Execute Rocket MV U2 Subroutine Asynchronously using C# (async\await) and U2 Toolkit for .NET. Convert Subroutine Multi-Value Output to Json/Objects/DataTable

# Overview

Asynchronous programming involves executing operations in the background so that the main thread can continue its own operations. This way the main thread can keep the user interface responsive while the background thread is processing the task at hand. .NET framework 4.5 introduced the C# (async/await) and VB.NET (Async\Await) keywords that simplify asynchronous programming. If you pull huge amounts of data from a database synchronously then the entire UI will block and the user is unable to do anything until all database operations finish. But if we do it asynchronously then the user is free to do their work while the data is loading in the background.

U2 Toolkit for .NET v2.2.0 implements ADO.NET’s asynchronous API such as

* ExecuteNonQueryAsync()
* ExecuteReaderAsync()
* ExecuteScalarAsync()

These APIs allows you to execute U2 Subroutines and return multi-value data as U2Parameter’s output value. You can also return multi-column multi-rows of data like result set (DataSet/DataTable) from U2 Subroutines (UniVerse Only).

You can use U2Parameter class to convert multi-value output data to Json or Objects or DataTable.

Asynchronous features can be used in the following applications

* ASP.NET MVC
* Web API and oData
* Node.js\Edge.js (Server Side) and Angular.js (Client Side)
* WPF, WinForm
* ASP.NET Web Form
* Mobile and Cloud Development

This article demonstrate the following walkthroughs

1. [Execute Simple U2 subroutine](#_Walkthrough_Subroutine)
2. [Execute U2 Subroutine, return multi-value data as output, convert multi-value data to DataTable](#_Walkthrough_Subroutine_DataTable)
3. [Execute U2 Subroutine, return multi-value data as output, convert multi-value data to Objects](#_Walkthrough_Subroutine_Objects)
4. [Execute U2 Subroutine, return multi-value data as output, convert multi-value data to Json](#_Walkthrough_Subroutine_Json)
5. [Execute U2 Subroutine, return result set (DataSet/DataTable) [UniVerse Only]](#_Walkthrough_Subroutine__SQLExecDire)

# Requirements

|  |  |
| --- | --- |
| **U2** | **Version** |
| UniVerse | **10.3 or later** |
| UniData | **7.1 or later** |
| U2 Toolkit for .NET | **2.2.0** |

|  |  |
| --- | --- |
| **Microsoft** | **Version** |
| Visual Studio | * **2012 Update 4** * **2013 Update 4** |

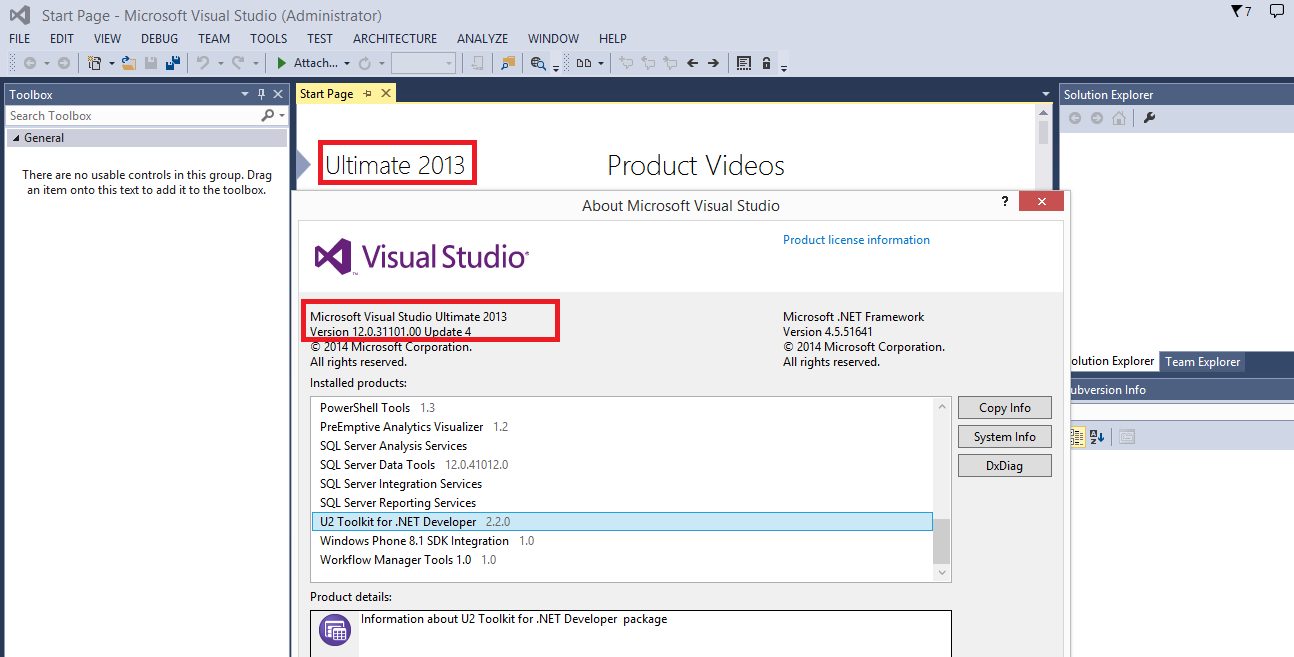
# Execute U2 Subroutine

In this article, we will provide the following C# Sample Code.

