(T21)討論 LinqToSql 的 CompiledQueryCompile、IdentityCache。比較 ExecuteQuery、ExecuteCommand
CourseGUID: 5ba9a6fe-7475-4b0c-8b99-bbcf7f5e2e1c
==
(T21)討論 LinqToSql 的 CompiledQueryCompile、IdentityCache。比較 ExecuteQuery、ExecuteCommand
==

==

0. Summary

1.
CompiledQuery.Compile(...)
Reference:
https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/language-reference/compiled-queries-linq-to-entities
https://msdn.microsoft.com/en-us/library/system.data.linq.compiledquery.compile(v=vs.110).aspx
1.1.
Linq query expression dynamically generates the TSQL statements
every time when a LINQ query is issued.
1.2.
CompiledQuery.Compile() can create a new delegate
to represent the compiled query,
so the application will use that delegate
instead of dynamically generate the TSQL statements again,
when a LINQ query is issued.

2.
ExecuteQuery V.S. ExecuteCommand
2.1.
ExecuteQuery is for Select.
ExecuteCommand is for Insert, Update, Delete or stored procedure.
2.2.
Try not to use ExecuteQuery and ExecuteCommand.
When using ExecuteQuery and ExecuteCommand,
we lose the advantage of strongly-typed variables in Linq queries.

3.
IdentityCache
SampleDataContext.Refresh
Reference:
https://msdn.microsoft.com/en-us/library/dd627203(v=vs.100).aspx
----
3.1.
E.g.

```
//dbContext2.Refresh(System.Data.Linq.RefreshMode.OverwriteCurrentValues, g2);
```

Refresh dbContext and then overwrite g2 variable

3.2.
IdentityCache
E.g.

```
//using (SampleDataContext dbContext = new SampleDataContext())
//{
//    Gamer g1 = dbContext.Gamers.FirstOrDefault(g => g.Id == 1);
//    Gamer g2 = dbContext.Gamers.FirstOrDefault(g => g.Id == 1);
//    Console.WriteLine("g1==g2 : {0}", object.ReferenceEquals(g1, g2));
//}
```

3.2.1.
We are using the same dataContext object
and the same identity search condition in Linq query
to retrieve 2 gamers.
When g1 Linq query is executed, the Linq query is converted to Tsql.
Then the Tsql is executed in the database, and return the result back to the application.
The application creates the Gamer g1 object and stores its data into g1 object properties.
The **object identity** is stored in the **Identity Cache**.
3.2.2.
When g2 Linq query is executed, the Linq query checks the **Identity Cache**
and returns a reference to the student object that already exists.
The application only request data from database once.
g1 and g2 are pointing to the same student object in memory.

3.3.

```
//using (SampleDataContext dbContext1 = new SampleDataContext())
//using (SampleDataContext dbContext2 = new SampleDataContext())
//{
//    //Write the generated sql query to the Console window
//    dbContext1.Log = Console.Out;
//    dbContext2.Log = Console.Out;
//    Console.WriteLine("4.2.1. -------------------------- ");
//    Gamer g1 = dbContext1.Gamers.FirstOrDefault(g => g.Id == 1);
//    Gamer g2 = dbContext2.Gamers.FirstOrDefault(g => g.Id == 1);
//    Console.WriteLine($"g1==g2 : {object.ReferenceEquals(g1, g2)}");
```

We are using 2 different dataContext object
and the same search condition in Linq query
to retrieve 2 gamers.
The application will request data from database twice.
The first call to the database and then application create Gamer g1 object.
The second call to the database and then application create Gamer g2 object.
g1 and g2 are pointing to the different student object in memory.

```
//    Console.WriteLine($"g1==g2 : {object.ReferenceEquals(g1, g2)}");
```

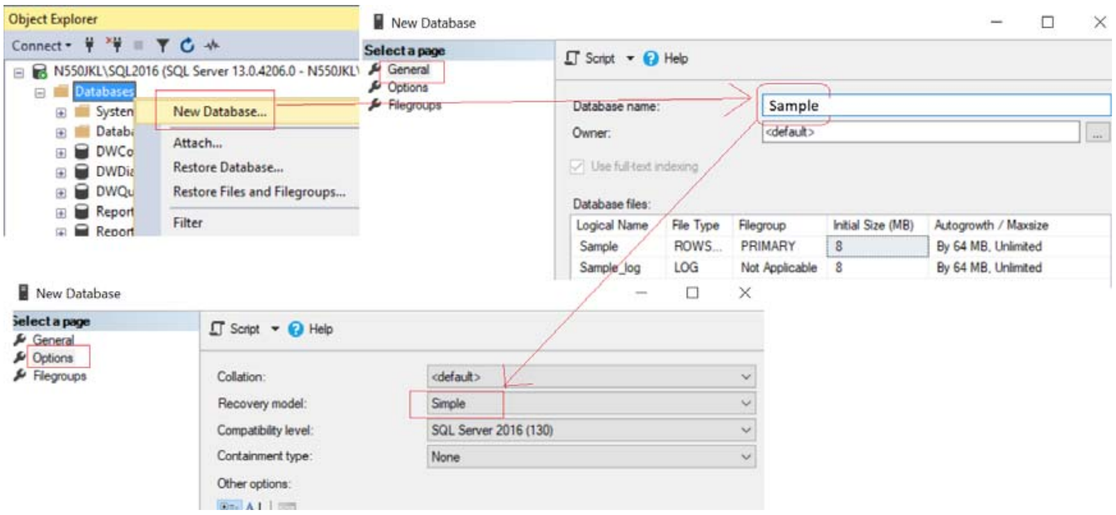
will return false.

==

1. Web Form Application - Linq Query

1.1. TSQL

Database --> Right Click --> New Database -->
Database Name : Sample
Options --> Recovery Model : Simple



Database name: Sample
Owner: <default>
Use full-text indexing
Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth / Maxsize
Sample	ROWS...	PRIMARY	8	By 64 MB, Unlimited
Sample_log	LOG	Not Applicable	8	By 64 MB, Unlimited

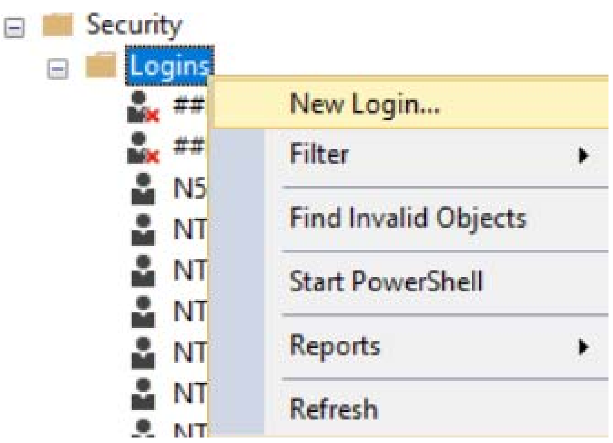
Collation: <default>
Recovery model: Simple
Compatibility level: SQL Server 2016 (130)
Containment type: None
Other options:

