(T19)比較LinqToSql的LazyLoading、EagerLoading  
CourseGUID: 5ba9a6fe-7475-4b0c-8b99-bbcf7f5e2e1c  
=======================================================================  
(T19)比較LinqToSql的LazyLoading、EagerLoading  
=======================================================================  
0. Summary

-----------

1. Web Form Application - Linq Query

1.1. TSQL

1.2. Set up SQL Authentication

-----------

2. Console App

2.1. Linq to SQL

2.1.1. Add Connection

2.1.2. Sample.dbml

2.2. Program.cs

-----------

3. Web Form App

3.1. Web.config

3.2. Linq to SQL

3.2.1. Add Connection

3.2.2. Sample.dbml

3.3. WebForm1.aspx

3.3.1. WebForm1.aspx

3.3.2. WebForm1.aspx.cs

-----------

4. LazyLoading V.S. EagerLoading  
=======================================================================

0. Summary

1.

LazyLoading V.S. EagerLoading

Reference:

[https://msdn.microsoft.com/en-us/library/jj574232(v=vs.113).aspx](https://msdn.microsoft.com/en-us/library/jj574232%28v=vs.113%29.aspx)

<https://stackoverflow.com/questions/97197/what-is-n1-select-query-issue>

1.1.

LazyLoading

1.1.1.

We retrieve just the amount of data that we need in a single query.

When we need more data, then it issues more queries to the database.

That means we might have to request the data from database many times,

and this might cost the performance.

1.1.2.

LazyLoading might cause N+1 select problem.

E.g.

One Team can have many Gamers.

One Gamer can have one Team.

This isOne-to-Many relationship.

When we have N teams, and when we select for the Teams,

and then additional selects to retrieve the Gamers **belonging to each Team.**

**That means we have to request the data from database additional N times.**

**This is N+1 select problem.**

1.2.

EagerLoading

We retrieve all data that we need in a single query,

and then be cached to improve the application performance.

That means we just have to request the data from database once,

but this cost memory consumption.

1.3.

Conclusion

1.3.1.

If you need only Team data,

then "lazy loading" works best.

If you choose to use "Eager loading" in this case,

it will cost memory consumption.

1.3.2.

However, if you need Team data and its Gamers data,

then "Eager loading" works best.

If you choose to use "lazy loading" in this case,

it will request the data from database too many times,

this cost application performance.

=============================================

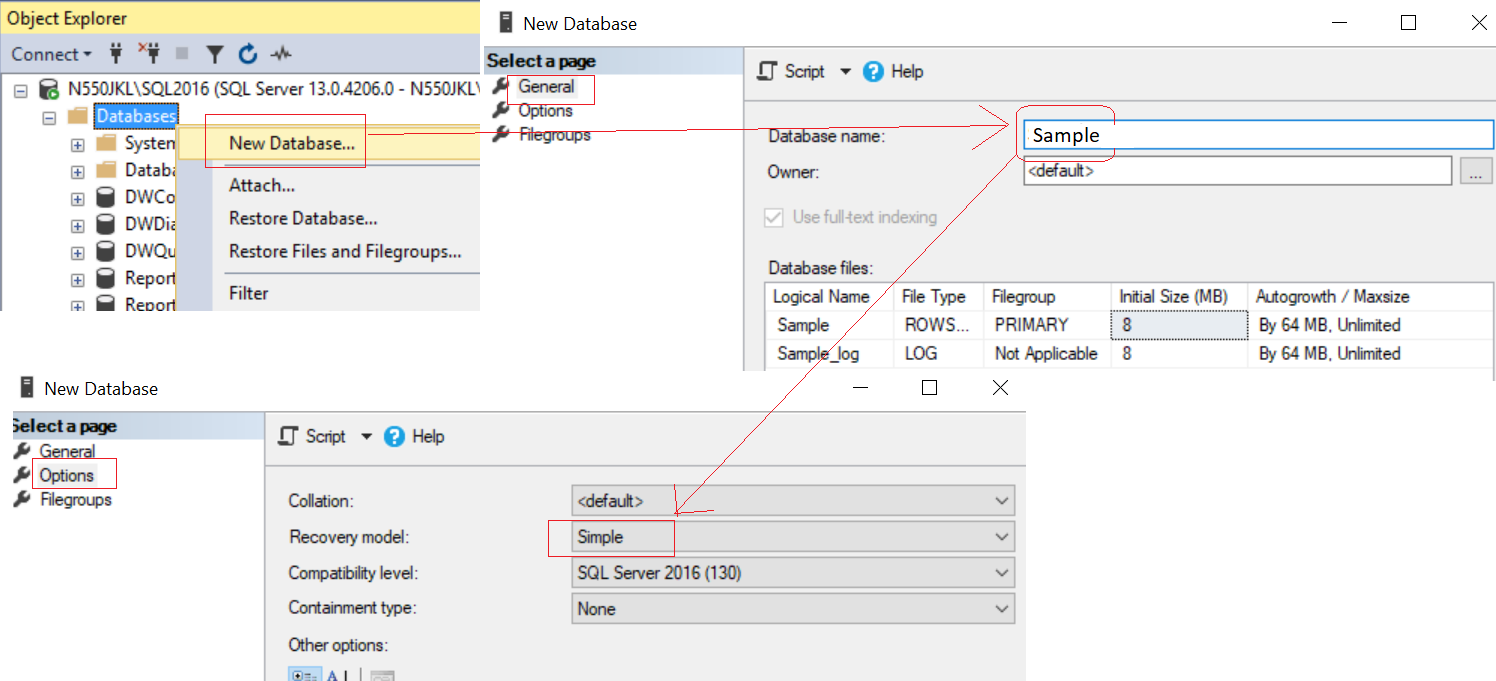
1. Web Form Application - Linq Query

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

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.





Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

2. Console App

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

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

Name: **Sample**

Graphical user interface, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated



2.1. Linq to SQL

2.1.1. Add Connection

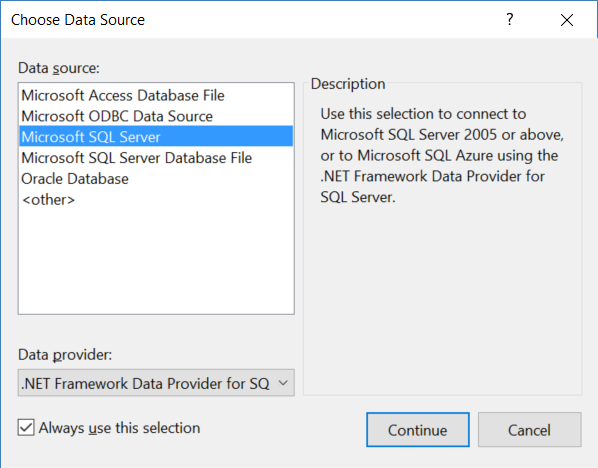
Server Explorer --> Data Connections --> Right click --> Add Connection...

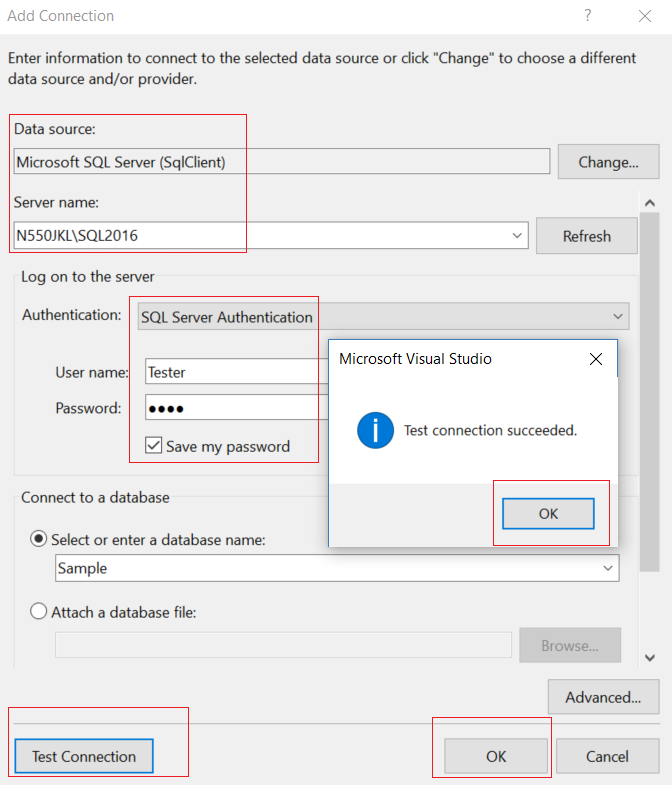
--> Microsoft SQL server -->

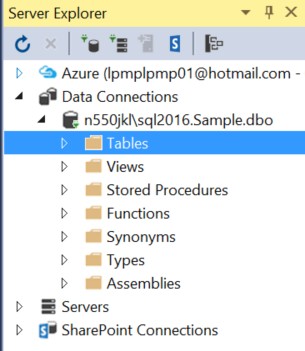
Enter your server and database details ....

Graphical user interface, application

Description automatically generated







2.1.2. Sample.dbml

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

--> Linq to SQL classes -->

Name : **Sample.dbml**

I name it as "Sample.dbml",

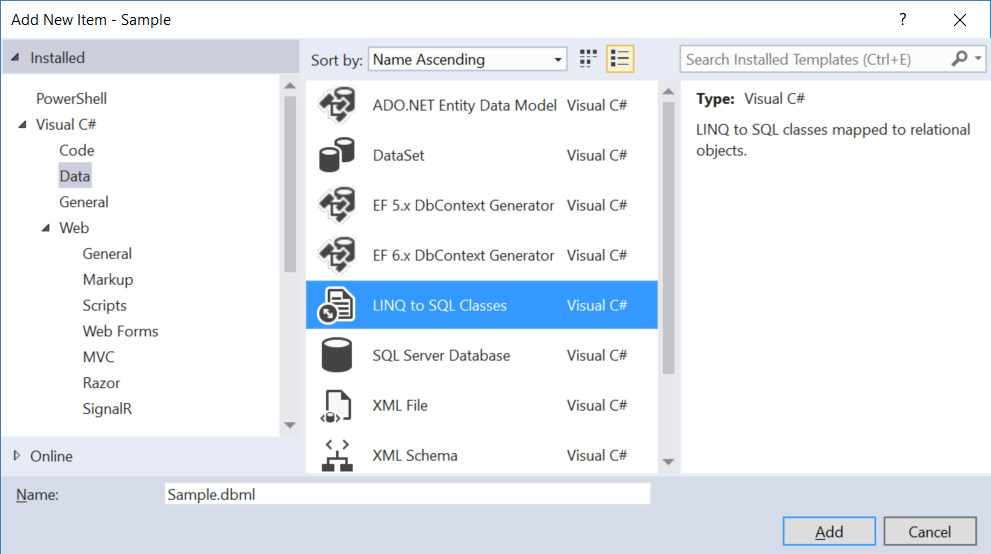
because I know this is for connection to "Sample" Database.