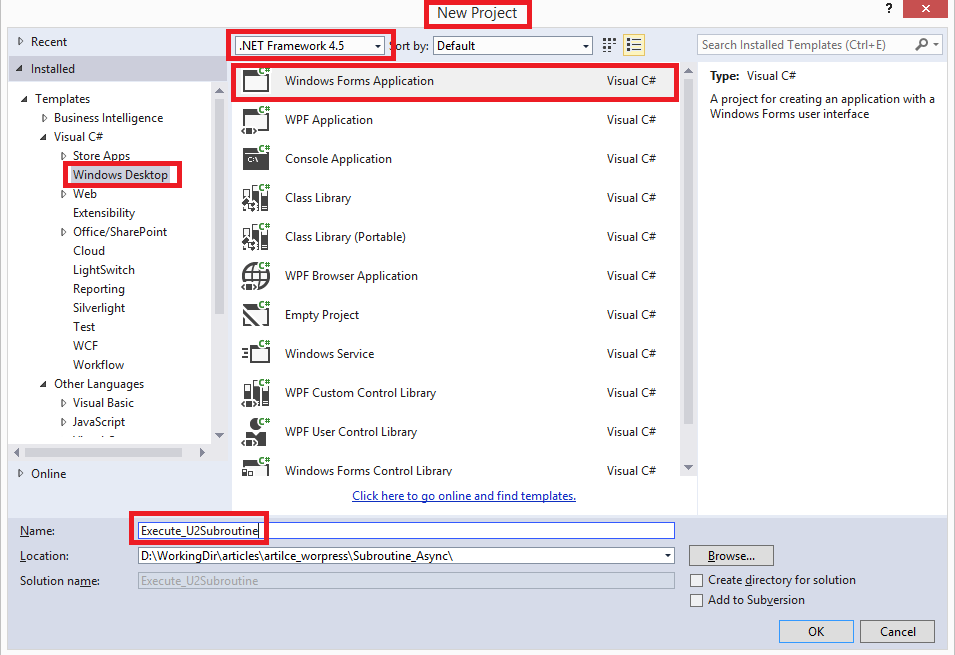
|  |  |
| --- | --- |
| **U2 Subroutine** | **Remarks** |
| Execute Subroutine Asynchronously | [See Walkthrough Subroutine](#_Walkthrough_Subroutine) |
| Execute Subroutine Asynchronously and Convert returned multi-value data to DataTable | [See Walkthrough Subroutine\_DataTable](#_Walkthrough_Subroutine_DataTable)   * Provide Schema (DataTable) for returned multi-value data |
| Execute Subroutine Asynchronously and Convert returned multi-value data to Objects | [See Walkthrough Subroutine\_Objects](#_Walkthrough_Subroutine_Objects)   * Provide Schema (POCO Class) for returned multi-value data |
| Execute Subroutine Asynchronously and Convert returned multi-value data to Json | [See Walkthrough Subroutine\_Json](#_Walkthrough_Subroutine_Json)   * Provide Schema (POCO Class) for returned multi-value data |
| Execute Subroutine Asynchronously and return multi-value data as multi-columns/multi-rows | [See Walkthrough Subroutine\_ SQLExecDirect](#_Walkthrough_Subroutine__SQLExecDire)   * UniVerse Only |

# Walkthrough Subroutine

## Start Visual Studio 2013

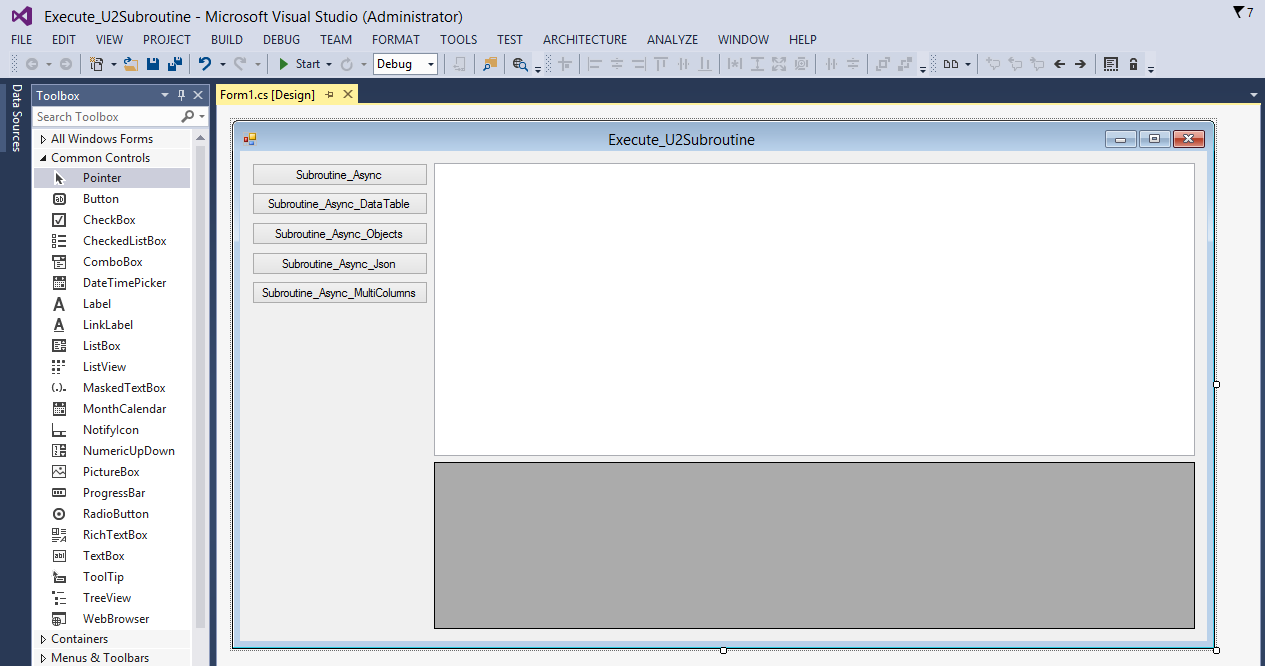


## Create Web Form C# Project

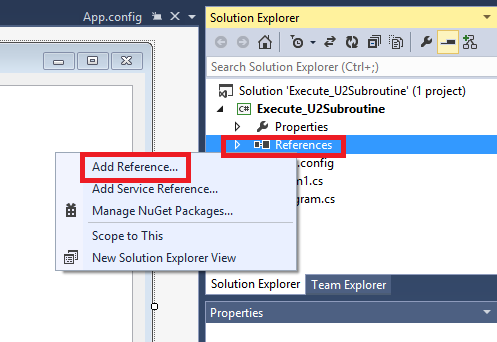


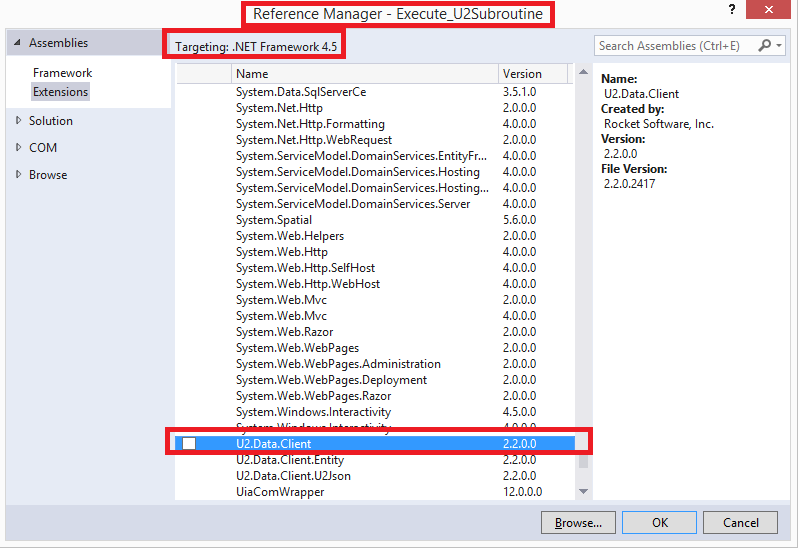
## Design Web Form

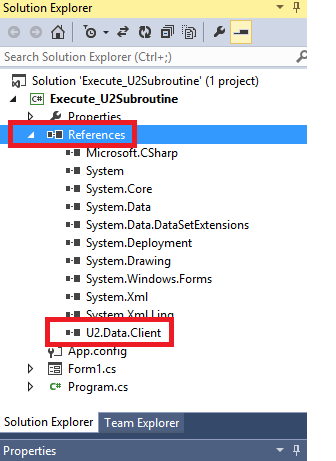
Drag and Drop Button Controls, Text Box Control and DataGridView Control into Form Designer



