ABOUT PRASAD SAYA

javax.sql.rowset.JdbcRowSet Example

Posted by: Prasad Saya in JdbcRowSet, sql February 3rd, 2015 0 340 Views

This article introduces the `JdbcRowSet` interface and its basic usage. This class is defined in the `javax.sql.rowset` package.

`JdbcRowSet` interface extends `RowSet` (and `Rowset` extends `ResultSet`).

A Jdbc rowset:

- It is a connected rowset.
- It is a wrapper around a `ResultSet` object; an enhanced `ResultSet` object. It maintains a connection to its data source, just as a `ResultSet` object does. As a consequence, a Jdbc rowset can, for example, be a component in a Java Swing application.
- It has a set of properties and a listener notification mechanism that make it a JavaBeans component.
- It can be used to make a `ResultSet` object scrollable and updateable when it does not otherwise have those capabilities.

1. JDBC Rowset

1.1. Connected Rowset

A rowset object may make a connection with a data source and maintain that connection throughout its life cycle, so it is called a connected rowset.

1.2. JavaBeans Properties

The `RowSet` interface provides a set of JavaBeans properties. This allows a `RowSet` instance to be configured to connect to a JDBC data source and read data from the data source: `setUrl()` ,

```
setUserName()
```

,

```
setDataSourceName()
```

,

```
setQueryTimeOut()
```

,

```
setReadOnly()
```

,

```
setTransactionIsolation()
```

,

```
setCommand()
```

, … and corresponding getter methods.

A group of setter methods (

```
setInt()
```

,

```
setByte()
```

,

```
setString()
```

, …) provide a way to pass input parameters to a rowset's command property.

For example:

```
JdbcRowSetImpl jrs = new JdbcRowSetImpl();
jrs.setCommand("SELECT * FROM books WHERE author = ?");
jrs.setURL("jdbc:myDriver:myAttribute"); // set method to connect to datasource (configure)
jrs.setUsername("myuser");
jrs.setPassword("mypwd");
jrs.setString(1, "Mark Twain"); // set method to pass input parameter
jrs.execute(); // fills this rowset object with data
```

1.3. JavaBeans Notification Mechanism

Rowset objects use the JavaBeans event model.

```
RowSetListener
```

is an interface that is implemented by a component that wants to be notified when a significant event happens in the life of a

```
RowSet
```

object. A component becomes a listener by being registered with a

```
RowSet
```

object via the method

```
RowSet.addRowSetListener()
```

.

There are three events trigger notifications (and handled by listener methods):

- Cursor movement:
  ```
  cursorMoved(RowSetEvent)
  ```
- Update, insert or delete of a row:
  ```
  rowChanged(RowSetEvent)
  ```
- Change to the entire rowset content:
  ```
  rowSetChanged(RowSetEvent)
  ```

1.4. Creating a JDBC Rowset

There are four ways to create a

```
JdbcRowSet
```

object. The

```
JdbcRowSet
```

object needs to connect to database and then be populated with data.

1.4.1. Reference implementation default constructor

```
JdbcRowSet jdbcRs = new JdbcRowSetImpl(); // create rowset object
jdbcRs.setCommand("select * from BOOKS"); // set properties, and
jdbcRs.setUrl(url); // connect to database
jdbcRs.setUserName(usr);
jdbcRs.setPassword(pwd);
jdbcRs.execute(); // populate with data
```

1.4.2. Constructor that takes a Connection object

```
JdbcRowSet jdbcRs = JdbcRowSetImpl(conn); // conn is the java.sql.Connection object
jdbcRs.setCommand("select * from BOOKS");
jdbcRs.execute();
```

1.4.3. Constructor that takes a ResultSet object

The

```
ResultSet
```

must be created as updateable and scrollable; otherwise the Jdbc rowset will not be updateable as well.

```
Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
ResultSet rs = stmt.executeQuery("select * from BOOKS");
JdbcRowSet jrs = new JdbcRowSetImpl(rs);
```

1.4.4. Using an instance of RowSetFactory

```
RowSetFactory rsf = RowSetProvider.newFactory();
JdbcRowSet jrs = rsf.createJdbcRowset();
// Set properties, connect to database and populate the rowset with data …
```

The example program in this article uses a

```
RowSetFactory
```

to create a jdbc rowset.

