(T18)討論CommandLine執行SqlMetal自動生成LinqToSql  
CourseGUID: 5ba9a6fe-7475-4b0c-8b99-bbcf7f5e2e1c  
=======================================================================  
(T18)討論CommandLine執行SqlMetal自動生成LinqToSql  
=======================================================================  
0. Introduction

-----------

1. Web Form Application - Linq Query

1.1. TSQL

1.2. Set up SQL Authentication

-----------

2. Using SqlMetal to create DBML

-----------

3. Asp.Net Web Application with existing DBML

3.1. Create Web Application

3.2. Create Web Application

3.3. Add existing DBML

3.4. WebForm1.aspx

3.4.1. WebForm1.aspx

3.4.2. WebForm1.aspx.cs  
=======================================================================

0. Introduction

In the previous tutorial, we use visual studio to create Linq to Sql classes.

This tutorial will demonstrate using SqlMetal to create Linq to Sql classes.

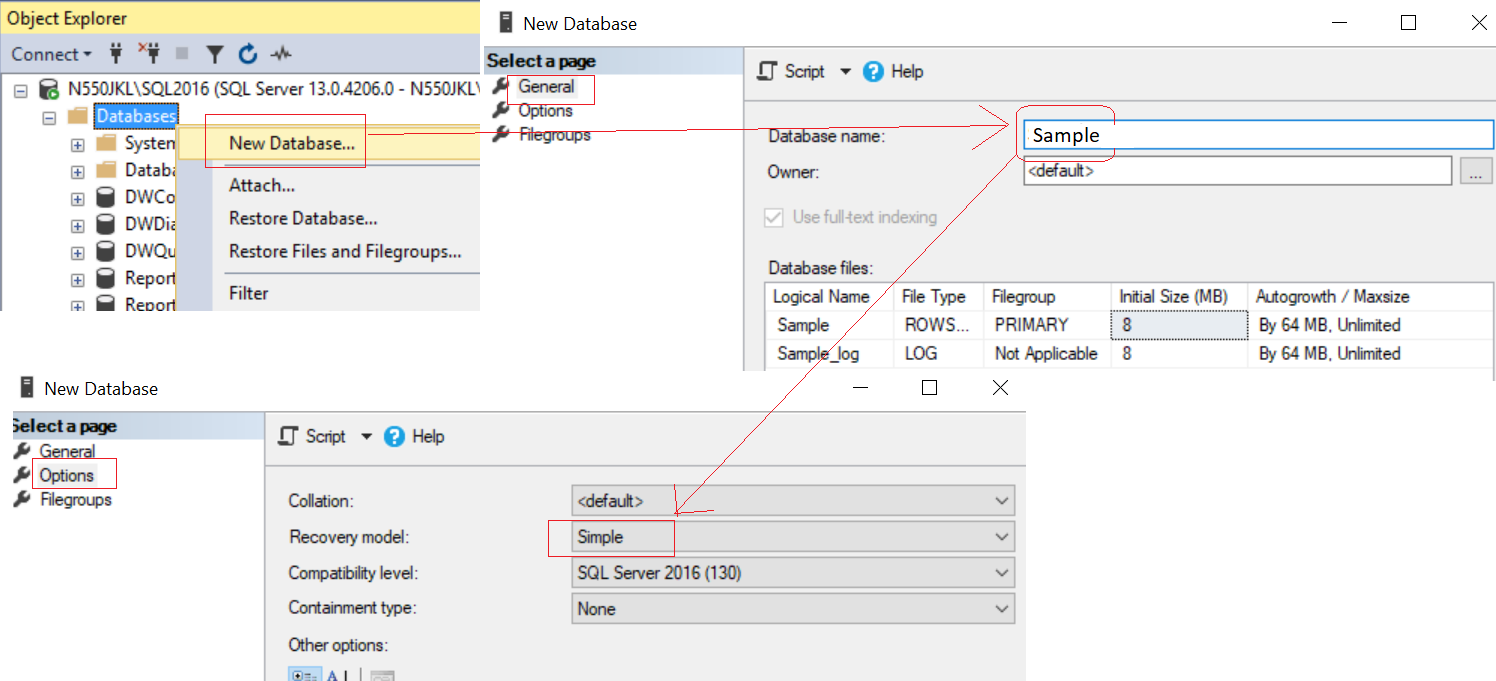
1. Web Form Application - Linq Query

1.1. TSQL

Database --> Right Click --> New Database -->

Database Name : Sample

Options --> Recovery Model : Simple



--Create a Sample DataBase and Run the following TSQL

/\*

1.

