TaeM Framework

User Guide

Document version 1.0.1.0

Feb 2020

(http://www.taemcorp.net/)

TaeM Framework User Guide

Document version 1.0.1.0

http://www.taemcorp.net/html/products/taem_framework.html

Welcome

Welcome to the TaeM Framework User guide. This TaeM Framework consists of functions and libraries for software engineer to easily make good software. This TaeM Framework version 1.0.1.0 includes ORM functionality. In the future, we plan to add new features that if often uses to make software.

License and Agreement

TaeM Framework End-User License Agreement (“Agreement”)

Last updated: Feb, 19, 2020

Please read this End-User License Agreement (“EULA”) carefully:

This End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and Author of this Software ("TaeM corp") for the Software Product "TaeM Framework", which includes computer software and may include associated media, printed materials, and "online" or electronic documentation ("Software Product"). The Software Product also includes any updates and supplements to original software provided to you.

By installing, copying, or otherwise using the Software Product, you agree to be bound by the terms of this EULA. If you do not agree to any of the terms of this EULA, you are not permitted to use the Software Product in any way, and all copies of it must be deleted from your system(s).

The Software Product is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The Software Product is licensed, not sold.

1. GRANT OF LICENSE

TaeM corp grants you the rights described in this EULA provided that you comply with all the terms and conditions of this EULA:

1.1 General Software License Grant.

You are granted a right to install the Software Product and use it in order to process your own software with the Software Product. TaeM corp grants the use of the Software Product according to one of the license types below as identified in the product title. Licenses purchased for development do not extend to third parties even if the software was developed for that third party. All entities must have a separate license.

1.1.1 Trial/Demo version (No costs)

Use of the Software Product without purchase of a License shall be limited to evaluation purposes only. Programs created using a trial license must be destroyed when the trial period has expired. Software protected using a trial license cannot be bought, sold, licensed, copied, traded or otherwise used by other individuals.

1.1.2 Full license (Perpetual license)

Any Software Product is available on a “per-user" or "per-machine” basis. You may install, activate, operate, and use the Software Product on ONE (1) computer per license purchased by you. While making an order in frames of the current EULA, you can purchase

the number of licenses you require.

1.1.3 Subscription-based license

TaeM corp also support subscription- based license. it also is available on a “per-user" or "per-machine” basis. You may install, activate, operate, and use the Software Product on ONE (1) computer per license purchased by you. This subscription-based license will be accessible for use only during the subscription term specified at purchase.

Once the subscription term is over, you will be able to either stop using the Software Product or renew the license for a new subscription term. While making an order in frames of the current EULA, you can purchase the number of licenses you require and choose the subscription term for which you want to purchase these licenses.

1.2 Redistribution.

1.2.1 You can redistribute in binary form any components of the Software Product explicitly marked as redistributable provided that you provide all technical support for the distribution. However, you must purchase a license for the system being redistributed.

You do not allow recipients to disassemble, decompile, or in any other way allowing them to gain separate access to the Software Product or any part of the Software Product.

1.2.2 TaeM corp is not obligated to provide support for works derived from the Software Product.

1.3 Disassembly.

You may not modify this product in any manner. You shall not, nor allow others to copy, in whole or in part, emulate, sub-license, sell, transfer, exploit, alter, modify or adapt the Software Product or decompile, dissemble or reverse engineer the same nor attempt to do such thing.

1.4 Reservation of Rights.

TaeM corp reserves all rights not expressly granted herein.

2. LIMITATIONS

Only legally registered users are licensed to use the Software, subject to all of the conditions of this Agreement. Usage of the Software is subject to the following restrictions.

2.1 You may not develop any applications that use or are based on the Software explicitly or implicitly without obtaining an appropriate license from TaeM corp. This includes, but is not limited to, enhancing, modifying, or developing applications, services, web applications, Integration Services packages, Analysis Services projects, or Reporting Services reports that use the Software.

2.2 You may not transfer, assign, or modify the Software, in whole or in part. In particular, the Software license is non-transferable, and you may not transfer the Software installation package.

2.3 You may not reverse engineer, decompile, or disassemble the Software.

2.4 You may not reproduce or distribute any Software documentation without express written permission from TaeM corp.

3. TERMINATION

TaeM corp may immediately terminate this Agreement without notice or judicial resolution in the event of any failure to comply with any provision of this Agreement. Upon such termination you must destroy the Software, all accompanying written materials, and all copies.

4. WARRENTY

The Software and documentation are provided "AS IS" without warranty of any kind. TaeM corp makes no warranties, expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or use.

5. SUBSCRIPTION AND SUPPORT

The Software is sold on a subscription basis. The Software subscription entitles you to download improvements and enhancement from TaeM corp's web site as they become available, during the active subscription period. The initial subscription period is one year from the date of purchase of the license. The subscription is automatically activated upon purchase, and may be subsequently renewed by TaeM corp, subject to receipt of applicable fees. Licensed users of the Software with an active subscription may request technical assistance with using the Software over email from the Software development team. TaeM corp shall use its reasonable endeavours to answer queries raised, but does not guarantee that your queries or problems will be fixed or solved.

6. COPYRIGHT

All title and copyrights in and to the Software Product (including but not limited to any images, photographs, text, and "apps" incorporated into the Software Product), the accompanying printed materials, and any copies of the Software Product are owned by TaeM corp. The Software Product is protected by copyright laws and international treaty provisions. Therefore, you must treat the Software Product like any other copyrighted material.

8. MARKETING

You agree to be identified as a customer of TaeM corp and that TaeM corp may refer to you by name, trade name and trademark, if applicable, and may briefly describe your business in TaeM corp's marketing materials, on TaeM corp's web site, in public or legal documents. You hereby grant TaeM corp a license to use your name and any of your trade names and trademarks solely pursuant to this marketing section.

9. DISCLAIMER

THIS SOFTWARE IS PROVIDED "AS IS", WITHOUT A WARRENTY OF ANY KIND. THE AUTHOR OF THIS SOFTWARE ("TAEM CORP") DON'T TAKE ANY RESPONSE FOR ANY DAMAGES SUFFERED AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE. IN NO EVENT WILL TAEM CORP BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF TAEM CORP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE AUTHOR DOES NOT WARRANT THAT TAEM FRAMEWORK IS FREE FROM BUGS, ERRORS, OR OTHER PROGRAM LIMITATIONS. LIMITATION OF LIABILITY AND DAMAGES TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE AUTHOR IS NOT LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: DAMAGES FOR LOSS OF BUSINESS, LOSS OF PROFITS OR INVESTMENT, OR THE LIKE), WHETHER BASED ON BREACH OF CONTRACT, BREACH OF WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, EVEN IF THE AUTHOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND EVEN IF A REMEDY SET FORTH HEREIN IS FOUND TO HAVE FAILED OF ITS ESSENTIAL PURPOSE. THE AUTHOR WILL NOT BE SUBJECT TO LIABILITY FOR ANY BUGS OR DAMAGES CAUSED BY CERTAIN TAEM FRAMEWORK FEATURES.

10. LANGUAGE OF AGREEMENT

You agree that this EULA be drafted in the English language.

Table of Contents

Introduction

Thank you for choosing TaeM Framework. Please be sure to read this document before using TaeM Framework.

TaeM Framework

The TaeM Framework is a .NET-based software library developed for all .NET technologies. At this moment, TaeM Framework version 1.0.1.0 includes ORM functionality and we will continue to add new features in the future.

Software Requirement

The following are the software requirements.

- The current installation package can be installed with Windows Installer (msi file) or UWP App Package (msix file or msixupload file). And you need .NET framework or similar. You can install it according to the version of Windows you are using.
- TaeM Framework supports the following .NET Framework. Currently supported versions for each framework are as follows.

.NET Framework
2.0
4.0
4.5
4.7

- For reference, other OS support such as Linux and Mac OS is also considered, but it is not possible at present. This other OS support requires a lot of verification and testing. We will consider this in later versions.

Release Notes

Currently Assembly version 1.0.1.0, File version 1.0.1.0.

The release notes for past releases are available at the following address.

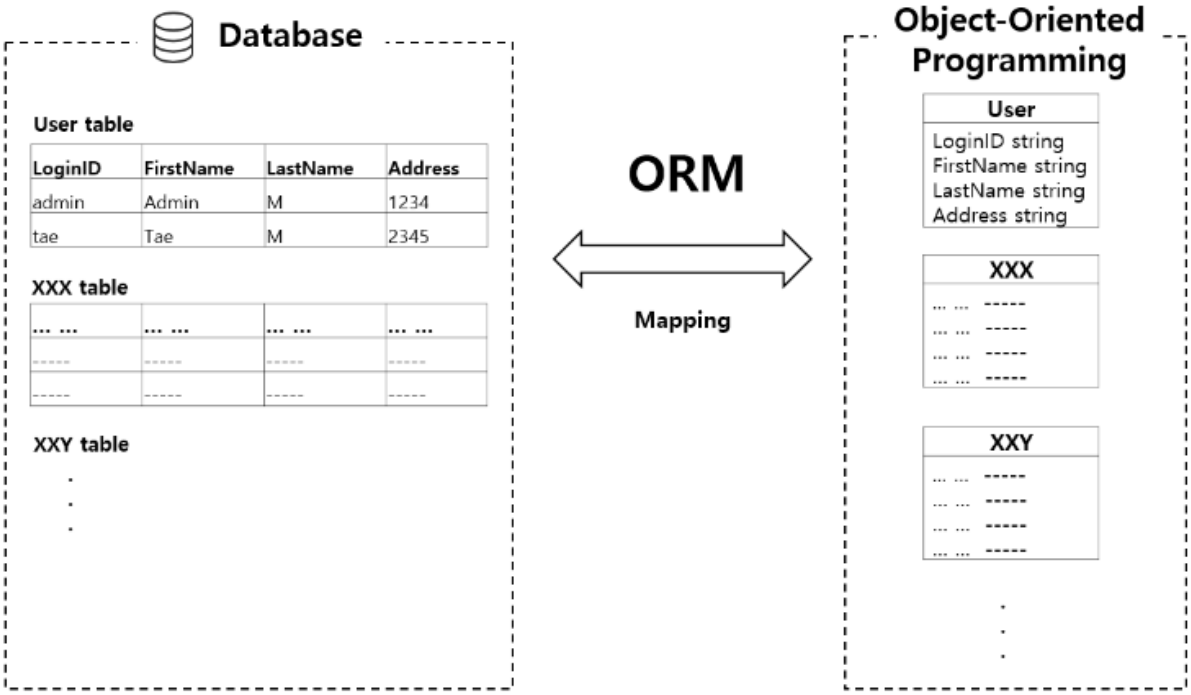
http://www.taemcorp.net/html/company/news.html

Product Features

The TaeM Framework provides the following features.

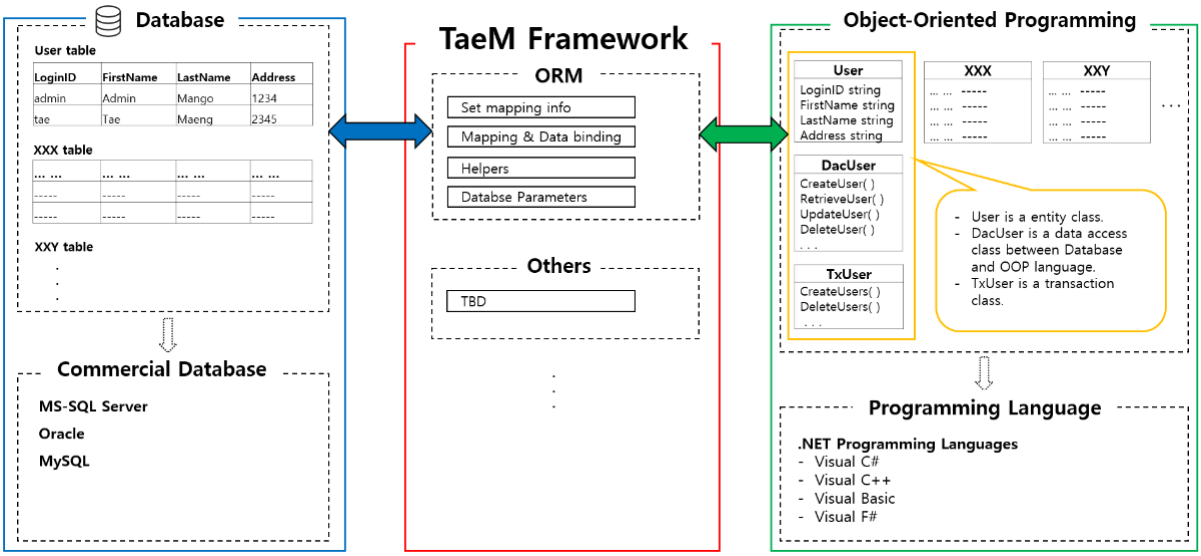
ORM(Object-Relational Mapping)

Object-Relational Mapping (ORM) is a programming technique for transforming incompatible data between a database and an object-oriented programming language. In other words, it is also called object relationship mapping.



[Figure 1 ORM concept]

The Object-Relational Mapping (ORM), as shown in picture, maps tables in Relational Database (RDB) to objects in Object-oriented programming language. There are several Object-Relational Mapping (ORM) frameworks related to this. TaeM Framework also includes Object-Relational Mapping (ORM) in version 1.0.1.0.



[Figure 2 ORM in TaeM Framework]

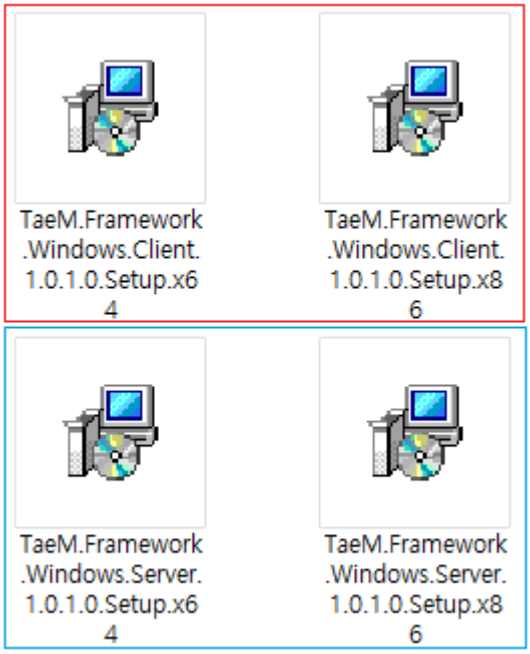
Above picture shows the ORM functions of the TaeM Framework, supported databases, and programming languages that can use the TaeM Framework. TaeM Framework provides object-relational mapping (ORM) functionality for commonly used relational database (RDB) products such as MS-SQL Server, Oracle, MySQL. We will support other relational databases and NoSQL databases in the future.

Install TaeM Framework

The current installation package is Windows Installer (.msi file). You can install the TaeM Framework using this Windows Installer (.msi file).

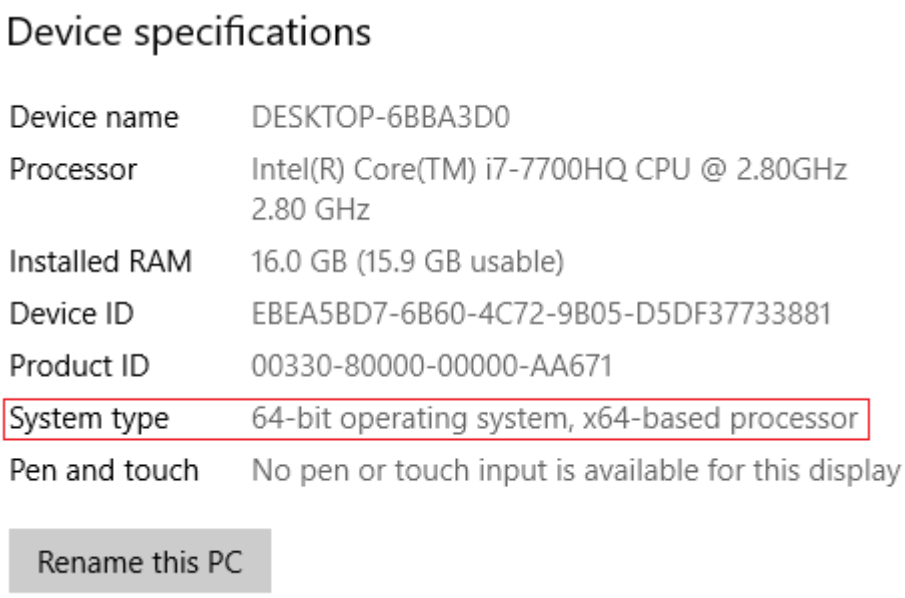
Windows Installer (.msi file)

You might get the following four types of Windows Installer (.msi file) file through various paths.



[Figure 3 TaeM Framework Windows Installer]

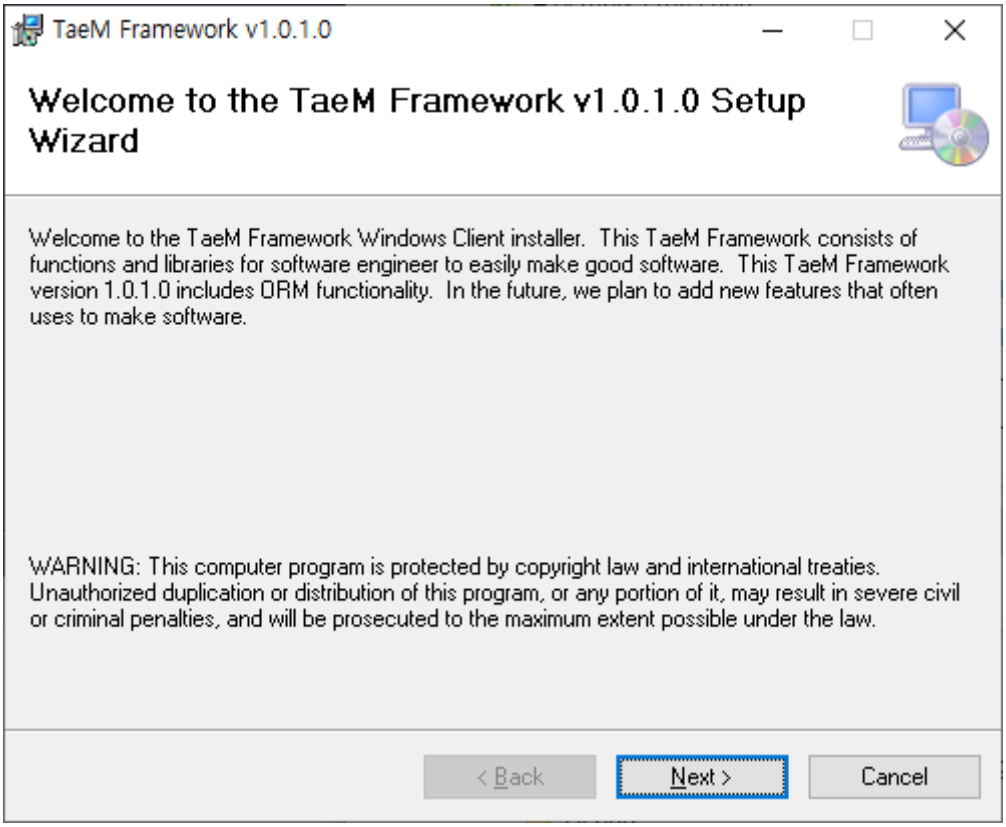
The files in the first line of Figure 3 above (TaeM.Framework.Windows.Client.1.0.1.0.Setup.x64.msi, TaeM.Framework.Windows.Client.1.0.1.0.Setup.x86.msi) are the installers for Windows Client family such as Windows 7, 8, 8.1, and 10. And the files in the second line (TaeM.Framework.Windows.Server.1.0.1.0.Setup.x64.msi, TaeM.Framework.Windows.Server.1.0.1.0.Setup.x86.msi) are the installers for Windows Server family like as Windows Server 2012, 2016 and 2019. In addition, depending on whether Windows is 64-bit or 32-bit, you can run the appropriate installation file. For reference, you can check the system information of your Windows that you use as follows.