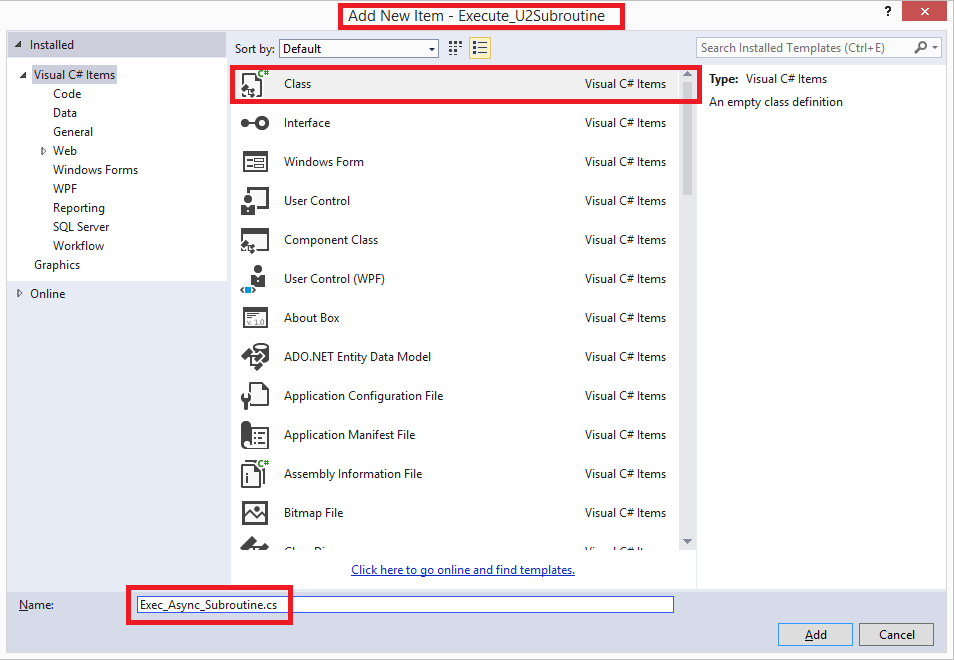
## Add Reference to U2.Data.Client.dll (v4.5)







## Create C# Class : Exec\_Async\_Subroutine



## Create Async Method

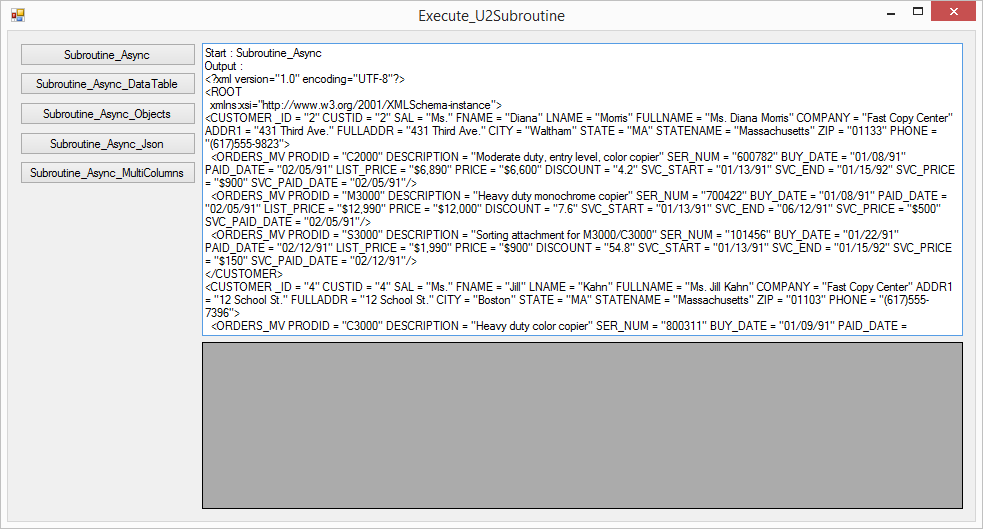
|  |
| --- |
| public static async Task<string> CallSubroutine()  {  // see GitHub for full source code  } |

## Call Async Method

Create Event Handler for Button Control “Subroutine\_Async”.

|  |
| --- |
| private async void button1\_Click(object sender, EventArgs e)  {  try  {  this.textBox1.AppendText("Start : Subroutine\_Async" +Environment.NewLine);  string lRetVal = await Exec\_Async\_Subroutine.CallSubroutine();  this.textBox1.AppendText("Output : " + Environment.NewLine);  this.textBox1.AppendText(lRetVal);  this.textBox1.AppendText("End : Subroutine\_Async" + Environment.NewLine);  }  catch (Exception e3)  {  this.textBox1.AppendText("Error : Subroutine\_Async:" + e3.Message + Environment.NewLine);  }  } |