One Team can have many Gamers

One Gamer can have One Team.

This is One to Many Relationship.

2.

Team Id==4 has no Gamer.

Gamer Id==7 has no Team.

\*/

--1 ----------------------------------------------------------

--Drop Table if it exists.

--IF OBJECT\_ID('Gamer') IS NOT NULL

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gamer' ) )

    BEGIN

        TRUNCATE TABLE Gamer;

        DROP TABLE Gamer;

    END;

GO -- Run the previous command and begins new batch

--Drop Table if it exists.

--IF OBJECT\_ID('Team') IS NOT NULL

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Team' ) )

    BEGIN

        TRUNCATE TABLE Team;

        DROP TABLE Team;

    END;

GO -- Run the previous command and begins new batch

--Create Tables

CREATE TABLE Team

    (

      Id INT PRIMARY KEY

             IDENTITY ,

      Name NVARCHAR(100) ,

      Type NVARCHAR(100)

    );

GO -- Run the previous command and begins new batch

CREATE TABLE Gamer

    (

      Id INT PRIMARY KEY

             IDENTITY ,

      Name NVARCHAR(50) ,

      Gender NVARCHAR(50) ,

      Score INT ,

      Type NVARCHAR(50) ,

      TeamId INT FOREIGN KEY REFERENCES Team ( Id )

    );

GO -- Run the previous command and begins new batch

--2 ----------------------------------------------------------

--Insert Data

INSERT  INTO Team

VALUES  ( 'Team1\_Guardian', 'Guardian' );

INSERT  INTO Team

VALUES  ( 'Team2\_Assassinator', 'Assassinator' );

INSERT  INTO Team

VALUES  ( 'Team3\_Soldier', 'Soldier' );

INSERT  INTO Team

VALUES  ( 'Team4\_Civilian', 'Civilian' );

GO -- Run the previous command and begins new batch

INSERT  INTO Gamer

VALUES  ( 'Name1 ABC', 'Male', 5000, 'Water', 1 );

INSERT  INTO Gamer

VALUES  ( 'Name2 ABCDE', 'Female', 4500, 'Fire', 3 );

INSERT  INTO Gamer

VALUES  ( 'Name3 EFGH', 'Male', 6500, 'Fire', 2 );

INSERT  INTO Gamer

VALUES  ( 'Name4 HIJKLMN', 'Female', 45000, 'Water', 2 );

INSERT  INTO Gamer

VALUES  ( 'Name5 NOP', 'Male', 3000, 'Wood', 1 );

INSERT  INTO Gamer

VALUES  ( 'Name6 PQRSTUVW', 'Male', 4000, 'Earth', 3 );

INSERT  INTO Gamer

VALUES  ( 'Name7 XYZ', 'Male', 4500, 'Metal', NULL );

GO -- Run the previous command and begins new batch

--3 ----------------------------------------------------------

--3.1. --------------------------------

--Drop Stored Procedure if it exists.

--IF OBJECT\_ID('spGetGamers') IS NOT NULL

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetGamers' ) )

    BEGIN

        DROP PROCEDURE spGetGamers;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spGetGamers

AS

    BEGIN

        SELECT  Id ,

                Name ,

                Gender ,

                Score ,

                Type ,

                TeamId

        FROM    Gamer;

    END;

GO -- Run the previous command and begins new batch

--3.2. --------------------------------

-- Update Stored Procedure

--Drop Stored Procedure if it exists.

--IF OBJECT\_ID('spInsertGamer') IS NOT NULL

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spInsertGamer' ) )

    BEGIN

        DROP PROCEDURE spInsertGamer;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spInsertGamer

    @name NVARCHAR(50) ,

    @gender NVARCHAR(50) ,

    @score INT ,

    @type NVARCHAR ,

    @teamId int

AS

    BEGIN

        INSERT  INTO Gamer

        VALUES  ( @name, @gender, @score, @type, @teamId );

    END;

GO

--3.3. --------------------------------

-- Update Stored Procedure

--Drop Stored Procedure if it exists.