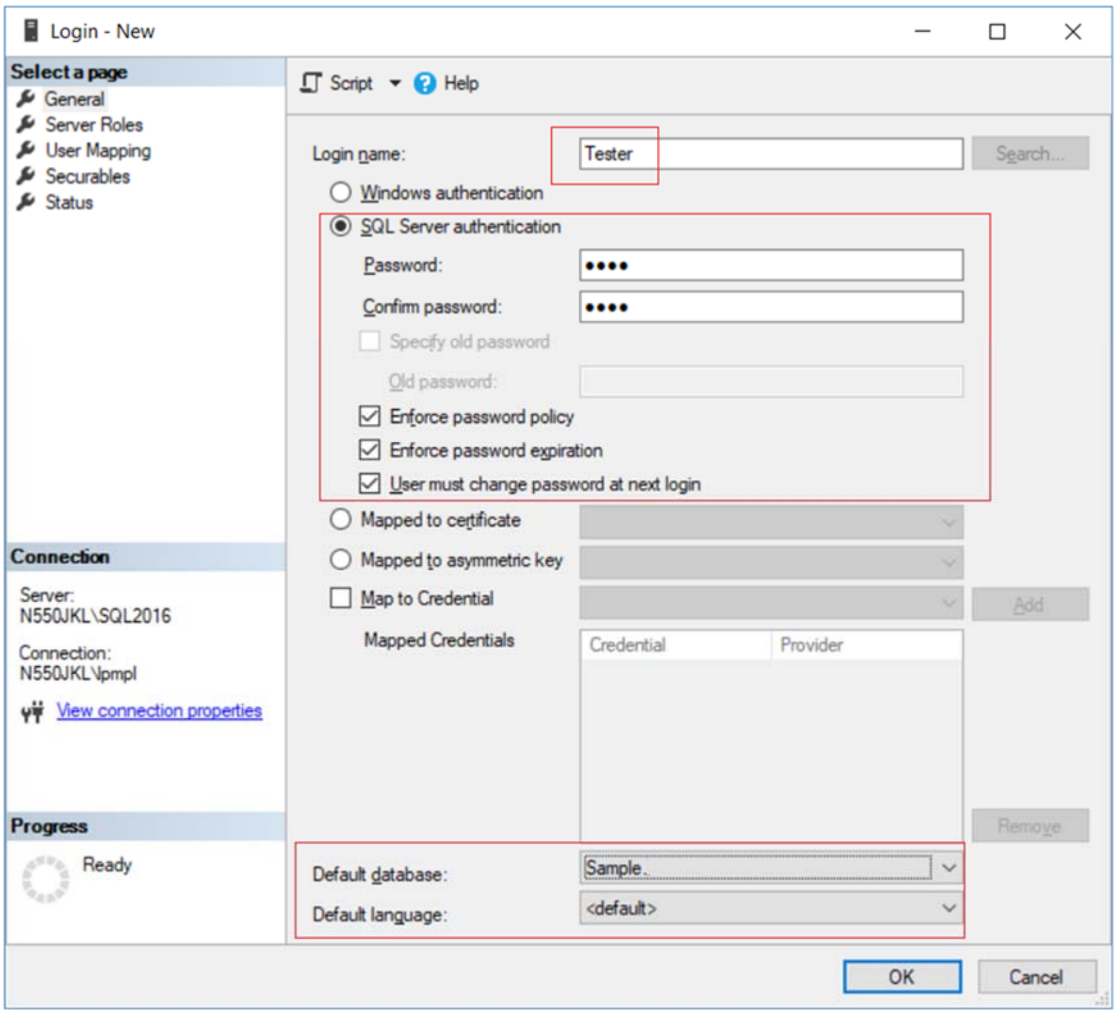
--Create a Sample DataBase and Run the following TSQL

```
--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
CREATE TABLE Gamer
    (
      Id INT PRIMARY KEY
             IDENTITY ,
     Name NVARCHAR(50) ,
      Gender NVARCHAR(50) ,
      Score INT ,
         Type NVARCHAR(50) ,
         CombatPower INT,
         MagicPower INT
    );
GO -- Run the previous command and begins new batch
--2 ----------------------------------------------------------
INSERT  INTO Gamer
VALUES  ( 'Name1 ABC', 'Male', 5000, 'Warrior', 500, NULL );
INSERT  INTO Gamer
VALUES  ( 'Name2 ABCDE', 'Female', 4500, 'Warrior', 350, NULL );
INSERT  INTO Gamer
VALUES  ( 'Name3 EFGH', 'Male', 6500, 'Magician', NULL, 600 );
INSERT  INTO Gamer
VALUES  ( 'Name4 HIJKLMN', 'Female', 45000, 'Magician', NULL, 650 );
INSERT  INTO Gamer
VALUES  ( 'Name5 NOP', 'Male', 3000, 'Magician', NULL, 700 );
INSERT  INTO Gamer
VALUES  ( 'Name6 PQRSTUVW', 'Male', 4000, 'Warrior', 450, NULL );
INSERT  INTO Gamer
VALUES  ( 'Name7 XYZ', 'Male', 4500, 'Warrior', 550, NULL );
GO -- Run the previous command and begins new batch
```

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.





Login name: Tester

Windows authentication

[✓ SQL Server authentication]

Password: ••••

Confirm password: ••••

Specify old password

Old password:

☑ Enforce password policy

☑ Enforce password expiration

☑ User must change password at next login

Mapped to certificate

Mapped to asymmetric key

☐ Map to Credential

Mapped Credentials

Credential	Provider

Default database: Sample

Default language: <default>

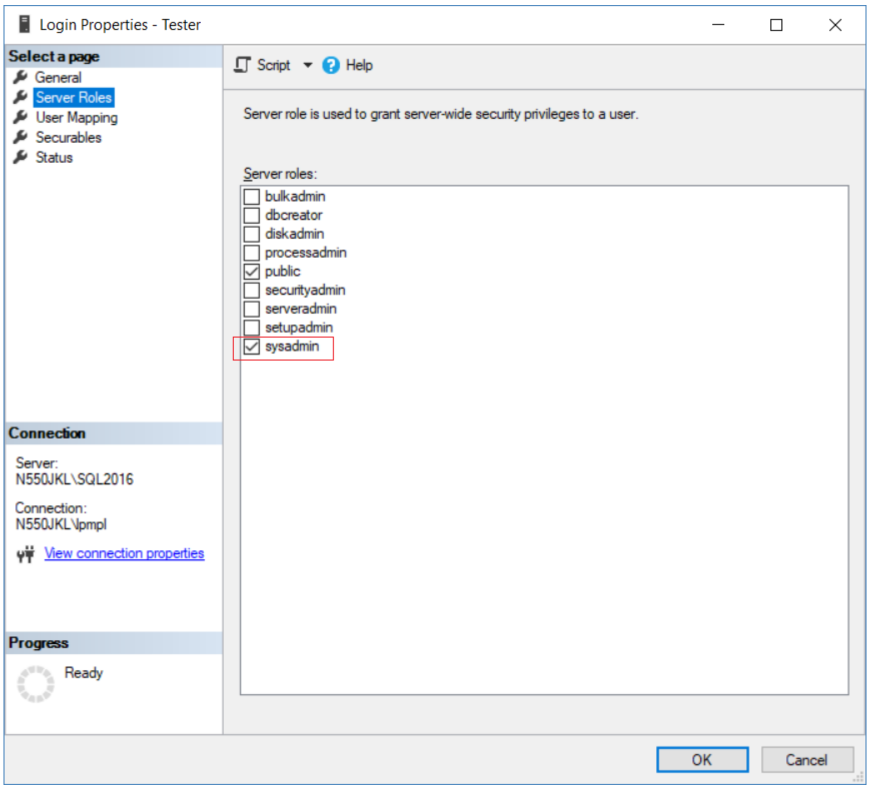
Connection

Server: N550JKL\SQL2016

Connection: N550JKL\pmpl

Progress

Ready


Login Properties - Tester

Select a page: General, Server Roles, User Mapping, Securables, Status

Server role is used to grant server-wide security privileges to a user.