1.5. Using JdbcRowSet objects

- Rowset can use all of the cursor movement methods defined in the

  ```
  ResultSet
  ```

  interface:

  ```
  absolute(int row)
  ```

  ,

  ```
  previous()
  ```

  ,

  ```
  relative(int rows)
  ```

  , …
- Rowset is updated (insert, update and delete) the same way data is updated in a

  ```
  ResultSet
  ```

  object.

2. The Example

The example program performs Create, Read, Update and Delete operations on a database table using the

```
JDBCRowset
```

.

- Create the JDBC rowset, configure it and connect to the database.
- Read all rows from the database table and populate the rowset.
- Query all rows from rowset.
- Insert a row into the rowset.
- Update a row in the rowset.

- Delete a row from the rowset.

The example uses MySQL version 5.5.20 server database. The details to create the example database and data are shown below.

2.1. Database SQL Scripts

The following MySQL SQL commands can be used to create the example database, table and insert some data into the table. The

```
mysql
```

command-line tool can be used.

2.1.1. Create database, verify and use it

```
CREATE DATABASE example_db;
SHOW DATABASES;
USE example_db;
```

2.1.2. Create table and verify it

```
CREATE TABLE books_table (title VARCHAR(40), author VARCHAR(40), PRIMARY KEY (title));
DESCRIBE books_table;
```

2.1.3. Insert data into the table

```
INSERT INTO books_table VALUES ('The Mysterious Affair at Styles', 'Agatha Christie');
INSERT INTO books_table VALUES ('The Count of Monte Cristo', 'Alexandre Dumas');
INSERT INTO books_table VALUES ('A Study in Scarlet', 'Arthur Conan Doyle');
INSERT INTO books_table VALUES ('Doctor Zhivago', 'Boris Pasternak');
INSERT INTO books_table VALUES ('Animal Farm', 'George Orwell');
```

2.1.4. Query the table data

```
SELECT * FROM books_table;
```

The output from the query should be as follows:

```
mysql> SELECT * FROM books_table;
+----------------------------------+------------------------+
| title                            | author                 |
+----------------------------------+------------------------+
| A Study in Scarlet               | Arthur Conan Doyle     |
| Animal Farm                      | George Orwell          |
| Doctor Zhivago                   | Boris Pasternak        |
| The Count of Monte Cristo        | Alexandre Dumas        |
| The Mysterious Affair at Styles  | Agatha Christie        |
+----------------------------------+------------------------+
5 rows in set (0.00 sec)
```