[Figure 4 Windows 10 System information]

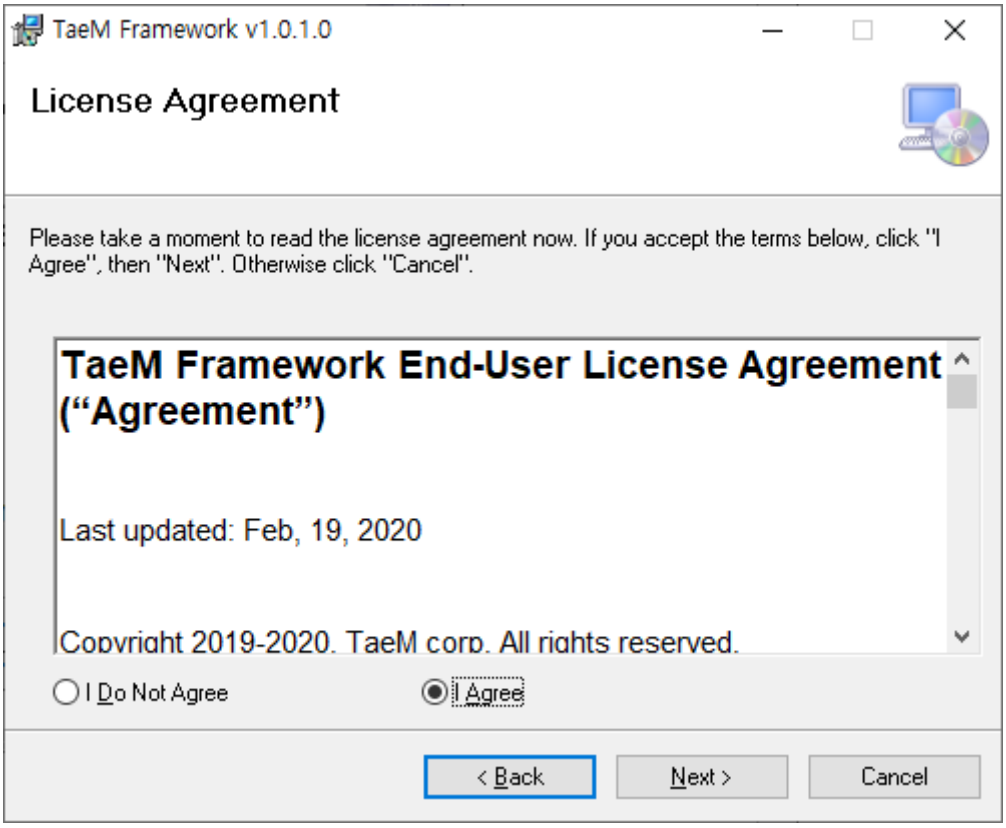
Here, we will explain the installation process using Windows Installer files in a 64-bit environment of Windows 10, but the installation process and method are the same for other Windows Installer files, but the installation location is slightly different.

When you execute the TaeM.Framework.Windows.Client.1.0.1.0.Setup.x64.msi file, the following TaeM Framework installation screen will be displayed.



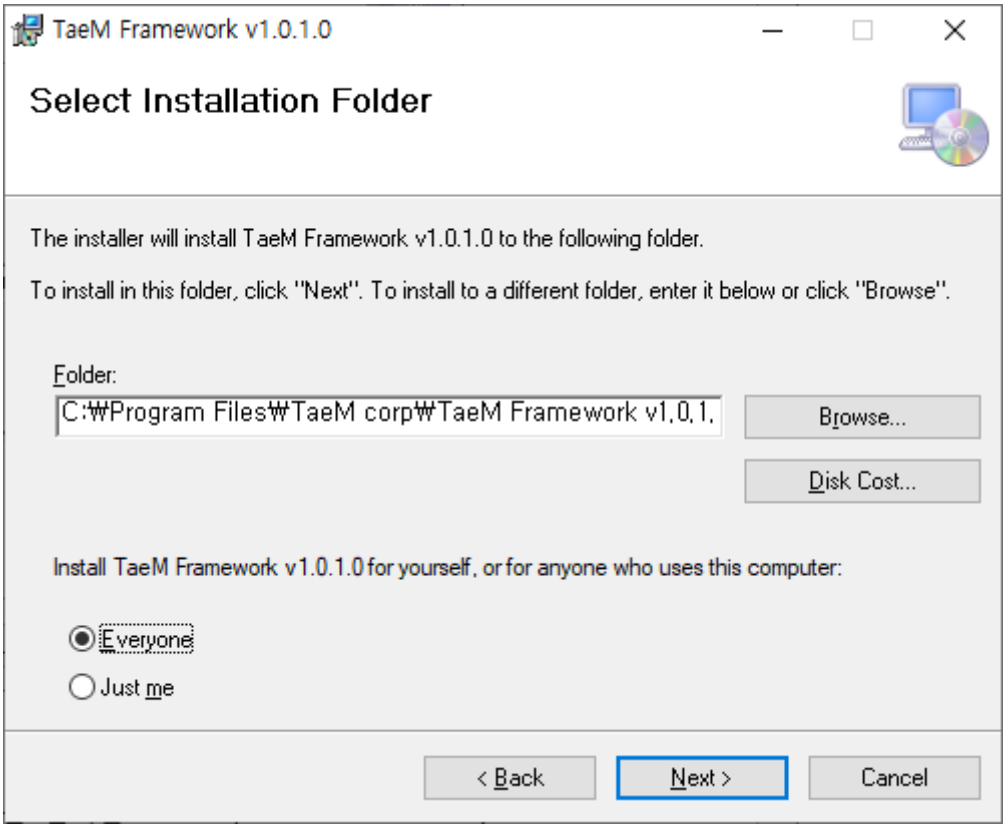
[Figure 5 TaeM Framework installation welcome]

The Welcome screen displays a brief description of the TaeM Framework and usage notes. Click the "Next" button to go to the next screen.



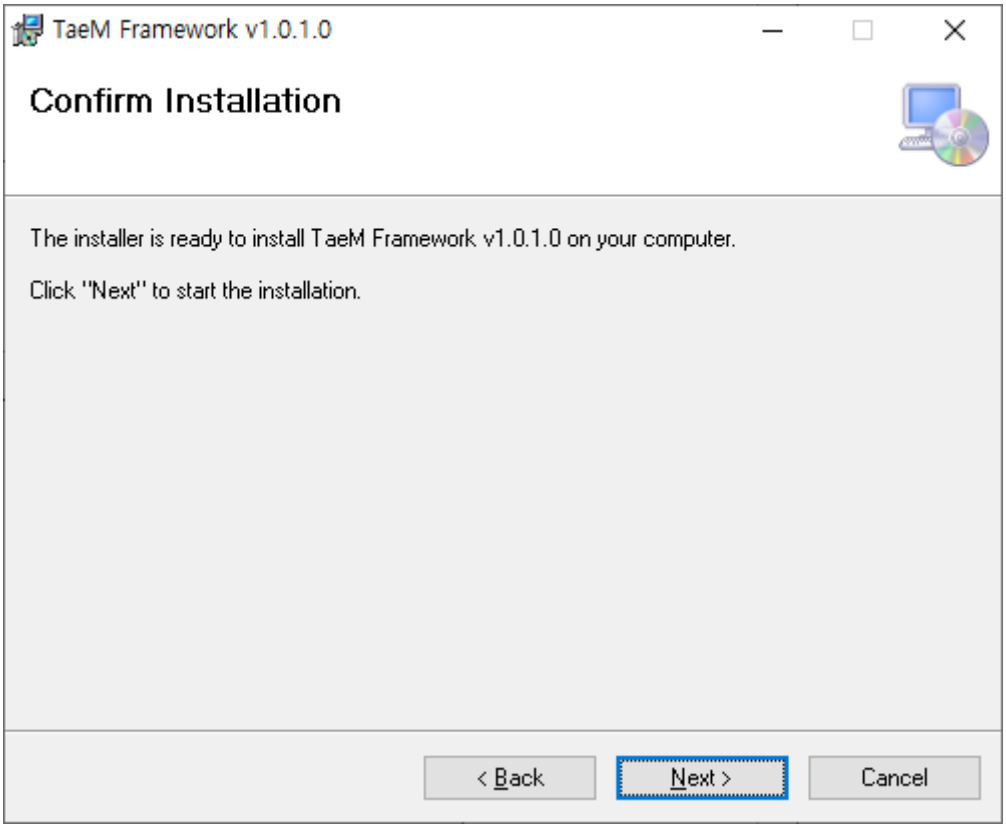
[Figure 6 TaeM Framework License description and its agreemnt]

The License Agreement screen for the TaeM Framework is displayed. Here you can make a choice, such as a description of Lincense and a consent ("I Agree") or ("I Do Not Agree"). However, you must accept the Lincense agreement before you can proceed with the installation. Therefore, we select "I Agree" and click "Next" button to continue installation.



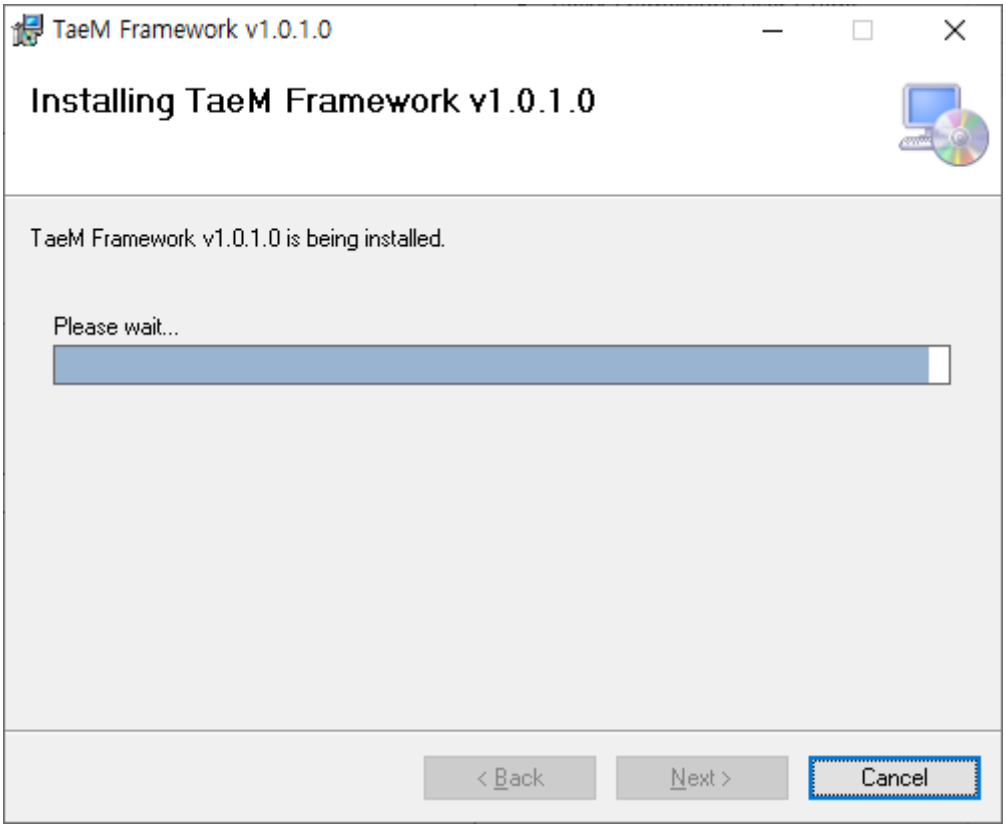
[Figure 7 Install folder]

Figure 7 above shows the default folder "C:\Program Files\TaeM corp\TaeM Framework v1.0.1.0\" to install the TaeM Framework, and you can change the installation path of the TaeM Framework by installing any other folder. In this case, we will install the TaeM Framework in the default path "C:\Program Files\TaeM corp\TaeM Framework v1.0.1.0\", so click the "Next" button without changing the installation folder path..



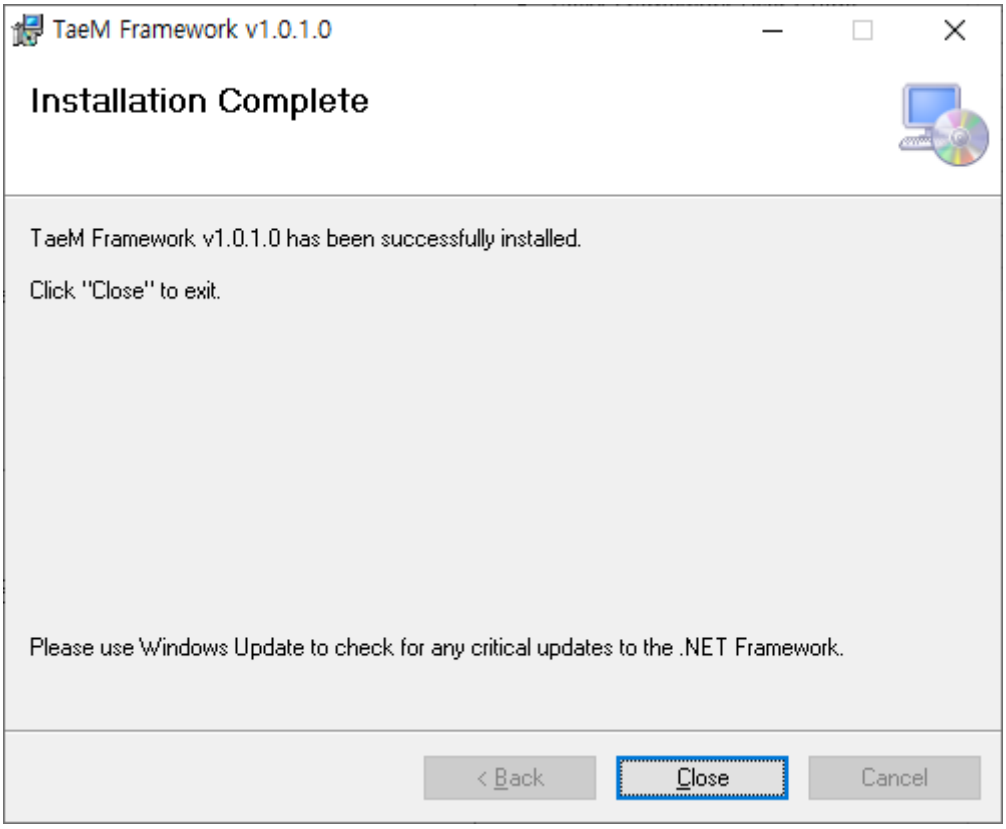
[Figure 8 Pre-Installation confirmation]

Before installing the TaeM Framework, a final confirmation screen is displayed. We will continue with the installation, so click "Next" button.



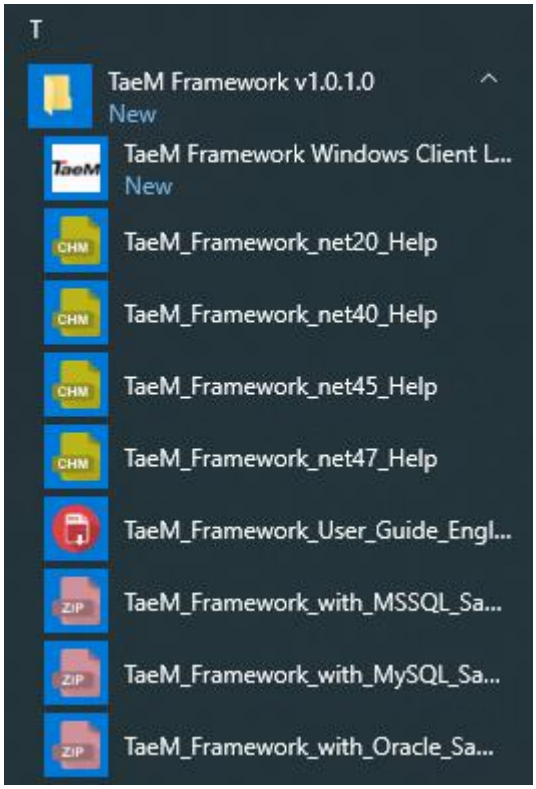
[Figure 9 Installation progress screen]

The installation of the TaeM Framework proceeds as shown above.



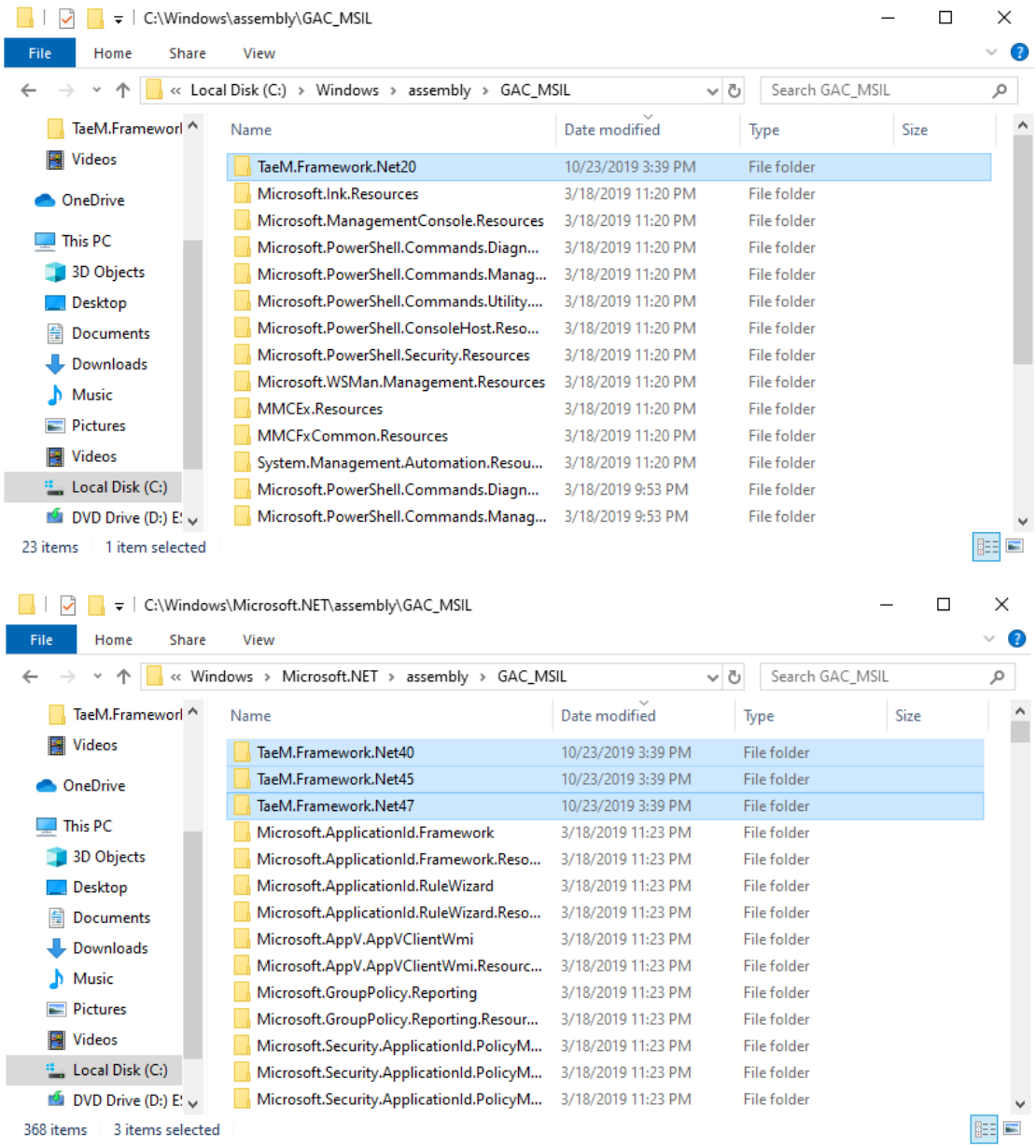
[Figure 10 Installation complete]

The installation of the TaeM Framework is complete. Click the "Close" button to exit Windows Installer.



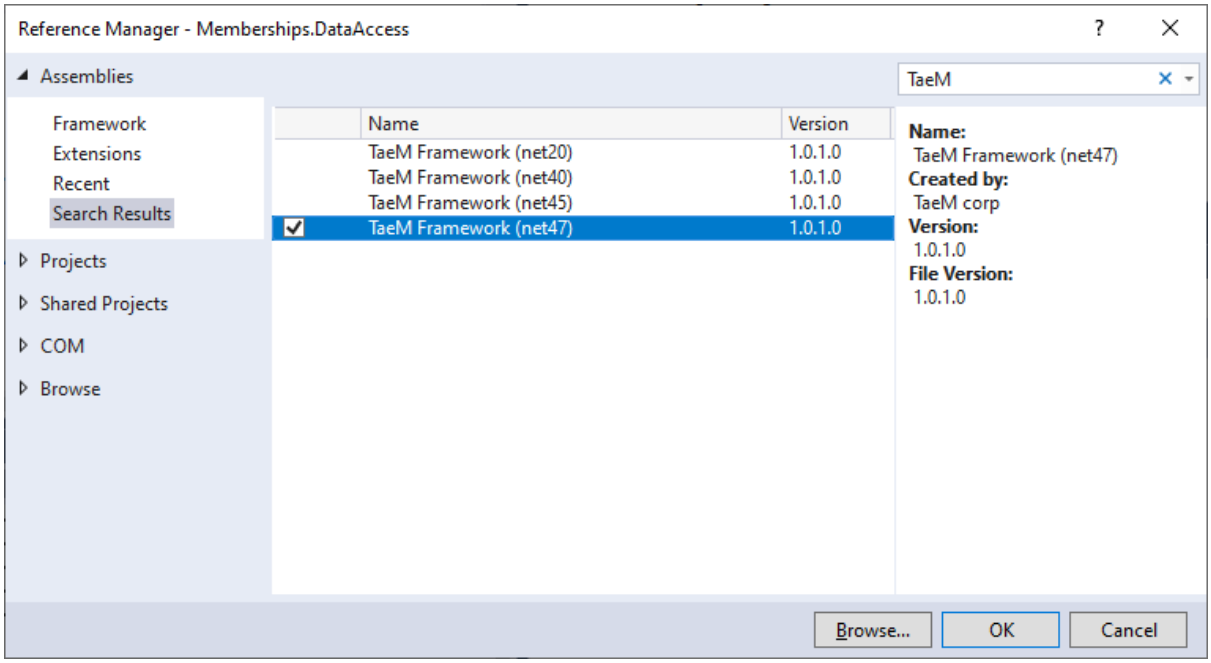
[Figure 11 TaeM Framework in the Windows program menu]

When the installation is complete, you can see that the TaeM Framework has been installed from the Windows program menu as shown above. You can see the TaeM Framework folder in the Windows program menu, which contains the API documentation, user guide and samples for each .NET Framework version.