Server roles:

- [ ] bulkadmin
- [ ] dbcreator
- [ ] diskadmin
- [ ] processadmin
- [x] public
- [ ] securityadmin
- [ ] serveradmin
- [ ] setupadmin
- [x] sysadmin

Connection

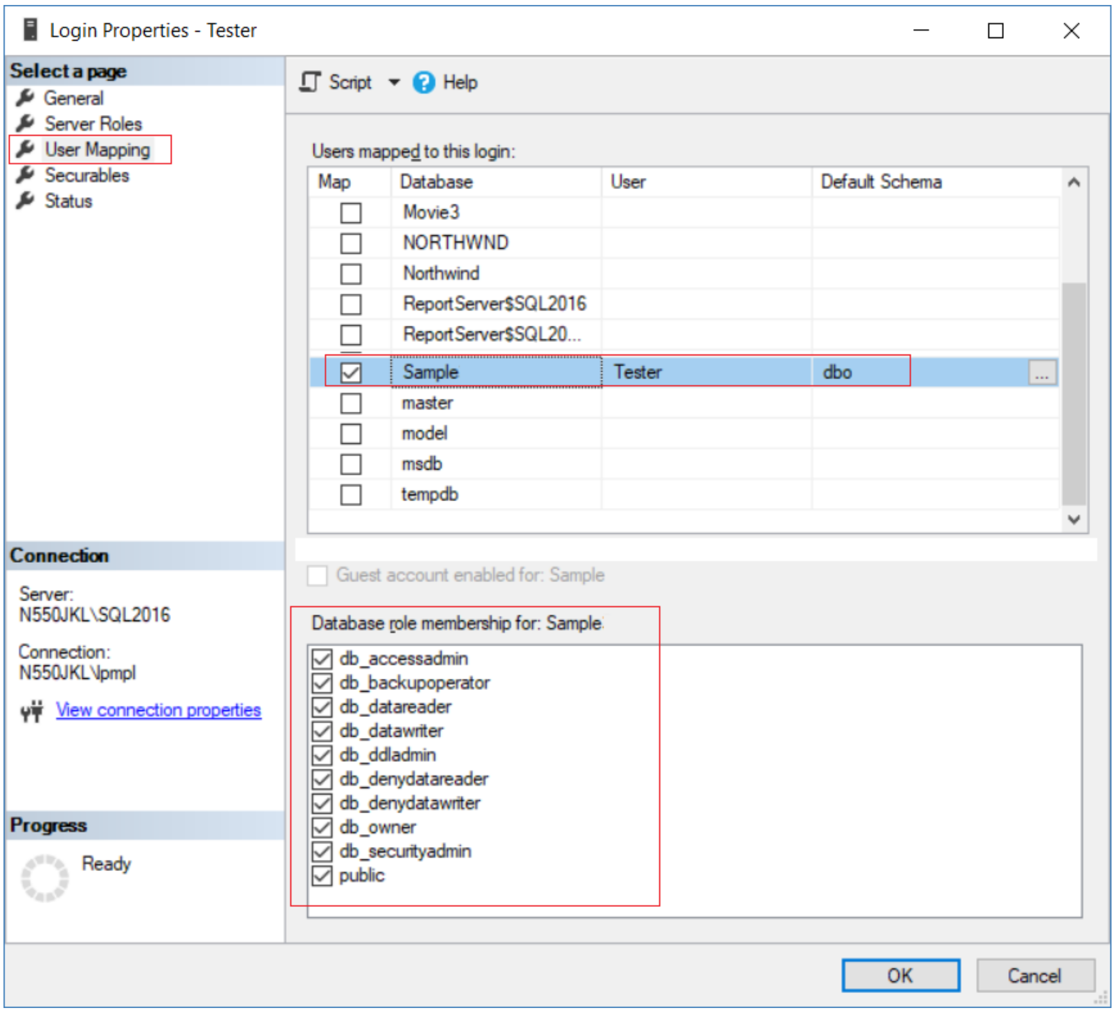
Server:
N550JKL\SQL2016

Connection:
N550JKL\lpmpl

View connection properties

Progress

Ready


Login Properties - Tester

Select a page: General, Server Roles, User Mapping, Securables, Status

Users mapped to this login:

Map	Database	User	Default Schema
☐	Movie3		
☐	NORTHWND		
☐	Northwind		
☐	ReportServer$SQL2016		
☐	ReportServer$SQL20...		
☑	Sample	Tester	dbo
☐	master		
☐	model		
☐	msdb		
☐	tempdb		

- [ ] Guest account enabled for: Sample

Database role membership for: Sample

- [x] db_accessadmin
- [x] db_backupoperator
- [x] db_datareader
- [x] db_datawriter
- [x] db_ddladmin
- [x] db_denydatareader
- [x] db_denydatawriter
- [x] db_owner
- [x] db_securityadmin
- [x] public