2.2. The Example Program Code

JDBCRowsetExample.java

```java
import javax.sql.rowset.RowSetProvider;
import javax.sql.rowset.RowSetFactory;
import javax.sql.rowset.JdbcRowSet;
import java.sql.SQLException;

public class JDBCRowsetExample {

    private int insertedRowNo;

    private final static String DB_URL = "jdbc:mysql://localhost:3306/example_db";
    private final static String USR = "root";
    private final static String PWD = "root";
    private final static String BOOKS_TABLE = "books_table";
    private final static String TITLE = "title";
    private final static String AUTHOR = "author";

    private final static String INSERT_ROW_TITLE = "Lady Chatterley's Lover";
    private final static String INSERT_ROW_AUTHOR = "D H Lawrence";
    private final static String UPDATE_ROW_AUTHOR = "D H LAWRENCE";

    public JDBCRowsetExample() {
    }

    public static void main(String [] args)
            throws Exception {

        JDBCRowsetExample pgm = new JDBCRowsetExample();

        JdbcRowSet jrs = pgm.getJDBCRowset();

        pgm.loadAllRows(jrs);
        pgm.printAllRows(jrs);
        pgm.insertRow(jrs);
        pgm.updateRow(jrs);
        pgm.deleteRow(jrs);

        jrs.close();

        System.out.println("- Close.");
    }

    private JdbcRowSet getJDBCRowset()
            throws SQLException {

        System.out.println("- Configure JDBC Rowset and connect to database: " + DB_URL);

        RowSetFactory rsFactory = RowSetProvider.newFactory();
        JdbcRowSet jRowset = rsFactory.createJdbcRowSet();

        jRowset.setUsername(USR);
        jRowset.setPassword(PWD);
        jRowset.setUrl(DB_URL);
        jRowset.setReadOnly(false); // make rowset updateable

        return jRowset;
    }

    private void loadAllRows(JdbcRowSet jrs)
            throws SQLException {

        // populate the rowset with table rows

        System.out.println("- Load (initial) all rows from database table: " + BOOKS_TABLE);
        String sql = "SELECT * FROM " + BOOKS_TABLE;
        jrs.setCommand(sql);
        jrs.execute();

        System.out.println("Total rows in table: " + getRowCount(jrs));
    }

    private int getRowCount(JdbcRowSet jrs)
            throws SQLException {

        jrs.last();
        return jrs.getRow();
    }

    private void printAllRows(JdbcRowSet jrs)
            throws SQLException {

        System.out.println("- Print all rows:");

        jrs.beforeFirst();

        while (jrs.next()) {

            int rowNo = jrs.getRow();
            String s1 = jrs.getString(TITLE);
            String s2 = jrs.getString(AUTHOR);
            System.out.println(rowNo + ": " + s1 + ", " + s2);
        }
    }

    private void insertRow(JdbcRowSet jrs)
            throws SQLException {

        System.out.println("- Insert row: ");

        jrs.moveToInsertRow();
        jrs.updateString(TITLE, INSERT_ROW_TITLE);
        jrs.updateString(AUTHOR, INSERT_ROW_AUTHOR);
        jrs.insertRow();

        insertedRowNo = jrs.getRow(); // Note: this is an instance variable
        String s1 = jrs.getString(TITLE);
        String s2 = jrs.getString(AUTHOR);
        System.out.println(insertedRowNo + ": " +  jrs.getString(TITLE) + ", " + jrs.getString(AUTHOR));
        System.out.println("Total rows in table: " + getRowCount(jrs));
    }

    private void updateRow(JdbcRowSet jrs)
            throws SQLException {

        System.out.println("- Update row " + insertedRowNo);

        jrs.absolute(insertedRowNo);
        String s1 = jrs.getString(TITLE);
        String s2 = jrs.getString(AUTHOR);
        System.out.println("Row (before update): " + s1 + ", " + s2);

        jrs.updateString("AUTHOR", UPDATE_ROW_AUTHOR);
        jrs.updateRow();

        s1 = jrs.getString(TITLE);
        s2 = jrs.getString(AUTHOR);
        System.out.println("Row (after update): " +  s1 + ", " + s2);
    }

    private void deleteRow(JdbcRowSet jrs)
            throws SQLException {

        jrs.absolute(insertedRowNo);
        String s1 = jrs.getString(TITLE);
        String s2 = jrs.getString(AUTHOR);
        System.out.println("- Delete row " + insertedRowNo + ": " + s1 + ", " + s2);

        jrs.deleteRow();

        System.out.println("Total rows in table: " + getRowCount(jrs));
    }
}
```

2.3. The Output

```
- Configure JDBC Rowset and connect to database: jdbc:mysql://localhost:3306/example_db
- Load (initial) all rows from database table: books_table
Total rows in table: 5
- Print all rows:
1: A Study in Scarlet, Arthur Conan Doyle
2: Animal Farm, George Orwell
3: Doctor Zhivago, Boris Pasternak
4: The Count of Monte Cristo, Alexandre Dumas
5: The Mysterious Affair at Styles, Agatha Christie
- Insert row:
6: Lady Chatterley's Lover, D H Lawrence
Total rows in table: 6
- Update row 6
Row (before update): Lady Chatterley's Lover, D H Lawrence
Row (after update): Lady Chatterley's Lover, D H LAWRENCE
- Delete row 6: Lady Chatterley's Lover, D H LAWRENCE
Total rows in table: 5
- Close.
```

From the output:

The program inserts a row with title "Lady Chatterley's Lover" and author "D H Lawrence". Next, it updates the same row's author to "D H LAWRENCE". Finally, it deletes the inserted (and updated) row.

3. Download Java Source Code

This was an example of

```
javax.sql.rowset.JdbcRowSet Example
```

Download
You can download the full source code of this example here: **JDBCRowsetExample.zip**