[Figure 12 TaeM Framework components registered in the GAC]

The figure above shows the TaeM Framework components installed in the Global Assembly Cache (GAC) in Windows Explorer.



[Figure 13 TaeM Framework component in the References dialog box of Visaul Studio]

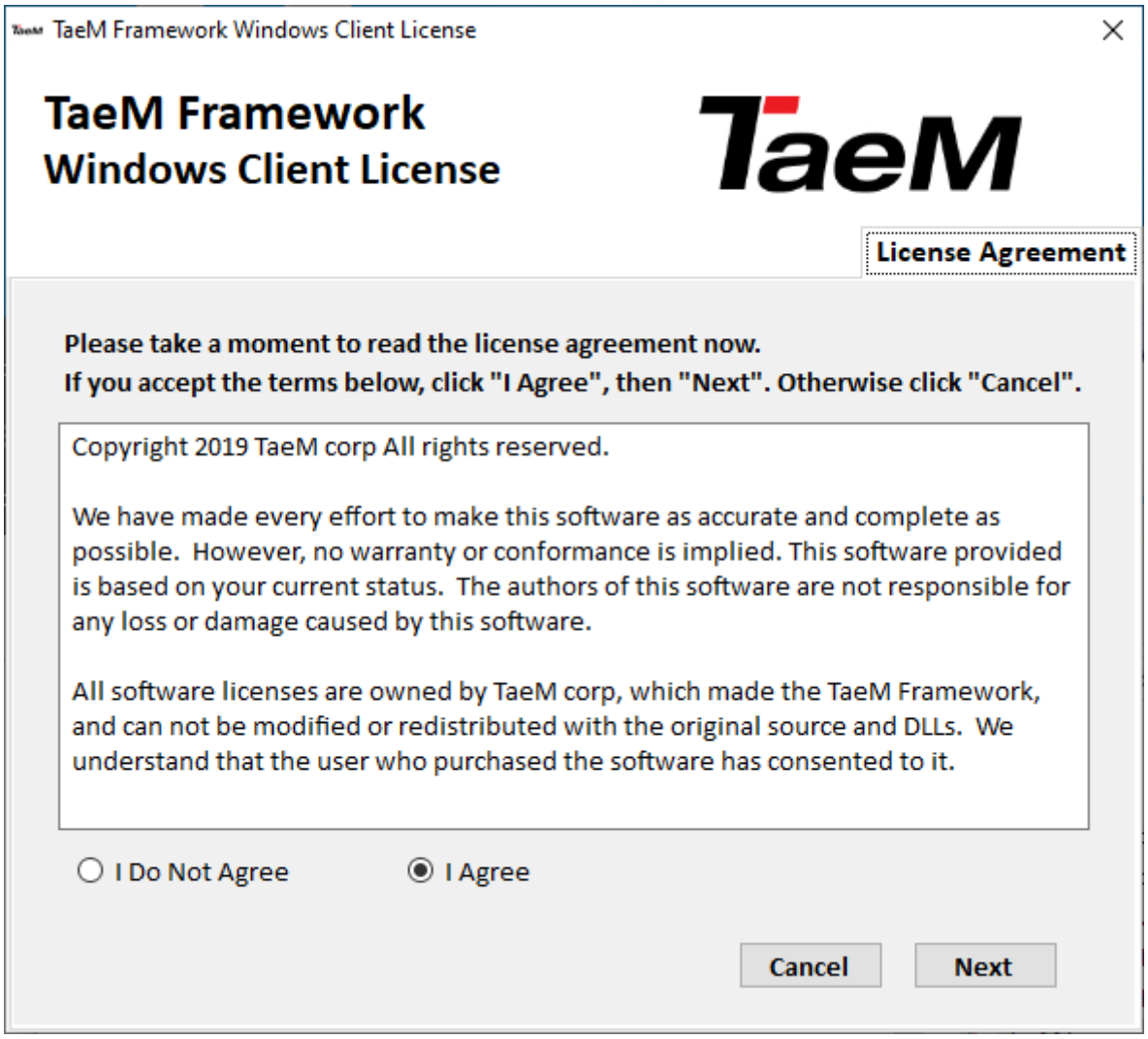
In addition, in the above installation process, but not selected, if you installed the TaeM Framework component display in the reference dialog box of Visual Studio and installed it, each .NET Framework version of TaeM Framework in the reference dialog box of Visaul Studio is displayed as shown above.

License App

The “TaeM Framework” you installed earlier can only run 100 times in 15 days without being certified. So, if you want to continue using it, you need to do the authentication process through the “TaeM Framework Windows Client License” and “TaeM Framework Windows Server License” app. Therefore, after installation, you should be authenticated as shown below.

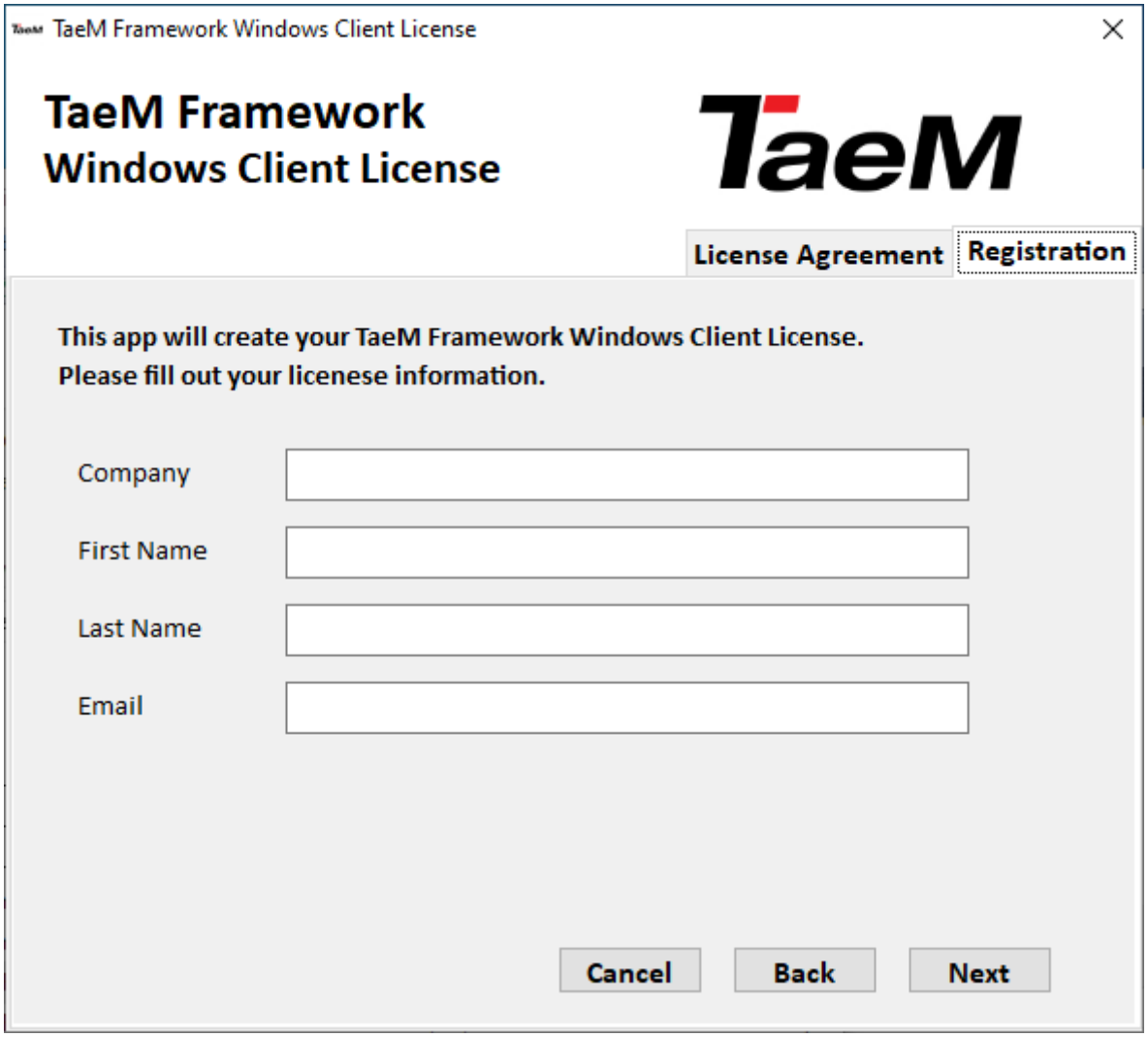
The “TaeM Framework Windows Client License” app is a licensed app that you must purchase when using the “TaeM Framework” on Windows Client family like Windows 7, 8, 8.1, 10. If you want to use “TaeM Framework” on Windows Server family such as Windows Server 2012, 2016, 2019, you need to purchase “TaeM Framework Windows Server License” app.

The figure below shows the screen of the “TaeM Framework Windows Client License” purchased from Windows 7, 8, 8.1, and 10. You must select "I Agree" here and click the "Next" button.



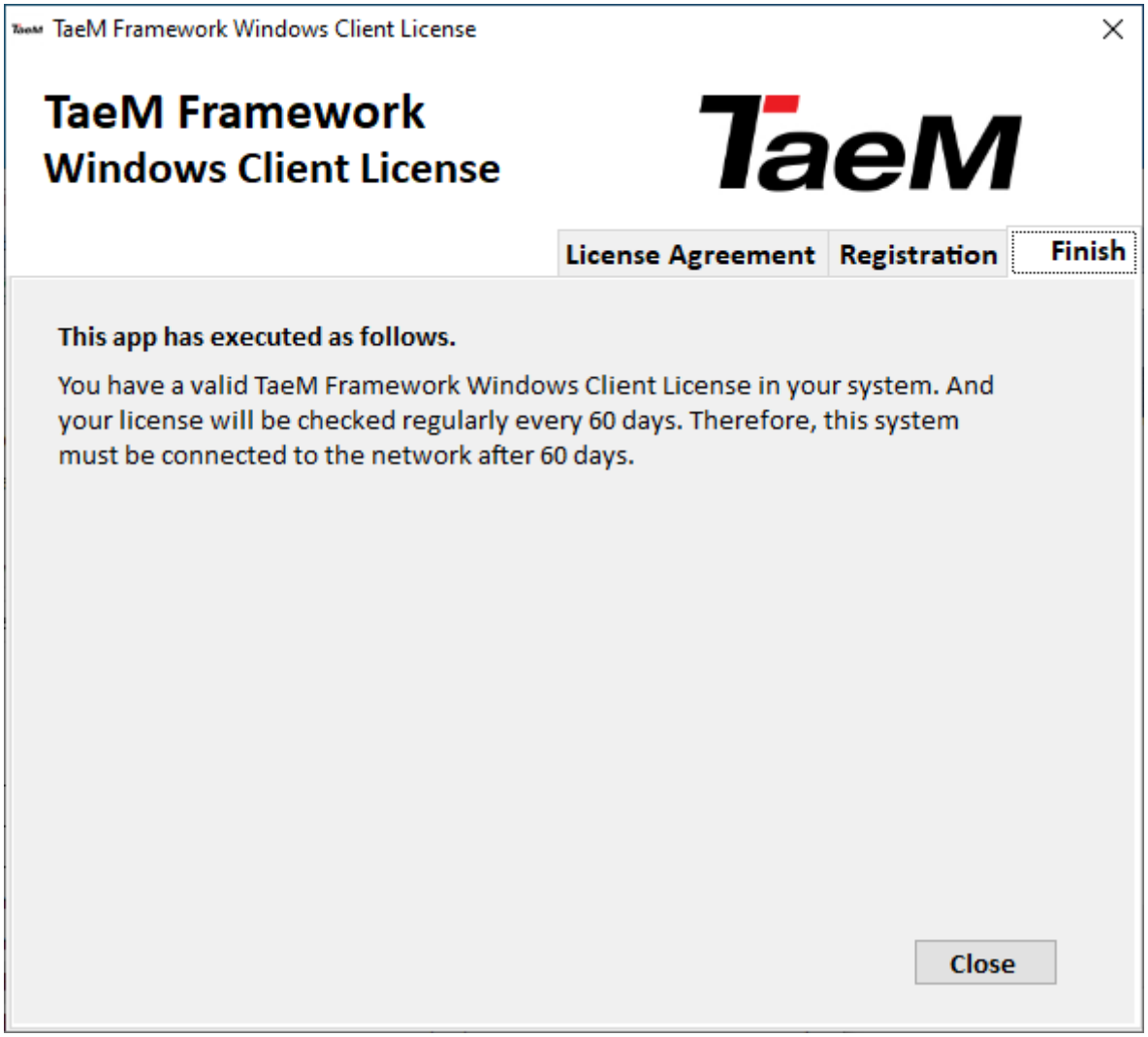
[Figure 14 The license description and its agreement of TaeM Framework Windows Client License]

The figure below shows the user information registration screen. You have to fill out the company name, user first and last name and e-mail, and click the “Next” button.



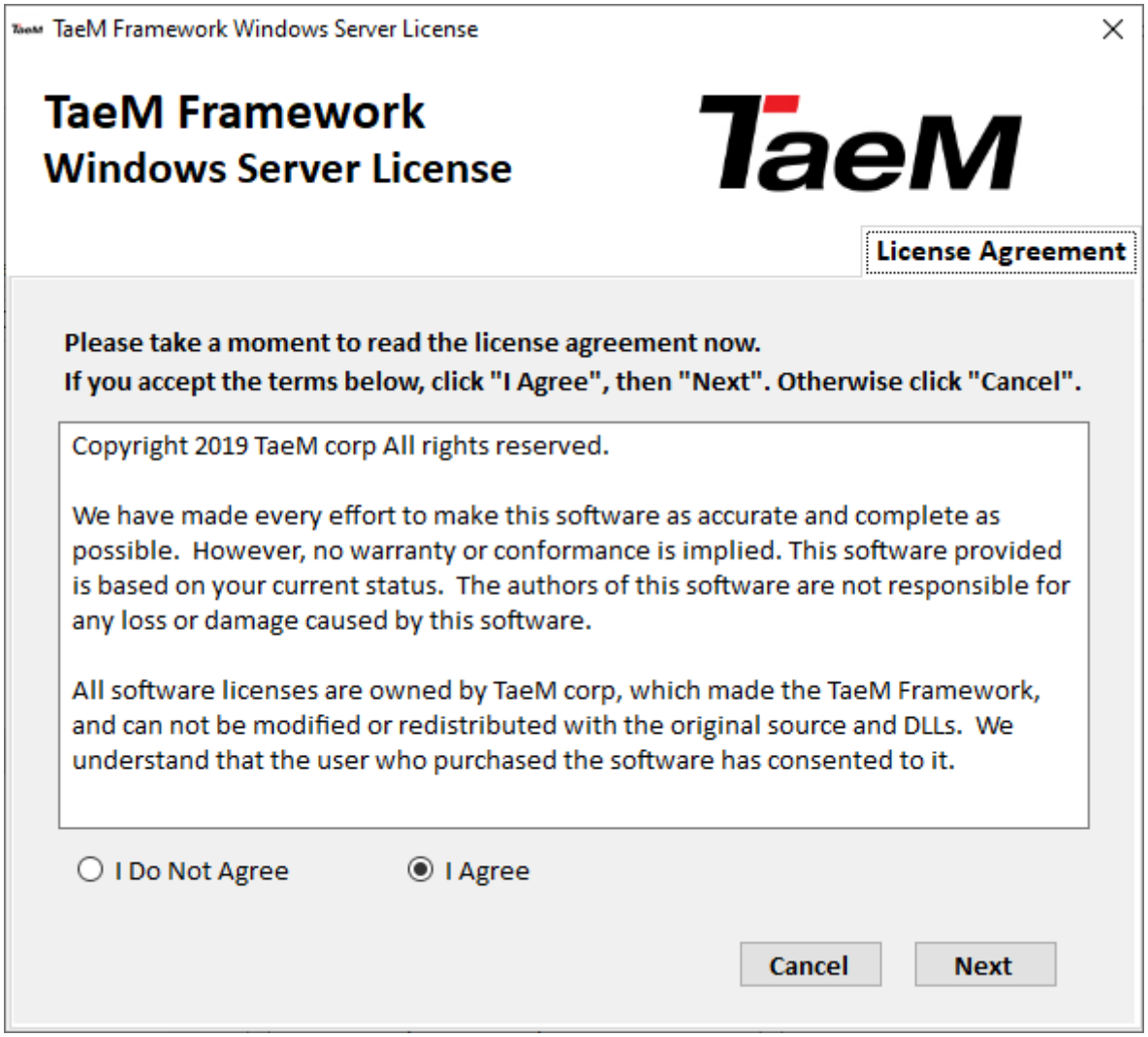
[Figure 15 The user information registration of TaeM Framework Windows Client License]

Now, when your registration is completed normally, a message indicating that Windows Client license registration is normally displayed as shown below. Also, note here that the installed system must be connected to the network, as the license verification process takes place every 60 days after installation to ensure that the licenses are maintained.



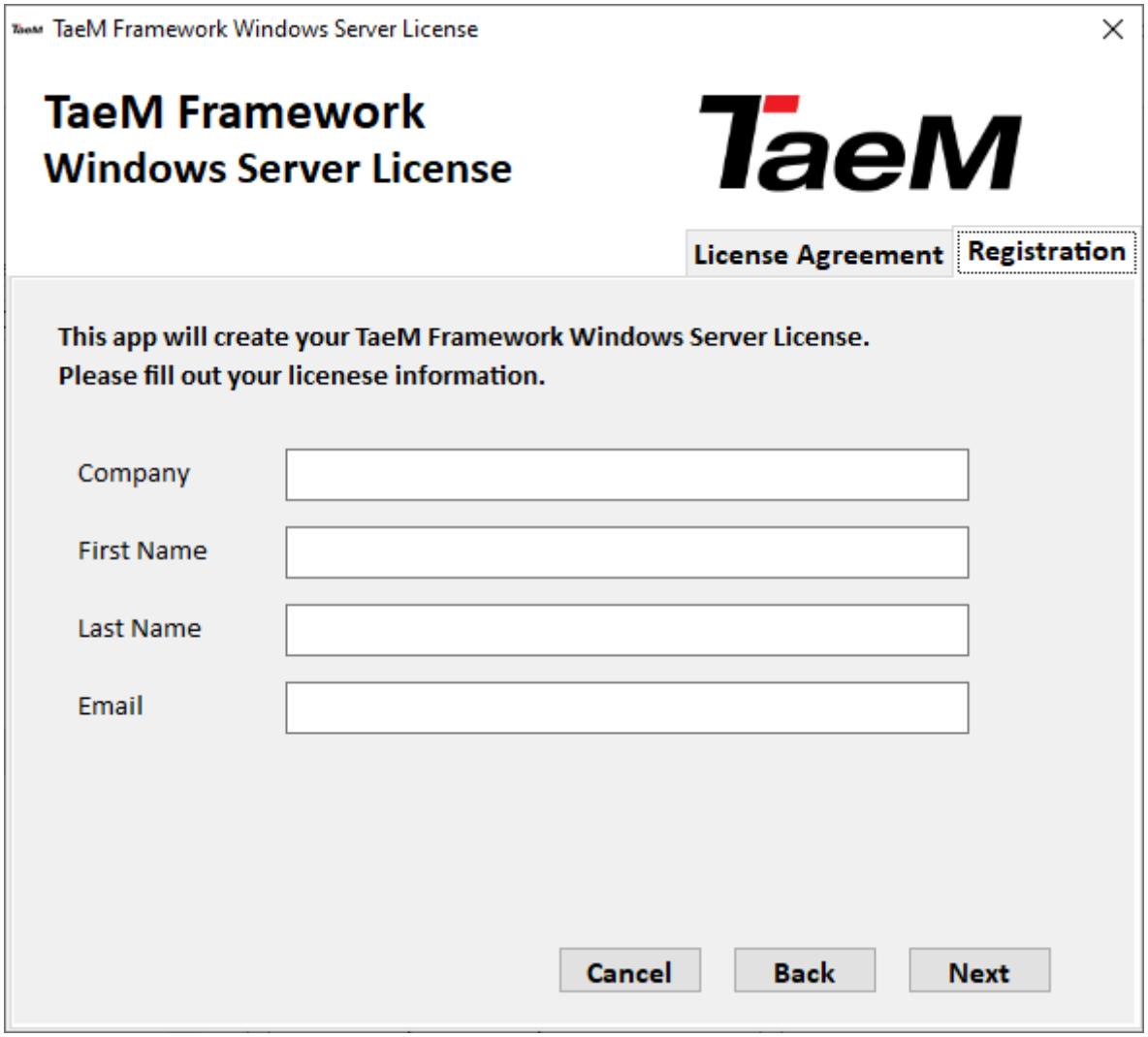
[Figure 16 The license confirmation of TaeM Framework Windows Client License]

To use “TaeM Framework” on Windows Server 2012, 2016, 2019, you need to purchase “TaeM Framework Windows Server License”. Then run the “TaeM Framework Windows Server License” app and the following screen will appear. You must select "I Agree" here and click the "Next" button.