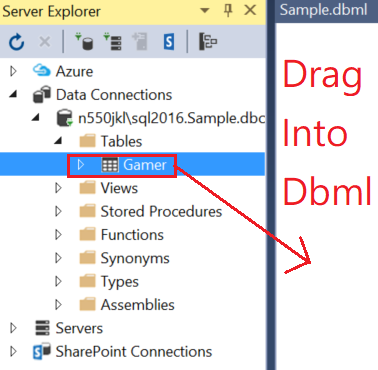
-->

Drag Table from Server Explorer into DBML

Graphical user interface, application

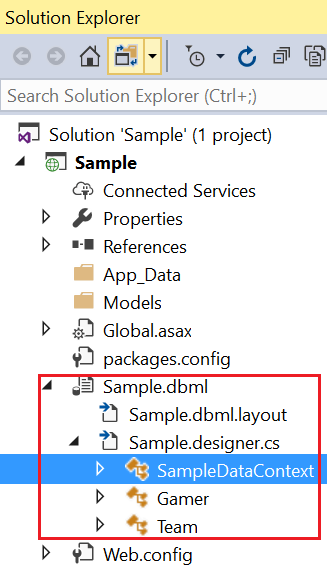
Description automatically generated





Save the dbml, it will generate the following files.

The DataContext context is the entry point to database.



2.2. Program.cs

using System;

using System.Collections.Generic;

using System.Data.Linq;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Sample

{

    class Program

    {

        static void Main(string[] args)

        {

            // 1. ===================================

            //LazyLoading()

            Console.WriteLine("1. LazyLoading() ================== ");

            LazyLoading();

            // 2. ===================================

            //EagerLoading()

            Console.WriteLine("2. EagerLoading() ================== ");

            EagerLoading();

            // 3. ===================================

            //EagerLoading2()

            Console.WriteLine("3. EagerLoading2() ================== ");

            EagerLoading2();

            Console.ReadLine();

        }

        // 1. ===================================

        //LazyLoading()

        static void LazyLoading()

        {

            using (SampleDataContext dbContext = new SampleDataContext())

            {

                //Write the generated sql query to the Console window

                dbContext.Log = Console.Out;

                foreach (Team team in dbContext.Teams)

                {

                    Console.WriteLine("Team -------------------------");

                    Console.WriteLine(team);

                    foreach (Gamer gamer in team.Gamers)

                    {

                        Console.WriteLine("Gamer -----------");

                        Console.WriteLine(gamer);

                    }

                    Console.WriteLine();

                }

            }

        }

        // 2. ===================================

        //EagerLoading()

        static void EagerLoading()

        {

            using (SampleDataContext dbContext = new SampleDataContext())

            {

                //Write the generated sql query to the Console window

                dbContext.Log = Console.Out;

                // Load related Employee entities along with the Department entity

                DataLoadOptions loadOptions = new DataLoadOptions();

                loadOptions.LoadWith<Team>(t => t.Gamers);

                dbContext.LoadOptions = loadOptions;

                foreach (Team team in dbContext.Teams)

                {

                    Console.WriteLine("Team -------------------------");

                    Console.WriteLine(team);

                    foreach (Gamer gamer in team.Gamers)

                    {

                        Console.WriteLine("Gamer -----------");

                        Console.WriteLine(gamer);

                    }

                    Console.WriteLine();

                }

            }

        }

        // 3. ===================================

        //EagerLoading2()

        static void EagerLoading2()

        {

            using (SampleDataContext dbContext = new SampleDataContext())

            {

                //Write the generated sql query to the Console window

                dbContext.Log = Console.Out;

                var linqQuery = from team in dbContext.Teams

                                select new { Team = team, Gamers = team.Gamers };

                foreach (var linqQueryItem in linqQuery)

                {

                    Console.WriteLine("Team -------------------------");

                    Console.WriteLine(linqQueryItem.Team);

                    foreach (Gamer gamer in linqQueryItem.Gamers)

                    {

                        Console.WriteLine("Gamer -----------");

                        Console.WriteLine(gamer);

                    }

                    Console.WriteLine();

                }

            }

        }

    }

    public partial class Gamer

    {

        public override string ToString()

        {

            return $"Id=={Id},Name=={Name},Gender=={Gender},Score=={Score},Type=={Type},TeamId=={TeamId}";

        }

    }

    public partial class Team

    {

        public override string ToString()