Connection

Server:
N550JKL\SQL2016

Connection:
N550JKL\lpmpl

View connection properties

Progress

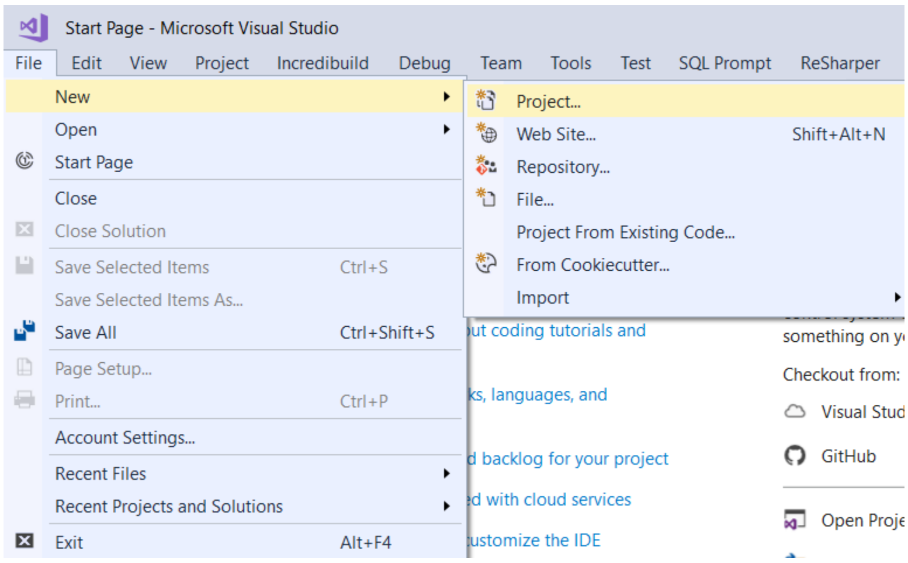
Ready

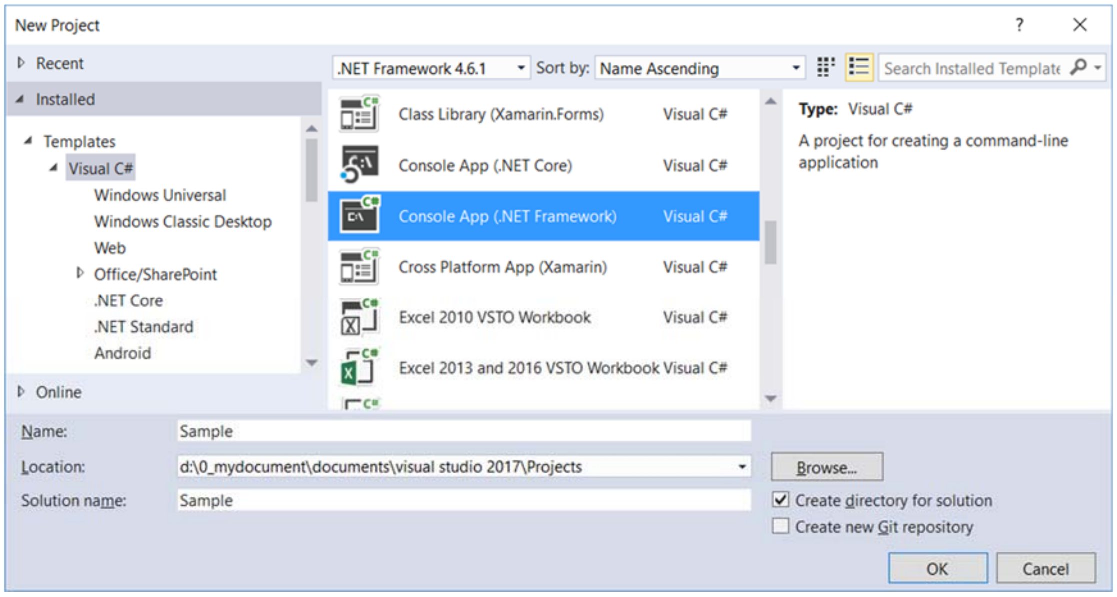
2. Console App

File --> New --> Project... -->

Visual C# --> **Console App (.Net Framework)** -->

Name: **Sample**


Start Page - Microsoft Visual Studio
File Edit View Project Incredibuild Debug Team Tools Test SQL Prompt ReSharper
New
Open
Start Page
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Page Setup...
Print... Ctrl+P
Account Settings...
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project...
Web Site... Shift+Alt+N
Repository...
File...
Project From Existing Code...
From Cookiecutter...
Import


New Project
Recent
Installed
Templates
Visual C#
Windows Universal
Windows Classic Desktop
Web
Office/SharePoint
.NET Core
.NET Standard
Android
Online
.NET Framework 4.6.1
Sort by: Name Ascending
Search Installed Templates
Class Library (Xamarin.Forms) Visual C#
Console App (.NET Core) Visual C#
Console App (.NET Framework) Visual C#
Cross Platform App (Xamarin) Visual C#
Excel 2010 VSTO Workbook Visual C#
Excel 2013 and 2016 VSTO Workbook Visual C#
Browse...
OK
Cancel

Type: Visual C#

A project for creating a command-line application

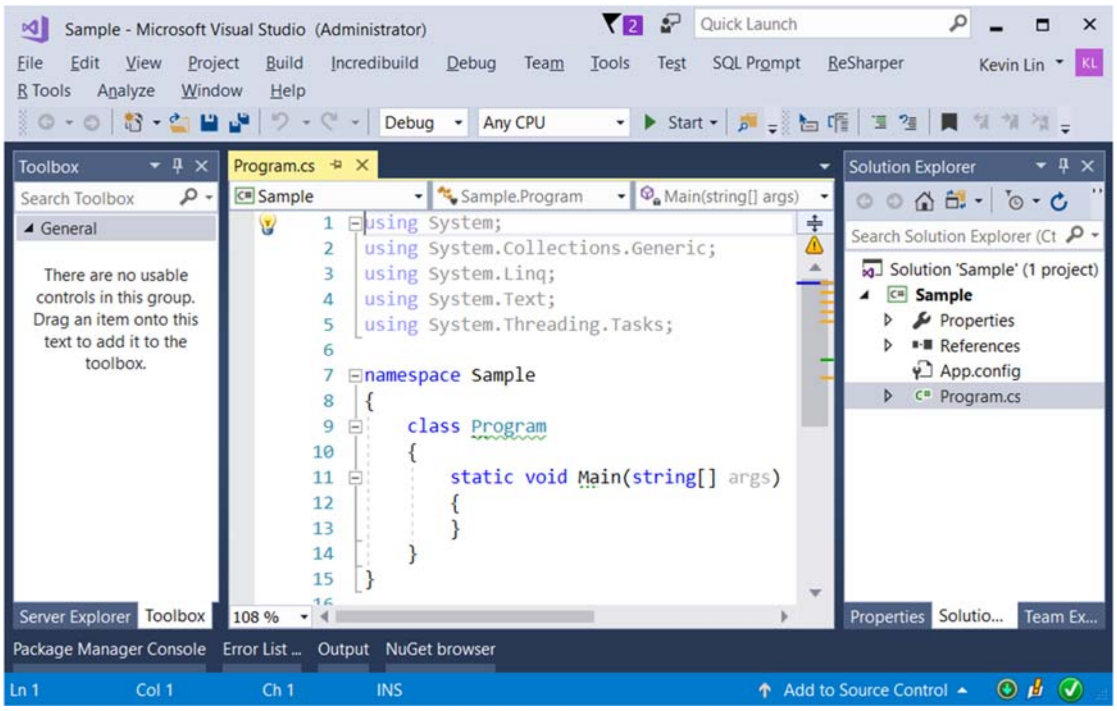
Name: Sample

Location: d:\0_mydocument\documents\visual studio 2017\Projects

Solution name: Sample

[✓] Create directory for solution

[ ] Create new Git repository

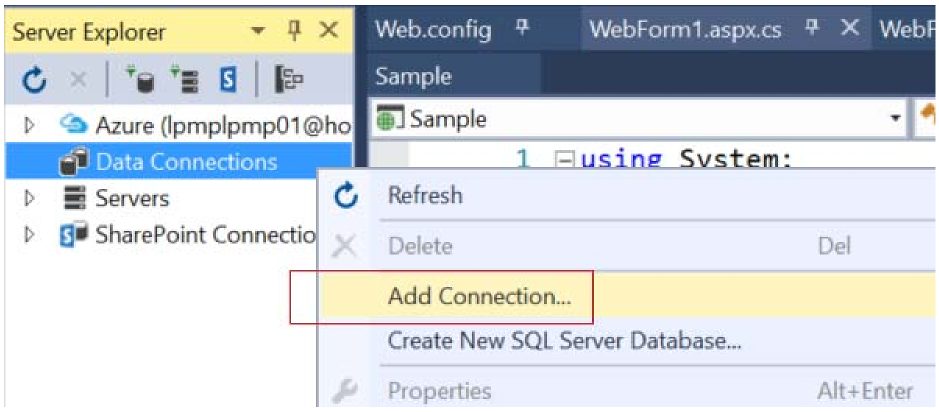


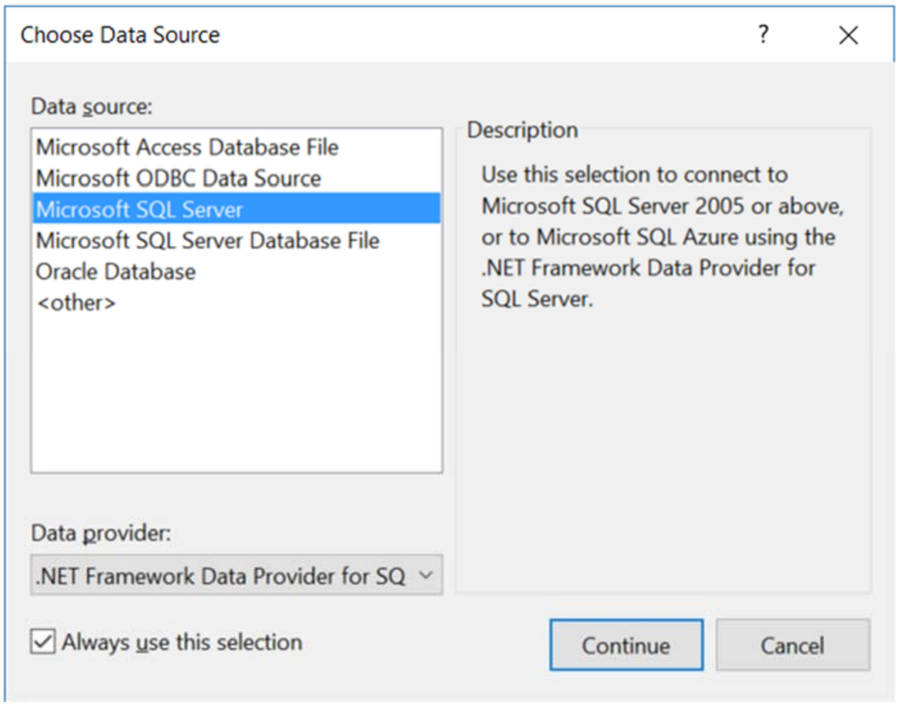
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

2.1. Linq to SQL

2.1.1. Add Connection

Server Explorer --> Data Connections --> Right click --> Add Connection...
--> Microsoft SQL server -->
Enter your server and database details ....