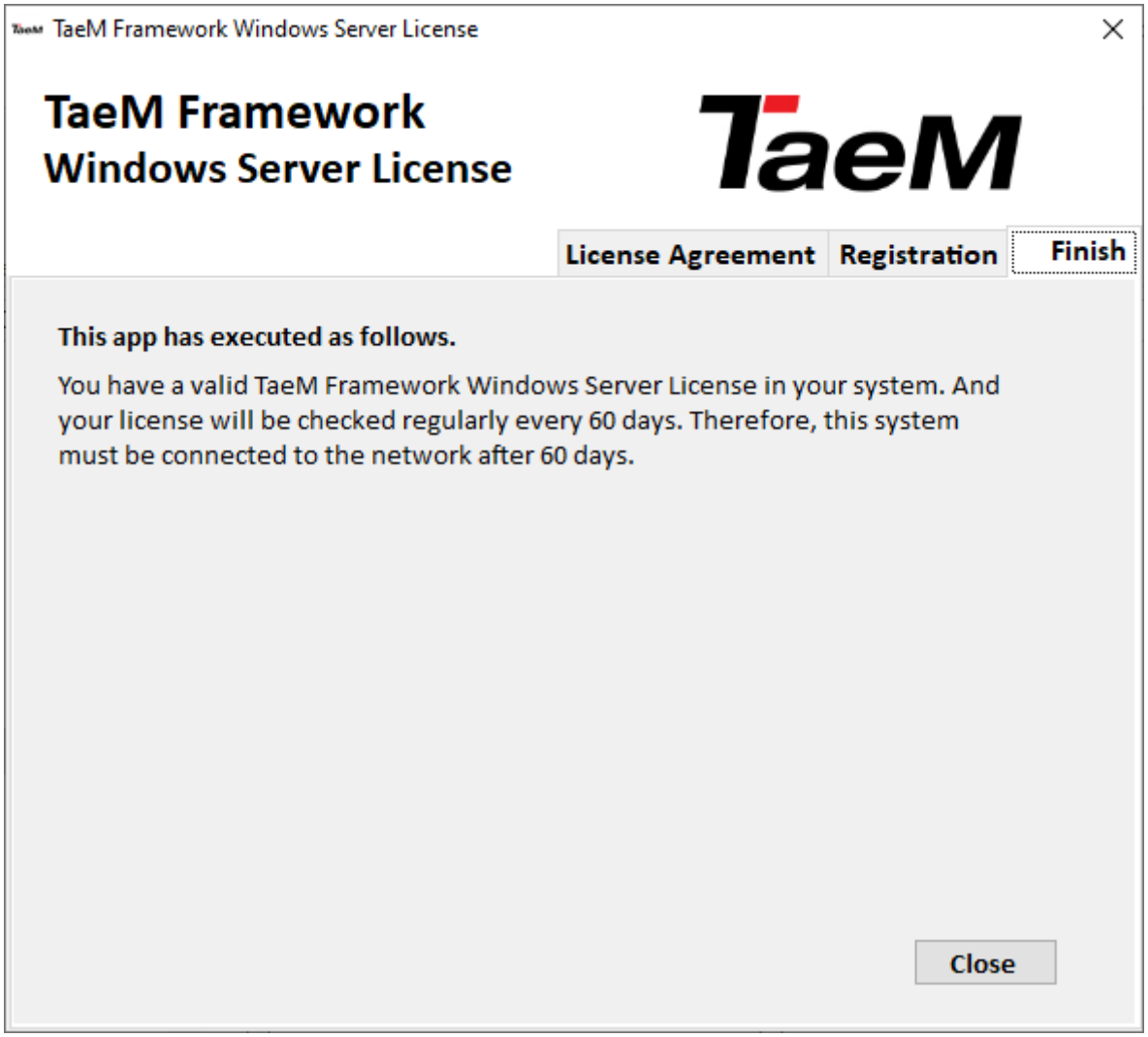
[Figure 17 The license description and its agreement of TaeM Framework Windows Server License]

The figure below shows the user information registration screen. You have to fill out the company name, user first and last name and e-mail, and click the “Next” button.



[Figure 18 The user information registration of TaeM Framework Windows Server License]

Now, when your registration is completed normally, a message indicating that Windows Server license registration is normally displayed as shown below. Also, note here that the installed system must be connected to the network, as the license verification process takes place every 60 days after installation to ensure that the licenses are maintained.



[Figure 19 The license confirmation of TaeM Framework Windows Server License]

TaeM Framework currently supports Windows Client family (Windows 7, 8, 8.1, 10) and Windows Server family (Windows Server 2012, 2016, 2019). In the future, we will support other OS such as Linux, and MacOS and shared VM environments (like AWS, Azure, …).

For reference, shared VM environments such as AWS and Azure can be installed and used for Windows family. However, since hardware information may change, there is a problem that a license malfunction case is confirmed, so we do not guarantee perfect operation.

Let's start using the TaeM Framework directly in the next chapter. In the following chapters, you will look at an example of how to apply the TaeM Framework in MS-SQL Server, Oracle, and MySQL. So it would be better in time for you to look at the options you choose for your own database.

How to use TaeM Framework with MS-SQL Server

Now, we will examine the process of applying TaeM Framework ORM function in MS-SQL Server. Since we have created it like the tutorial format, you can follow the steps in order to easily develop software based on MS-SQL Server by using TaeM Framework ORM function. Note that the below source codes and other samples are made in Microsoft Visual Studio 2017. And you can use other version of Visual Studio because Microsoft Visual Studio supports compatibility to another versions.

Database Table Schema & Entity Class

It's a chicken-and-egg problem. You can create an entity class and create a table of the database later or you can create a table of the database and create entity class later. So, we have created a database table first.

(1) Database Table Schema

We have defined Member and MemberHistory table used in common development. This Member table has member information and This MemberHistory tables keeps records of changes in Member table.

(A) Member table

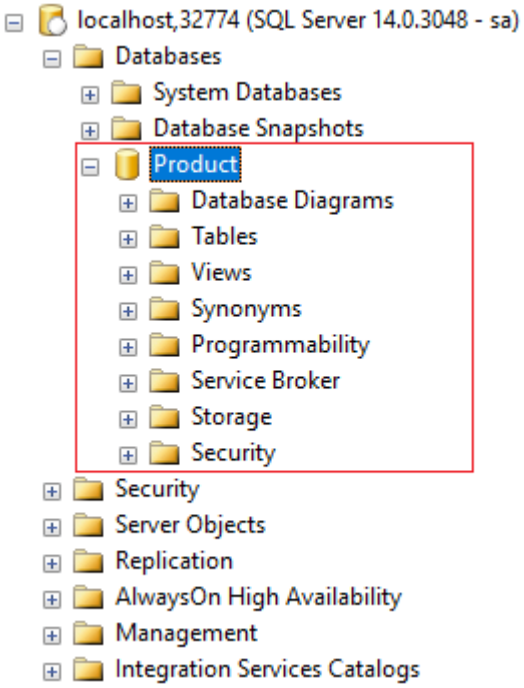
Table Name	Member				
This table has each member information in the system.					
Field Name	**Date type**	**Nullable**	**Default value**	**Primary key**	**Others**
MemberID	INT	NOT NULL		Primary key	IDENTITY(1, 1)
MemberName	NVARCHAR(100)	NULL			
IsAvailable	BIT	NULL			
Email	NVARCHAR(100)	NULL			
PhoneNumber	NVARCHAR(100)	NULL			
Address	NVARCHAR(1024)	NULL			
InsertedDate	DATETIME	NULL			
UpdatedDate	DATETIME	NULL			

(B) MemberHistory table

Table Name	MemberHistory				
This table has member transaction history. It means that when the member table changes in each field the change history is stored in this table.					
Field Name	**Date type**	**Nullable**	**Default value**	**Primary key**	**Others**
Sequence	INT	NOT NULL		Primary key	IDENTITY(1, 1)
MemberID	INT	NOT NULL			
MemberName	NVARCHAR(100)	NULL			
Successful	BIT	NULL	0		
Message	NVARCHAR(1024)	NULL			
InsertedDate	DATETIME	NULL			

(C) Table creation query

In MS-SQL Server, you need to create the Product database as follows.



Then, the following table creation query is executed to create a Member and a MemberHistory table. You can see this table creation query in the sample project.

In TaeMFrameworkwithMSSQL [Database > Table creation > Create_Table_Queries.sql]

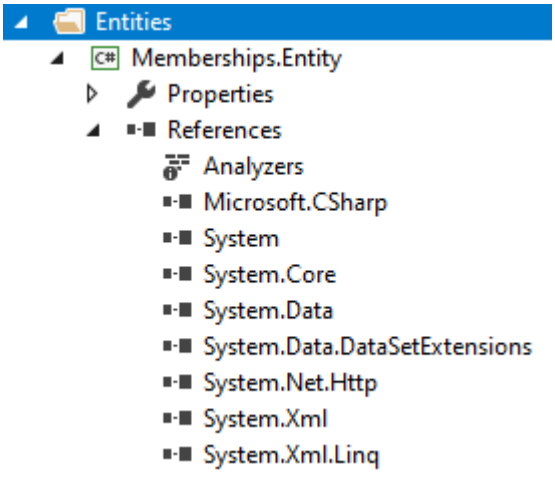
```
use Product;

-- Member Table Creation Query
CREATE TABLE [dbo].[Member]
(
	[MemberID] INT IDENTITY (1, 1) NOT NULL PRIMARY KEY,
	[MemberName] NVARCHAR(100) NULL,
	[IsAvailable] BIT NULL,
	[Email] NVARCHAR(100) NULL,
	[PhoneNumber] NVARCHAR(100) NULL,
	[Address] NVARCHAR(1024) NULL,
	[InsertedDate] DATETIME NULL,
	[UpdatedDate] DATETIME NULL
);

-- Drop table
-- drop table dbo.Member;

-- MemberHistory Table Creation Query
CREATE TABLE [dbo].[MemberHistory]
(
	[Sequence] INT IDENTITY (1, 1) NOT NULL PRIMARY KEY,
	[MemberID] INT NOT NULL,
	[MemberName] NVARCHAR(100) NULL,
	[Successful] BIT NULL DEFAULT 0,
	[Message] NVARCHAR(1024) NULL,
	[InsertedDate] DATETIME NULL
);

-- Drop table
-- drop table.dbo.MemberHistory;
```

(2) Entity Class

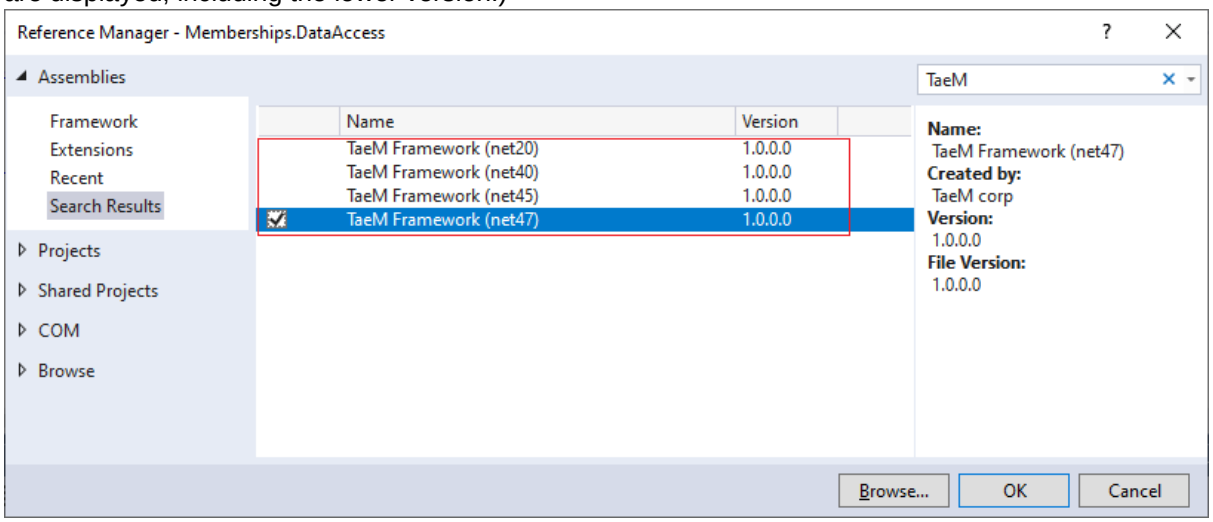
Through the definition of the Member table and the MemberHistory table, each entity classes can be created as follows. In this case, we will map all the fields of the Member table and the MemberHistory table as they are, so we have defined that each classes have same properties with all the fields of the Member table and the MemberHistory table.

(A) Creation of project and addition of TaeM Framework dll as reference

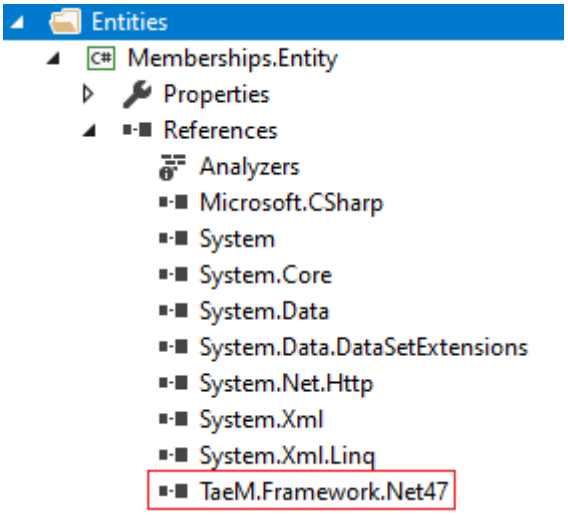
Before you define the entity class, you must create a project to contain the entity class. You can make a project with Visual Studio that you are using. You can create a Memberships.Entity project through Visual Studio's [New> Project]. The figure below shows the project information after creating the Memberships.Entity project.



Then add TaeM.Framework.Net47.dll as a reference to the Memberships.Entity project. Because you installed the TaeM Framework in the previous chapter, here you will see the following "Reference Manager" when you click "Add Reference" in the Memberships.Entity project. When you search "Reference Manager" for "TaeM", you will see four .NET Framework versions of TaeM.Framework.dll. (Note that the target framework for the current project is the .NET Framework 4.7, so all four versions are displayed, including the lower version.)



In this case, select "TaeM Framework (net47)". So, if you add a reference, you should add TaeM.Framework.Net47.dll to the Memberships.Entity project as shown below.



(B) Member entity class

You created a project and added TaeM.Framework.Net47.dll as a reference to the project. Then we will make the Member entity class to the class Memberships.Entity project. In Visual Studio, you can add the Member.cs class file through [New> File].

After you create the Member.cs class file, you need to add the TaeM.Framework.Data.MsSql code using the using statement as follows. TaeM.Framework.Data.MsSql has an attribute class defined as MsSqlDataBinder. This MsSqlDataBinder attribute class automatically binds the result of the DB execution to the field (or variable) or property of the specified class when this attribute is specified for the field (or variable) or property of the class.

```
using System;
using TaeM.Framework.Data.MsSql;
```

Then, we can define the Member entity class as shown below. The Member entity class has constructors and a properties. For each property, MsSqlDataBinder attribute can be specified as follows. The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Entities> Memberships.Entity> Member.cs]

```
using System;
using TaeM.Framework.Data.MsSql;

namespace Memberships.Entity
{
    public class Member
    {
        public Member() : this(string.Empty, false,
            string.Empty, string.Empty, string.Empty)
        {
        }

        public Member(string memberName, bool isAvailable,
            string email, string phoneNumber, string address)
            : this(-1, memberName, isAvailable, email, phoneNumber, address, DateTime.MinValue,
DateTime.MinValue)
        {
        }

        public Member(int memberID, string memberName, bool isAvailable,
            string email, string phoneNumber, string address,
```

```
                DateTime insertedDate, DateTime updatedDate)
        {
            this.MemberID = memberID;
            this.MemberName = memberName;
            this.IsAvailable = isAvailable;

            this.Email = email;
            this.PhoneNumber = phoneNumber;
            this.Address = address;

            this.InsertedDate = insertedDate;
            this.UpdatedDate = updatedDate;
        }

        [MsSqlDataBinder("MemberID")]
        public int MemberID { get; set; }

        [MsSqlDataBinder("MemberName")]
        public string MemberName { get; set; }

        [MsSqlDataBinder("IsAvailable")]
        public bool IsAvailable { get; set; }

        [MsSqlDataBinder("Email")]
        public string Email { get; set; }

        [MsSqlDataBinder("PhoneNumber")]
        public string PhoneNumber { get; set; }

        [MsSqlDataBinder("Address")]
        public string Address { get; set; }

        [MsSqlDataBinder("InsertedDate")]
        public DateTime InsertedDate { get; set; }

        [MsSqlDataBinder("UpdatedDate")]
        public DateTime UpdatedDate { get; set; }
    }
}
```

In other words, an important part of the above Member entity class definition is the MsSqlDataBinder attribute which specifies the field name of the database table (exactly the field name of the query result from the database) and this is specified in each property of the Member entity class for the mapping from each field in the database table. So, MsSqlDataBinder attribute is used to map a table field of MS-SQL Server to each field or property of Member entity class. The field names of the table to be mapped are specified in the MsSqlDataBinder attribute parameter.

How to use MsSqlDataBinder attribute is as follows.

```
MsSqlDataBinder("Field name of the Dababase table to be mapped")
```

In ORM of TaeM Framework, it is not necessary to create separate XML file differently from other ORM solution. Just when declaring entity class, you only need to declare Database data mapping information by declaring attribute. In this case, the MsSqlDataBinder attribute is used as the mapping declaration.

(C) MemberHistory Entity class

This time we define the MemberHistory Entity class. You need to create a MemberHistory class, add a using statement, and assign an MsSqlDataBinder attribute to the Property or Field. The defined MemberHistory.cs class file is as follows.

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Entities> Memberships.Entity> MemberHistory.cs]

```
using System;
using TaeM.Framework.Data.MsSql;

namespace Memberships.Entity
{
    public class MemberHistory
    {
        public MemberHistory()
            : this(-1, string.Empty, false, string.Empty)
        {
        }

        public MemberHistory(int memberID, string memberName, bool isSuccess, string message)
            : this(-1, memberID, memberName, isSuccess, message, DateTime.MinValue)
        {
        }

        public MemberHistory(int seq, int memberID, string memberName,
            bool isSuccess, string message, DateTime insertedDate)
        {
            this.Seq = seq;
            this.MemberID = memberID;
            this.MemberName = memberName;

            this.IsSuccess = isSuccess;
            this.Message = message;
            this.InsertedDate = insertedDate;
        }

        [MsSqlDataBinder("Sequence")]
        public int Seq { get; set; }

        [MsSqlDataBinder("MemberID")]
        public int MemberID { get; set; }

        [MsSqlDataBinder("MemberName")]
        public string MemberName { get; set; }

        [MsSqlDataBinder("Successful")]
        public bool IsSuccess { get; set; }

        [MsSqlDataBinder("Message")]
        public string Message { get; set; }

        [MsSqlDataBinder("InsertedDate")]
        public DateTime InsertedDate { get; set; }
    }
}
```

Data Access

Next, we will defines the class of the Data Access part. The classes in the Data Access part connect to the actual database, execute a query on the database, and bind the resultant value to the entity class. Of course, you might get the number of rows affected by the query type in the database, or you might want to import the first column of the first row.

At First, the functions of MS-SQL Server related helper classes provided by TaeM Framework 1.0.1.0 are as follows.

(1) MsSqlDataHelperFactory class

	Method name	Description
1	SelectSingleEntity<T>()	SelectSingleEntity<T> method executes query and return their single row result with a type of return object, an SQL command type, an SQL query, and SQL parameters.
2	SelectMultipleEntities<T>()	SelectMultipleEntities<T> method executes query and return their multiple rows result with a type of return object, an SQL command type, an SQL query, and SQL parameters.
3	SelectScalar()	SelectScalar method executes query and return the value of first row and first column with an SQL command type, an SQL query, and SQL parameters.
4	Execute()	Execute method executes query and return the number of affected row from the SQL query execution result with an SQL command type, an SQL query, and SQL parameters.

(2) MsSqlParameterHelperFactory class

	Property name	Description
1	ProviderName	ProviderName is a property used to set or get the DB provider name. For MS-SQL Server, this value is usually "System.Data.SqlClient", and "System.Data.dll" is used.