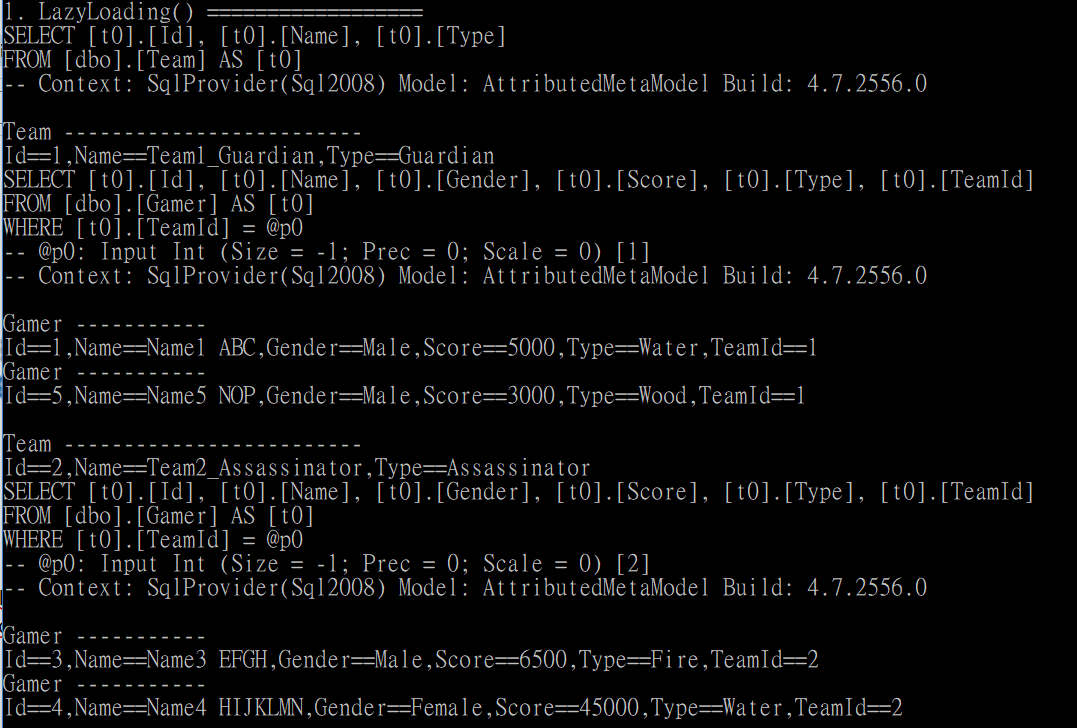
        {

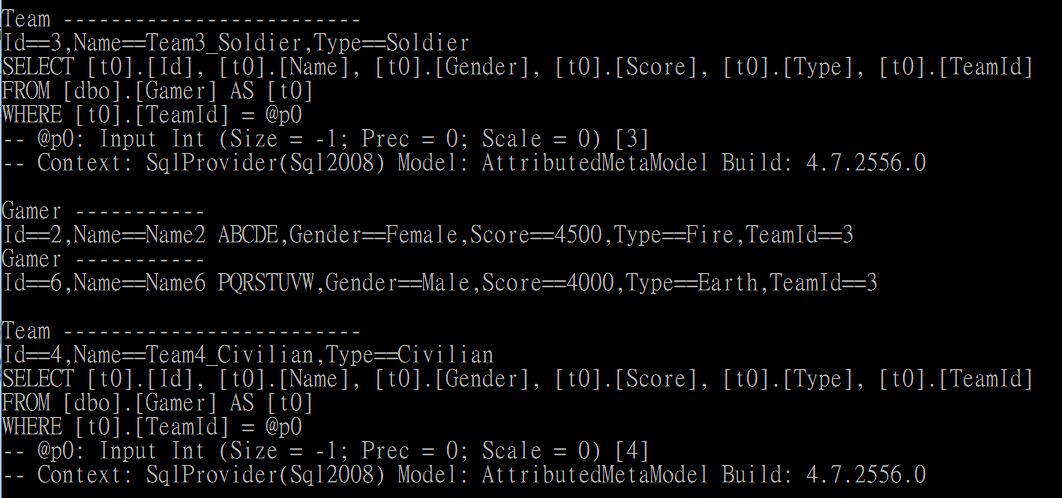
            return $"Id=={Id},Name=={Name},Type=={Type}";

