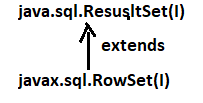
1.Def:- Row set is extension of Result Set. Row set is an alternative to Result set.



By Default, rowset is scrollable and Updatable.

Limitations of ResultSet:-

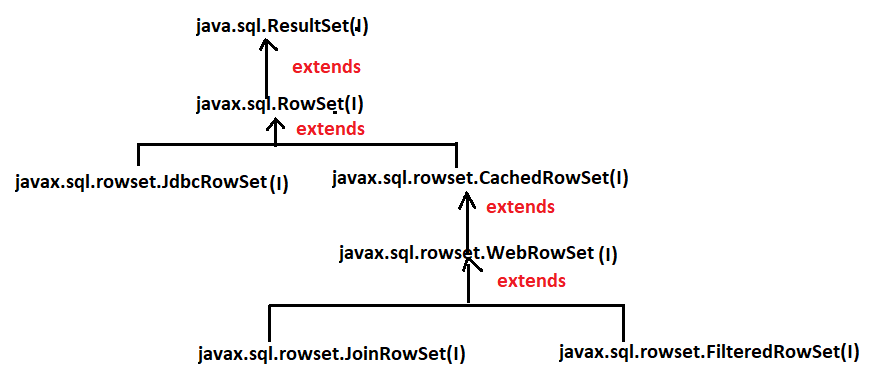
1. Resultset object is not serializable object. So we can not send the result set object over the network.
2. Result set object is connected object i.e If the JDBC app loss connection with DB software, JDBC application can’t process the resultset object.
3. Result set do not support the Event handling.

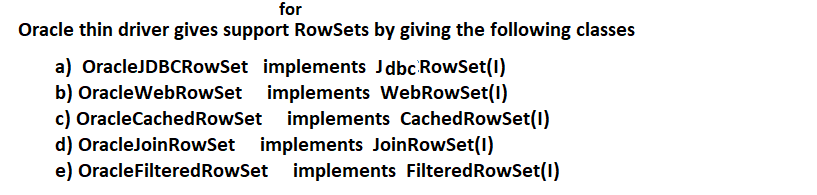
To overcome those problems , use Rowset.

Advantages of Rowset:-

1. Rowset object is serializable object. So we can send the rowset object over network.
2. Rowset supports the Event handling.
3. Result set is disconnected object i.e JDBC application can process the rowset object even though JDBC app close the connection with DB software.

**2. RowSet Types**:-





* 1. JdbcRowSet:-

1. It is Connected object.
2. JdbcRowSet object is not Serialized object.
3. It does not support EventHandling.

Example:-

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.Statement;

**import** javax.sql.rowset.JdbcRowSet;

**import** oracle.jdbc.rowset.OracleJDBCRowSet;

**public** **class** Sample {

**public** **static** String *qur*="select eno,ename from emp";

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

**try** {

JdbcRowSet jr=**new** OracleJDBCRowSet();

jr.setUrl("jdbc:oracle:thin:@localhost:1521:xe");

jr.setUsername("sukumar");

jr.setPassword("sukumar");

jr.setCommand("select \* from emp");

jr.execute();

**while**(jr.next()) {

System.***out***.println(jr.getInt(1)+" "+jr.getString(2));

}

}

**catch**(Exception e) {

System.***out***.println(e.getMessage());

}

}

}

Output:

1 suku

2 samba

3 veena

4 suma

5 suha

6 sula

7 subbarao

* 1. CachedRowSet:-

1. Cache means buffer.
2. It is disconnected object.
3. It is Serialized object.

Example:-

* 1. **WebRowSet:-**

1. WebRowSet object is disconnected object.
2. WebRowSet object is serializable object.
3. Converting DB table records to XML file content is very useful to send DB table data over the network to another app as XML content.

writeXml(FileWriter/Stream object);

This method present in webRowSet interface. This method converts table records as XML file content and publish XML file content on console or to .xml file.

readXml(FileReader object)

This method present in webRowSet interface.This method read xml file content and this method writes xml file content to rowset.

Example:-1

**import** java.io.FileWriter;

**import** javax.sql.rowset.WebRowSet;

**import** oracle.jdbc.rowset.OracleWebRowSet;

**public** **class** Sample {

**public** **static** String *qur*="select eno,ename from emp";

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

**try** {

WebRowSet jr=**new** OracleWebRowSet();

jr.setUrl("jdbc:oracle:thin:@localhost:1521:xe");

jr.setUsername("sukumar");

jr.setPassword("sukumar");

jr.setCommand("select \* from emp where eno=1 or eno=2");

jr.execute();

//jr.writeXml(System.out);

// Above statement publish xml content on console.

FileWriter fw=**new** FileWriter("D:\\family.xml");

jr.writeXml(fw);

**while**(jr.next()) {

System.***out***.println(jr.getInt(1)+" "+jr.getString(2));

}

}

**catch**(Exception e) {

System.***out***.println(e.getMessage());

}

}

}

Example:2

**import** java.io.FileReader;

**import** java.sql.SQLException;

**import** javax.sql.rowset.WebRowSet;

**import** oracle.jdbc.rowset.OracleWebRowSet;

**public** **class** Sample {

**public** **static** String *qur*="select eno,ename from emp";

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

**try** {

WebRowSet jr=**new** OracleWebRowSet();

jr.setUrl("jdbc:oracle:thin:@localhost:1521:xe");

jr.setUsername("sukumar");

jr.setPassword("sukumar");

jr.setCommand("select \* from emp where eno=3");

jr.execute();

FileReader fw=**new** FileReader("D:\\family.xml");

jr.readXml(fw);

jr.acceptChanges();

**while**(jr.next()) {

System.***out***.println(jr.getInt(1)+" "+jr.getString(2));

}

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

* 1. JoinRowSet:-

1. JoinRowSet object is serializable object and disconnected object.
2. If we want to join rows from different rowsets into a single rowset based on matched column(common column) then we should go for joinRowSet.

Example:

**package** com.raos.jdbc;

**import** javax.sql.rowset.CachedRowSet;

**import** javax.sql.rowset.JoinRowSet;

**import** oracle.jdbc.rowset.OracleCachedRowSet;

**import** oracle.jdbc.rowset.OracleJoinRowSet;

**public** **class** Sample {

**public** **static** String *qur*="select eno,ename from emp";

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

**try** {

CachedRowSet jr=**new** OracleCachedRowSet();

jr.setUrl("jdbc:oracle:thin:@localhost:1521:xe");

jr.setUsername("sukumar");

jr.setPassword("sukumar");

jr.setCommand("select \* from emp where eno=1 or eno=2");

jr.execute();

CachedRowSet jr1=**new** OracleCachedRowSet();

jr1.setUrl("jdbc:oracle:thin:@localhost:1521:xe");

jr1.setUsername("sukumar");

jr1.setPassword("sukumar");

jr1.setCommand("select \* from student ");

jr1.execute();

JoinRowSet jr2=**new** OracleJoinRowSet();

jr2.addRowSet(jr,1);

jr2.addRowSet(jr1,1);

**while**(jr2.next()) {

System.***out***.println(jr2.getString(1)+" "+jr2.getString(2)+" "+jr2.getString(3));

}

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

Output:-

1 suku rock

2 samba hhh