will reduce the performance of Jdbc application.

In the above context,to improve the performance of Jdbc applications,we have to use an alternative where we have to perform Query Processing one time inorder to perform or execute the same SQL Query in the next sequence.

To achieve the above alternative,we have to use PreparedStatement over Statement.

If we want to use PreparedStatement in Jdbc applications we have to use the following steps.