## Run App and press Button Control “Subroutine\_Async”



# Walkthrough Subroutine\_DataTable

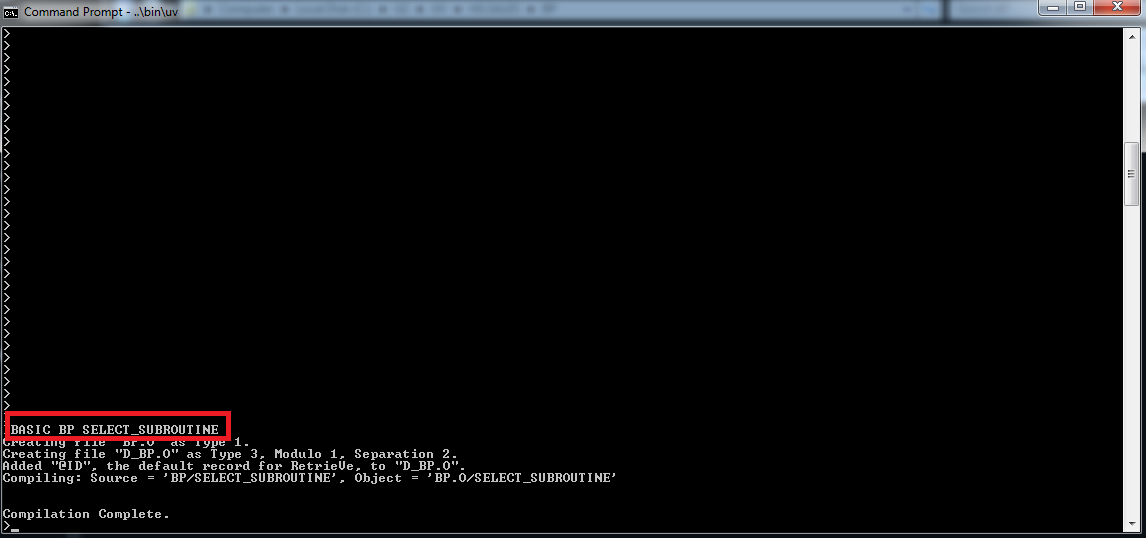
## Create Simple U2 Subroutine

This subroutines returns multi-value data using ARG\_OUTPUT argument

|  |
| --- |
| SUBROUTINE SELECT\_SUBROUTINE(ARG\_INPUT,ARG\_OUTPUT)  x = ARG\_INPUT  ARG\_OUTPUT = "100":@VM:"101":@VM:"102":@VM:"103":@FM:"Nancy":@VM:"Andrew":@VM:"Janet":@VM:"Margaret":@FM:"01/06/1991":@VM:"06/07/1996":@VM:"11/08/1999":@VM:"12/10/2001"  RETURN |

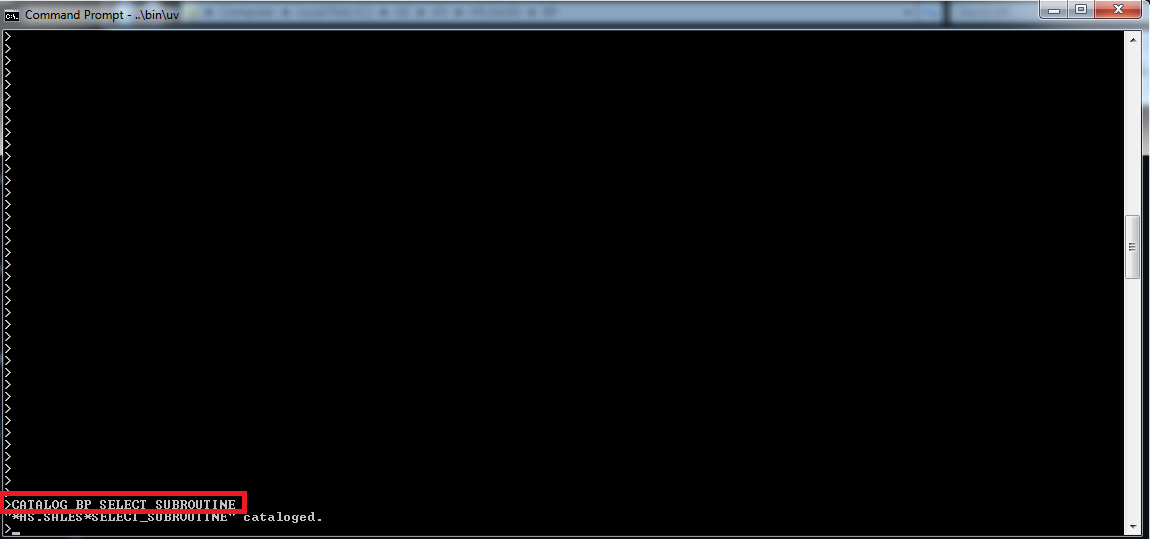
## Compile Simple U2 Subroutine

>BASIC BP SELECT\_SUBROUTINE

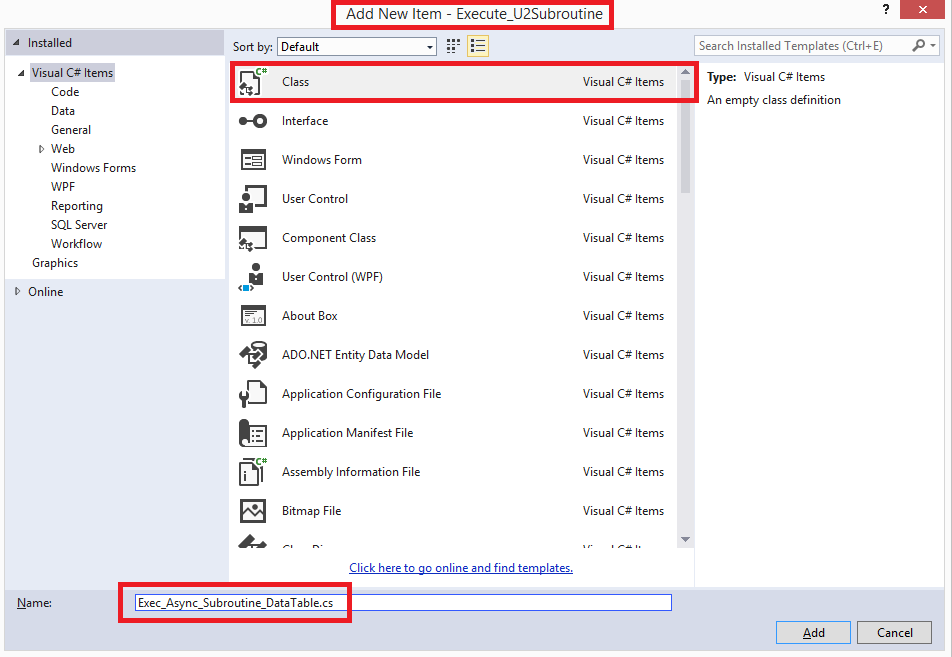


## Catalog Simple U2 Subroutine

>CATALOG BP SELECT\_SUBROUTINE



## Create C# Class : Exec\_Async\_Subroutine\_DataTable Class



## Create Schema of multi-value data

|  |
| --- |
| DataTable lRetDT = new DataTable("EmpTable");  lRetDT.Columns.Add("ID", typeof(Int32));  lRetDT.Columns.Add("Name", typeof(string));  lRetDT.Columns.Add("HireDate", typeof(DateTime)); |

