

Introduction to M Programming



Agenda

- M Programming Syntax
- Writing M Code
- Query Folding
- Custom Connectors



Let Statement

- Queries created using let statement

The screenshot displays the 'Advanced Editor' interface for a query. The main editor area shows a 'Simple Let Statement' with the following code:

```
let  
  var1 = "Hello",  
  var2 = "World",  
  output = var1 & " " & var2  
in  
  output
```

Three red boxes highlight the variable assignments and the final output line, with red arrows pointing from each box to the 'QUERY SETTINGS' panel on the right. The 'QUERY SETTINGS' panel has a close button (X) in the top right corner. It contains two sections:

- PROPERTIES**
 - Name: Simple Let Statement
 - [All Properties](#)
- APPLIED STEPS**
 - var1
 - var2
 - output** (highlighted with a yellow bar and a small X icon)

At the bottom of the editor, a green checkmark indicates 'No syntax errors have been detected.' Below this are 'Done' and 'Cancel' buttons.



M Datatypes

```
let

  // primitives
  var1 = 123,      // number
  var2 = true,     // boolean
  var3 = "hello",  // text
  var4 = null,     // null

  // creating lists
  list1 = {1, 2, 3},      // list of three numbers

  // accessing list elements
  var5 = list1{1},

  // create records
  record1 = [ FirstName="Soupy", LastName="Sales", ID=3 ],

  // accessing records
  var6 = record1[FirstName],

  // table
  table1 = #table( {"A", "B"}, { {1, 2}, {3, 4} } ),

  // creating function
  function1 = (x) => x * 2,

  // calling function
  output = function1(var1)

in
  output
```



Initializing Dates and Times

```
// time
var1 = #time(09,15,00),

// date
var2 = #date(2013,02,26),

// date and time
var3 = #datetime(2013,02,26, 09,15,00),

// date and time in specific timezone
var4 = #datetimezone(2013,02,26, 09,15,00, 09,00),

// time durection
var5 = #duration(0,1,30,0),
```



Combination Operator (&)

- Used to combine strings, arrays and records

```
// text concatenation: "ABC"  
var1 = "A" & "BC",  
  
// list concatenation: {1, 2, 3}  
var2 = {1} & {2, 3},  
  
// record merge: [ a = 1, b = 2 ]  
var3 = [ a = 1 ] & [ b = 2 ],
```



For Each

```
let
  Source = #table( {"Col1", "Col2"},
                  { {1, 2}, {3, 4} } ),
  AddColumn = Table.AddColumn(Source, "Col3", each [Col1] + [Col2] )
in
  AddColumn
```



SQL Server and Query Folding

```
let
    Source = Sql.Database("cpt.database.windows.net", "wingtipSalesDB"),
    CustomersTableIdentifier = [ Item="Customers" ],
    CustomersTable = Source{CustomersTableIdentifier}[Data],
    FilteredRows = Table.SelectRows(CustomersTable, each [CustomerId] <= 10),
    ColumnsToRetrieve = {"CustomerId", "FirstName", "LastName"},
    SelectedColumns = Table.SelectColumns(FilteredReaders, ColumnsToRetrieve),
    MergedColumns = Table.CombineColumns(SelectedColumns,
                                         {"FirstName", "LastName"},
                                         Combiner.CombineTextByDelimiter(" ", QuoteStyle.None),
                                         "Customer"),
    Output = MergedColumns
in
    Output
```