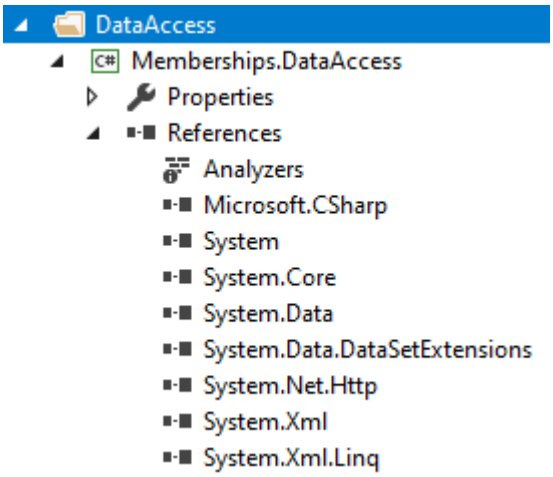
	Method name	Description
1	CreateParameter()	CreateParameter method creates and returns an object of MS-SQL DB Parameter with provider name, parameter name, parameter value, direction of parameter.
2	CreateParameterWOProviderName()	CreateParameter method creates and returns an object of MS-SQL DB Parameter with parameter name, parameter value, the direction of parameter. This should only be used if you have set a value for the ProviderName property.
3	CreateParameter<T>()	CreateParameter method creates and returns an object of MS-SQL DB Parameter with provider name, parameter name, parameter data type, parameter value, direction of parameter.
4	CreateParameterWOProviderName<T>()	CreateParameter method creates and returns an object of MS-SQL DB Parameter with parameter name, parameter value, the direction of parameter. This should only be used if you have set a value for the ProviderName property.

When using MS-SQL Server as a database, you can easily query the MS-SQL Server query by using the above two Helper classes MsSqlDataHelperFactory and MsSqlParameterHelper class and get the

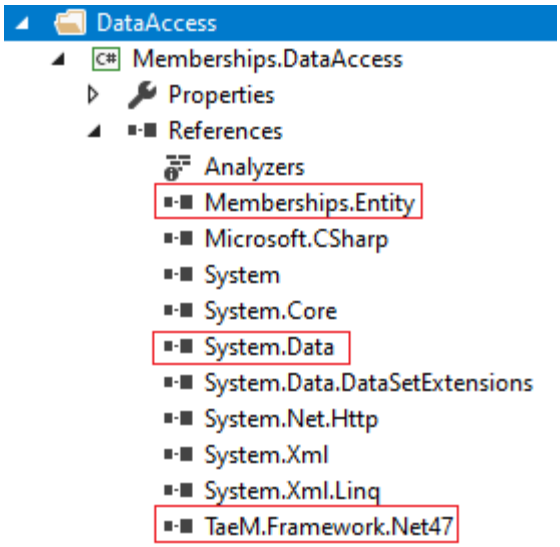
result. Using these two classes, we will code the CRUD ("Create: Create," "Retrieve or Read," "Update: Update," and "Delete or Destroy", A term that refers to a function that a user interface must have.) function with the database.

(3) Making the CRUD

Because it is responsible for Data Access, we will define the class with a name of type DacXXX. In this case, we will define a class name DacMember because it is responsible for the data access of the Member. And we will make a Membership.DataAccess project that is a collection of data access classes, and we will define the DacMember class in this project. We can make the Memberships.DataAccess project as shown below through [New> Project] in Visual Studio. In this case, for the readability and similar property is separated into separate projects, but it is possible to put them together in one project according to the coding rules you want or use.



Since we created a new project, we need to add TaeM.Framework.Net47.dll as a reference to our Memberships.DataAccess project. Of course, if you use the same proejct before, you have already added TaeM.Framework.Net47.dll as a reference. Select "Add Reference" and add TaeM.Framework.Net47.dll in "Reference Manager". The following figure shows TaeM.Framework.Net47.dll and Memberships.Entity.dll added to Memberships.DataAccess project. We also added System.Data.dll, a .NET Data Provider for MS-SQL Server, as a reference. This System.Data.dll is a database driver for MS-SQL Server.



We made Memberships.DataAccess project and added TaeM.Framework.Net47.dll and System.Data.dll to the project as references. Next, we will try adding the DacMember.cs class to the Memberships.DataAccess project. The Addition of the DacMember.cs class file through Visual Studio's [New> File]. After you added the DacMember.cs class file, you need to use the using statement to add the TaeM.Framework.Data code and the Memberships.Entity namespace of the entity class you created before as follows.

TaeM.Framework.Data has the MsSqlDataHelperFactory and MsSqlParameterHelper classes we discussed earlier. Therefore, we will add code using two helper classes in this class. And Memberships.Entity has a Member.cs class which is the entity class that we defined. We also defined the using statement to use System.Data and System.Data.SqlClient to use the .NET Data Provider for MS-SQL Server.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;
```

Then, we can define the DacMember.cs class as shown below. The DacMember.cs class defines a PROVIDER_NAME that specifies the default DB provider, a CONNECTION_STRING that specifies the default MS-SQL Server connection string, a providerName with DB provider information, and a connection field with SqlConnection information, and three constructors and six methods.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMember
    {
        private static readonly string PROVIDER_NAME = "System.Data.SqlClient";

        private static readonly string CONNECTION_STRING
            = "Data Source=localhost,32774;Initial Catalog=Product;Persist Security Info=True;User ID=sa;Password=qwert12345!;Connect Timeout=60";


        private string providerName;
        private SqlConnection connection;


        public DacMember() ...
        public DacMember(string providerName, string connectionString)...
        public DacMember(string providerName, SqlConnection connection)...


        /// <summary> InsertMember method - Insert Member table row from member informat ...
        public Member InsertMember(Member member)...

        /// <summary> SelectMember method - Select Member table row by memberID
        public Member SelectMember(int memberID)...

        /// <summary> SelectMembers method - Select Member table rows by memberName
        public List<Member> SelectMembers(string memberName)...

        /// <summary> SelectNumsOfMembers method - Select number of Member table rows by ...
        public int SelectNumsOfMembers(string memberName)...

        /// <summary> UpdateMember method - Update Member table row by member informatio ...
        public bool UpdateMember(Member member)...

        /// <summary> DeleteMember() method - Delete Member table row by memberID
        public bool DeleteMember(int memberID)...
    }
}
```

The above six methods need to execute the following kind of query on the Member table and return the result value.

	Query type	Method name
1	Retrieve a single data row	SelectMember(…)
2	Retrieve multiple data row	SelectMembers(…)
3	Creation of the data row	InsertMember(…)
4	Modification of the data row	UpdateMember(…)
5	Deletion of the data row	DeleteMember(…)
6	Retrieve the number of data row	SelectNumsOfMembers(…)

To implement the method according to the above query type, you can use the following method defined in the MsSqlDataHelperFactory class mentioned earlier in this section.

	Query type	Method name	The Method of MsSqlDataHelperFactory class
1	Retrieve a single data row	SelectMember(…)	Using the SelectSingleEntity <T> () method
2	Retrieve multiple data row	SelectMembers(…)	Using the SelectMultipleEntities <T> () method
3	Creation of the data row	InsertMember(…)	Using the Execute () method or the SelectSingleEntity <T> () method
4	Modification of the data row	UpdateMember(…)	Using the Execute () method
5	Deletion of the data row	DeleteMember(…)	Using the Execute () method
6	Retrieve the number of data row	SelectNumsOfMembers(…)	Using the SelectScalar () method

Particularly here is the InsertMember (...) method, you can use both the Execute () method and the SelectSingleEngity <T> () method. For the InsertMember (...) method, you must use the Execute () method to execute a query that only generates a row in the Member table. But, if you want to create a row in the Member table and get the row information of the generated table, you need to use the SelectSingleEngity <T> () method to get the data row type as the result.

The MsSqlParameterHelper class is a class that generalizes the parameters of a database and has one property and four methods.

So, we can make the DacMember.cs class which implements CRUD using MsSqlDataHelperFactory and MsSqlParameterHelper class as follows.

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [DataAccess> Memberships.DataAccess> DacMember.cs]

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMember
    {
        private static readonly string PROVIDER_NAME = "System.Data.SqlClient";

        private static readonly string CONNECTION_STRING
            = "Data Source=localhost,32774;Initial Catalog=Product;Persist Security Info=True;User
ID=sa;Password=qwert12345!;Connect Timeout=60";
```

```
        private string providerName;
        private SqlConnection connection;


        public DacMember() : this(PROVIDER_NAME, CONNECTION_STRING)
        {
        }
        public DacMember(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connection = new SqlConnection(connectionString);
        }
        public DacMember(string providerName, SqlConnection connection)
        {
            this.providerName = providerName;
            this.connection = connection;
        }


        /// <summary>
        /// InsertMember method
        /// - Insert Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member InsertMember(Member member)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    Member ret =
(Member)MsSqlDataHelperFactory.SelectSingleEntity<Member>(connection,
                                typeof(Member),
                                CommandType.Text,
                                @"INSERT INTO Product.dbo.Member " +
                                @"( MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate,
UpdatedDate ) " +
                                @"VALUES " +
                                @"( @MemberName, @IsAvailable, @Email, @PhoneNumber, @Address,
GETDATE(), NULL ) " +
                                @" " +
                                @"SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber,
Address, InsertedDate, UpdatedDate " +
                                @"FROM Product.dbo.Member " +
                                @"WHERE MemberName = @MemberName AND IsAvailable = @IsAvailable
AND Email = @Email " +
                                @"    AND PhoneNumber = @PhoneNumber AND Address = @Address " +
                                @"ORDER BY InsertedDate DESC ",
                                MsSqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", member.MemberName, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@IsAvailable",
member.IsAvailable, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@Email",
member.Email, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName,
"@PhoneNumber", member.PhoneNumber, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@Address",
member.Address, ParameterDirection.Input)
                                );

                    return ret;
```

```
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// SelectMember method
        /// - Select Member table row by memberID
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public Member SelectMember(int memberID)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    Member ret =
(Member)MsSqlDataHelperFactory.SelectSingleEntity<Member>(connection,
                        typeof(Member),
                        CommandType.Text,
                        @"SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber,
Address, InsertedDate, UpdatedDate " +
                        @"FROM Product.dbo.Member " +
                        @"WHERE MemberID = @MemberID ",
                        MsSqlParameterHelperFactory.CreateParameter<SqlDbType>(providerName,
"@MemberID", memberID, SqlDbType.Int, ParameterDirection.Input)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// SelectMembers method
        /// - Select Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public List<Member> SelectMembers(string memberName)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    List<Member> ret =
(List<Member>)MsSqlDataHelperFactory.SelectMultipleEntities<Member>(connection,
                        typeof(Member),
                        CommandType.Text,
                        @"SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber,
Address, InsertedDate, UpdatedDate " +
                        @"FROM Product.dbo.Member " +
                        @"WHERE MemberName like '%' + @MemberName + '%' ",
                        MsSqlParameterHelperFactory.CreateParameter(providerName,
```

```
"@MemberName", memberName, ParameterDirection.Input)
                                );

                        return ret;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// SelectNumsOfMembers method
            /// - Select number of Member table rows by memberName
            /// </summary>
            /// <param name="memberName"></param>
            /// <returns></returns>
            public int SelectNumsOfMembers(string memberName)
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        int ret = (int)MsSqlDataHelperFactory.SelectScalar(connection,
                            CommandType.Text,
                            @"SELECT COUNT(*) " +
                            @"FROM Product.dbo.Member " +
                            @"WHERE MemberName like '%' + @MemberName + '%' ",
                            MsSqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", memberName, ParameterDirection.Input)
                            );

                        return ret;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// UpdateMember method
            /// - Update Member table row by member information
            /// </summary>
            /// <param name="member">Member information</param>
            /// <returns></returns>
            public bool UpdateMember(Member member)
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        int ret = MsSqlDataHelperFactory.Execute(connection,
                            CommandType.Text,
                            @"UPDATE Product.dbo.Member " +
                            @"SET MemberName = @MemberName, IsAvailable = @IsAvailable, Email =
@Email, " +
                            @"    PhoneNumber = @PhoneNumber, Address = @Address " +
                            @"WHERE MemberID = @MemberID ",
                            MsSqlParameterHelperFactory.CreateParameter(providerName,
```

```
"@MemberName", member.MemberName, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@IsAvailable",
member.IsAvailable, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@Email",
member.Email, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName,
"@PhoneNumber", member.PhoneNumber, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@Address",
member.Address, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@MemberID",
member.MemberID, ParameterDirection.Input)
                                );

                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// DeleteMember() method
            /// - Delete Member table row by memberID
            /// </summary>
            /// <param name="memberID">Member ID</param>
            /// <returns></returns>
            public bool DeleteMember(int memberID)
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        int ret = MsSqlDataHelperFactory.Execute(connection,
                            CommandType.Text,
                            @"DELETE FROM Product.dbo.Member " +
                            @"WHERE MemberID = @MemberID ",
                            MsSqlParameterHelperFactory.CreateParameter<SqlDbType>(providerName,
"@MemberID", memberID, SqlDbType.Int, ParameterDirection.Input)
                            );

                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
        }
}
```

Here you can define the DacMemberHistory class. We have created the DacMemberHistory class, added the using statement, and defined methods in the DacMemberHistory.cs class as shown below.

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [DataAccess > Memberships.DataAccess > DacMemberHistory.cs]

```
using System;
using System.Collections.Generic;
```

```
using System.Data;
using System.Data.SqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
	public class DacMemberHistory
	{
		private static readonly string PROVIDER_NAME = "System.Data.SqlClient";

		private static readonly string CONNECTION_STRING
			= "Data Source=localhost,32774;Initial Catalog=Product;Persist Security Info=True;User
ID=sa;Password=qwert12345!;Connect Timeout=60";


		private string providerName;
		private SqlConnection connection;


		public DacMemberHistory() : this(PROVIDER_NAME, CONNECTION_STRING)
		{
		}
		public DacMemberHistory(string providerName, string connectionString)
		{
			this.providerName = providerName;
			this.connection = new SqlConnection(connectionString);
		}
		public DacMemberHistory(string providerName, SqlConnection connection)
		{
			this.providerName = providerName;
			this.connection = connection;
		}


		/// <summary>
		/// InsertMemberHistory method
		/// - Insert MemberHistory table row from member history information
		/// </summary>
		/// <param name="member">Member history information</param>
		/// <returns></returns>
		public MemberHistory InsertMemberHistory(MemberHistory memberHistory)
		{
			try
			{
				using (connection)
				{
					connection.Open();

					return
(MemberHistory)MsSqlDataHelperFactory.SelectSingleEntity<MemberHistory>(connection,
						typeof(MemberHistory),
						CommandType.Text,
						@"INSERT INTO Product.dbo.MemberHistory " +
						@"( MemberID, MemberName, Successful, Message, InsertedDate ) " +
						@"VALUES " +
						@"( @MemberID, @MemberName, @Successful, @Message, GETDATE() ) " +
						@" " +
						@"SELECT Sequence, MemberID, MemberName, Successful, Message,
InsertedDate " +
						@"FROM Product.dbo.MemberHistory " +
						@"WHERE MemberID = @MemberID AND MemberName = @MemberName " +
						@"    AND Successful = @Successful AND Message = @Message " +
						@"ORDER BY InsertedDate DESC ",
						MsSqlParameterHelperFactory.CreateParameter<SqlDbType>(providerName,
```

```
"@MemberID", memberHistory.MemberID, SqlDbType.Int, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", memberHistory.MemberName, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@Successful",
memberHistory.IsSuccess, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@Message",
memberHistory.Message, ParameterDirection.Input)
                                );
                        }
                    }
                    catch (Exception ex)
                    {
                        throw ex;
                    }
                }

                /// <summary>
                /// SelectMemberHistories() method
                /// - Select MemberHistory table row by fromDate and toDate
                /// </summary>
                /// <param name="fromDate">From date</param>
                /// <param name="toDate">To date</param>
                /// <returns></returns>
                public List<MemberHistory> SelectMemberHistories(DateTime fromDate, DateTime toDate)
                {
                    try
                    {
                        using (connection)
                        {
                            connection.Open();

                            return
(List<MemberHistory>)MsSqlDataHelperFactory.SelectMultipleEntities<MemberHistory>(connection,
                                typeof(MemberHistory),
                                CommandType.Text,
                                @"SELECT Sequence, MemberID, MemberName, Successful, Message,
InsertedDate " +
                                @"FROM Product.dbo.MemberHistory " +
                                @"WHERE InsertedDate >= @FromDate AND InsertedDate <= @ToDate " +
                                @"ORDER BY InsertedDate DESC ",
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@FromDate",
fromDate, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@ToDate",
toDate, ParameterDirection.Input)
                                );
                        }
                    }
                    catch (Exception ex)
                    {
                        throw ex;
                    }
                }
            }
        }
```

(4) Using stored procedure

In the DacMember.cs and DacMemberHistory.cs class definitions above, we created a pure query in text format to execute the query directly to MS-SQL Server. However, you can use the stored procedure as shown below for performance.

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [DataAccess > Memberships.DataAccess > DacMemberSP.cs]

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMemberSP
    {
        private static readonly string PROVIDER_NAME = "System.Data.SqlClient";

        private static readonly string CONNECTION_STRING
            = "Data Source=localhost,32774;Initial Catalog=Product;Persist Security Info=True;User
ID=sa;Password=qwert12345!;Connect Timeout=60";


        private string providerName;
        private SqlConnection connection;


        public DacMemberSP() : this(PROVIDER_NAME, CONNECTION_STRING)
        {
        }
        public DacMemberSP(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connection = new SqlConnection(connectionString);
        }
        public DacMemberSP(string providerName, SqlConnection connection)
        {
            this.providerName = providerName;
            this.connection = connection;
        }


        /// <summary>
        /// InsertMember method
        /// - InsertMember Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member InsertMember(Member member)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    Member ret =
(Member)MsSqlDataHelperFactory.SelectSingleEntity<Member>(connection,
                        typeof(Member),
                        CommandType.StoredProcedure,
                        "Product.dbo.sp_Memberships_Insert_Member",
                        MsSqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", member.MemberName, ParameterDirection.Input),
                        MsSqlParameterHelperFactory.CreateParameter(providerName, "@IsAvailable",
member.IsAvailable, ParameterDirection.Input),
                        MsSqlParameterHelperFactory.CreateParameter(providerName, "@Email",
member.Email, ParameterDirection.Input),
                        MsSqlParameterHelperFactory.CreateParameter(providerName,
"@PhoneNumber", member.PhoneNumber, ParameterDirection.Input),
                        MsSqlParameterHelperFactory.CreateParameter(providerName, "@Address",
```

```
member.Address, ParameterDirection.Input)
                                    );

                                return ret;
                            }
                        }
                        catch (Exception ex)
                        {
                            throw ex;
                        }
                    }

                    /// <summary>
                    /// SelectMember method
                    /// - Select Member table row by memberID
                    /// </summary>
                    /// <param name="memberID">Member ID</param>
                    /// <returns></returns>
                    public Member SelectMember(int memberID)
                    {
                        try
                        {
                            using (connection)
                            {
                                connection.Open();

                                Member ret =
(Member)MsSqlDataHelperFactory.SelectSingleEntity<Member>(connection,
                                    typeof(Member),
                                    CommandType.StoredProcedure,
                                    "Product.dbo.sp_Memberships_Select_Member_By_MemberID",
                                    MsSqlParameterHelperFactory.CreateParameter<SqlDbType>(providerName,
"@MemberID", memberID, SqlDbType.Int, ParameterDirection.Input)
                                    );

                                return ret;
                            }
                        }
                        catch (Exception ex)
                        {
                            throw ex;
                        }
                    }

                    /// <summary>
                    /// SelectMembers method
                    /// - Select Member table rows by memberName
                    /// </summary>
                    /// <param name="memberName">Member name</param>
                    /// <returns></returns>
                    public List<Member> SelectMembers(string memberName)
                    {
                        try
                        {
                            using (connection)
                            {
                                connection.Open();

                                List<Member> ret =
(List<Member>)MsSqlDataHelperFactory.SelectMultipleEntities<Member>(connection,
                                    typeof(Member),
                                    CommandType.StoredProcedure,
                                    "Product.dbo.sp_Memberships_Select_Members_By_MemberName",
                                    MsSqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", memberName, ParameterDirection.Input)
                                    );
```

```
                return ret;
            }
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    /// <summary>
    /// SelectNumsOfMembers method
    /// - Select number of Member table rows by memberName
    /// </summary>
    /// <param name="memberName"></param>
    /// <returns></returns>
    public int SelectNumsOfMembers(string memberName)
    {
        try
        {
            using (connection)
            {
                connection.Open();

                int ret = (int)MsSqlDataHelperFactory.SelectScalar(connection,
                    CommandType.StoredProcedure,
                    "Product.dbo.sp_Memberships_Select_Num_Of_Members_By_MemberName",
                    MsSqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", memberName, ParameterDirection.Input)
                    );

                return ret;
            }
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    /// <summary>
    /// UpdateMember method
    /// - Update Member table row by member information
    /// </summary>
    /// <param name="member">Member information</param>
    /// <returns></returns>
    public bool UpdateMember(Member member)
    {
        try
        {
            using (connection)
            {
                connection.Open();

                int ret = MsSqlDataHelperFactory.Execute(connection,
                    CommandType.StoredProcedure,
                    "Product.dbo.sp_Memberships_Update_Member",
                    MsSqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", member.MemberName, ParameterDirection.Input),
                    MsSqlParameterHelperFactory.CreateParameter(providerName, "@IsAvailable",
member.IsAvailable, ParameterDirection.Input),
                    MsSqlParameterHelperFactory.CreateParameter(providerName, "@Email",
member.Email, ParameterDirection.Input),
                    MsSqlParameterHelperFactory.CreateParameter(providerName,
"@PhoneNumber", member.PhoneNumber, ParameterDirection.Input),
                    MsSqlParameterHelperFactory.CreateParameter(providerName, "@Address",
```

```
member.Address, ParameterDirection.Input),
                                MsSqlParameterHelperFactory.CreateParameter(providerName, "@MemberID",
member.MemberID, ParameterDirection.Input)
                                );

                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// DeleteMember() method
            /// - Delete Member table row by memberID
            /// </summary>
            /// <param name="memberID">Member ID</param>
            /// <returns></returns>
            public bool DeleteMember(int memberID)
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        int ret = MsSqlDataHelperFactory.Execute(connection,
                            CommandType.StoredProcedure,
                            "Product.dbo.sp_Memberships_Delete_Member_By_MemberID",
                            MsSqlParameterHelperFactory.CreateParameter<SqlDbType>(providerName,
"@MemberID", memberID, SqlDbType.Int, ParameterDirection.Input)
                            );

                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
        }
}
```

The stored procedures used in DacMemberSP.cs are as follows.