## Create Async Method

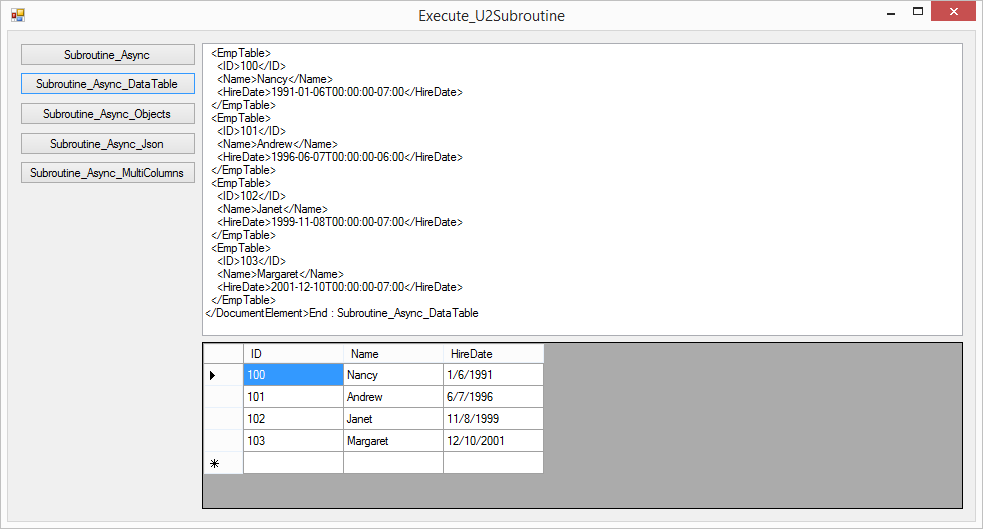
|  |
| --- |
| public static async Task<DataTable> CallSubroutine()  {  // see GitHub for full source code  } |

## Call Async Method

Create Event Handler for Button Control “Subroutine\_Async\_DataTable”.

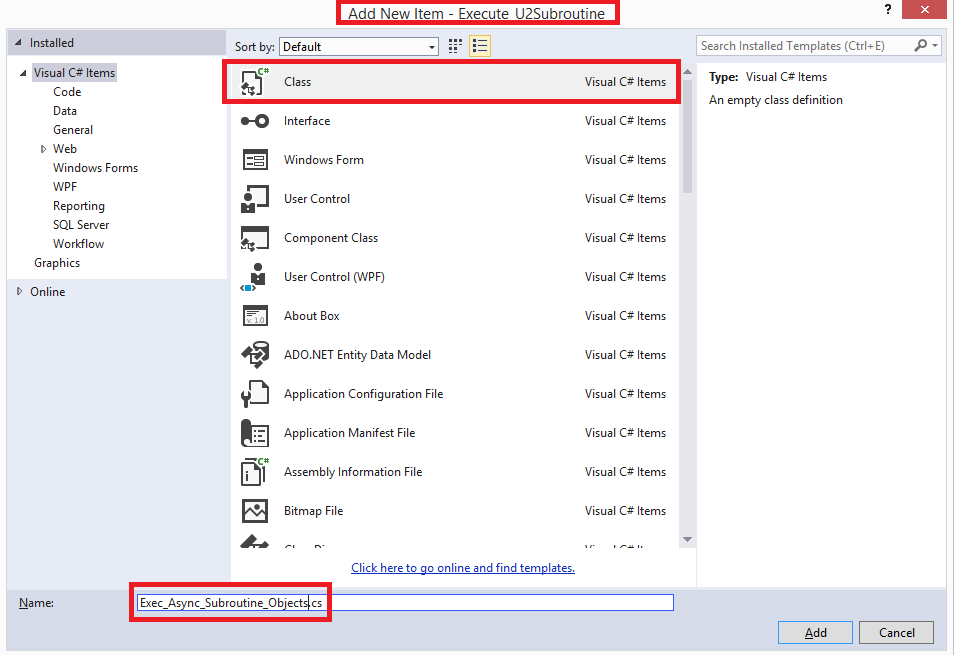
|  |
| --- |
| private async void button2\_Click(object sender, EventArgs e)  {  try  {  this.textBox1.AppendText("Start : Subroutine\_Async\_DataTable" + Environment.NewLine);  DataTable lRetValDT = await Exec\_Async\_Subroutine\_DataTable.CallSubroutine();  this.textBox1.AppendText("Output : " + Environment.NewLine);  System.IO.StringWriter writer = new System.IO.StringWriter();  lRetValDT.WriteXml(writer, true);  this.textBox1.AppendText(writer.ToString());  this.dataGridView1.DataSource = lRetValDT;  this.textBox1.AppendText("End : Subroutine\_Async\_DataTable" + Environment.NewLine);  }  catch (Exception e3)  {  this.textBox1.AppendText("Error : Subroutine\_Async\_DataTable:" + e3.Message + Environment.NewLine);  }  } |

## Run App and press Button Control “Subroutine\_Async\_DataTable”



# Walkthrough Subroutine\_Objects

## Create C# Class : Exec\_Async\_Subroutine\_DataTable Class



## Create Schema of multi-value data

|  |
| --- |
| public class Employee  {  public int ID { get; set; }  public string Name { get; set; }  public DateTime HireDate { get; set; }  } |

## Create Async Method

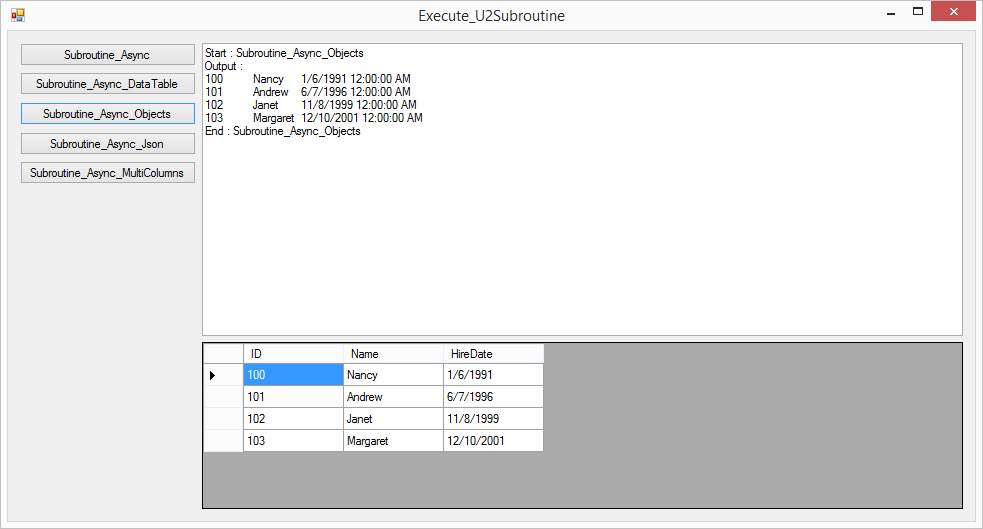
|  |
| --- |
| public static async Task<Employee> CallSubroutine()  {  // see GitHub for full source code  } |

## Call Async Method

Create Event Handler for Button Control “Subroutine\_Async\_Objects”.