Choose Data Source

Data source:

- Microsoft Access Database File
- Microsoft ODBC Data Source
- Microsoft SQL Server
- Microsoft SQL Server Database File
- Oracle Database
- <other>

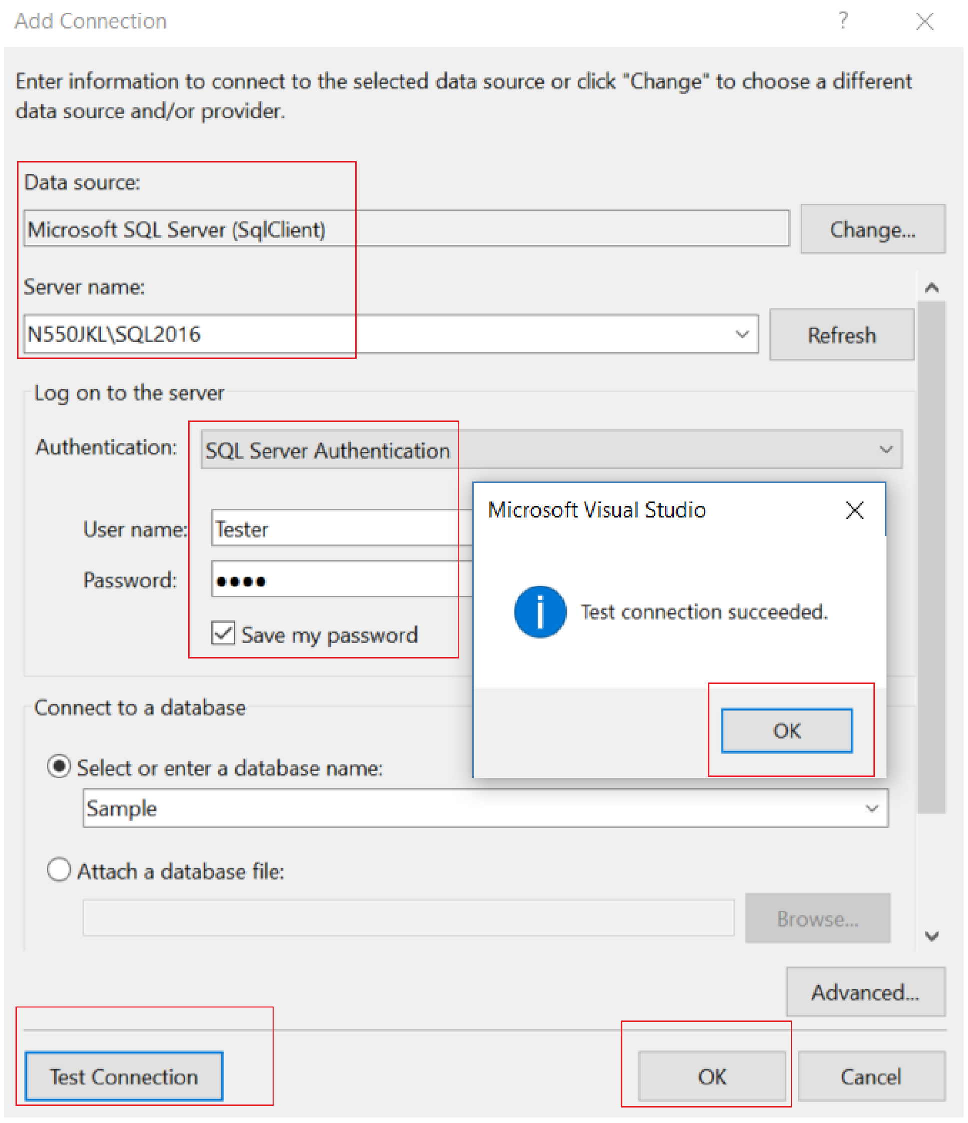
Description

Use this selection to connect to Microsoft SQL Server 2005 or above, or to Microsoft SQL Azure using the .NET Framework Data Provider for SQL Server.

Data provider: .NET Framework Data Provider for SQ

[✓] Always use this selection

Continue | Cancel


Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient) | Change...

Server name: N550JKL\SQL2016 | Refresh

Log on to the server

Authentication: SQL Server Authentication

User name: Tester

Password: ●●●●

[✓] Save my password

Microsoft Visual Studio

Test connection succeeded.

OK

Connect to a database

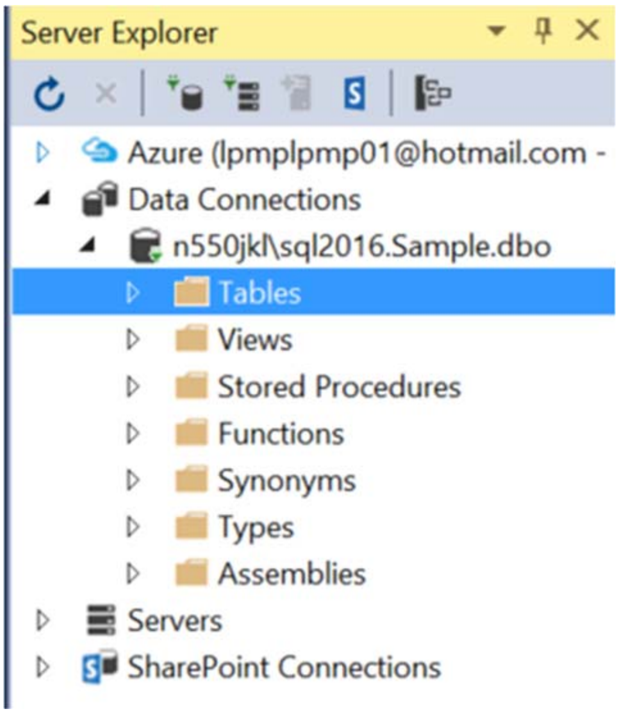
(●) Select or enter a database name: Sample

( ) Attach a database file:

Browse...

Advanced...