- sp_Memberships_Insert_Member.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Database > Stored procedure > Member > sp_Memberships_Insert_Member.sql]

```
CREATE PROCEDURE [dbo].[sp_Memberships_Insert_Member]
            @MemberName nvarchar(100),
            @IsAvailable bit,
            @Email nvarchar(100),
            @PhoneNumber nvarchar(100),
            @Address nvarchar(1024)
AS
```

```
	SET NOCOUNT ON;
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

	BEGIN TRANSACTION

		INSERT INTO dbo.Member
		( MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate, UpdatedDate )
		VALUES
		( @MemberName, @IsAvailable, @Email, @PhoneNumber, @Address, GETDATE(),
NULL )

		SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber, Address,
InsertedDate, UpdatedDate
		FROM dbo.Member
		WHERE MemberName = @MemberName AND IsAvailable = @IsAvailable AND Email =
@Email
			AND PhoneNumber = @PhoneNumber AND Address = @Address
		ORDER BY InsertedDate DESC

	COMMIT

RETURN
```

- sp_Memberships_Select_Member_By_MemberID.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Database > Stored procedure > Member > sp_Memberships_Select_Member_By_MemberID.sql]

```
CREATE PROCEDURE [dbo].[sp_Memberships_Select_Member_By_MemberID]
	@MemberID int
AS
	SET NOCOUNT ON;
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

	SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate,
UpdatedDate
	FROM dbo.Member
	WHERE MemberID = @MemberID

RETURN
```

- sp_Memberships_Select_Members_By_MemberName.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Database > Stored procedure > Member > sp_Memberships_Select_Members_By_MemberName.sql]

```
CREATE PROCEDURE [dbo].[sp_Memberships_Select_Members_By_MemberName]
	@MemberName nvarchar(100)
AS
	SET NOCOUNT ON;
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

	SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate,
UpdatedDate
	FROM dbo.Member
	WHERE MemberName like '%' + @MemberName + '%'

RETURN
```

- sp_Memberships_Select_Num_Of_Members_By_MemberName.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Database > Stored procedure > Member > sp_Memberships_Select_Num_Of_Members_By_MemberName.sql]

```
CREATE PROCEDURE [dbo].[sp_Memberships_Select_Num_Of_Members_By_MemberName]
	@MemberName nvarchar(100)
AS
	SET NOCOUNT ON;
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

	SELECT COUNT(*)
	FROM dbo.Member
	WHERE MemberName like '%' + @MemberName + '%'

RETURN
```

- sp_Memberships_Update_Member.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Database > Stored procedure > Member > sp_Memberships_Update_Member.sql]

```
CREATE PROCEDURE [dbo].[sp_Memberships_Update_Member]
	@MemberID int,
	@MemberName nvarchar(100),
	@IsAvailable bit,
	@Email nvarchar(100),
	@PhoneNumber nvarchar(100),
	@Address nvarchar(1024)
AS
	SET NOCOUNT OFF;
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

	UPDATE dbo.Member
	SET MemberName = @MemberName, IsAvailable = @IsAvailable, Email = @Email,
		PhoneNumber = @PhoneNumber, Address = @Address, UpdatedDate = GETDATE()
	WHERE MemberID = @MemberID

RETURN
```

- sp_Memberships_Delete_Member_By_MemberID.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Database > Stored procedure > Member > sp_Memberships_Delete_Member_By_MemberID.sql]

```
CREATE PROCEDURE [dbo].[sp_Memberships_Delete_Member_By_MemberID]
	@MemberID int
AS
	SET NOCOUNT OFF;
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

	DELETE FROM dbo.Member
	WHERE MemberID = @MemberID

RETURN
```

There is one thing to note when using the above stored procedure. sp_Memberships_Update_Member.sql and sp_Memberships_Delete_Member_By_MemberID.sql are set to SET NOCOUNT OFF when defining a stored procedure and the remaining stored procedures are defined as SET NOCOUNT ON. The difference is that there is a different way that Transaction-SQL statements such as Insert, Update, and Delete return the number of affected rows as part of the result value. Therefore, if the stored procedure is set to SET NOCOUNT ON, the Execute method of the MsSqlDataHelperFactory class can not get the number of affected rows in the Transaction-SQL statement because the number of rows affected by the Transaction-SQL statement is not returned. So you normally run a stored procedure, but the number of affected rows is always -1.

DacMemberHistory.cs has also changed the DacMemberHistorySP.cs file to use the stored procedure as follows.

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [DataAccess > Memberships.DataAccess > DacMemberHistorySP.cs]

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMemberHistorySP
    {
        private static readonly string PROVIDER_NAME = "System.Data.SqlClient";

        private static readonly string CONNECTION_STRING
            = "Data Source=localhost,32774;Initial Catalog=Product;Persist Security Info=True;User
ID=sa;Password=qwert12345!;Connect Timeout=60";


        private string providerName;
        private SqlConnection connection;


        public DacMemberHistorySP() : this(PROVIDER_NAME, CONNECTION_STRING)
        {
        }
        public DacMemberHistorySP(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connection = new SqlConnection(connectionString);
        }
        public DacMemberHistorySP(string providerName, SqlConnection connection)
        {
            this.providerName = providerName;
            this.connection = connection;
        }


        /// <summary>
        /// InsertMemberHistory method
        /// - Insert MemberHistory table row from member history information
        /// </summary>
        /// <param name="member">Member history information</param>
        /// <returns></returns>
        public MemberHistory InsertMemberHistory(MemberHistory memberHistory)
        {
```

```
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        return
(MemberHistory)MsSqlDataHelperFactory.SelectSingleEntity<MemberHistory>(connection,
                            typeof(MemberHistory),
                            CommandType.StoredProcedure,
                            "Product.dbo.sp_Memberships_Insert_MemberHistory",
                            MsSqlParameterHelperFactory.CreateParameter(providerName, "@MemberID",
memberHistory.MemberID, ParameterDirection.Input),
                            MsSqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", memberHistory.MemberName, ParameterDirection.Input),
                            MsSqlParameterHelperFactory.CreateParameter(providerName, "@Successful",
memberHistory.IsSuccess, ParameterDirection.Input),
                            MsSqlParameterHelperFactory.CreateParameter(providerName, "@Message",
memberHistory.Message, ParameterDirection.Input)
                            );
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// SelectMemberHistories() method
            /// - Select MemberHistory table row by fromDate and toDate
            /// </summary>
            /// <param name="fromDate">From date</param>
            /// <param name="toDate">To date</param>
            /// <returns></returns>
            public List<MemberHistory> GetMemberHistories(DateTime fromDate, DateTime toDate)
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        return
(List<MemberHistory>)MsSqlDataHelperFactory.SelectMultipleEntities<MemberHistory>(connection,
                            typeof(MemberHistory),
                            CommandType.StoredProcedure,
                            "Product.dbo.sp_Memberships_Select_MemberHistory_By_FromToDate",
                            MsSqlParameterHelperFactory.CreateParameter(providerName, "@FromDate",
fromDate, ParameterDirection.Input),
                            MsSqlParameterHelperFactory.CreateParameter(providerName, "@ToDate",
toDate, ParameterDirection.Input)
                            );
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
        }
}
```

The stored procedure in DacMemberHistorySP.cs above is as follows.

- sp_Memberships_Insert_MemberHistory.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Database > Stored procedure > Member > sp_Memberships_Insert_MemberHistory.sql]

```
CREATE PROCEDURE [dbo].[sp_Memberships_Insert_MemberHistory]
	@MemberID int,
	@MemberName nvarchar(100),
	@Successful bit,
	@Message nvarchar(1024)
AS
	SET NOCOUNT ON;
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

	BEGIN TRANSACTION

		INSERT INTO dbo.MemberHistory
		( MemberID, MemberName, Successful, Message, InsertedDate )
		VALUES
		( @MemberID, @MemberName, @Successful, @Message, GETDATE() )

		SELECT Sequence, MemberID, MemberName, Successful, Message, InsertedDate
		FROM dbo.MemberHistory
		WHERE MemberID = @MemberID AND MemberName = @MemberName
			AND Successful = @Successful AND Message = @Message
		ORDER BY InsertedDate DESC

	COMMIT

RETURN
```

- sp_Memberships_Select_MemberHistory_By_FromToDate.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Database > Stored procedure > Member > sp_Memberships_Select_MemberHistory_By_FromToDate.sql]

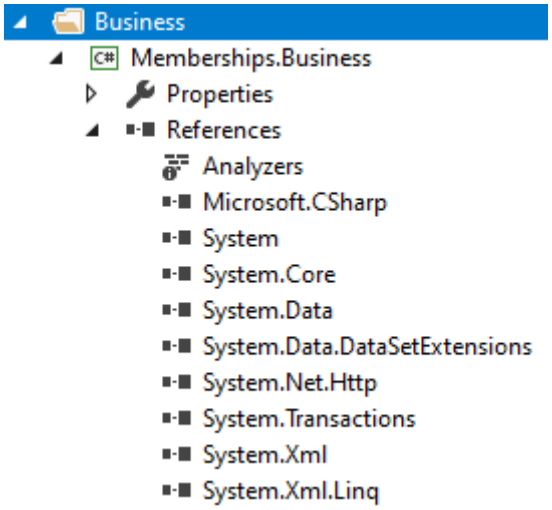
```
CREATE PROCEDURE [dbo].[sp_Memberships_Select_MemberHistory_By_FromToDate]
	@FromDate datetime,
	@ToDate datetime
AS
	SET NOCOUNT ON;
	SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

	SELECT Sequence, MemberID, MemberName, Successful, Message, InsertedDate
	FROM dbo.MemberHistory
	WHERE InsertedDate >= @FromDate AND InsertedDate <= @ToDate
	ORDER BY InsertedDate DESC

RETURN
```

We know that it is better to use the stored procedure in the performance. However, if the stored procedure is used, the DB query part is not included in the source code that we manage, and in case of the actual stored procedure, it needs to be stored in the database server beforehand. This is so uncomfortable. In this case, you can select the method you need. Of course, if you develop a pure query and there is a performance issue, switching to a stored procedure can be a good choice for your development. So we can always choose a step-by-step approach because we have to finish our work at a set time.

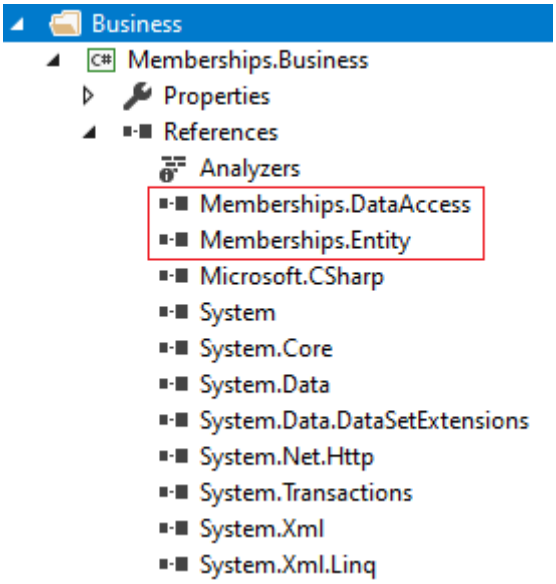
Transaction & Business

There are many ways to handle transactions in the database. This can be handled either on the DB side, in the .NET code side, or in the COM transaction. Transaction processing on the DB side has seen the handling of transactions in the stored procedure as in the "sp_Memberships_Insert_Member.sql" file or the "sp_Memberships_Insert_MemberHistory.sql" file, and here is how to deal with transactions using TransactionScope in .NET code and you can see as the following code.

We are going to create a new project to separate the transaction and business processing parts from other code. So, we can create the Memberships.Business project through [New> Project] in Visual Studio as below.



We have added a reference to the Memberships.Entity and Memberships.DataAccess project we created earlier, as shown below. Of course, if you have defined Entity classes or Data Access classes in one project, you do not need to add references.



And we will create the BizMemberShip.cs class file and added the code below.

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Business > Memberships.Business > BizMembership.cs]

```
using System;
using System.Collections.Generic;
using System.Transactions;

using Memberships.Entity;
using Memberships.DataAccess;

namespace Memberships.Business
{
    public class BizMemberShip
    {
        private string providerName;
        private string connectionString;


        public BizMemberShip() : this(string.Empty, string.Empty)
        {
        }
        public BizMemberShip(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connectionString = connectionString;
        }


        /// <summary>
        /// CreateMember method
        /// - Create Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member CreateMember(Member member)
        {
            Member newMember = null;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    newMember = new DacMember(providerName,
connectionString).InsertMember(member);

                    if (newMember != null)
                    {
                        // Success
                        new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(newMember.MemberID, newMember.MemberName,
true,
                                string.Format("Create new member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    newMember.MemberID, newMember.MemberName,
newMember.IsAvailable,
                                    newMember.Email, newMember.PhoneNumber,
newMember.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                    member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );
```

```
                    }
                }
                catch (Exception ex)
                {
                    // Fail
                    new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                        new MemberHistory(member.MemberID, member.MemberName, false,
                            string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                member.MemberName, member.IsAvailable,
                                member.Email, member.PhoneNumber, member.Address)
                        )
                    );

                    throw ex;
                }
                finally
                {
                    scope.Complete();
                }
            }

            return newMember;
        }

        /// <summary>
        /// GetMember method
        /// - Get Member table row by memberID
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public Member GetMember(int memberID)
        {
            Member ret = null;

            try
            {
                ret = new DacMember(providerName, connectionString).SelectMember(memberID);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        /// <summary>
        /// GetMembers method
        /// - Get Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public List<Member> GetMembers(string memberName)
        {
            List<Member> ret = null;

            try
            {
                ret = new DacMember(providerName, connectionString).SelectMembers(memberName);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
```

```
        }

        /// <summary>
        /// GetNumsOfMembers method
        /// - Get number of Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public int GetNumsOfMembers(string memberName)
        {
            int ret = -1;

            try
            {
                ret = new DacMember(providerName,
connectionString).SelectNumsOfMembers(memberName);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        /// <summary>
        /// SetMember method
        /// - Set Member table row by member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public bool SetMember(Member member)
        {
            bool? ret;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    ret = new DacMember(providerName, connectionString).UpdateMember(member);

                    if (ret != null)
                    {
                        // Success
                        new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, true,
                                string.Format("Update member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    member.MemberID, member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    member.MemberID, member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );
                    }
                }
                catch (Exception ex)
                {
```

```
                        // Fail
                        new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    member.MemberID, member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );

                        throw ex;
                    }
                    finally
                    {
                        scope.Complete();
                    }

                    return (ret == true) ? true : false;
                }
            }

            /// <summary>
            /// RemoveMember() method
            /// - Remove Member table row by memberID
            /// </summary>
            /// <param name="memberID">Member ID</param>
            /// <returns></returns>
            public bool RemoveMember(int memberID)
            {
                bool? ret;
                Member removedMember = null;

                using (TransactionScope scope = new TransactionScope())
                {
                    try
                    {
                        removedMember = new DacMember(providerName,
connectionString).SelectMember(memberID);

                        ret = new DacMember(providerName, connectionString).DeleteMember(memberID);

                        if (ret != null && ret == true)
                        {
                            // Success
                            new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                                new MemberHistory(removedMember.MemberID,
removedMember.MemberName, true,
                                    string.Format("Remove member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                        removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                                )
                            );
                        }
                        else
                        {
                            // Fail
                            new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                                new MemberHistory(removedMember.MemberID,
removedMember.MemberName, false,
                                    string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                        removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                                )
```

```
                                );
                            }
                        }
                        catch (Exception ex)
                        {
                            // Fail
                            new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                                new MemberHistory(removedMember.MemberID, removedMember.MemberName,
false,
                                    string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                        removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                                )
                            );

                            throw ex;
                        }
                        finally
                        {
                            scope.Complete();
                        }

                        return (ret == true) ? true : false;
                    }
                }
            }
}
```

In BizMemberShip.cs, we call pure query data access classes and BizMemberShipSP.cs which calls stored procedure as below.

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Business > Memberships.Business > BizMembershipSP.cs]

```
using System;
using System.Collections.Generic;
using System.Transactions;

using Memberships.Entity;
using Memberships.DataAccess;

namespace Memberships.Business
{
    public class BizMemberShipSP
    {
        private string providerName;
        private string connectionString;


        public BizMemberShipSP() : this(string.Empty, string.Empty)
        {
        }
        public BizMemberShipSP(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connectionString = connectionString;
        }


        /// <summary>
        /// CreateMember method
```

```
        /// - Create Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member CreateMember(Member member)
        {
            Member newMember = null;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    newMember = new DacMemberSP(providerName,
connectionString).InsertMember(member);

                    if (newMember != null)
                    {
                        // Success
                        new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(newMember.MemberID, newMember.MemberName,
true,
                                string.Format("Create new member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    newMember.MemberID, newMember.MemberName,
newMember.IsAvailable,
                                    newMember.Email, newMember.PhoneNumber,
newMember.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                    member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );
                    }
                }
                catch (Exception ex)
                {
                    // Fail
                    new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                        new MemberHistory(member.MemberID, member.MemberName, false,
                            string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                member.MemberName, member.IsAvailable,
                                member.Email, member.PhoneNumber, member.Address)
                        )
                    );

                    throw ex;
                }
                finally
                {
                    scope.Complete();
                }
            }

            return newMember;
        }

        /// <summary>
        /// GetMember method
        /// - Get Member table row by memberID
```

```
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public Member GetMember(int memberID)
        {
            Member ret = null;

            try
            {
                ret = new DacMemberSP(providerName, connectionString).SelectMember(memberID);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        /// <summary>
        /// GetMembers method
        /// - Get Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public List<Member> GetMembers(string memberName)
        {
            List<Member> ret = null;

            try
            {
                ret = new DacMemberSP(providerName, connectionString).SelectMembers(memberName);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        /// <summary>
        /// GetNumsOfMembers method
        /// - Get number of Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public int GetNumsOfMembers(string memberName)
        {
            int ret = -1;

            try
            {
                ret = new DacMemberSP(providerName,
connectionString).SelectNumsOfMembers(memberName);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        /// <summary>
        /// SetMember method
```