--IF OBJECT\_ID('spUpdateGamer') IS NOT NULL

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spUpdateGamer' ) )

    BEGIN

        DROP PROCEDURE spUpdateGamer;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spUpdateGamer

    @id INT ,

    @name NVARCHAR(50) ,

    @gender NVARCHAR(50) ,

    @score INT ,

    @type NVARCHAR ,

    @teamId int

AS

    BEGIN

        UPDATE  Gamer

        SET     Name = @name ,

                Gender = @gender ,

                Score = @score ,

                Type = @type ,

                TeamId = @teamId

        WHERE   Id = @id;

    END;

GO -- Run the previous command and begins new batch

--3.4. --------------------------------

-- Delete Stored Procedure

--Drop Stored Procedure if it exists.

--IF OBJECT\_ID('spDeleteGamer') IS NOT NULL

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spDeleteGamer' ) )

    BEGIN

        DROP PROCEDURE spDeleteGamer;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spDeleteGamer @Id int

AS

    BEGIN

        DELETE  FROM dbo.Gamer

        WHERE   Id = @Id;

    END;

GO

--3.5. --------------------------------

--3.5.1. --------------------------------

-- Delete Stored Procedure

--Drop Stored Procedure if it exists.

--IF OBJECT\_ID('spGetGamerByTeam') IS NOT NULL

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spGetGamerByTeam' ) )

    BEGIN

        DROP PROCEDURE spGetGamerByTeam;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spGetGamerByTeam

    @teamId INT ,

    @teamName NVARCHAR(100) OUT

AS

    BEGIN

        SELECT  @teamName = Name

        FROM    dbo.Team

        WHERE   Id = @teamId;

        SELECT  \*

        FROM    dbo.Gamer

        WHERE   TeamId = @teamId;

    END;

GO

--3.5.2. --------------------------------

--Test

DECLARE @teamName NVARCHAR(50);

EXECUTE spGetGamerByTeam 1, @teamName OUT;

SELECT  @teamName;

1.2. Set up SQL Authentication

In SQL server

Object Explorer --> Security --> Logins --> New Logins

-->

General Tab

Login Name :

**Tester**

Password:

**1234**

Default Database:

**Sample**

-->

Server Roles Tab

Select

**sysadmin**

-->

User Mapping Tab

Select **Sample**

Select every Roles.









2. Using SqlMetal to create DBML

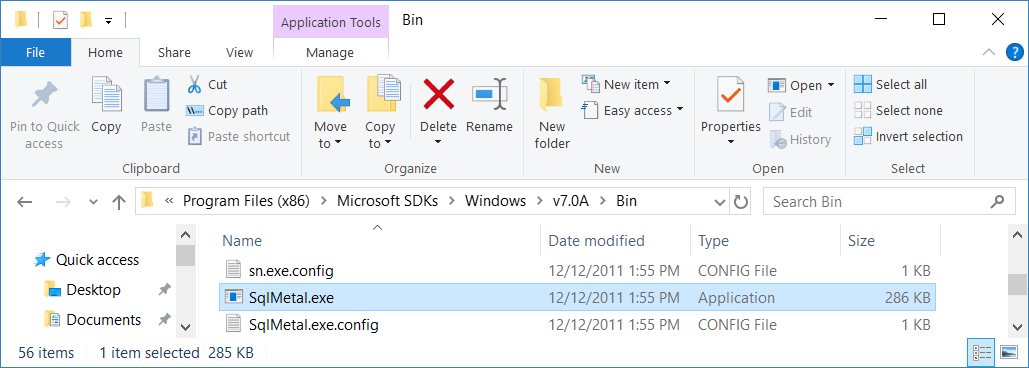
In the previous tutorial, we use visual studio to create Linq to Sql classes.

This tutorial will demonstrate using SqlMetal to create Linq to Sql classes.

SqlMetal is location in

**C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bin\SqlMetal.exe**

please do not double-click it.