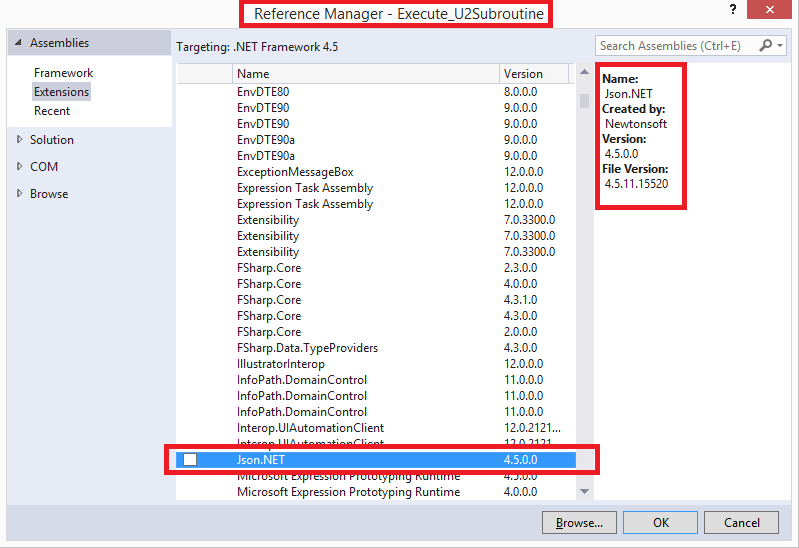
|  |
| --- |
| private async void button3\_Click(object sender, EventArgs e)  {  try  {  this.textBox1.AppendText("Start : Subroutine\_Async\_Objects" + Environment.NewLine);  List<Employee> lRetEmpList = await Exec\_Async\_Subroutine\_Objects.CallSubroutine();  this.textBox1.AppendText("Output : " + Environment.NewLine);  foreach (var item in lRetEmpList)  {  this.textBox1.AppendText(item.ID+"\t"+item.Name+"\t"+item.HireDate + Environment.NewLine);  }    this.dataGridView1.DataSource = lRetEmpList;  this.textBox1.AppendText("End : Subroutine\_Async\_Objects" + Environment.NewLine);  }  catch (Exception e3)  {  this.textBox1.AppendText("Error : Subroutine\_Async\_Objects:" + e3.Message + Environment.NewLine);  }  } |

## Run App and press Button Control “Subroutine\_Async\_Objects”

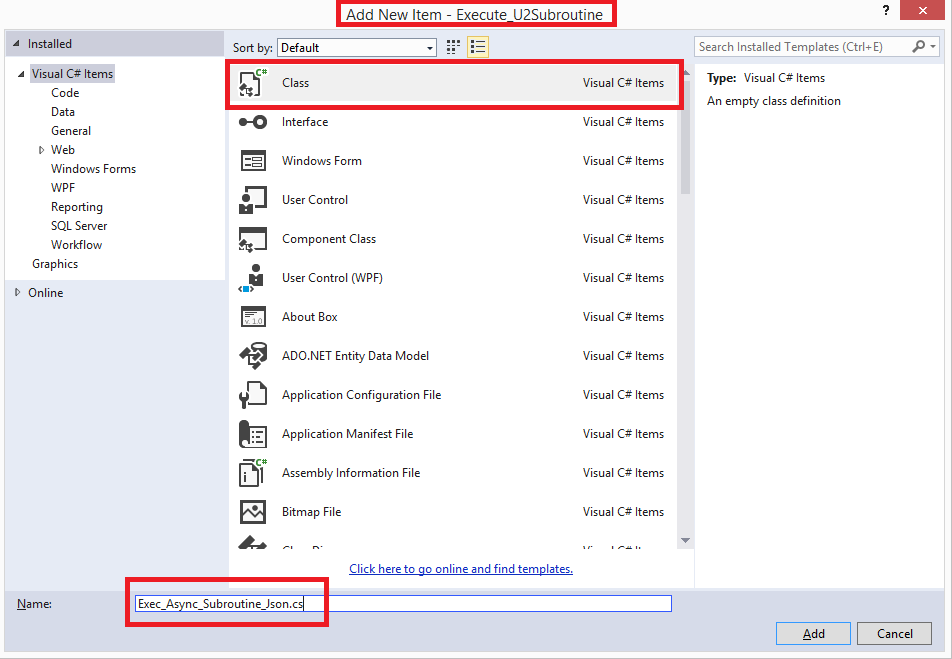


# Walkthrough Subroutine\_Json

## Add Reference Json.NET



## Create C# Class : Exec\_Async\_Subroutine\_Json Class



## Create Schema of multi-value data

|  |
| --- |
| public class Employee  {  public int ID { get; set; }  public string Name { get; set; }  public DateTime HireDate { get; set; }  } |

## Create Async Method

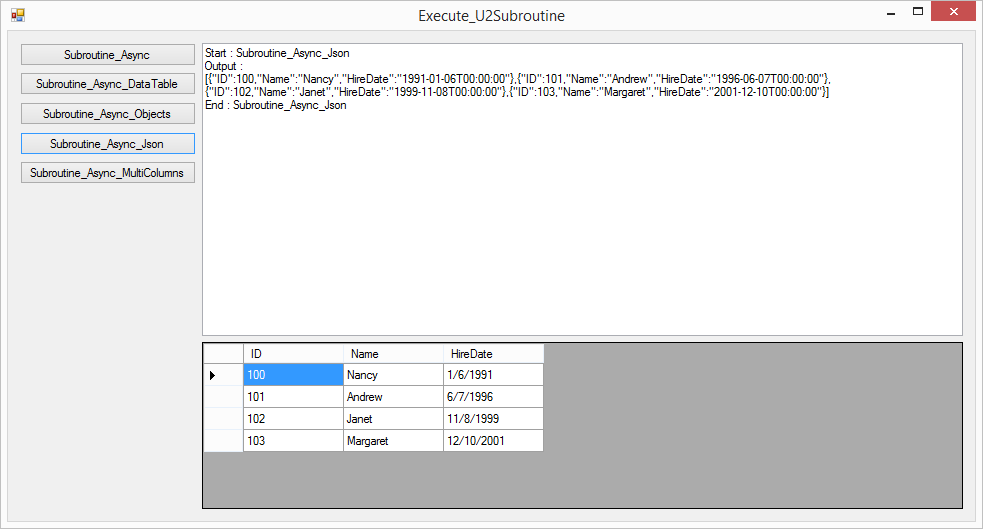
|  |
| --- |
| public static async Task<string> CallSubroutine()  {  // see GitHub for full source code  } |