```
        /// - Set Member table row by member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public bool SetMember(Member member)
        {
            bool? ret;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    ret = new DacMemberSP(providerName, connectionString).UpdateMember(member);

                    if (ret != null)
                    {
                        // Success
                        new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, true,
                                string.Format("Update member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    member.MemberID, member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    member.MemberID, member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );
                    }
                }
                catch (Exception ex)
                {
                    // Fail
                    new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                        new MemberHistory(member.MemberID, member.MemberName, false,
                            string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                member.MemberID, member.MemberName, member.IsAvailable,
                                member.Email, member.PhoneNumber, member.Address)
                        )
                    );

                    throw ex;
                }
                finally
                {
                    scope.Complete();
                }

                return (ret == true) ? true : false;
            }
        }

        /// <summary>
        /// RemoveMember() method
        /// - Remove Member table row by memberID
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public bool RemoveMember(int memberID)
```

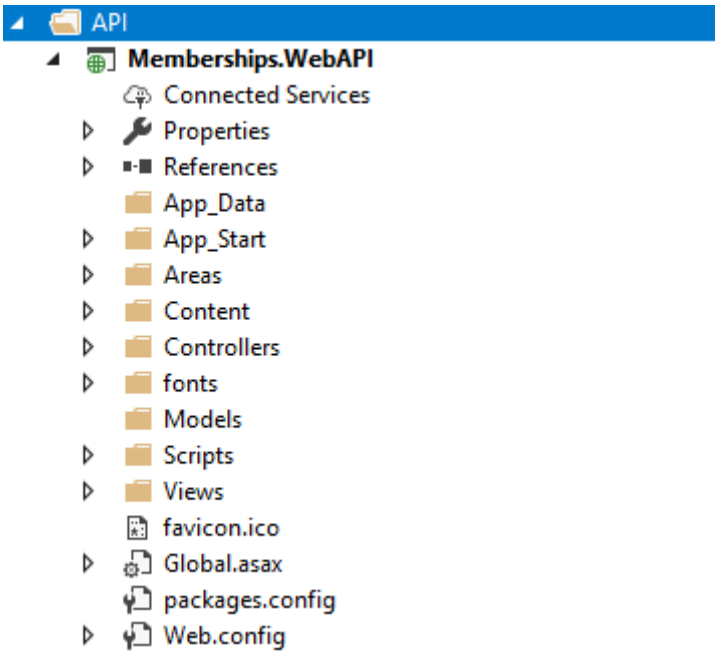
```
        {
            bool? ret;
            Member removedMember = null;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    removedMember = new DacMemberSP(providerName,
connectionString).SelectMember(memberID);

                    ret = new DacMemberSP(providerName, connectionString).DeleteMember(memberID);

                    if (ret != null && ret == true)
                    {
                        // Success
                        new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(removedMember.MemberID,
removedMember.MemberName, true,
                                string.Format("Remove member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                    removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(removedMember.MemberID,
removedMember.MemberName, false,
                                string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                    removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                            )
                        );
                    }
                }
                catch (Exception ex)
                {
                    // Fail
                    new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                        new MemberHistory(removedMember.MemberID, removedMember.MemberName,
false,
                            string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                        )
                    );

                    throw ex;
                }
                finally
                {
                    scope.Complete();
                }

                return (ret == true) ? true : false;
            }
        }
```

```
    }
}
```

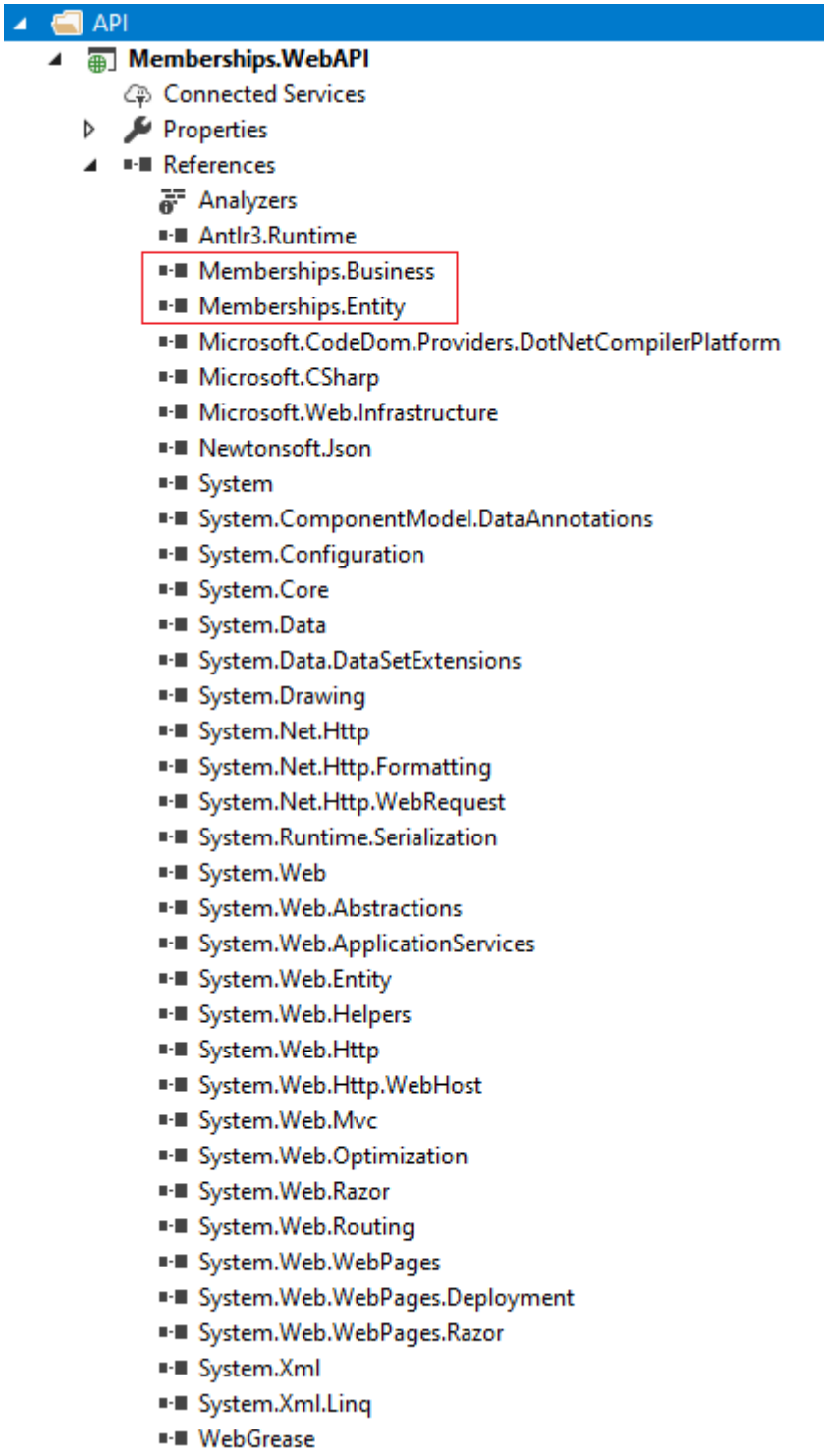
API

We will make a web service API by directly utilizing Entity classes, Data Access classes and Business classes defined above. This is similar to the real web service API associated with membership information.

We will create an ASP.NET WebAPI project and check the actual IIS services. We can create a Memberships.WebAPI ASP.NET Web Application project through Visual Studio's [New> Project]. Note that when you create a project, you must create a project by selecting the Web API project template, and in the addition folder and core reference, you need to select the MVC and Web API. This will create a Memberships.WebAPI project as shown below.



You need to add Memberships.Business as references to the Memberships.WebAPI project. Of course, you can use Memberships.DataAccess as a reference instead of Memberships.Business. We will use Memberships.Business instead of Memberships.DataAccess because the transaction was implemented in Memberships.Business.



Then, you need to open the Web.config file and add the connection string to connect MS-SQL Server as follows. Note that the default connection string is defined in the DacMember.cs or DacMemberHistory.cs, DacMemberSP.cs, and DacMemberHistorySP.cs files of the DataAccess class, but we will define and use the DB connection string that is commonly used by the WebAPI service.

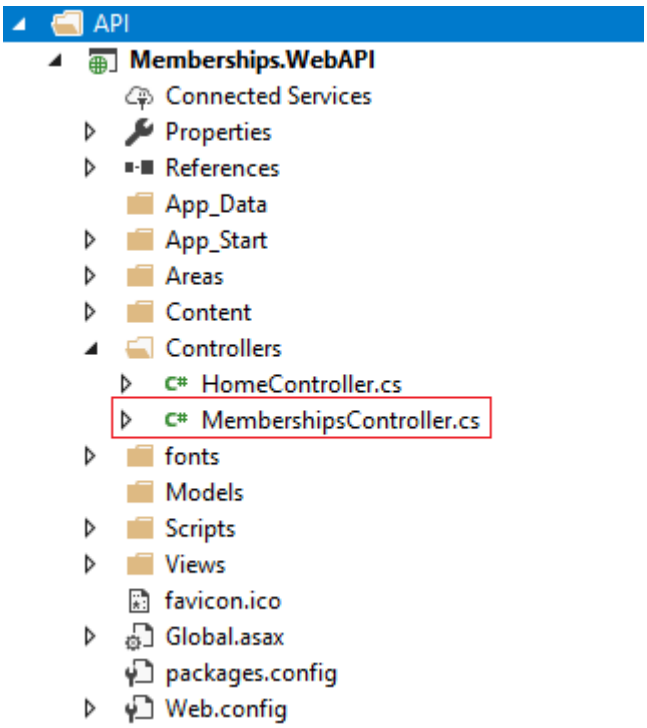
The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [API> Memberships.API> Web.config]

```
<configuration>
  <connectionStrings>
```

```
    <add name="MSSQLDB"
         connectionString="Data Source=localhost,32774;Initial Catalog=Product;Persist Security
Info=True;User ID=sa;Password=qwert12345!;Connect Timeout=60"
         providerName="System.Data.SqlClient" />
  </connectionStrings>
  …
  …
</configuration>
```

Next, you need to add the MembershipsController.cs file to the Controllers folder of the Memberships.WebAPI project. As you can see, you can create an API just by implementing a Controller in a WebAPI project.



Then you need to add the following code to the MembershipsController.cs.

The code below can be found in the sample project.

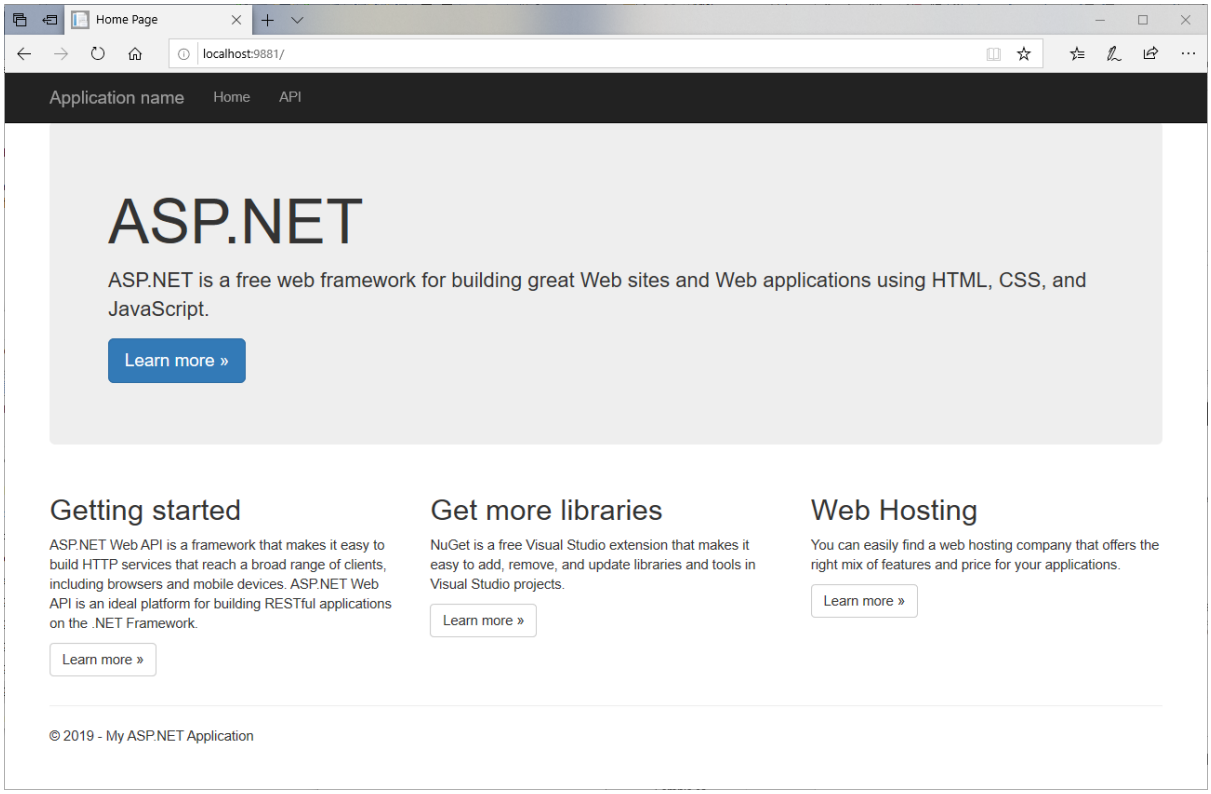
In TaeMFrameworkwithMSSQL [API> Memberships.WebAPI> Controllers> MembershipsController.cs]

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;

using Memberships.Business;
using Memberships.Entity;

namespace Memberships.WebAPI.Controllers
{
    public class MembershipsController : ApiController
    {
        private static string MS_SQL_PROVIDER_NAME =
ConfigurationManager.ConnectionStrings["MSSQLDB"].ProviderName;
```

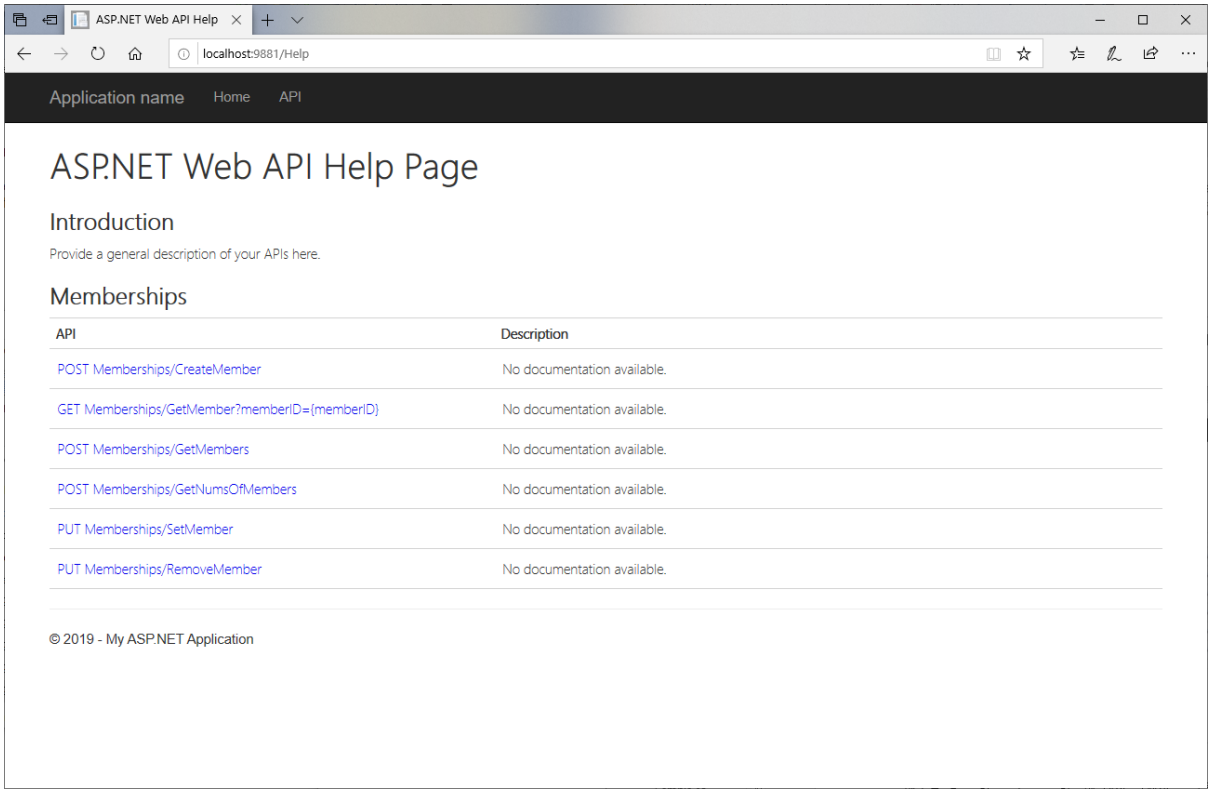
```
        private static string MS_SQL_CONN_STR =
ConfigurationManager.ConnectionStrings["MSSQLDB"].ConnectionString;


        [HttpPost]
        [Route("Memberships/CreateMember")]
        public Member CreateMember(Member member)
        {
            Member ret = null;

            try
            {
                ret = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).CreateMember(member);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        [HttpGet]
        [Route("Memberships/GetMember")]
        public Member GetMember(int memberID)
        {
            Member ret = null;

            try
            {
                ret = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMember(memberID);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        [HttpPost]
        [Route("Memberships/GetMembers")]
        public List<Member> GetMembers([FromBody]string memberName)
        {
            List<Member> ret = null;

            try
            {
                ret = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMembers(memberName);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        [HttpPost]
        [Route("Memberships/GetNumsOfMembers")]
        public int GetNumsOfMembers([FromBody]string memberName)
        {
            int ret = -1;
```

```
                try
                {
                    ret = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetNumsOfMembers(memberName);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            [HttpPut]
            [Route("Memberships/SetMember")]
            public bool SetMember(Member member)
            {
                bool ret = false;

                try
                {
                    ret = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).SetMember(member);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            [HttpPut]
            [Route("Memberships/RemoveMember")]
            public bool RemoveMember([FromBody]int memberID)
            {
                bool ret = false;

                try
                {
                    ret = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).RemoveMember(memberID);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }
        }
}
```

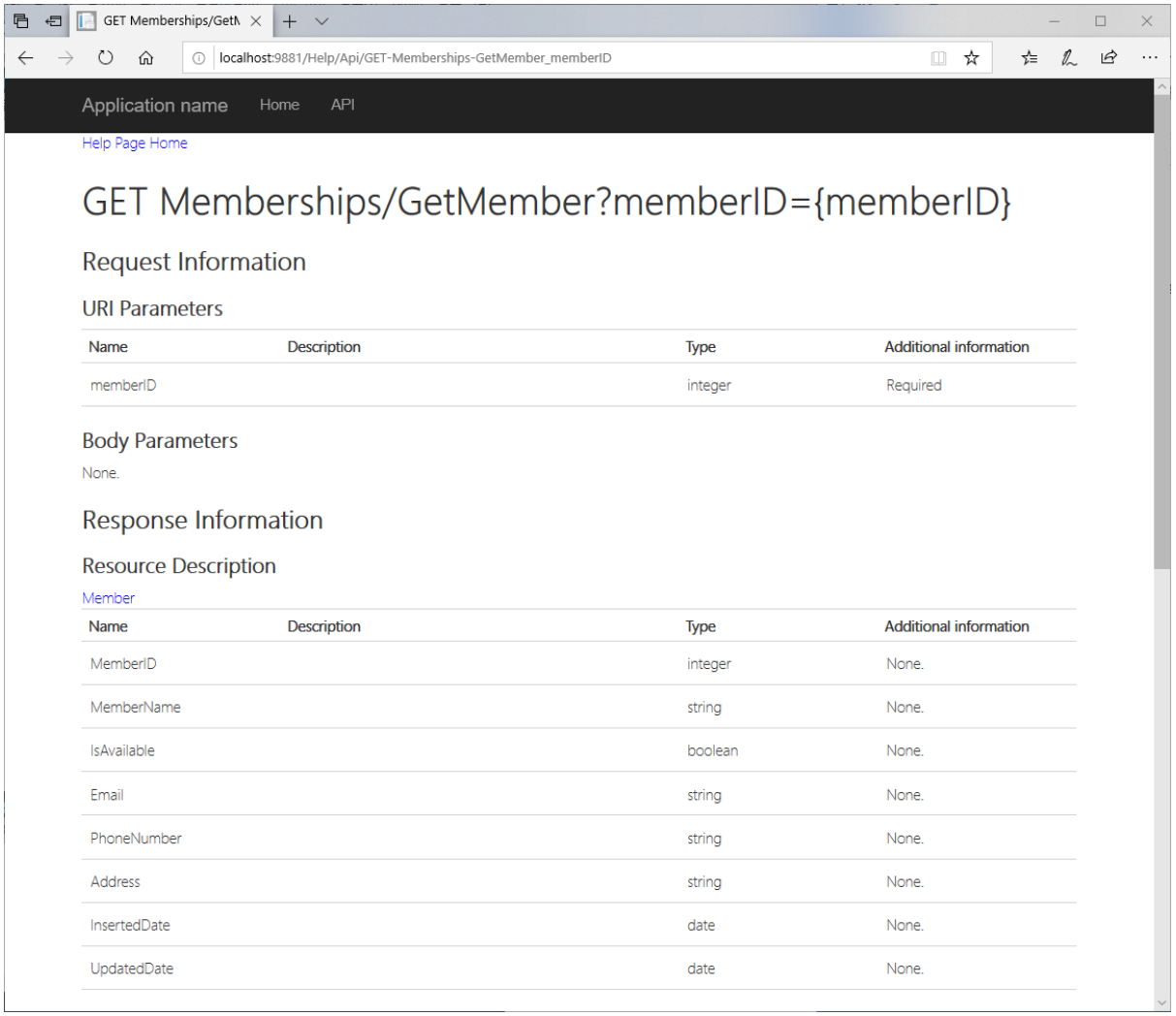
After adding the Controller, build and run the Memberships.WebAPI project. Execution of the Memberships.WebAPI project can be performed on IIS Express, which is the Visual Studio's default WebAPI project execution environment, or on a separate IIS after setting the WebAPI in IIS. The first screen you have run is shown below.