**Step1:**

Create folder "**C:\LinqToSqlClasses**"

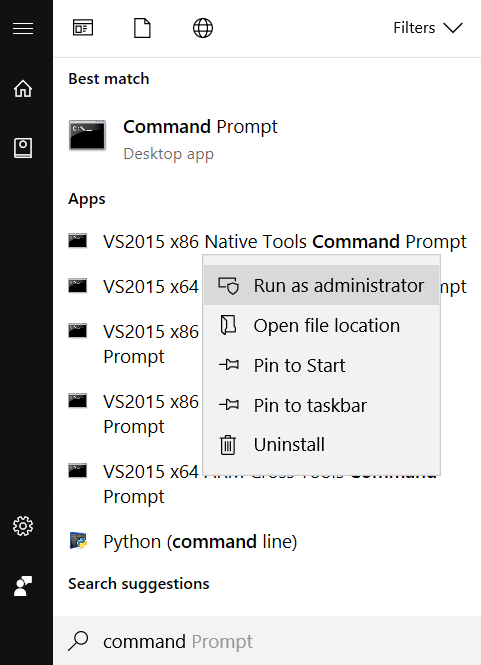
**Step2:**

Run "**Visual Studio Command prompt**" as an "**Administrator**"

In windows 10 -->

Start --> search "**command**"  --> VS2015 x86 Native Tools Command Prompt

--> Right click --> Run as administrator



**sqlmetal.exe /?**

this command will show the options documentation.

If you want to know more, you can visit MSDN

[http://msdn.microsoft.com/en-gb/library/vstudio/bb386987(v=vs.100).aspx](http://msdn.microsoft.com/en-gb/library/vstudio/bb386987%28v=vs.100%29.aspx)

**Step3:**

**SqlMetal.exe /server:N550JKL\SQL2016 /database:Sample /namespace:Sample /dbml:C:\LinqToSqlClasses\Sample.dbml /Context:SampleDataContext**

**/server:N550JKL\SQL2016**

This is the database server name on my current machine.

**/database:Sample**

I want to connect to "Sample" database.

**/namespace:Sample**

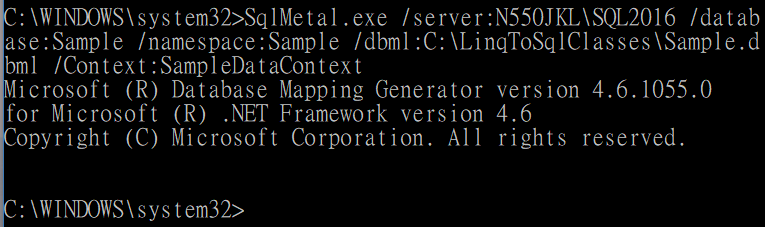
When we create Linq to Sql classes, the namespace is "**Sample**"

**/dbml:C:\LinqToSqlClasses\Sample.dbml**

I want to create **Sample.dbml** in **C:\LinqToSqlClasses**

**/Context:SampleDataContext**

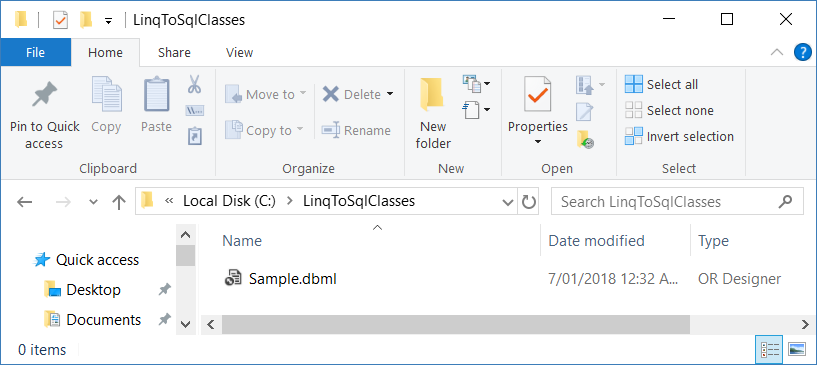
When we create Linq to Sql classes, the data context class name is "**SampleDataContext**"



**Step4:**

Go and check if the dbml has been created.

**C:\LinqToSqlClasses\Sample.dbml**



3. Asp.Net Web Application with existing DBML

3.1. Create Web Application

Open Visual Studio, I am currently using VS2017

If you don't have it, you may follow the instruction here to download.