## Call Async Method

Create Event Handler for Button Control “Subroutine\_Async\_Json”.

|  |
| --- |
| private async void button4\_Click(object sender, EventArgs e)  {  try  {  this.textBox1.AppendText("Start : Subroutine\_Async\_Json" + Environment.NewLine);  string lRetVal = await Exec\_Async\_Subroutine\_Json.CallSubroutine();  this.textBox1.AppendText("Output : " + Environment.NewLine);  this.textBox1.AppendText(lRetVal);  var result = Newtonsoft.Json.JsonConvert.DeserializeObject<List<Employee>>(lRetVal);  this.dataGridView1.DataSource = result;  this.textBox1.AppendText( Environment.NewLine);  this.textBox1.AppendText("End : Subroutine\_Async\_Json" + Environment.NewLine);  }  catch (Exception e3)  {  this.textBox1.AppendText("Error : Subroutine\_Async\_Json:" + e3.Message + Environment.NewLine);  }  } |

## Run App and press Button Control “Subroutine\_Async\_Json”



# Walkthrough Subroutine\_ SQLExecDirect

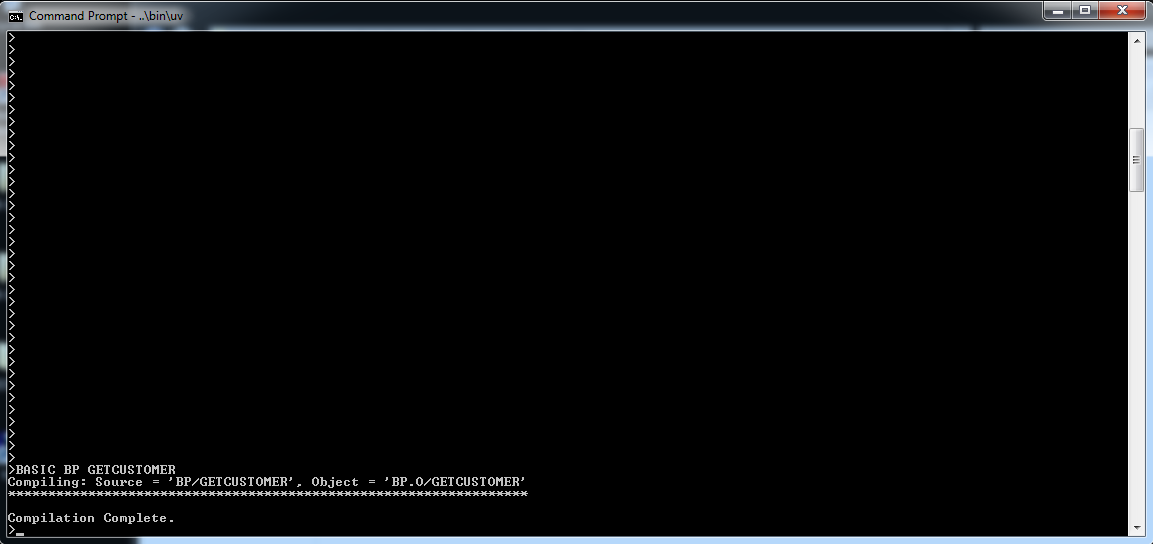
## Create Simple U2 Subroutine

This subroutines returns multi-columns result set (Universe Only)

|  |
| --- |
| SUBROUTINE GETCUSTOMER  $INCLUDE UNIVERSE.INCLUDE ODBC.H  SELSTMT = "SELECT \* FROM CUSTOMER"  ST = SQLExecDirect(@HSTMT, SELSTMT)  RETURN |