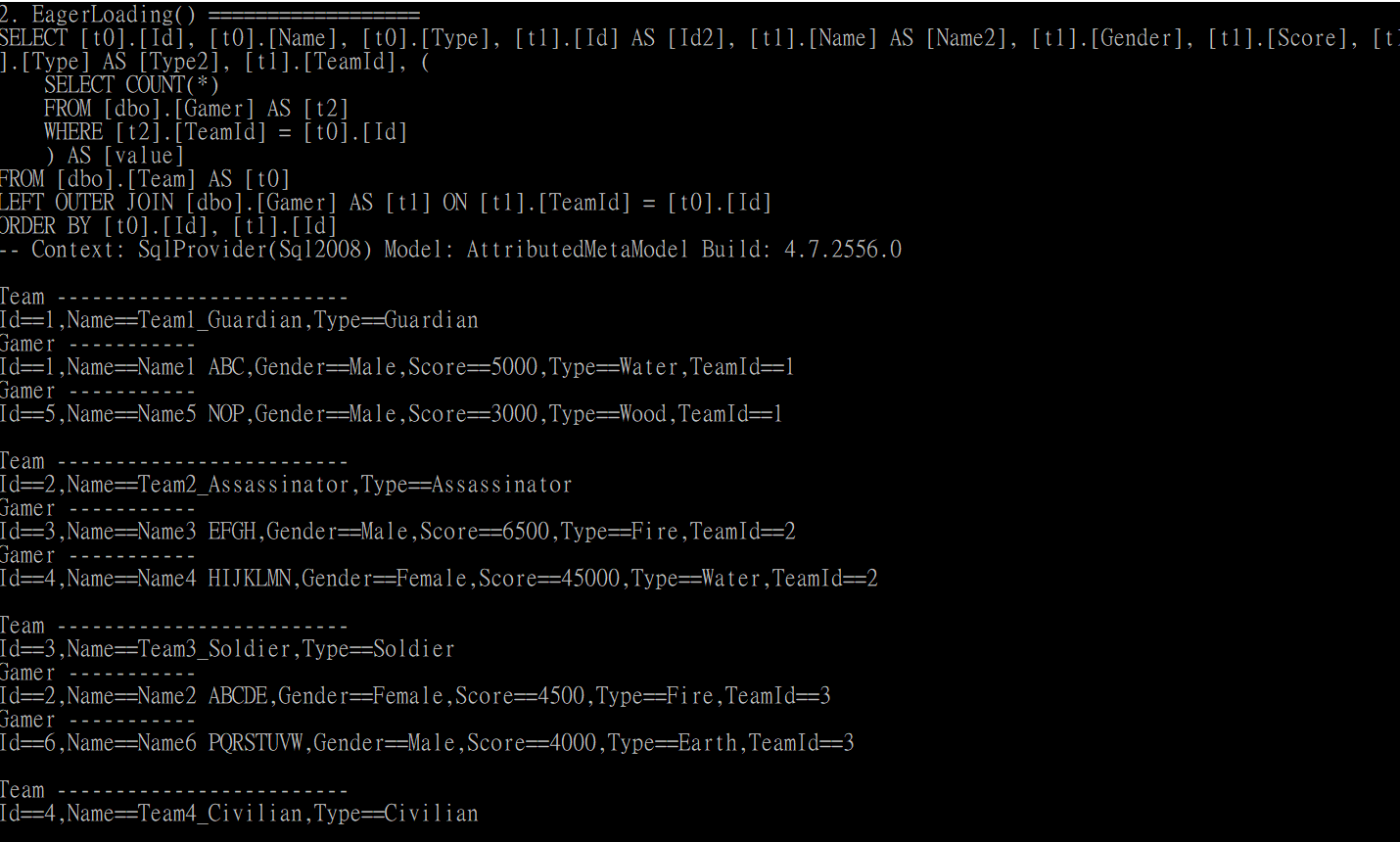
        }

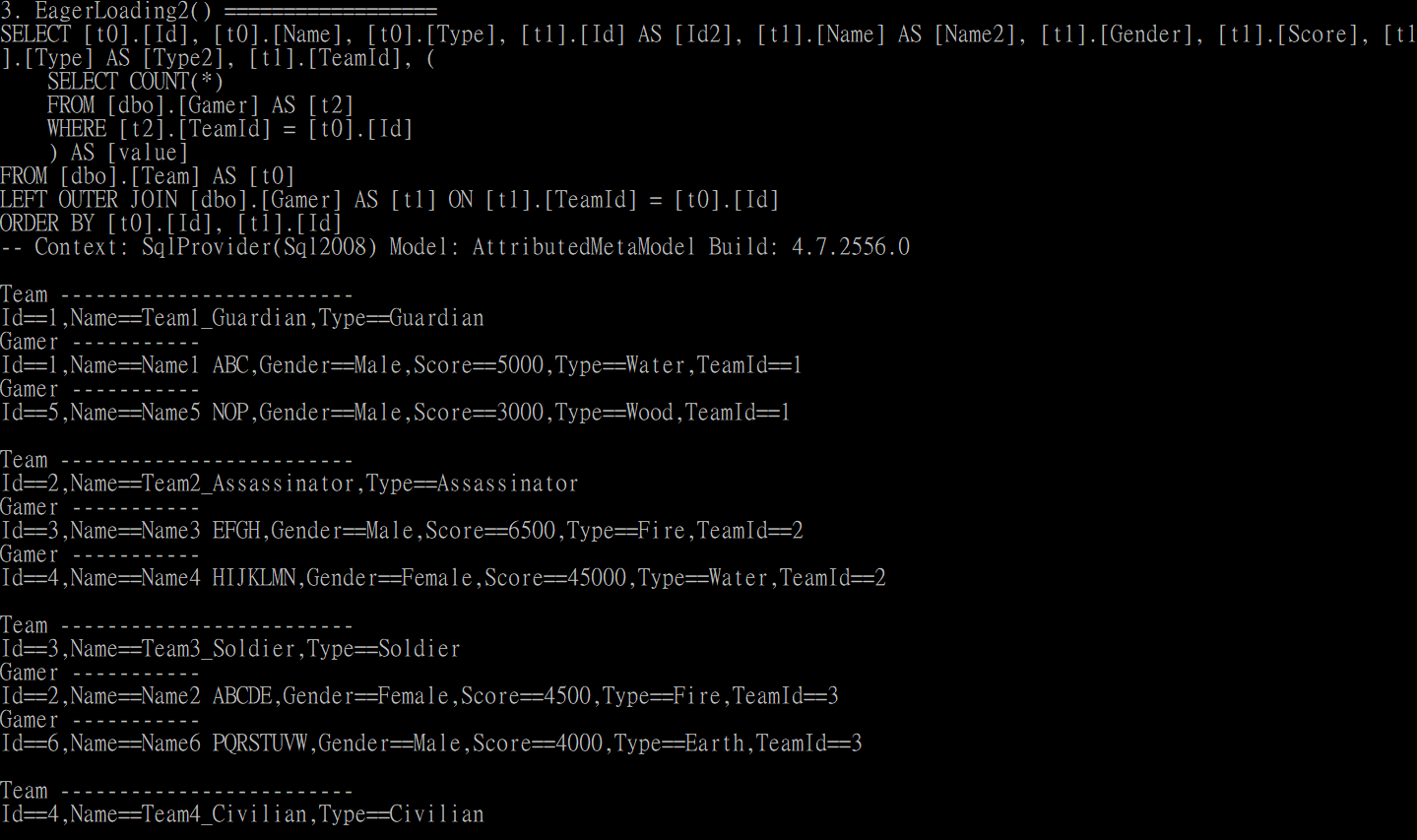
    }

}









3. Web Form App

Open Visual Studio, I am currently using VS2017

If you don't have it, you may following 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

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

3.1. Web.config

Add connection String

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

<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

3.2. Linq to SQL

3.2.1. Add Connection

Server Explorer --> Data Connections --> Right click --> Add Connection...

--> Microsoft SQL server -->

Enter your server and database details ....

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

3.2.2. 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

