# [[1]](#footnote-1)#[[2]](#footnote-2)$Using the SqlResultSet

You can use the ADO.NET SqlResultSet to retrieve an updateable, scrollable cursor that resides on the data server. Instead of marshaling an entire result set from the server to the client, the SqlResultSet retrieves data for a single row at a time. Once created, a SqlResultSet can be used to view, edit, insert, and delete rows from a data source.

In This Section

Creating a SqlResultSetcpconCreatingSqlResultSet

Accessing Data with a SqlResultSetcpconAccessingDataWithSqlResultSet

## [[3]](#footnote-3)#[[4]](#footnote-4)$Creating a SqlResultSet

To obtain a SqlResultSet, use the ExecuteResultSet method of the SqlCommand object. The behavior of the SqlResultset cursor is determined by the value passed in the Options parameter of ExecuteResultSet method. These values can be combined. For example, you can request an updateable, scrollable cursor, which is sensitive to changes made to the data source. The following table provides you with a description of ResultSetOptions enumerations.

|  |  |  |
| --- | --- | --- |
| Member Name | Value | Description |
| None | 0 | Default is to scroll? [**NOTE: The default behavior was not defined when this document was written.**] |
| Updatable | 1 | Data can be updated using the SqlResultSet. |
| Scrollable | 16 | The cursor can be used to access rows randomly. |
| Sensitive | 256 | The SqlResultSet is sensitive to changes made to the data source. |
| Insensitive | 4096 | The rows returned by the SqlResultSet are not affected by changes made to the data source. |

The following sample creates an instance of SqlResultSet.

[Visual Basic]

' Initialize the command and connection objects.

Dim sqlConn As New SqlConnection(strConn)

Dim sqlCmd As SqlCommand

sqlCmd = sqlConn.CreateCommand()

sqlCmd.CommandText = strSQL

' Create the SqlResultSet.

Dim sqlResults As SqlResultSet

sqlResults = sqlConn.ExecuteResultSet( \_

  ResultSetOptions.Updatable | ResultSetOptions.Scrollable)

[C#]

// Initialize the command and connection objects.

SqlConnection sqlConn = new SqlConnection(strConn);

SqlCommand sqlCmd = sqlConn.CreateCommand();

sqlCmd.CommandText = strSQL;

// Create the SqlResultSet.

sqlResultSet sqlResults =   sqlCmd.ExecuteResultSet(ResultSetOptions.Updatable |   ResultSetOptions.Scrollable);

See Also

SqlResultSet Class!ALink("frlrfSystemDataSqlClientSqlResultSetClassTopic") | Using the SqlResultSetcpconUsingSqlResultSet | Accessing Data with a SqlResultSetcpconAccessingDataWithSqlResultSet

## [[5]](#footnote-5)#[[6]](#footnote-6)$Accessing Data with the SqlResultSet

A SqlResultSet object can be used to view, edit, insert, and delete rows from a data source. It provides a scrollable, updateable cursor that is created and maintained on the server. The ExecuteResultSet method of the SqlCommand object returns a SqlResultSet.

Once you have obtained the SqlResultSet object from ExecuteResultSet, you can access the data a single row at a time. The SqlResultSet does not return all of the records that match the selection criteria of the command. Instead, you access individual rows using the SqlResultSet object's read methods as listed in the following table.

|  |  |
| --- | --- |
| Method | Description |
| ReadFirst | Reads the first row in the result set. The method returns true if the read was successful. |
| ReadLast | Reads the last row in the result set. The method returns true if the read was successful. |
| Read | Reads the next row in the result set. The method returns true if the read was successful. |
| ReadPrevious | Reads the previous row in the result set. The method returns true if the read was successful. If the read would move the cursor to a position before the first record, ReadPrevious returns false. |
| ReadRelative | Moves the cursor forward or backward by the number of rows specified in the Position parameter. |
| ReadAbsolute | Moves the cursor to the position indicated by the Position parameter. |

The following sample obtains a SqlResultSet. It scrolls to the last row, using ReadLast, and then displays the values for each row by moving backwards, using ReadPrevious.

[Visual Basic]

' Initialize the command and connection objects.

Dim sqlConn As New SqlConnection(strConn)

Dim sqlCmd As SqlCommand

sqlCmd = sqlConn.CreateCommand()

sqlCmd.CommandText = strSQL

' Create the SqlResultSet.

Dim sqlResults As SqlResultSet

sqlResults = sqlConn.ExecuteResultSet( \_

  ResultSetOptions.Updatable | ResultSetOptions.Scrollable)

If sqlResults.MoveLast() Then

  Dim i

  Do While(sqlResults.MovePrevious())

    For i = 0 To sqlResults.FieldCount - 1

      Console.WriteLine("Column: " & sqlResults.GetName(i) & \_

        "Value: " & sqlResults.GetValue(i)

    Next i

  Loop

End If

sqlResults.Close()

[C#]

// Initialize the command and connection objects.

SqlConnection sqlConn = new SqlConnection(strConn);

SqlCommand sqlCmd = sqlConn.CreateCommand();

sqlCmd.CommandText = strSQL;

// Create the SqlResultSet

sqlResultSet sqlResults =   sqlCmd.ExecuteResultSet(ResultSetOptions.Updatable |   ResultSetOptions.Scrollable);

// Display the values in the last row.

if(sqlResults.MoveLast())

  for(int i = 0; i < sqlResults.FieldCount; i++)

    Console.WriteLine("Column: " + sqlResults.GetName(i) +

      "Value: " + sqlResults.GetValue(i);

// Move through the rows backwards, displaying

// the values for each row.

  while(sqlResults.MovePrevious())

    for(int i = 0; i < sqlResults.FieldCount; i++)

      Console.WriteLine("Column: " + sqlResults.GetName(i) +

        "Value: " + sqlResults.GetValue(i);

sqlResults.Close();

See Also

SqlResultSet Class!ALink("frlrfSystemDataSqlClientSqlResultSetClassTopic") | Using the SqlResultSetcpconUsingSqlResultSet | Creating a SqlResultSetcpconCreatingSqlResultSet

1. # cpconUsingSqlResultSet [↑](#footnote-ref-1)
2. $ Using the SqlResultSet [↑](#footnote-ref-2)
3. # cpconCreatingSqlResultSet [↑](#footnote-ref-3)
4. $ Creating a SqlResultSet [↑](#footnote-ref-4)
5. # cpconAccessingDataWithSqlResultSet [↑](#footnote-ref-5)
6. $ Accessing Data with the SqlResultSet [↑](#footnote-ref-6)