<http://ithandyguytutorial.blogspot.com/2017/10/ch00install-visual-studio-2017-offline.html>

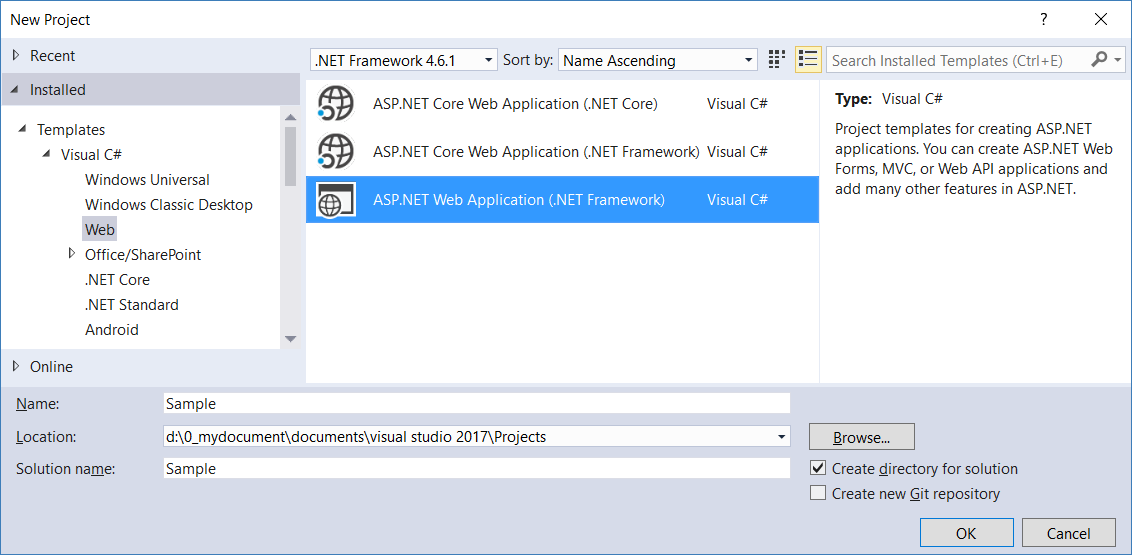
New Project --> Web -->[ASP.NET](http://asp.net/)**Web Application (.Net Framework)**

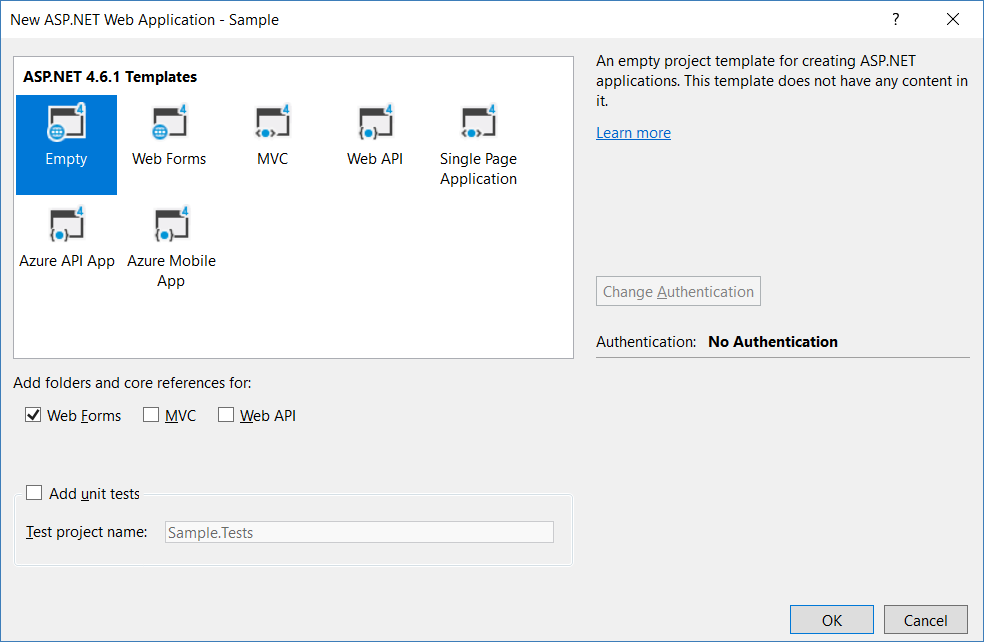
-->

Name:

**Sample**

--> **Empty** --> Select "**Web Forms**"  --> OK





3.2. Create Web Application

Add connection String

If you use Linq to Sql, you don't have to set this connection string.

