# Splice Machine ADO.NET Provider

This directory contains the source code and configuration files of the ADO.NET provider which will be used to connect with Splice DB through .NET framework.

## **Installation on Visual Studio 2019**

* Download the sdk files from this link or copy paste the following url to the browser.
* Paste the extracted files to a secure folder in your project.
* Right click the references folder and choose the directory where the files are placed. Select the SpliceMachine.Provider.dll.

## **Establishing the Connection**

* SpliceDbConnection class is the connection class where we specify the connection properties like host, userid, password and port.
* Connection string will be in the format: “uid=admin;pwd=password;host=localhost;port=1527;”.
* Most used members of connection object are as follows:

1. Open()
2. Close()
3. Commit()
4. Rollback()
5. IsAutoCommit(Bool)

* Open() and Close() methods will be used to open and close the connections.(Ref. CODE SAMPLE #0001)
* Commit() and Rollback() will be used for committing or rolling back a transaction.(Ref. CODE SAMPLE #0002,#0002#A)
* IsAutoCommit is by default set to true and the commit is handled automatically while executing the queries. When it is set to false the commit() should be called to commit a transaction at known scope.(Ref. CODE SAMPLE #0002)

CODE SAMPLE #0001:

using (var connection = new SpliceDbConnection("uid=splice;pwd=admin;host=localhost;port=1527"))

{

connection.Open();

using (var command = connection.CreateCommand())

{

command.CommandText = "CREATE TABLE TESTTABLE(a int)";

var count = command.ExecuteNonQuery();

}

connection.Close();

}

CODE SAMPLE #0002:

using (var connection = new SpliceDbConnection("uid=splice;pwd=admin;host=localhost;port=1527"))

{

connection.Open();

connection.IsAutoCommit = false;

using (var command = connection.CreateCommand())

{

command.CommandText = "CREATE TABLE TESTTABLE(a int)";

var count = command.ExecuteNonQuery();

connection.Commit();

}

connection.Close();

}

CODE SAMPLE #0002#A1:

using (var connection = new SpliceDbConnection("uid=splice;pwd=admin;host=localhost;port=1527"))

{

const string QryCreateTable = "CREATE TABLE TESTTABLE(COL1 BIGINT,COL2 INT,COL3 DECIMAL(16,5))";

const string QryInsertValue = "INSERT INTO TESTTABLE VALUES(?,?,?)";

connection.Open();

using (var command = connection.CreateCommand())

{

connection.IsAutoCommit = false;

command.CommandText = QryCreateTable;

var result1 = command.ExecuteNonQuery();

command.CommandText = QryInsertValue;

command.Parameters.Add(new SpliceDbParameter() { Value = 456});

command.Parameters.Add(new SpliceDbParameter() { Value = 789});

command.Parameters.Add(new SpliceDbParameter() { Value = 32.21 });

var result2 = command.ExecuteNonQuery();

connection.Rollback();

}

}

## **Creating a command**

* Creating a command can be done using SpliceDbCommand class.
* Most used members of the class are below:

1. CommandText(Ref. CODE SAMPLE #0002)
2. ExecuteNonQuery()(Ref. CODE SAMPLE #0002)
3. ExecuteReader()
4. Parameters

* CommandText is the string property which will be assigned with query.
* ExecuteNonQuery will return the number of rows getting affected in the db after the query execution.
* ExecuteReader will be used to fetch the data from Db.
* Parameters will be used to add parameters to a command.

## **Using Parameters**

* SpliceDbParameter class can be used to add parameters to a command.(Ref. CODE SAMPLE #0003)
* Parameters inside the query should be indicated with a ‘?’ symbol.

CODE SAMPLE #0003:

using (var connection = new SpliceDbConnection("uid=splice;pwd=admin;host=localhost;port=1527"))

{

connection.Open();

connection.IsAutoCommit = false;

using (var command = connection.CreateCommand())

{

command.CommandText = "INSERT INTO TESTTABLE VALUES(?)";

var parameter1 = new SpliceDbParameter();

parameter1.Value = 1;

command.Parameters.Add(parameter1);

var count = command.ExecuteNonQuery();

connection.Commit();

}

connection.Close();

}

**Data Retrieval**

* Data retrieval is possible using two ways,

1. ExecuteReader()(Ref. CODE SAMPLE #0004)
2. SpliceDbAdapter class(Ref. CODE SAMPLE #0005)

* ExecuteReader() will fetch the data from the table every time we call up the read() method.
* SpliceDbAdapter will fetch the entire data from the table and populate it into a DataTable.
* Stored Procedures/Functions can be invoked only through ExecuteReader(). SpliceDbAdapter does not support StoredProcedures/Functions.

CODE SAMPLE #0004:

using (var connection = new SpliceDbConnection("uid=splice;pwd=admin;host=localhost;port=1527"))

{

connection.Open();

using (var command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM TESTTABLE";

var dataReader = command.ExecuteReader();

while(dataReader.read())

{

//Every row data will be present in the dataReader object. Column index can be indicated inside square brackets.

var data1 = dataReader[0];

}

}

connection.Close();

}

CODE SAMPLE #0005:

using (var connection = new SpliceDbConnection("uid=splice;pwd=admin;host=localhost;port=1527"))

{

connection.Open();

using (var command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM TESTTABLE";

DataTable myData = new DataTable();

SpliceDbAdapter adapter = new SpliceDbAdapter();

adapter.SelectCommand = command;

adapter.Fill(myData);

//myData will have the rows and columns of the result set

}

connection.Close();

}

**Procedure/Functions execution**

* Procedure/Function can be called/invoked simply using the ExecuteNonQuery() method.
* In case of the data retrieval through function/procedure we can use a data reader.
* Refer CODE SAMPLE #0006 for the data retrieval through proceedures/functions.

CODE SAMPLE #0006:

using (var connection = new SpliceDbConnection("uid=splice;pwd=admin;host=localhost;port=1527"))

{

connection.Open();

using (var command = connection.CreateCommand())

{

command.CommandText = "CALL SYSCS\_UTIL.SYSCS\_GET\_ALL\_PROPERTIES()";

var dataReader = command.ExecuteReader();

while(dataReader.read())

{

//Every row data will be present in dataReader object. Column index can be indicated inside square brackets.

var data1 = dataReader[0];

}

}

connection.Close();

}