**ROWSETS**

* Introduced in JDBC 2.0 (supports from JDK 1.4 onwards)
* Rowset is the child interface of Resultset.
* Rowset is defined in javax.sql package
* Rowset is the advanced version of Resultset used to manage the list of records received from the table

What is the difference between Resultset and Rowset?

|  |  |
| --- | --- |
| ResultSet | Rowset |
| Always moves in forward direction by default | Rowset is the scrollable by default |
| Result set allows read only by default | Row set allows update operation by default (updatable) |
| Result set is not serializable. So not possible to send the object across the network | Rowset is serializable by default |
| Result set is always connected   * that means DB connection is mandatory to retrieve the data from the resultset. * If DB connection is closed, then we can’t retrieve the data from resultset | Rowset is either connected or disconnected.  JDBCRowset 🡪 connected rowset  CachedRowset/Webrowset/Filtered rowset are disconnected rowset |

Types of Rowsets

|  |  |
| --- | --- |
| ConnectedRowset | DisconnectedRowset |
| * Similar to resultset * We can’t serialize the rowset object because we are not allowed to retrieve the data without DB connection | * Serializable – we can send this object across the network because disconnected rowset won’t require DB connection to retrieve the data |
| * JDBCRowset | * + - CachedRowset     - WebRowset     - JoinRowset     - FilterRowset |

Hierarchy of Rowset

Steps to create Rowset objects:

Step 1:

RowsetFactory rsf = RowSetProvider.newFactory();

Step 2: using rsf reference, we can create any type of rowset object

JdbcRowSet jrs = rsf.createJdbcRowSet();

CachedRowSet crs = rsf.createCachedRowSet();

WebRowSet wrs = rsf.createWebRowSet();

JoinRowSet jrs = rsf.createJoinRowSet();

FilteredRowSet frs = rsf.createFilteredRowset();

Step 3: provide the URL, username, password and select query using setter methods.

jrs.setUrl("jdbc:oracle:thin:@localhost:1521:orcl");

jrs.setUsername("system");

jrs.setPassword("manager");

jrs.setCommand("select htno,sname from student");

Step 4: call the execute method to get the rowset from database

jrs.execute()

Step 5: process the obtained rowset using reference variable of rowset.

while(jrs.next())

{

System.out.println(jrs.getInt(1) +"::" +

jrs.getString(2));

}

Note:

we can use the methods like absolute(), first(), last() and so on to navigate the control

Step 6: close the rowset using close()

|  |  |
| --- | --- |
| CachedRowSet | * The main advantage of CachedRowSet is we can send this RowSet object for multiple people across the network and all those people can access RowSet data without having DB Connection. * If we perform any update operations(like insert,delete and update) to the CachedRowSet,to reflect those changes compulsary Connection should be established. Once Connection established then only those changes will be reflected in Database. |

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

**import** java.sql.SQLException;

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

**import** javax.sql.rowset.RowSetFactory;

**import** javax.sql.rowset.RowSetProvider;

**public** **class** JDBCRowSetDemo {

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

**try**

{

RowSetFactory rsf = RowSetProvider.*newFactory*();

JdbcRowSet jdbc = rsf.createJdbcRowSet();

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

jdbc.setUsername("advdec23");

jdbc.setPassword("advdec23");

jdbc.setCommand("select pid,pname,mrp from product");

jdbc.execute();

**while**(jdbc.next()) {

System.***out***.printf("%10d%20s%10.2f\n", jdbc.getInt(1),jdbc.getString(2),jdbc.getDouble(3));

}

System.***out***.println("Data in Reverse Order...");

jdbc.afterLast();

**while**(jdbc.previous()) {

System.***out***.printf("%10d%20s%10.2f\n", jdbc.getInt(1),jdbc.getString(2),jdbc.getDouble(3));

}

}**catch**(SQLException e) {

System.***out***.println("SQL Exception.....");

}

}

}

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

**import** java.sql.SQLException;

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

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

**import** javax.sql.rowset.RowSetFactory;

**import** javax.sql.rowset.RowSetProvider;

**public** **class** CachedRowSetDemo {

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

**try**

{

RowSetFactory rsf = RowSetProvider.*newFactory*();

CachedRowSet jdbc = rsf.createCachedRowSet();

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

jdbc.setUsername("advdec23");

jdbc.setPassword("advdec23");

jdbc.setCommand("select pid,pname,mrp from product");

jdbc.execute();

**while**(jdbc.next()) {

System.***out***.printf("%10d%20s%10.2f\n", jdbc.getInt(1),jdbc.getString(2),jdbc.getDouble(3));

}

System.***out***.println("Data in Reverse Order...");

jdbc.afterLast();

**while**(jdbc.previous()) {

System.***out***.printf("%10d%20s%10.2f\n", jdbc.getInt(1),jdbc.getString(2),jdbc.getDouble(3));

}

}**catch**(SQLException e) {

System.***out***.println("SQL Exception.....");

}

}

}