Test Connection | OK | Cancel

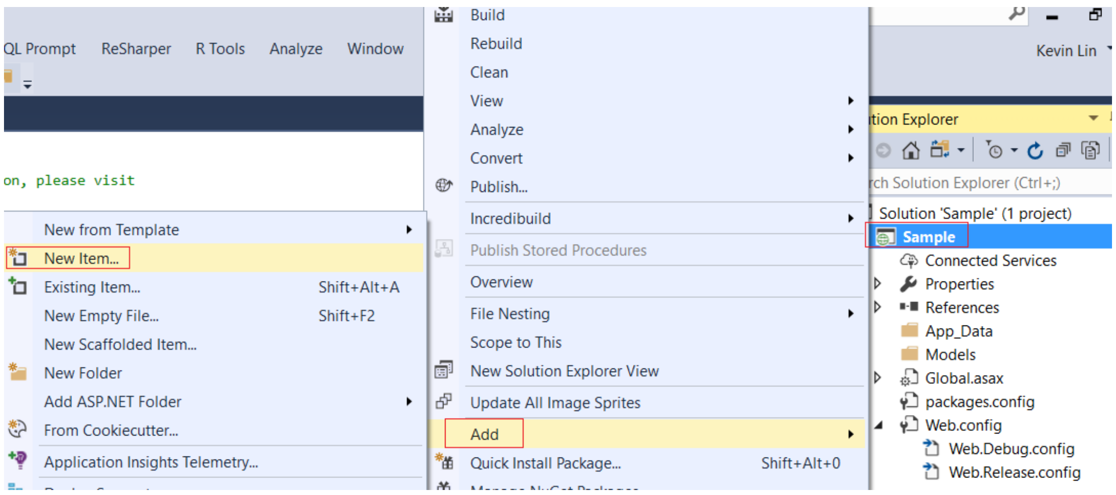


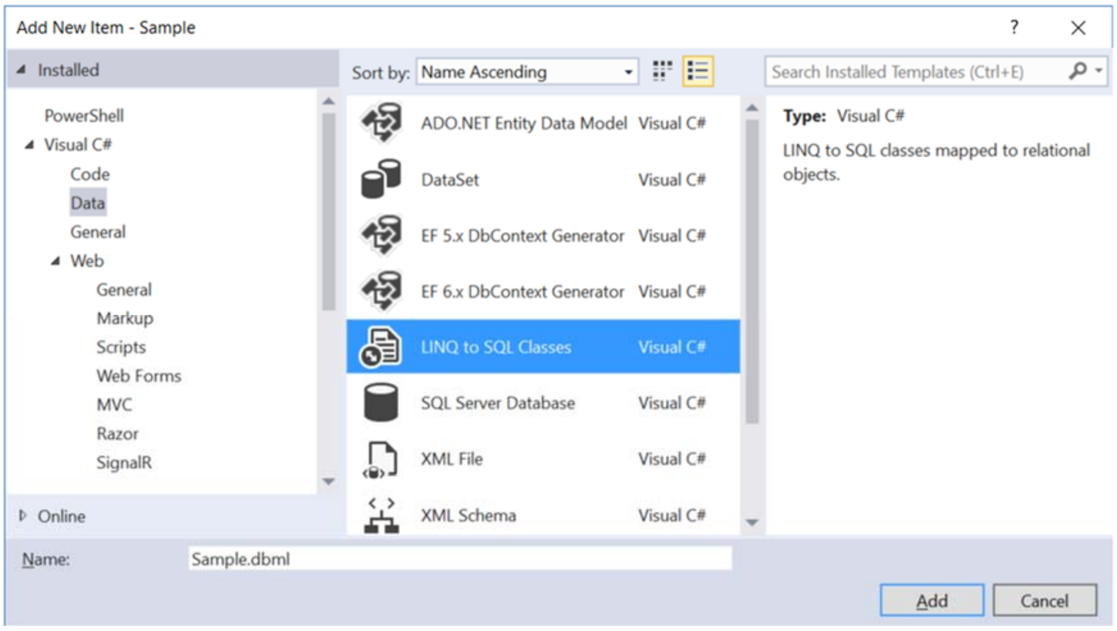
2.1.2. Sample.dbml

ProjectName --> Right Click --> Add --> New Item...
--> Linq to SQL classes -->
Name : **Sample.dbml**

I name it as "Sample.dbml",
because I know this is for connection to "Sample" Database.

-->
Drag Table from Server Explorer into DBML

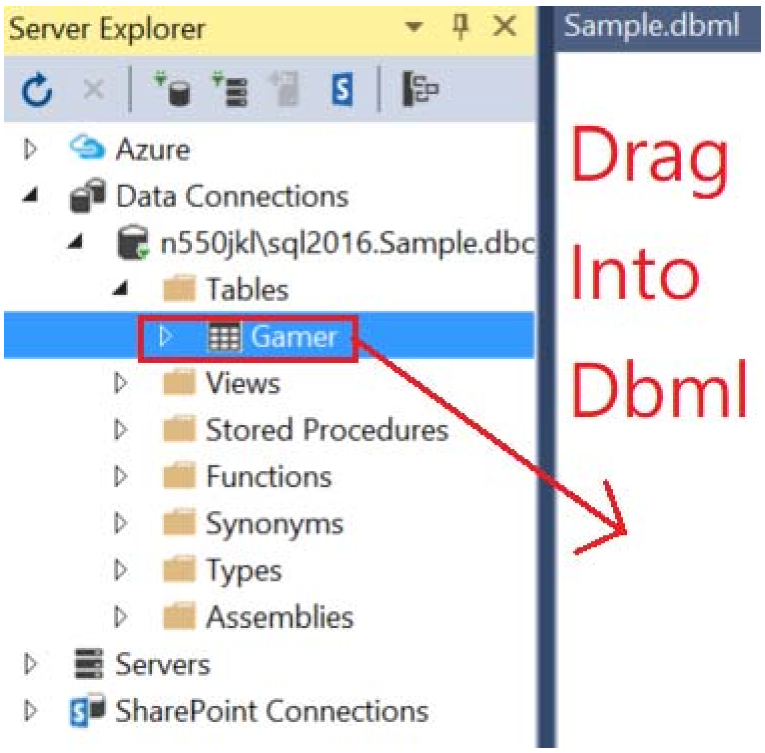


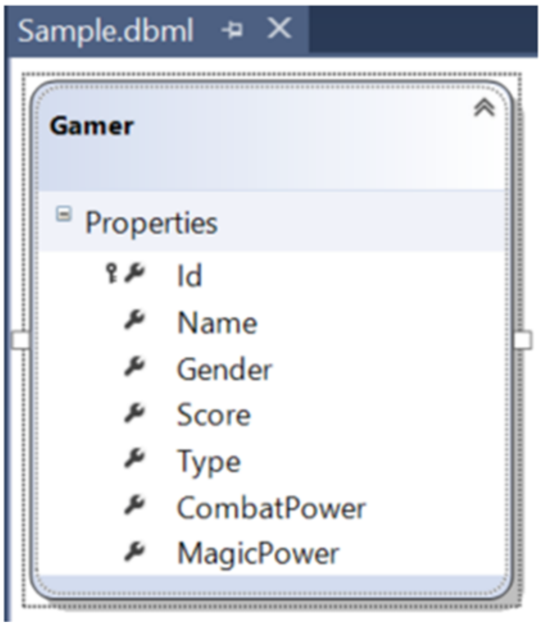


Type: Visual C#

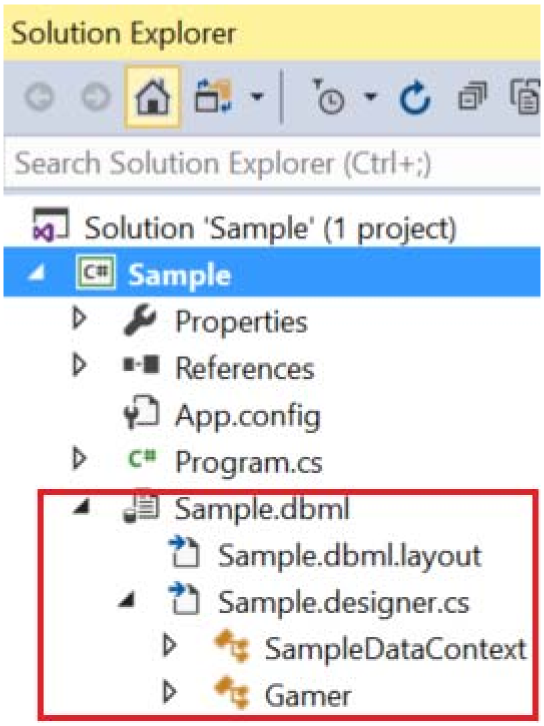
LINQ to SQL classes mapped to relational objects.