I personally already get used to set it by myself.

<configuration>

  <connectionStrings>

    <add name="SampleConnectionString" connectionString="Data Source=N550JKL\SQL2016;Initial Catalog=Sample;User ID=Tester;Password=1234"

        providerName="System.Data.SqlClient" />

  </connectionStrings>

Graphical user interface, text, application

Description automatically generated

3.3. Add existing DBML

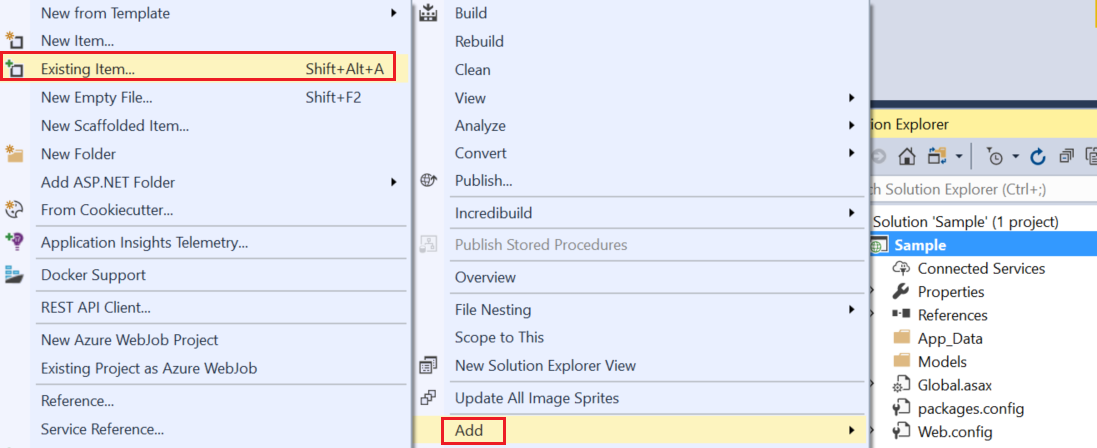
Project Name --> Right click --> Add --> Existing Item...

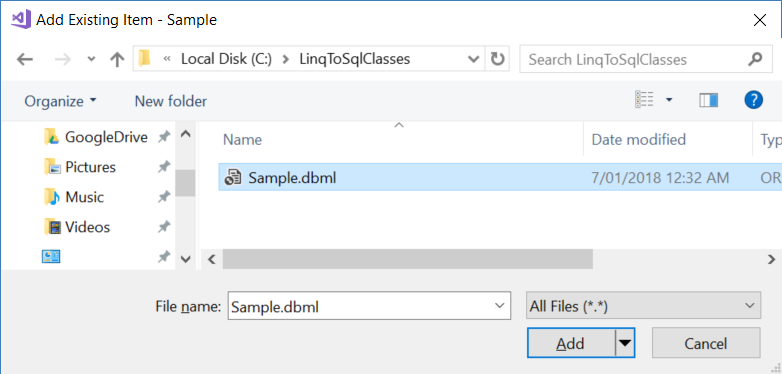
--> select

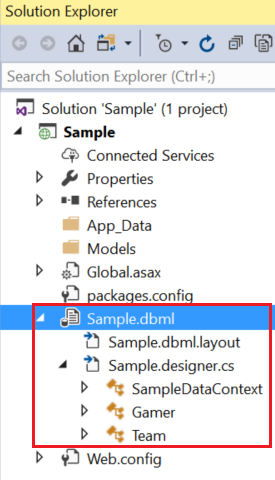
**C:\LinqToSqlClasses\Sample.dbml**

When add existing dbml, it will copy that dbml to the project folder.

After you add dbml, it will generate the classes.







