SQream .NET Connector

SQream Technologies

Version 1.2.0

Table of Contents

.NET Connector

Version 1.2.0

This guide describes:

- The required prerequisistes, and installation of the SQream .NET driver.
- How to create and maintain a connection to SQream DB for a .NET application.

Prerequisites

Install .NET Framework 4.0 or newer.

Install the SQream .NET Driver

- Download the SQream Node.js driver file: sqream-connector-[version].tgz.
- Unzip it.
- From your .NET project, add a reference to SqreamNet.dll.

> **TIP:**
>
> To upgrade the provider for an existing project, just replace the exiting DLL file, or point the project to the new driver location.

Create a connection to SQream DB

Connecting to SQream DB is established by creating a SqreamConnection object with a connection string.

Import the SqreamNet namespace to your source code.

The following example shows how to connect to a server that listens on port 5000 on the local machine.

Example:

```
using SqreamNet;

var connectionString  =  "Data
Source=127.0.0.1,5000;User=sqream;Password=sqream;Initial \
Catalog=master;Integrated Security=true";
var connection  = new SqreamConnection(connectionString);
connection.Open();
// Tip: Validate the connection was opened
if  (connection.State != System.Data.ConnectionState.Open)
{
```

```
    throw new SqreamException(string.Format("failed to connect to
sqream"));
}
```

Run a Query

Reading data from SQream DB is achieved by using the SqreamDataReader Object which implements the IDataReader interface.

The following example shows how to read all the records in the **books** table.

```
public class Book
{
    public int Id { get; set;  }
    public string Title { get; set;  }
    public string Author { get; set;  }
}

public List<Book> GetAllBooks(SqreamConnection connection)
{
    var sql  =  "select id , title , author from books;";
    var command  = new SqreamCommand(sql, connection);
    var reader  =  (SqreamDataReader)command.ExecuteReader();
    var result  = new List<Book>();
    while  (reader.Read())

{
        var book  = new Book
        {
            Id  = reader.GetInt32(0),
            Title  = reader.GetString(1),
            Author  = reader.GetString(2)
        };

        result.Add(book);
    }
    return result;
}
```

Run a Query with Dynamic Data

When the data structure of the object is unknown, saving pre-defined indexes will not help. It is therefore recommended that you implement an extension method, which is an ability the .NET framework provides.

The following sample suggests a possible implementation to “wrap” the reader object so it would read the data, independent from query or data types.

Classes and list object are used to illustrate the functionality. This should be adjusted in your own program according to your needs.

```
    /// <summary>    /// Extension to a collection class which
contains rows of data
    /// </summary>    public class SqreamCollectionEx

{
        public List<SqreamRowEx> Rows;
    }

    /// <summary>    /// Row of data which contains columns data
    /// </summary>    public class SqreamRowEx

{
        public List<SqreamColumnEx> Columns;
    }

    /// <summary>    /// Column Data
    /// </summary>
    [DebuggerDisplay("Index: {Index} , Name: {Name} , Type:
{TypeName} , Value: {Value}")]
    public class SqreamColumnEx

{
        public int Index { get; set;  }
        public string Name { get; set;  }
        public string TypeName { get; set;  }
        public object Value { get; set;  }
    }

    /// <summary>    /// Reads the collection, extends sqream data
reader
    /// </summary>    public static class SqreamNetEx

{
        public static SqreamCollectionEx ReadCollectionEx(this
SqreamDataReader reader)

{
            var result  = new SqreamCollectionEx();
            result.Rows = new List<SqreamRowEx>();
            while  (reader.Read())

{
                var row  = new SqreamRowEx();
                row.Columns = new List<SqreamColumnEx>();
                for  (int i  =  0; i < reader.FieldCount; i++)

{
                    var column  = new SqreamColumnEx();
                    column.Index = i;
                    column.TypeName = reader.GetDataTypeName(i);
                    column.Name = reader.GetName(i);
```

```
                    switch (reader.GetDataTypeName(i))
                    {
                        case "Boolean":
                            column.Value = reader.GetBoolean(i);
                            break;
                        case "Int32":
                            column.Value = reader.GetInt32(i);
                            break;
                        case "Int64":
                            column.Value = reader.GetInt64(i);
                            break;
                        case "Single":
                            column.Value = reader.GetFloat(i);
                            break;
                        case "Double":
                            column.Value = reader.GetDouble(i);
                            break;
                        case "DateTime":
                            column.Value = reader.GetDateTime(i);
                            break;
                        case "String":
                            column.Value = reader.GetString(i);
                            break;
                        default:
                            throw new SqreamException(string.Format
("invalid type in index: {0}", i));
                    }
                    row.Columns.Add(column);
                }
                result.Rows.Add(row);
            }
            return result;
        }
}

// This code uses the extension method to retrieve the data:
var sql  =  "select id , title , author from books;";
var command  = new SqreamCommand(sql, connection);
var reader  =  (SqreamDataReader)command.ExecuteReader();
var result  = reader.ReadCollectionEx();
```

Error Handling

When working with the provider, it is recommended to implement the error handling on a per-project level.

Common Exceptions are:

Unsupported Functionalities – NotSupportedException Functionality that was not implemented – NotImplementedException SQream DB parsing or runtime errors – SqreamException