Using IBM Database Add-ins for Visual Studio 2013 in DB2 Cancun (10.5 Fix Pack 4)

Rajendra Kamath (rajkamat@in.ibm.com)
Engineering Manager
IBM

Malarvizhi Kaliyakrishnan (malarvizhi@in.ibm.com)
Systems Software Engineer, Informix CSDK Team, India
Software Lab
IBM

Archana Soni (archsoni@in.ibm.com)
Senior Software Engineer
IBM

This tutorial explains the key new capabilities in IBM Database Add-ins for Visual Studio 2013 available with the DB2® 10.5 Fix Pack 4. The authors explain support of the Microsoft Visual Studio 2013 feature set with IBM data servers (DB2 for z/OS®; DB2 for i; DB2 for Linux, UNIX, and Windows; and Informix®).

Overview

With DB2 10.5 Fix Pack 4, IBM Database Add-ins for Visual Studio supports Microsoft Visual Studio 2013, allowing applications connecting to IBM database servers to work with Visual Studio 2013. We will introduce key features; provide an overview of schema filter and DbName filter, which help improve performance; and offer detailed information about using stored procedures.

Schema filter

While working with Visual Studio, one challenge is the duration it takes to fetch information from large schemas. Using the Schema filter feature set, it is possible to filter and fetch only the required objects — by filtering out unimportant DB objects. The usefulness of Schema filter is apparent in database environments with large schemas, which is common in typical enterprise database ecosystems. For instance, when connecting to DB2 for i, if we do not filter the objects, all schemas in QSYS and QSYS2 will be fetched, affecting performance.

When using IBM Database Add-ins, a Schema filter can be set while adding a connection or while modifying an existing connection. Here is a step-by-step guide for setting the Schema filter for
the above-mentioned scenarios. It is assumed that the system already has a schema named INVENTORY and RICK.

A. Setting the Schema filter when adding a new connection

1. In Visual Studio 2013, right-click the Data Connections node of Server Explorer, and select Add Connection.

2. When the Change Data Source wizard appears, select IBM DB2 and IDS Servers as the data source. Ensure that IBM DB2 and IDS Data Provider for .NET is selected as the data provider, then click OK.

3. The Add Connection dialog box appears, where you'll enter the following connection information:
   a. Server name
   b. Port number
   c. User name
   d. Password
   e. Database name
**Note**: Server name and port number should be of this format `<ServerName>:\<Port>`.

**Figure 3. Add Connection dialog box with server name and port number format information**

4. Click to expand **Specify filtering options** and enter the schema name in the Schema filter text box.
Figure 4. Add Connection dialog box with Schema filter information

In the Schema filter, a user can enter multiple schema names separated by space or comma.
Figure 5. Multiple schema

Note: Schema filter is case-sensitive, so to get correct filtering output, the user must specify Schema name in the appropriate case.

5. Once you click OK, a new connection will be added in the Server Explorer window, and only filtered objects will be listed.

Figure 6. Expanded connection node

Setting the Schema filter in an existing connection

To set the Schema filter when a connection already exists, right-click on the respective connection node and select Modify Connection from the context menu.
Follow steps 4-5 of Scenario A to filter out any database schemas that are not of interest.

**DbName filter**

With DB2 for z/OS, there is an additional filtering option called DbName filter. The DbName filter helps to reduce the number of tables enumerated in the Server Explorer window inside the Visual Studio IDE. The DbName filter not only helps to filter out unwanted tables but ensures that any table creation operations are applied only to the specified database.

```sql
create database DBZ4EVR
```

**Setting the DbName filter**

Similar to the Schema filter, the DbName filter must be set in the Add Connection dialog box.
On clicking **OK**, the DbName filter is applied to table enumeration in Server Explorer and the reduced number of tables listed.

**Figure 9. Table enumeration**

Creating tables with DbName DBZ4EVR

In the Data Connections node in Server Explorer, expand the connection. Right-click the **Tables** folder and select **Add New Table with Designer**.
In the Table Designer window, fill required fields and click Save. The table is created with DbName DBZ4EVR and is enumerated in the Tables folder in Server Explorer.

DbName Filter can also be used in conjunction with Schema Filter.

**Working on stored procedures with DataSet**

With the help of IBM Database Add-ins and Visual Studio Rapid Application Development tools, fetching data via stored procedure is simplified. Adding data set in the project helps perform the DML operation on a table using the stored procedure.

The next section illustrates how to perform the INSERT and FETCH (SELECT) operations on a table through stored procedure using Visual Studio 2013. This is illustrated with a Windows Forms application as an example. It is assumed that the system has procedures with names
MYFETCHPROCEDURE and MYINSERTPROCEDURE. The scripts for the stored procedures and table used in the illustration is an example.