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

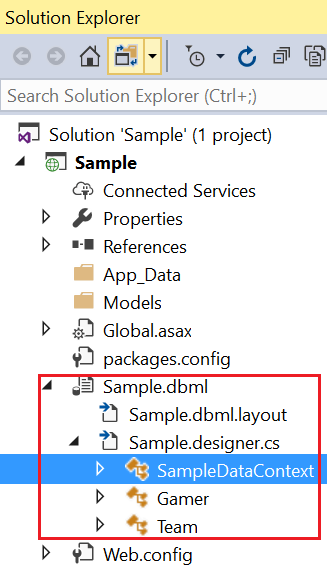
Description automatically generated

Graphical user interface, diagram

Description automatically generated

Save the dbml, it will generate the following files.

The DataContext context is the entry point to database.



3.3. WebForm1.aspx

3.3.1. WebForm1.aspx

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

-->

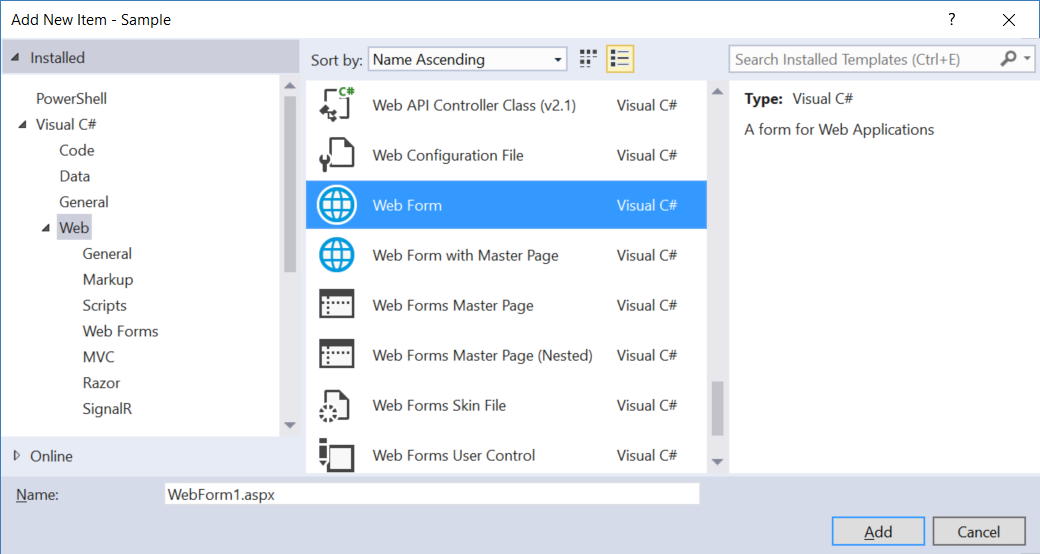
**WebForm**

Name :

**WebForm1.aspx**

Graphical user interface, application

Description automatically generated



<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.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="gvTeams" runat="server" AutoGenerateColumns="False">