Name: Sample.dbml




Sample.dbml
Gamer
Properties
Id
Name
Gender
Score
Type
CombatPower
MagicPower

Save the dbml, it will generate the following files.
The DataContext context is the entry point to the database.



Solution 'Sample' (1 project)
Sample
Properties
References
App.config
Program.cs
Sample.dbml
Sample.dbml.layout
Sample.designer.cs
SampleDataContext
Gamer

2.2. Program.cs

```
using System;
using System.Collections.Generic;
using System.Data.Linq;
using System.Linq;
namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
            // 1. =====================================
            //CompiledQuerySample();
            Console.WriteLine("1. CompiledQuerySample() ============== ");
             CompiledQuerySample();
            // 2. =====================================
            //ExecuteQuerySample();
            Console.WriteLine("2. ExecuteQuerySample() ============== ");
             ExecuteQuerySample();
            // 3. =====================================
            //ExecuteCommandSample();
            Console.WriteLine("3. ExecuteCommandSample() ============== ");
             ExecuteCommandSample();
            // 4. =====================================
            //IdentityCacheRefreshSample();
            Console.WriteLine("4. IdentityCacheRefreshSample() =========== ");
             IdentityCacheRefreshSample();
            Console.ReadLine();
        }

        // 1. =====================================
        //CompiledQuerySample();
        static void CompiledQuerySample()
        {
            //1.
            //CompiledQuery.Compile(...)
            //Reference:
            //https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/language-
reference/compiled-queries-linq-to-entities
            //https://msdn.microsoft.com/en-
us/library/system.data.linq.compiledquery.compile(v=vs.110).aspx
            //1.1.
            //Linq query expression dynamically generate the TSQL statements
            //every time when a LINQ query is issued.
            //1.2.
            //CompiledQuery.Compile() can create a new delegate
            //to represent the compiled query,
            //so the application will use that delegate
            //instead of dynamically generate the TSQL statements again,
            //when a LINQ query is issued.
            //1.1. ----------------------------------------------
            //Dynamically generate the required T-SQL statements
            Console.WriteLine("1.1. Dynamically generate the required T-SQL statements ------------ ");
            using (SampleDataContext dbContext = new SampleDataContext())
            {
                Gamer gamerId1 = (from g in dbContext.Gamers
                                   where g.Id == 1
                                   select g).Single();
                Console.WriteLine(gamerId1);
            }
            //1.2. ----------------------------------------------
            //CompiledQuery.Compile(...)
            Console.WriteLine("1.2. CompiledQuery.Compile(...) ------------ ");
            Func<SampleDataContext, int, Gamer> compiledQuery = CompiledQuery.Compile(
                            (SampleDataContext dataContext, int gamerId) =>
                                (from g in dataContext.Gamers
                                 where g.Id == gamerId
```

```
                              select g).Single());
        using (SampleDataContext dbContext = new SampleDataContext())
        {
            if (compiledQuery != null)
            {
                Gamer gamer = compiledQuery(dbContext, 1);
                Console.WriteLine(gamer);
            }
        }
    }
    // 1.1. Dynamically generate the required T-SQL statements ------------
    // Id==1,Name==Name1 ABC,Gender==Male,Score==5000,Type==Warrior,CombatPower==500,MagicPower==
    // 1.2. CompiledQuery.Compile(...) ------------
    // Id==1,Name==Name1 ABC,Gender==Male,Score==5000,Type==Warrior,CombatPower==500,MagicPower==



   // 2. =====================================
    //ExecuteQuerySample();
    static void ExecuteQuerySample()
    {
        //ExecuteQuery is for Select.
        //ExecuteCommand is for Insert, Update, Delete or stored procedure.
        using (SampleDataContext dbContext = new SampleDataContext())
        {
            //2.1. ExecuteQuery ------------------------------------
            Console.WriteLine("2.1. ExecuteQuery ------------ ");
            IEnumerable<Gamer> gamers = dbContext.ExecuteQuery<Gamer>(
                "Select * from Gamer where Gender='Male'");
            foreach (Gamer gamer in gamers)
            {
                Console.WriteLine(gamer);
            }
            //2.2. ParameterizeQuery ------------------------------------
            Console.WriteLine("2.2. ParameterizeQuery ------------ ");
            IEnumerable<Gamer> gamers2 = dbContext.ExecuteQuery<Gamer>(
                "Select * from Gamer where Gender={0} AND Type={1}","Male", "Warrior");
            foreach (Gamer gamer2 in gamers2)
            {
                Console.WriteLine(gamer2);
            }
        }
    }
    // 2.1. ExecuteQuery ------------
    // Id==1,Name==Name1 ABC,Gender==Male,Score==5000,Type==Warrior,CombatPower==500,MagicPower==
    // Id==3,Name==Name3 EFGH,Gender==Male,Score==6500,Type==Magician,CombatPower==,MagicPower==600
    // Id==5,Name==Name5 NOP,Gender==Male,Score==3000,Type==Magician,CombatPower==,MagicPower==700
    // Id==6,Name==Name6 PQRSTUVW,Gender==Male,Score==4000,Type==Warrior,CombatPower==450,MagicPower==
    // Id==7,Name==Name7 XYZ,Gender==Male,Score==4500,Type==Warrior,CombatPower==550,MagicPower==
    // 2.2. ParameterizeQuery ------------
    // Id==1,Name==Name1 ABC,Gender==Male,Score==5000,Type==Warrior,CombatPower==500,MagicPower==
    // Id==6,Name==Name6 PQRSTUVW,Gender==Male,Score==4000,Type==Warrior,CombatPower==450,MagicPower==
    // Id==7,Name==Name7 XYZ,Gender==Male,Score==4500,Type==Warrior,CombatPower==550,MagicPower==
    // 3. =====================================
    //ExecuteCommandSample();
    static void ExecuteCommandSample()
```

```
        {
            using (SampleDataContext dbContext = new SampleDataContext())
            {
                //ExecuteQuery is for Select.
                //ExecuteCommand is for Insert, Update, Delete or stored procedure.
                int count = dbContext.ExecuteCommand(
                    "Update Gamer set Score=5555 where Id=1");
                Console.WriteLine("Rows Updated = {0}", count);
                Gamer gamerId1 = dbContext.ExecuteQuery<Gamer>(
                    "Select * from Gamer where Id=1").FirstOrDefault();
                Console.WriteLine(gamerId1);
            }
        }
        // Rows Updated = 1
        // Id==1,Name==Name1 ABC,Gender==Male,Score==5555,Type==Warrior,CombatPower==500,MagicPower==


        // 4. =====================================
        //IdentityCacheRefreshSample();
        static void IdentityCacheRefreshSample()
        {
            // 4.1. -----------------------------------------------------
            //3.2.
            //IdentityCache
            //3.2.1.
            //We are using the same dataContext object
            //and the same identity search condition in Linq query
            //to retrieve 2 gamers.
            //When g1 Linq query is executed, the Linq query is converted to Tsql.
            //Then the Tsql is executed in the database, and return the result back to the application.
            //The application creates the Gamer g1 object and stores its data into g1 object properties.
            //The object identity is stored in the Identity Cache.