3.4. WebForm1.aspx

3.4.1. WebForm1.aspx

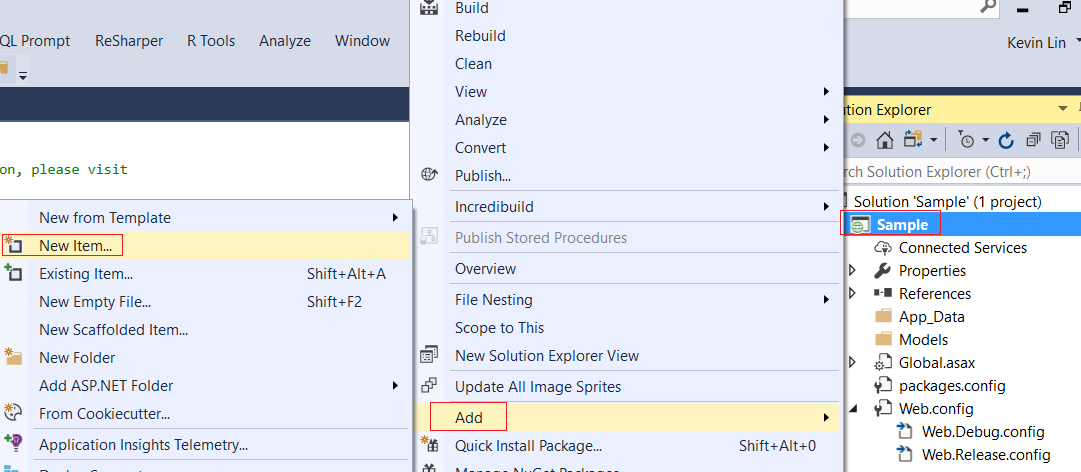
ProjectName --> Right Click --> Add --> New Item...

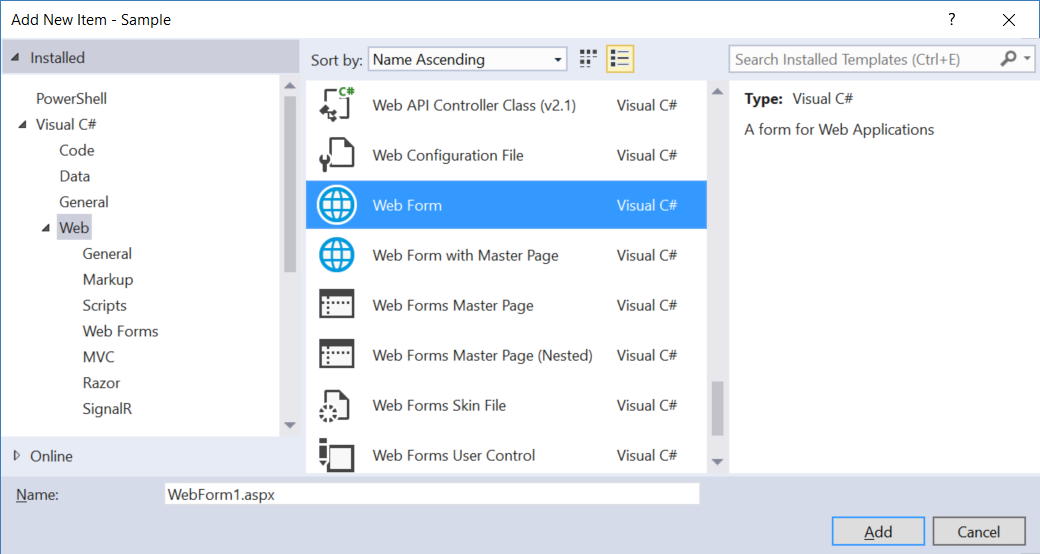
-->

**WebForm**

Name :

**WebForm1.aspx**





<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="Sample.WebForm1" %>

<!DOCTYPE html>

<html xmlns="<http://www.w3.org/1999/xhtml>">

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

        <div>

            <asp:GridView ID="GridView1" runat="server"></asp:GridView>

        </div>

    </form>

</body>

</html>

3.4.2. WebForm1.aspx.cs

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Sample

{

    public partial class WebForm1 : System.Web.UI.Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        {

            GetData();

        }

        private void GetData()

        {

            string cs = ConfigurationManager.

               ConnectionStrings["SampleConnectionString"].ConnectionString;

            using (SampleDataContext dbContext = new SampleDataContext(cs))

            {

                IQueryable<Gamer> gamerQueryable =

                    from gamer in dbContext.Gamer

                    select gamer;

                GridView1.DataSource = gamerQueryable;

                GridView1.DataBind();

            }

        }

    }

}