                <Columns>

                    <asp:BoundField HeaderText="TeamId" DataField="Id" />

                    <asp:BoundField HeaderText="TeamName" DataField="Name" />

                    <asp:BoundField HeaderText="TeamType" DataField="Type" />

                    <asp:TemplateField HeaderText="Gamers">

                        <ItemTemplate>

                            <asp:GridView ID="gvGamers" runat="server"

                                AutoGenerateColumns="false"

                                DataSource='<%# Eval("Gamers") %>'>

                                <Columns>

                                    <asp:BoundField DataField="Id" HeaderText="Id" />

                                    <asp:BoundField DataField="Name" HeaderText="Name" />

                                    <asp:BoundField DataField="Gender" HeaderText="Gender" />

                                    <asp:BoundField DataField="Score" HeaderText="Score" />

                                    <asp:BoundField DataField="Type" HeaderText="Type" />

                                </Columns>

                            </asp:GridView>

                        </ItemTemplate>

                    </asp:TemplateField>

                </Columns>

            </asp:GridView>

            <br />

            <br />

            <asp:GridView ID="gvTeams2" runat="server" AutoGenerateColumns="False">

                <Columns>

                    <asp:BoundField HeaderText="TeamId" DataField="Id" />

                    <asp:BoundField HeaderText="TeamName" DataField="Name" />

                    <asp:BoundField HeaderText="TeamType" DataField="Type" />

                    <asp:TemplateField HeaderText="Gamers">

                        <ItemTemplate>

                            <asp:GridView ID="gvGamers2" runat="server"

                                AutoGenerateColumns="false"

                                DataSource='<%# Eval("Gamers") %>'>

                                <Columns>

                                    <asp:BoundField DataField="Id" HeaderText="Id" />

                                    <asp:BoundField DataField="Name" HeaderText="Name" />

                                    <asp:BoundField DataField="Gender" HeaderText="Gender" />

                                    <asp:BoundField DataField="Score" HeaderText="Score" />

                                    <asp:BoundField DataField="Type" HeaderText="Type" />

                                </Columns>

                            </asp:GridView>

                        </ItemTemplate>

                    </asp:TemplateField>

                </Columns>

            </asp:GridView>

            <br />

            <br />

            <asp:Label ID="lbl3" runat="server" Text=""></asp:Label>

            <asp:GridView ID="gvTeams3" runat="server" AutoGenerateColumns="False">

                <Columns>

                    <asp:BoundField HeaderText="TeamId" DataField="Id" />

                    <asp:BoundField HeaderText="TeamName" DataField="Name" />

                    <asp:BoundField HeaderText="TeamType" DataField="Type" />

                    <asp:TemplateField HeaderText="Gamers">

                        <ItemTemplate>

                            <asp:GridView ID="gvGamers3" runat="server"

                                AutoGenerateColumns="false"

                                DataSource='<%# Eval("Gamers") %>'>

                                <Columns>

                                    <asp:BoundField DataField="Id" HeaderText="Id" />

                                    <asp:BoundField DataField="Name" HeaderText="Name" />

                                    <asp:BoundField DataField="Gender" HeaderText="Gender" />

                                    <asp:BoundField DataField="Score" HeaderText="Score" />

                                    <asp:BoundField DataField="Type" HeaderText="Type" />

                                </Columns>

                            </asp:GridView>

                        </ItemTemplate>

                    </asp:TemplateField>

                </Columns>

            </asp:GridView>

            <br />

            <br />

        </div>

    </form>

</body>

</html>

3.3.2. WebForm1.aspx.cs

using System;

using System.Data.Linq;

using System.Linq;

namespace WebApplication1

{

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

    {

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

        {

            // 1. ===================================

            LazyLoading();

            // 2. ===================================

            EagerLoading();

            // 3. ===================================

            EagerLoading2();

        }

        // 1. ===================================

        private void LazyLoading()

        {

            using (SampleDataContext dbContext = new SampleDataContext())

            {

                ////Write the generated sql query to the Console window

                //dbContext.Log = Console.Out;

                //Write the generated sql query to the webform

                dbContext.Log = Response.Output;

                //IQueryable<Team> linqQuery =

                //    from team in dbContext.Teams

                //    select team;

                //Response.Write($"<br/>dbContext.GetCommand(linqQuery).CommandText<br/>{dbContext.GetCommand(linqQuery).CommandText}<br/><br/>");

                gvTeams.DataSource = dbContext.Teams;

                gvTeams.DataBind();

            }

        }

        // 2. ===================================

        private void EagerLoading()

        {

            using (SampleDataContext dbContext = new SampleDataContext())

            {

                Response.Write("<br/><br/>gvTeams2==========================<br/>");

                //Write the generated sql query to the webform

                dbContext.Log = Response.Output;

                DataLoadOptions loadOptions = new DataLoadOptions();

                loadOptions.LoadWith<Team>(t => t.Gamers);

                dbContext.LoadOptions = loadOptions;

                gvTeams2.DataSource = dbContext.Teams;

                gvTeams2.DataBind();

            }

        }

        // 3. ===================================

        private void EagerLoading2()

        {

            using (SampleDataContext dbContext = new SampleDataContext())