            //3.2.2.
            //When g2 Linq query is executed, the Linq query checks the Identity Cache
            //and returns a reference to the student object that already exists.
            //The application only request data from the database once.
            //g1 and g2 are pointing to the same student object in memory.
            Console.WriteLine("4.1. -------------------------- ");
            using (SampleDataContext dbContext = new SampleDataContext())
            {
                //Write the generated sql query to the Console window
                 dbContext.Log = Console.Out;
                Gamer g1 = dbContext.Gamers.FirstOrDefault(g => g.Id == 1);
                Gamer g2 = dbContext.Gamers.FirstOrDefault(g => g.Id == 1);
                Console.WriteLine("g1==g2 : {0}", object.ReferenceEquals(g1, g2));
            }
            // SELECT TOP (1) [t0].[Id], [t0].[Name], [t0].[Gender], [t0].[Score], [t0].[Type],
[t0].[CombatPower], [t0].[MagicPower]
            // FROM [dbo].[Gamer] AS [t0]
            // WHERE [t0].[Id] = @p0
            // -- @p0: Input Int (Size = -1; Prec = 0; Scale = 0) [1]
            // -- Context: SqlProvider(Sql2008) Model: AttributedMetaModel Build: 4.7.2556.0
            // g1==g2 : True
           // 4.2.1. -----------------------------------------------------
            //3.3.
            // We are using 2 different dataContext object
            // and the same search condition in Linq query
            // to retrieve 2 gamers.
            // The application will request data from database twise.
            // First call to database and then application create Gamer g1 object.
            // Second call to database and then application create Gamer g2 object.
            // g1 and g2 are pointing to the different student object in memory.
            // //    Console.WriteLine($"g1==g2 : {object.ReferenceEquals(g1, g2)}");
            // will return false.
            Console.WriteLine("4.2. -------------------------- ");
            using (SampleDataContext dbContext1 = new SampleDataContext())
            using (SampleDataContext dbContext2 = new SampleDataContext())
            {
                //Write the generated sql query to the Console window
                 dbContext1.Log = Console.Out;
                 dbContext2.Log = Console.Out;
                Console.WriteLine("4.2.1. -------------------------- ");
                Gamer g1 = dbContext1.Gamers.FirstOrDefault(g => g.Id == 1);
                Gamer g2 = dbContext2.Gamers.FirstOrDefault(g => g.Id == 1);
                Console.WriteLine($"g1==g2 : {object.ReferenceEquals(g1, g2)}");
                Console.WriteLine($"g1: {g1}");
                Console.WriteLine($"g2: {g2}");
                // SELECT TOP (1) [t0].[Id], [t0].[Name], [t0].[Gender], [t0].[Score], [t0].[Type],
[t0].[CombatPower], [t0].[MagicPower]
                // FROM [dbo].[Gamer] AS [t0]
                // WHERE [t0].[Id] = @p0
                // -- @p0: Input Int (Size = -1; Prec = 0; Scale = 0) [1]
                // -- Context: SqlProvider(Sql2008) Model: AttributedMetaModel Build: 4.7.2556.0
                // SELECT TOP (1) [t0].[Id], [t0].[Name], [t0].[Gender], [t0].[Score], [t0].[Type],
[t0].[CombatPower], [t0].[MagicPower]
                // FROM [dbo].[Gamer] AS [t0]
                // WHERE [t0].[Id] = @p0
                // -- @p0: Input Int (Size = -1; Prec = 0; Scale = 0) [1]
                // -- Context: SqlProvider(Sql2008) Model: AttributedMetaModel Build: 4.7.2556.0
                // g1==g2 : False
                // g1: Id==1,Name==Name1
ABC,Gender==Male,Score==5555,Type==Warrior,CombatPower==500,MagicPower==
                // g2: Id==1,Name==Name1
ABC,Gender==Male,Score==5555,Type==Warrior,CombatPower==500,MagicPower==


            // 4.2.2. -----------------------------------------------------
                //g1 is created by dbContext1.
                //We update g1 Name and then submit the change to the database.
                //g2 are created by dbContext2 and still use the old Name.
                Console.WriteLine("4.2.2. -------------------------- ");
                if (g1 != null) g1.Name += "_New";
                 dbContext1.SubmitChanges();
                Console.WriteLine("g1.Name += \"_New\"; dbContext1.SubmitChanges();");
                 g2 = dbContext2.Gamers.FirstOrDefault(g => g.Id == 1);
                Console.WriteLine($"g1: {g1}");
```

```
                Console.WriteLine($"g2: {g2}");
                // UPDATE [dbo].[Gamer]
                // SET [Name] = @p6
                // WHERE ([Id] = @p0) AND ([Name] = @p1) AND ([Gender] = @p2) AND ([Score] = @p3) AND
([Type] = @p4) AND ([CombatPower] = @p5) AND ([MagicPower] IS NULL)
                // -- @p0: Input Int (Size = -1; Prec = 0; Scale = 0) [1]
                // -- @p1: Input NVarChar (Size = 4000; Prec = 0; Scale = 0) [Name1 ABC]
                // -- @p2: Input NVarChar (Size = 4000; Prec = 0; Scale = 0) [Male]
                // -- @p3: Input Int (Size = -1; Prec = 0; Scale = 0) [5555]
                // -- @p4: Input NVarChar (Size = 4000; Prec = 0; Scale = 0) [Warrior]
                // -- @p5: Input Int (Size = -1; Prec = 0; Scale = 0) [500]
                // -- @p6: Input NVarChar (Size = 4000; Prec = 0; Scale = 0) [Name1 ABC_New]
                // -- Context: SqlProvider(Sql2008) Model: AttributedMetaModel Build: 4.7.2556.0
                // g1.Name += "_New"; dbContext1.SubmitChanges();
                // g1: Id==1,Name==Name1
ABC_New,Gender==Male,Score==5555,Type==Warrior,CombatPower==500,MagicPower==
                // g2: Id==1,Name==Name1
ABC,Gender==Male,Score==5555,Type==Warrior,CombatPower==500,MagicPower==
                // 4.2.3. -----------------------------------------------------
                //g1 is created by dbContext1.
                //We update g1 Name and then submit change to database.
                //g2 are created by dbContext2 and still use the old Name.
                //So we have to refresh the data of g2.
                ////dbContext2.Refresh(System.Data.Linq.RefreshMode.OverwriteCurrentValues, g2);
                //Refresh dbContext and then overwrite g2 variable
                Console.WriteLine("4.2.3. -------------------------- ");
                if (g2 != null) dbContext2.Refresh(System.Data.Linq.RefreshMode.OverwriteCurrentValues,
g2);
                //Refresh dbContext and then overwrite g2 variable
                Console.WriteLine("dbContext2.Refresh(RefreshMode.OverwriteCurrentValues, g2)");
                Console.WriteLine($"g1: {g1}");
                Console.WriteLine($"g2: {g2}");
                // SELECT [t0].[Id], [t0].[Name], [t0].[Gender], [t0].[Score], [t0].[Type],
[t0].[CombatPower], [t0].[MagicPower]
                // FROM [dbo].[Gamer] AS [t0]
                // WHERE [t0].[Id] = @p0
                // -- @p0: Input Int (Size = -1; Prec = 0; Scale = 0) [1]
                // -- Context: SqlProvider(Sql2008) Model: AttributedMetaModel Build: 4.7.2556.0
                // dbContext2.Refresh(RefreshMode.OverwriteCurrentValues, g2)
                // g1: Id==1,Name==Name1
ABC_New,Gender==Male,Score==5555,Type==Warrior,CombatPower==500,MagicPower==
                // g2: Id==1,Name==Name1
ABC_New,Gender==Male,Score==5555,Type==Warrior,CombatPower==500,MagicPower==
            }
        }
    }

    public partial class Gamer
    {
        public override string ToString()
        {
            return $"Id=={Id},Name=={Name},Gender=={Gender},Score=={Score},Type=={Type},CombatPower=={Comb
atPower},MagicPower=={MagicPower}";
        }
    }
}
```