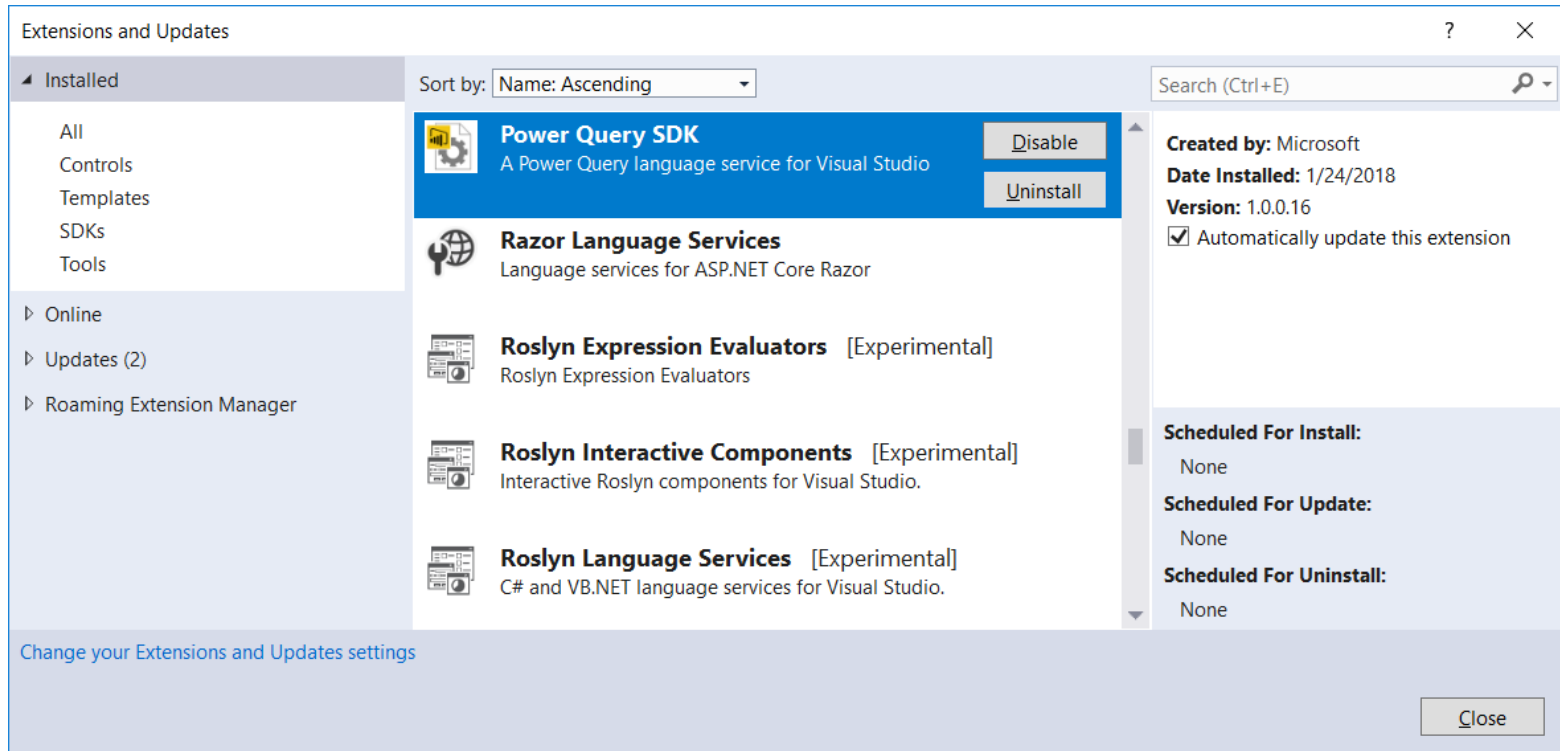
Native Queries

- No query folding occurs after native query

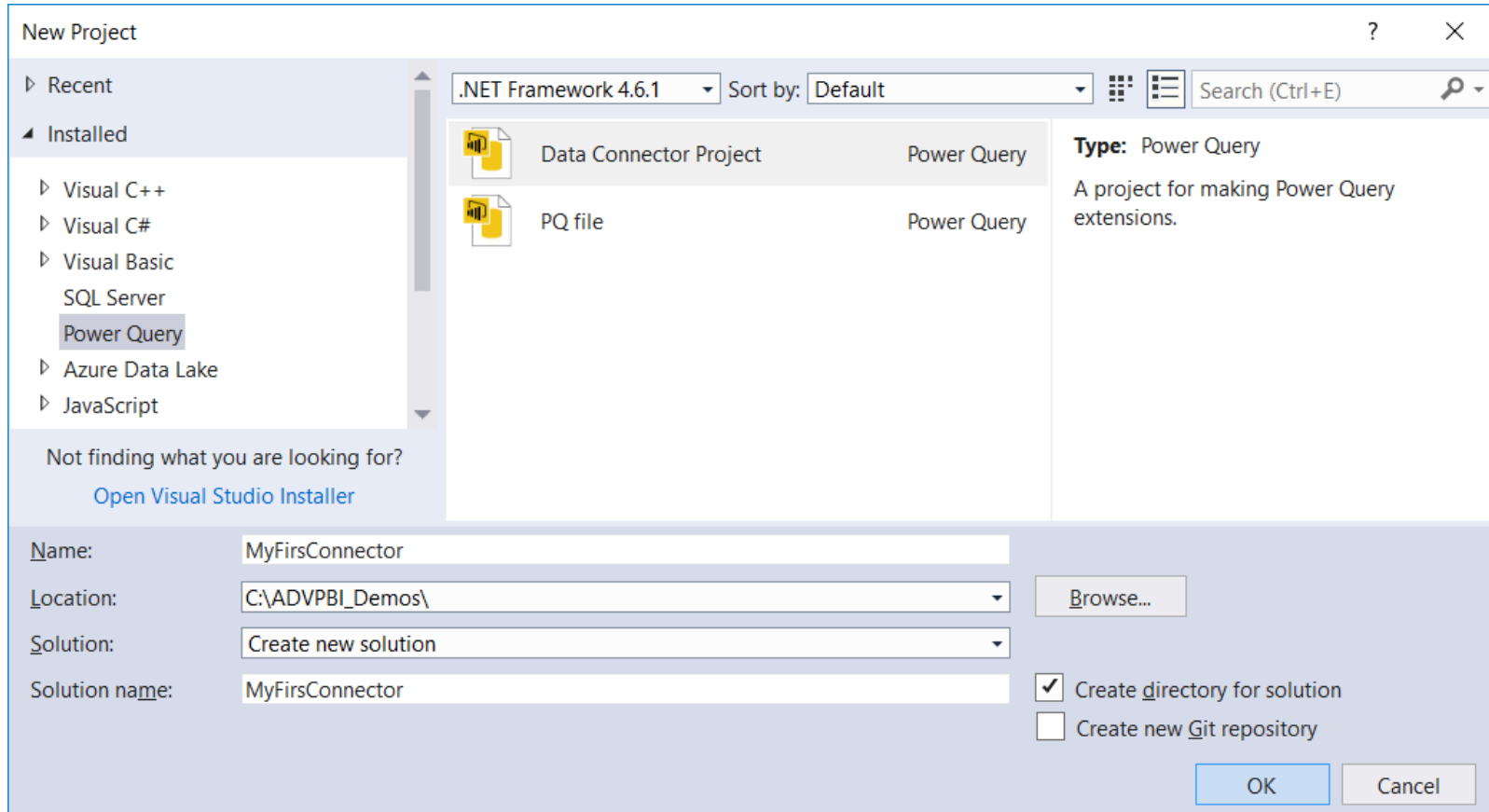
```
let
    DatabaseServer = "cpt.database.windows.net",
    DatabaseName = "WingtipSalesDB",
    SQL = "SELECT CustomerId, FirstName, LastName" &
        " FROM Customers" &
        " WHERE CustomerId <= 10" &
        " ORDER BY LastName, FirstName" ,
    Source = Sql.Database( DatabaseServer, DatabaseName , [Query=SQL] ),
    output = Source
in
    output
```



Power Query SDK



Creating a New Data Connector Project



The screenshot shows the 'New Project' dialog box in Visual Studio. The 'Installed' section on the left lists various project types, with 'Power Query' selected. The main pane displays two options: 'Data Connector Project' and 'PQ file', both categorized as 'Power Query'. The 'Type: Power Query' description on the right states: 'A project for making Power Query extensions.' The bottom section contains fields for project configuration: 'Name' (MyFirsConnector), 'Location' (C:\ADVPBI_Demos\), 'Solution' (Create new solution), and 'Solution name' (MyFirsConnector). There are also checkboxes for 'Create directory for solution' (checked) and 'Create new Git repository' (unchecked), along with 'Browse...', 'OK', and 'Cancel' buttons.

New Project

Recent

Installed

- Visual C++
- Visual C#
- Visual Basic
- SQL Server
- Power Query**
- Azure Data Lake
- JavaScript

Not finding what you are looking for?
[Open Visual Studio Installer](#)

.NET Framework 4.6.1 Sort by: Default Search (Ctrl+E)

Icon	Project Name	Type
	Data Connector Project	Power Query
	PQ file	Power Query

Type: Power Query
A project for making Power Query extensions.

Name: MyFirsConnector

Location: C:\ADVPBI_Demos\ Browse...

Solution: Create new solution

Solution name: MyFirsConnector

☒ Create directory for solution
☐ Create new Git repository

OK Cancel



Summary

- M Programming Syntax
- Writing M Code
- Query Folding
- Custom Connectors