            {

                var linqQuery = from team in dbContext.Teams

                                select new { Id = team.Id, Name=team.Name, Type=team.Type, Gamers = team.Gamers };

                lbl3.Text = $"<br/>dbContext.GetCommand(linqQuery).CommandText<br/>{dbContext.GetCommand(linqQuery).CommandText}<br/><br/>";

                gvTeams3.DataSource = linqQuery;

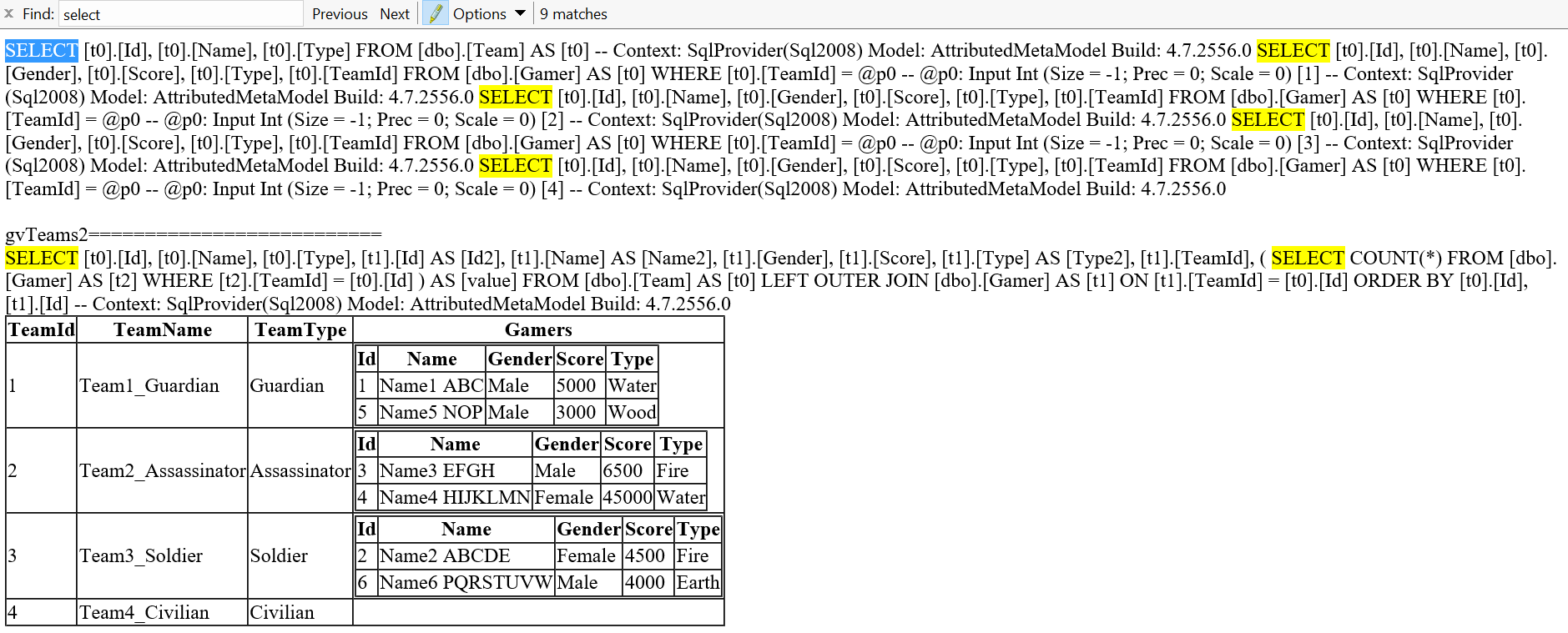
                gvTeams3.DataBind();

            }

        }

    }

}



Table

Description automatically generated

Table

Description automatically generated with low confidence

4. LazyLoading V.S. EagerLoading

1.

LazyLoading V.S. EagerLoading

Reference:

[https://msdn.microsoft.com/en-us/library/jj574232(v=vs.113).aspx](https://msdn.microsoft.com/en-us/library/jj574232%28v=vs.113%29.aspx)

<https://stackoverflow.com/questions/97197/what-is-n1-select-query-issue>

1.1.

LazyLoading

1.1.1.

We retrieve just the amount of data that we need in a single query.

When we need more data, then it issues more queries to the database.

That means we might have to request the data from database many times,

and this might cost the performance.

1.1.2.

LazyLoading might cause N+1 select problem.

E.g.

One Team can have many Gamers.

One Gamer can have one Team.

This isOne-to-Many relationship.

When we have N teams, and when we select for the Teams,

and then additional selects to retrieve the Gamers belonging to each Team.

That means we have to request the data from database additional N times.

This is N+1 select problem.

1.2.

EagerLoading

We retrieve all data that we need in a single query,

and then be cached to improve the application performance.

That means we just have to request the data from database once,

but this cost memory consumption.

1.3.

Conclusion

1.3.1.

If you need only Team data,

then "lazy loading" works best.

If you choose to use "Eager loading" in this case,

it will cost memory consumption.

1.3.2.

However, if you need Team data and its Gamers data,

then "Eager loading" works best.

If you choose to use "lazy loading" in this case,

it will request the data from database too many times,

this cost application performance.