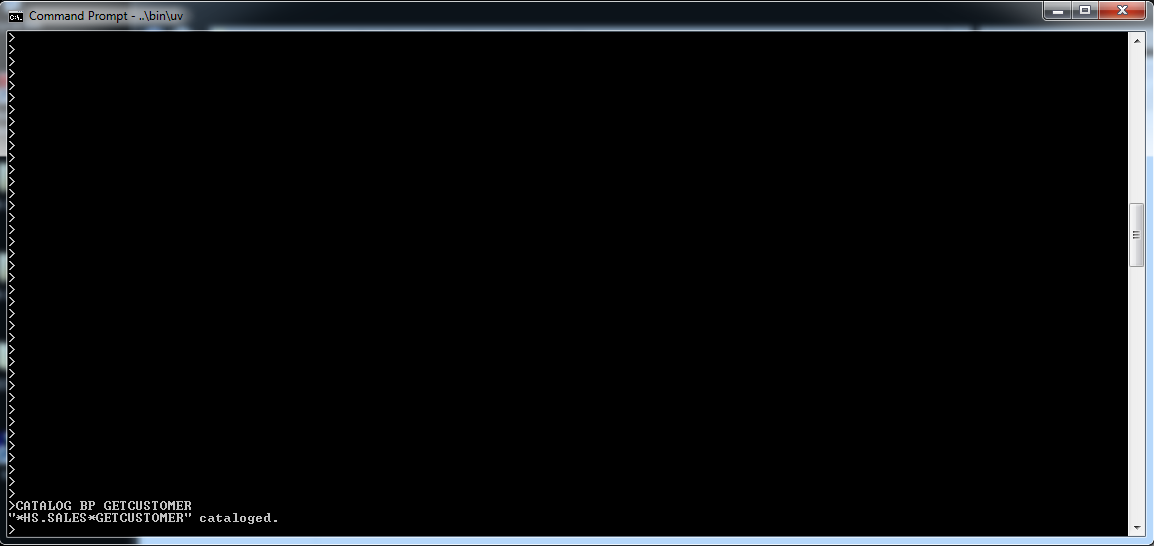
## Compile Simple U2 Subroutine

>BASIC BP GETCUSTOMER

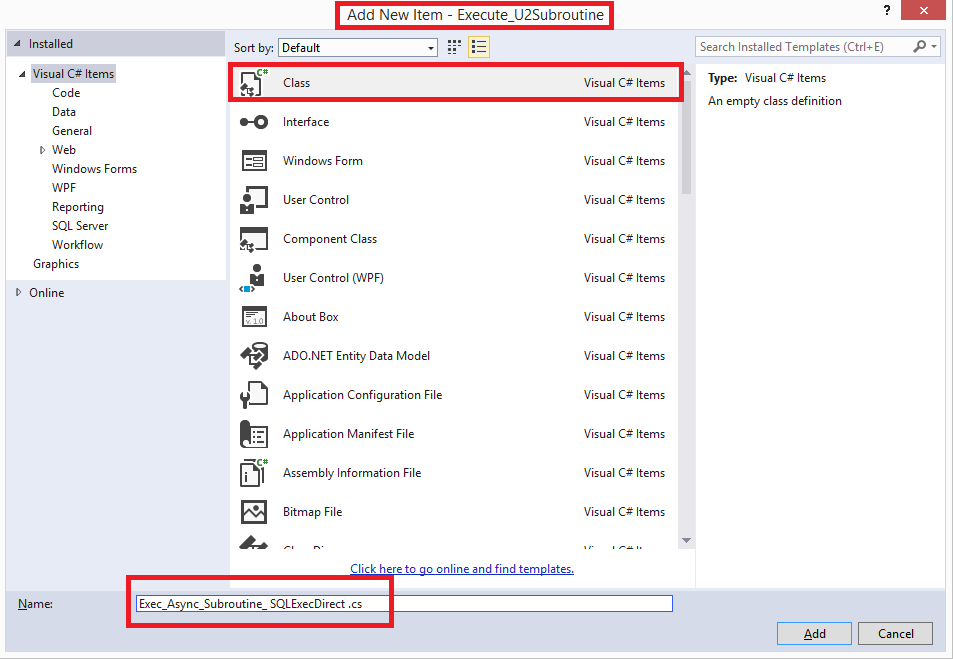


## Catalog Simple U2 Subroutine

>CATALOG BP GETCUSTOMER



## Create C# Class : Exec\_Async\_Subroutine\_ SQLExecDirect Class



## Create Schema of multi-value data

|  |
| --- |
| public class Customer  {  public int CustomerId { get; set; }  public string FirstName { get; set; }  public string LastName { get; set; }  } |

## Create Async Method

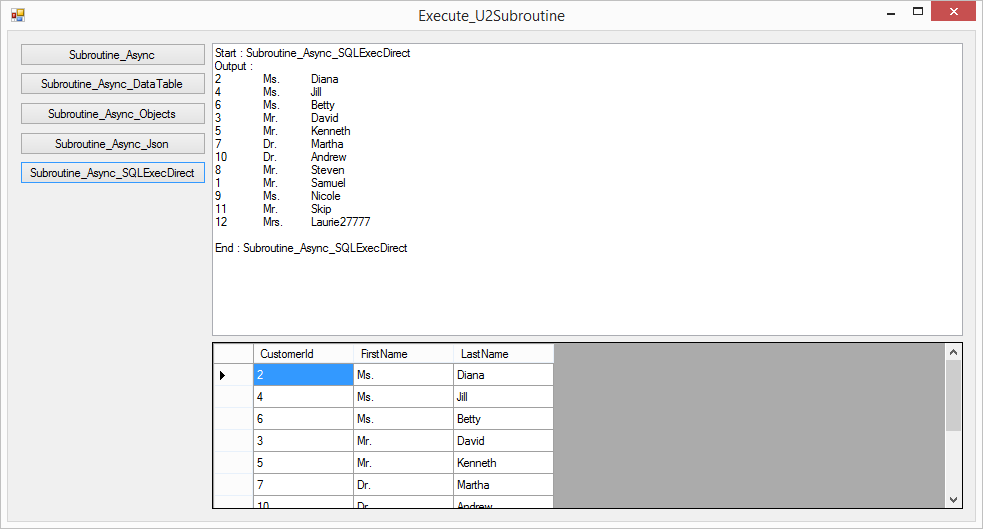
|  |
| --- |
| public static async Task<List<Customer>> CallSubroutine()  {  // see GitHub for full source code  } |

## Call Async Method

Create Event Handler for Button Control “Subroutine\_Async\_ SQLExecDirect”.

|  |
| --- |
| private async void button5\_Click(object sender, EventArgs e)  {  try  {  this.textBox1.AppendText("Start : Subroutine\_Async\_SQLExecDirect" + Environment.NewLine);  List<Customer> lRetEmpList = await Exec\_Async\_Subroutine\_SQLExecDirect.CallSubroutine();  this.textBox1.AppendText("Output : " + Environment.NewLine);  foreach (var item in lRetEmpList)  {  this.textBox1.AppendText(item.CustomerId + "\t" + item.FirstName + "\t" + item.LastName + Environment.NewLine);  }  this.dataGridView1.DataSource = lRetEmpList;  this.textBox1.AppendText(Environment.NewLine);  this.textBox1.AppendText("End : Subroutine\_Async\_SQLExecDirect" + Environment.NewLine);  }  catch (Exception e3)  {  this.textBox1.AppendText("Error : Subroutine\_Async\_SQLExecDirect:" + e3.Message + Environment.NewLine);  }  } |

## Run App and press Button Control “Subroutine\_Async\_ SQLExecDirect”



# Conclusion

.NET Framework 4.5, ADO.NET 4.5, Entity Framework 6.1 and Visual Studio 2013 introduced support for asynchronous operations. U2 Toolkit for .NET v2.2.0 implemented required ADO.NET Asynchronous API specification. Using these asynchronous methods along with the async/await pattern you can query the U2 Database or execute U2 Subroutine Asynchronously. You can also save the changes to the database in asynchronous manner. Asynchronous U2 Database calls will gain so much more responsiveness to our applications.

# Reference

[Asynchronous Programming](http://msdn.microsoft.com/en-us/library/hh211418(v=vs.110).aspx)

[Using SqlDataReader’s new async methods in .Net 4.5 (Part 1)](http://blogs.msdn.com/b/adonet/archive/2012/04/20/using-sqldatareader-s-new-async-methods-in-net-4-5-beta.aspx)

[Using SqlDataReader’s new async methods in .Net 4.5 (Part 2)](http://msdn.microsoft.com/en-us/library/vstudio/hh191443.aspx)

[Asynchronous Programming with Async and Await (C# and Visual Basic)](http://msdn.microsoft.com/en-us/library/vstudio/hh191443.aspx)

# Source Code Download

[Download Source Code from GitHub](https://github.com/RocketSoftware/u2-servers-lab/tree/master/U2-Toolkit)