**Listing 1. Scripts for stored procedures and table**

### MYTABLE:

```sql
CREATE TABLE INVENTORY.MYTABLE (
    ID INTEGER NOT NULL,
    NAME VARCHAR (35),
    MOBILE BIGINT,
    ADDRESS VARCHAR (50),
    EMAIL VARCHAR (40),
    CONSTRAINT MYTABLE_ID_PK PRIMARY KEY (ID))
```

### MYFETCHPROCEDURE:

```sql
CREATE PROCEDURE INVENTORY.MYFETCHPROCEDURE() RESULT SETS 1 LANGUAGE SQL MODIFIES SQL DATA
P1: BEGIN
    DECLARE DB2_SP_SQL1 CURSOR WITH RETURN FOR
    Select ID, NAME, MOBILE, ADDRESS, EMAIL FROM INVENTORY.MYTABLE;
    OPEN DB2_SP_SQL1;
END P1
```

### MYINSERTPROCEDURE:

```sql
CREATE PROCEDURE INVENTORY.MYINSERTPROCEDURE(IN ID INTEGER, IN NAME VARCHAR (35),
IN MOBILE BIGINT, IN ADDRESS VARCHAR (50), IN EMAIL VARCHAR (40)) LANGUAGE SQL MODIFIES SQL DATA
P1: BEGIN
    INSERT INTO INVENTORY.MYTABLE VALUES (ID, NAME, MOBILE, ADDRESS, EMAIL);
END P1
```

### A. Fetching data from a table via stored procedure

1. Launch Visual Studio 2013 and choose **File > New > Project**. From the Installed templates section of the New Project Wizard, select **Visual C# > Windows Forms Application**. Type in the project name (MyWinFormApp, for example) and click **OK**.
2. In the Solution explorer, right-click on Project node and go to **Add > New Item**, select **DataSet**, type the name as **DB2DataSet**, and click **Add**.
3. From the Server Explorer, drag and drop **MYFETCHPROCEDURE** procedure onto **DB2DataSet Designer**.

**Figure 14. Adding MYFETCHPROCEDURE onto DB2DataSet Designer**

4. From the tool box, drag and drop **DataGridView** onto **Form1 Designer**.
5. Click on **DataGridView** tasks and in **Choose Data Source**, browse for **Other Data Sources** > **Project Data Sources** > **DB2DataSet** and select **MYFETCHPROCEDURE**.

**Figure 16. Selecting data source for DataGridView**

Once the user selects a procedure, the DataGridView will have all the columns returned by the procedure.
Figure 17. DataGridView with MYFETCHPROOCEDURE Data Source

The user can hide the columns by editing DataGridView. Also without running the project, the user can see resulting data returned by the procedure by using the Preview Data option.

Figure 18. DataGridView Tasks

6. To execute the stored procedure, press **F5**.

Figure 19. Executed stored procedure results
B. Inserting data into a table via stored procedure

1. Drag and drop stored procedure onto **DB2DataSet Designer**. Below, we have dragged and dropped **MYINSERTPROCEDURE**.

   **Figure 20. Adding MYINSERTPROCEDURE onto DB2DataSet Designer**

2. Right-click on **MYINSERTPROCEDURE TableAdapter** and choose **Configure**.

   **Figure 21. Configure MYINSERTPROCEDURE**

3. In the TableAdapter Configuration Wizard, select **MYINSERTPROCEDURE** from the Insert drop-down list.
Follow the wizard instructions and click **Finish**.

4. On the Form Designer, drag and drop another **DataGridView** and add new columns, through which we will insert data.

**Figure 23. Add Columns into DataGridView through Add Column menu**

For example, we have now added Columns, which are the **INPUT** parameter of **MYINSERTPROCEDURE**.
5. Create a button, and on click event, we will insert data. Optionally, we can add one more buttons to refresh DataGridView. Finally, the UI is as shown below.

**Figure 25. Form1 Designer**

6. On click event of INSERT button, add the following code:

```csharp
this.myINSERTPROCEDURETableAdapter.Insert(Convert.ToInt32(row.Cells[0].Value.ToString()),
row.Cells[1].Value.ToString(),
```

7. On click event of the REFRESH button, add:

```csharp
this.myFETCHPROCEDURETableAdapter.Fill(this.db2DataSet.MYFETCHPROCEDURE);
```

8. To execute the application, press **F5**.
Figure 26. Fetching data from the table

9. To insert data, input data in second DataGridView and click **INSERT**.

Figure 27. Inserting data in the table

10. To verify that data is inserted, click **REFRESH**.
We can now perform select and insert operations with IBM Database Add-Ins for Visual Studio in a Windows Forms application.

**Conclusion**

.NET developers working with the Visual Studio 2013 environment can now use the IBM Database Add-Ins for Visual Studio with Visual Studio 2013 Support (10.5 Fix Pack 4 or higher) to connect to IBM data servers. This makes it possible to leverage Visual Studio 2013 capabilities for building next-generation .NET applications targeting IBM data servers.
Resources

Learn

- Check out Guidelines for Improving Performance with the IBM Database Add-Ins Cache.
- The Information Management area on developerWorks provides resources for architects, developers, and engineers.
- Stay current with developer technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Follow developerWorks on Twitter.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.

Get products and technologies

- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, use a product in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement Service Oriented Architecture efficiently.

Discuss

- Get involved in the developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.
About the authors

Rajendra Kamath

Rajendra Kamath currently leads the IBM Common Client and Tooling team that includes .NET, CLI/ODBC, and Visual Studio Integration for applications targeting IBM Data Servers. His primary interests is on data driven application and tooling with IBM Data Servers.

Malarvizhi Kaliyakrishnan

Malarvizhi Kaliyakrishnan is a Systems Software Engineer working at IBM India Software Lab, with experience in .NET application development and Informix client components namely .NET and ODBC.

Archana Soni

Archana Soni is a senior software engineer working at IBM India Software Lab. She works on the development and support of IBM Database Add-Ins for Visual Studio.

© Copyright IBM Corporation 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)