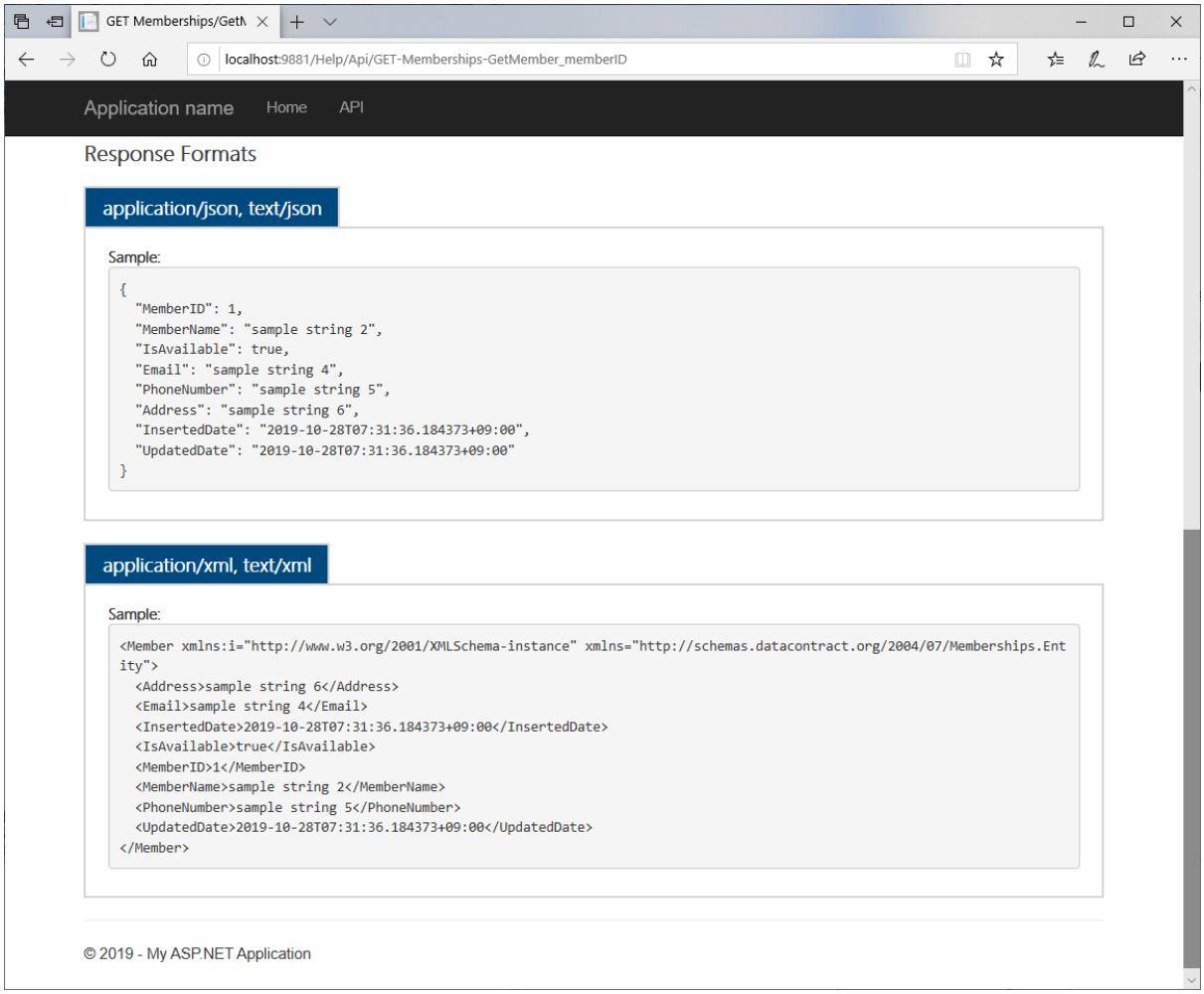


When this screen is displayed, click the "API" item. So you will see a list of the individual methods of the API implemented in the Memberships Controller, as shown below.



Here we will try to click on the GetMember method as GET format. So you can get a detailed description of the GetMember method request format and response format.

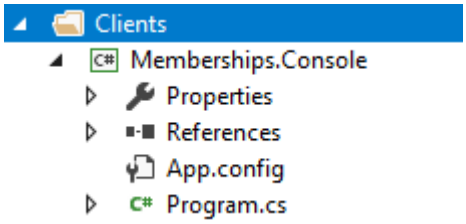

GET Memberships/GetM
localhost:9881/Help/Api/GET-Memberships-GetMember_memberID
Application name
Home
API
Help Page Home
GET Memberships/GetMember?memberID={memberID}
Request Information
URI Parameters
Name
Description
Type
Additional information
memberID
integer
Required
Body Parameters
None.
Response Information
Resource Description
Member
Name
Description
Type
Additional information
MemberID
integer
None.
MemberName
string
None.
IsAvailable
boolean
None.
Email
string
None.
PhoneNumber
string
None.
Address
string
None.
InsertedDate
date
None.
UpdatedDate
date
None.


GET Memberships/GetM
localhost:9881/Help/Api/GET-Memberships-GetMember_memberID
Application name
Home
API
Response Formats
application/json, text/json
Sample:
{
"MemberID": 1,
"MemberName": "sample string 2",
"IsAvailable": true,
"Email": "sample string 4",
"PhoneNumber": "sample string 5",
"Address": "sample string 6",
"InsertedDate": "2019-10-28T07:31:36.184373+09:00",
"UpdatedDate": "2019-10-28T07:31:36.184373+09:00"
}
application/xml, text/xml
Sample:
<Member xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/Memberships.Entity">
<Address>sample string 6</Address>
<Email>sample string 4</Email>
<InsertedDate>2019-10-28T07:31:36.184373+09:00</InsertedDate>
<IsAvailable>true</IsAvailable>
<MemberID>1</MemberID>
<MemberName>sample string 2</MemberName>
<PhoneNumber>sample string 5</PhoneNumber>
<UpdatedDate>2019-10-28T07:31:36.184373+09:00</UpdatedDate>
</Member>
© 2019 - My ASP.NET Application

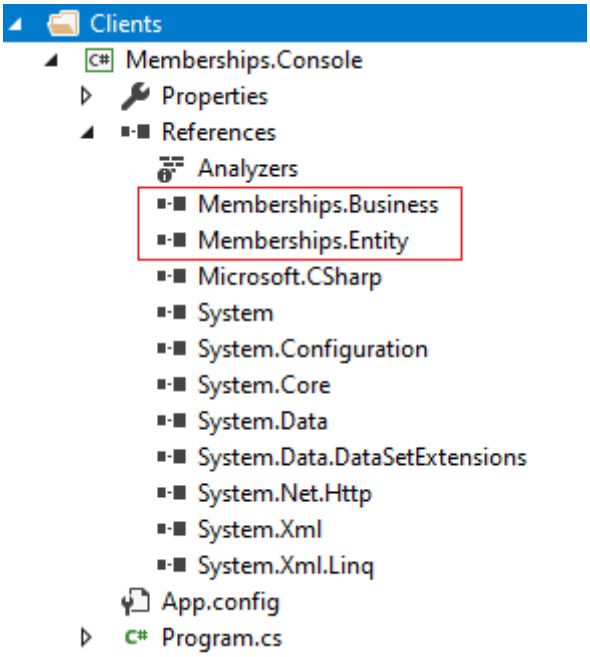
Test client

We made the design of database table, Entity class, Data Access classes, Business classes, and WebAPI. So we will create a client console application to check that the classes we created are working properly.

You can create a Memberships.Console project in Console Application (.NET Framework) project type through [New> Project] in Visual Studio. This will be created the Memberships.Console project as shown below.



You need to add Memberships.Entity, Memberships.Business as references in the Memberships.Console project as follows.



You need to add the DB connection string and WebAPI connection information to the App.config file as follows.

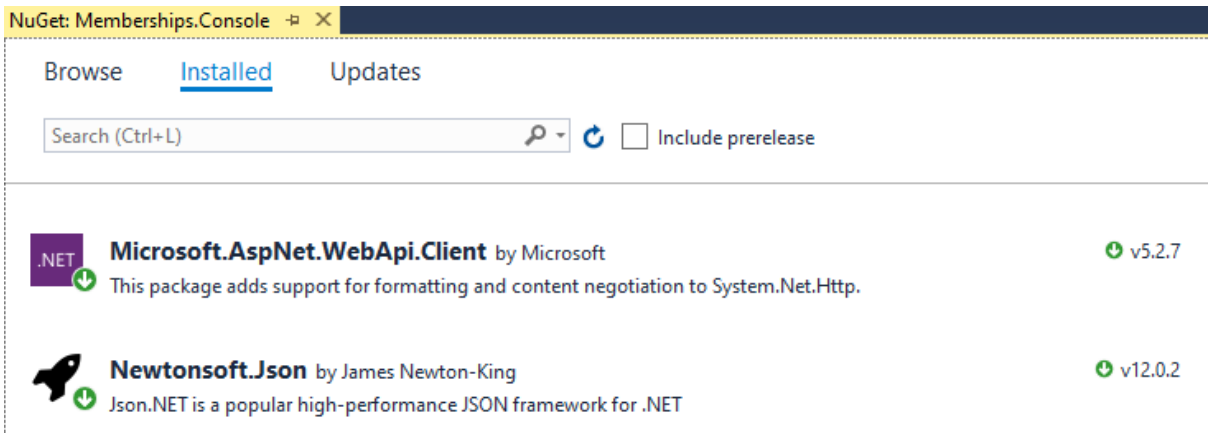
The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Clients> Memberships.Console> App.config]

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7" />
  </startup>
  <connectionStrings>
    <add name="MSSQLDB"
         connectionString="Data Source=localhost,32774;Initial Catalog=Product;Persist Security
Info=True;User ID=sa;Password=qwert12345!;Connect Timeout=60"
         providerName="System.Data.SqlClient" />
  </connectionStrings>
```

```
  <appSettings>
    <!-- VS IIS Express Uri -->
    <add key="WebAPI_IIS_Express_Uri" value="http://localhost:9881/"/>
    <!-- End -->

    <!-- IIS Uri -->
    <add key="WebAPI_IIS" value="http://localhost/Memberships.WebAPI.9881/"/>
    <!-- End -->

    <add key="HeaderType" value="application/json" />
  </appSettings>
</configuration>
```

And you need to add the following WebAPICaller.cs file to be used when calling to the WebAPI client. The WebAPICaller.cs uses several extension methods to facilitate WebAPI calls. So you need to install two NuGet packages: Microsoft.AspNet.WebApi.Client and Newtonsoft.Json.



We installed two NuGet packages and defined WebAPICaller.cs as shown below.

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Clients> Memberships.Console> WebAPICaller.cs]

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Net.Http.Formatting;
using System.Threading;

using Memberships.Entity;

namespace Memberships.Console
{
    public class WebAPICaller
    {
        public static Member CallCreateMember(string uri, Member member)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PostAsJsonAsync(
                "Memberships/CreateMember",
                member).Result;
```

```
            if (response.IsSuccessStatusCode)
            {
                Member ret = response.Content.ReadAsAsync<Member>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call CreateMember API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return null;
        }

        public static Member CallGetMember(string uri, int memberID)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.GetAsync(
                string.Format("Memberships/GetMember?memberID={0}", memberID)
                ).Result;

            if (response.IsSuccessStatusCode)
            {
                Member ret = response.Content.ReadAsAsync<Member>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call GetMember API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return null;
        }

        public static List<Member> CallGetMembers(string uri, string memberName)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PostAsJsonAsync(
                "Memberships/GetMembers",
                memberName).Result;

            if (response.IsSuccessStatusCode)
            {
                List<Member> ret = response.Content.ReadAsAsync<List<Member>>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call GetMembers API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return null;
        }

        public static int CallGetGetNumsOfMembers(string uri, string memberName)
        {
```

```
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PostAsJsonAsync(
                "Memberships/GetNumsOfMembers",
                memberName).Result;

            if (response.IsSuccessStatusCode)
            {
                int ret = response.Content.ReadAsAsync<int>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call GetNumsOfMembers API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return -1;
        }

        public static bool CallSetMember(string uri, Member member)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PutAsJsonAsync(
                "Memberships/SetMember",
                member).Result;

            if (response.IsSuccessStatusCode)
            {
                bool ret = response.Content.ReadAsAsync<bool>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call SetMember API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return false;
        }

        public static bool CallRemoveMember(string uri, int memberID)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PutAsJsonAsync(
                "Memberships/RemoveMember",
                memberID).Result;

            if (response.IsSuccessStatusCode)
            {
                bool ret = response.Content.ReadAsAsync<bool>().Result;
                return ret;
            }
            else
            {
```

```
                    System.Console.WriteLine("{0} : {1} ({2})", "Call SetMember API method",
(int)response.StatusCode, response.ReasonPhrase);
                }

                return false;
            }
        }
}
```

Finally, you need to add the following code to the Program.cs to put the code that calls the Business class directly and the WebAPI directly.

The code below can be found in the sample project.

In TaeMFrameworkwithMSSQL [Clients > Memberships.Console > Program.cs]

```
using System.Collections.Generic;
using System.Configuration;

using Memberships.Business;
using Memberships.Entity;

namespace Memberships.Console
{
    class Program
    {
        private static string MS_SQL_PROVIDER_NAME =
ConfigurationManager.ConnectionStrings["MSSQLDB"].ProviderName;
        private static string MS_SQL_CONN_STR =
ConfigurationManager.ConnectionStrings["MSSQLDB"].ConnectionString;


        static void Main(string[] args)
        {
            // Call Business class - Pure query
            CallBusiness();

            // Call Business class - Stored procedure
            CallBusinessSP();

            // Call WebAPI
            CallWebAPI();
        }

        private static void CallBusiness()
        {
            // Create new member
            Member newMember = new Member("John Doe", true, "john@taemcorp.net", "080-00-1234-
5678", "anywhere");
            Member createdMemberInDB = new BizMemberShip(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).CreateMember(newMember);
            WriteMember(createdMemberInDB);

            // Select member
            Member selectedMemberInDB = new BizMemberShip(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMember(createdMemberInDB.MemberID);
            WriteMember(selectedMemberInDB);


            // Create new member
            Member newMember2 = new Member("Jane Doe", true, "jane@taemcorp.net", "080-00-0234-
5378", "anywhere");
            Member createdMemberInDB2 = new BizMemberShip(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).CreateMember(newMember2);
```

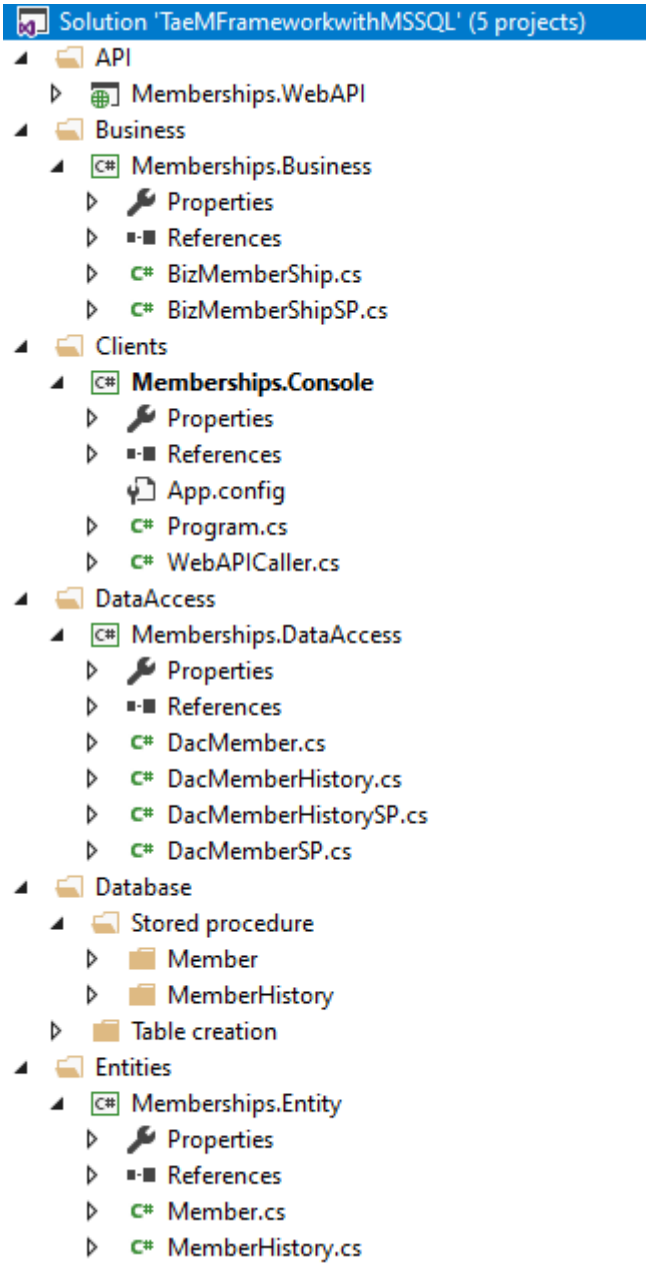
```
            WriteMember(createdMemberInDB2);

            // Get member
            Member selectedMemberInDB2 = new BizMemberShip(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMember(createdMemberInDB2.MemberID);
            WriteMember(selectedMemberInDB2);


            // Get members
            List<Member> currentMembers = new BizMemberShip(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMembers(string.Empty);
            WriteMembers(currentMembers);

            // Get numboer of members
            int numOfCurrentMembers = new BizMemberShip(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetNumsOfMembers(string.Empty);
            WriteCurrentNumberOfMemers(numOfCurrentMembers);

            // Update Member
            createdMemberInDB.MemberName = "John and Jane Doe";
            createdMemberInDB.Address = "John and Jane's home";
            createdMemberInDB.Email = "johnnjane.doe@taemcorp.net";

            bool updateResult = new BizMemberShip(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).SetMember(createdMemberInDB);

            if (updateResult)
            {
                Member updatedMember = new BizMemberShip(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMember(createdMemberInDB.MemberID);
                WriteMember(updatedMember);
            }

            // Delete Member
            bool deleteResult = new BizMemberShip(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).RemoveMember(createdMemberInDB.MemberID);

            List<Member> afterDeletedMembers = new BizMemberShip(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMembers(string.Empty);
            WriteMembers(afterDeletedMembers);
        }

        private static void CallBusinessSP()
        {
            // Create new member
            Member newMember = new Member("John Doe", true, "john@taemcorp.net", "080-00-1234-
5678", "anywhere");
            Member createdMemberInDB = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).CreateMember(newMember);
            WriteMember(createdMemberInDB);

            // Select member
            Member selectedMemberInDB = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMember(createdMemberInDB.MemberID);
            WriteMember(selectedMemberInDB);


            // Create new member
            Member newMember2 = new Member("Jane Doe", true, "jane@taemcorp.net", "080-00-0234-
5378", "anywhere");
            Member createdMemberInDB2 = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).CreateMember(newMember2);
            WriteMember(createdMemberInDB2);

            // Get member
            Member selectedMemberInDB2 = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
```

```
MS_SQL_CONN_STR).GetMember(createdMemberInDB2.MemberID);
                WriteMember(selectedMemberInDB2);


                // Get members
                List<Member> currentMembers = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMembers(string.Empty);
                WriteMembers(currentMembers);

                // Get numboer of members
                int numOfCurrentMembers = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetNumsOfMembers(string.Empty);
                WriteCurrentNumberOfMemers(numOfCurrentMembers);

                // Update Member
                createdMemberInDB.MemberName = "John and Jane Doe";
                createdMemberInDB.Address = "John and Jane's home";
                createdMemberInDB.Email = "johnnjane.doe@taemcorp.net";

                bool updateResult = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).SetMember(createdMemberInDB);

                if (updateResult)
                {
                    Member updatedMember = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMember(createdMemberInDB.MemberID);
                    WriteMember(updatedMember);
                }

                // Delete Member
                bool deleteResult = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).RemoveMember(createdMemberInDB.MemberID);

                List<Member> afterDeletedMembers = new BizMemberShipSP(MS_SQL_PROVIDER_NAME,
MS_SQL_CONN_STR).GetMembers(string.Empty);
                WriteMembers(afterDeletedMembers);
            }

            private static void CallWebAPI()
            {
                string uri = ConfigurationManager.AppSettings["WebAPI_IIS_Express_Uri"];

                // Create new member
                Member newMember = new Member("John Doe", true, "john@taemcorp.net", "080-00-1234-
5678", "anywhere");
                Member createdMemberInDB = WebAPICaller.CallCreateMember(uri, newMember);
                WriteMember(createdMemberInDB);

                // Select member
                Member selectedMemberInDB = WebAPICaller.CallGetMember(uri,
createdMemberInDB.MemberID);
                WriteMember(selectedMemberInDB);


                // Create new member
                Member newMember2 = new Member("Jane Doe", true, "jane@taemcorp.net", "080-00-0234-
5378", "anywhere");
                Member createdMemberInDB2 = WebAPICaller.CallCreateMember(uri, newMember2);
                WriteMember(createdMemberInDB2);

                // Get member
                Member selectedMemberInDB2 = WebAPICaller.CallGetMember(uri,
createdMemberInDB2.MemberID);
                WriteMember(selectedMemberInDB2);
```

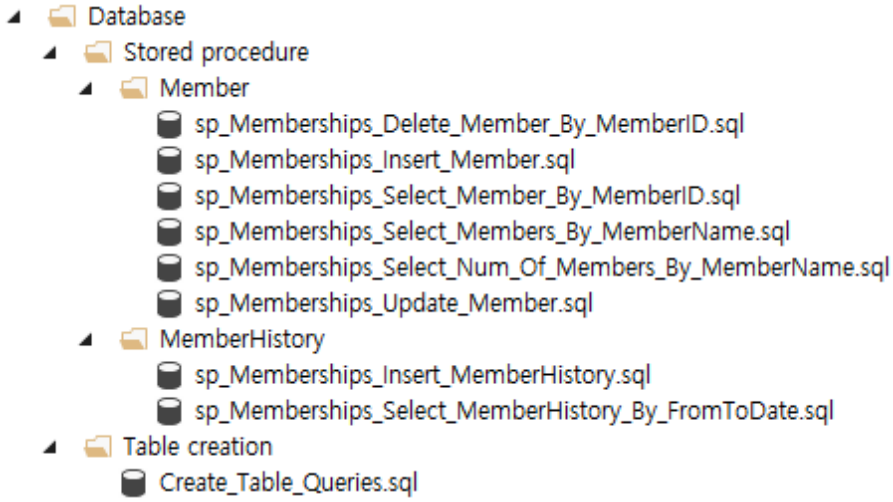
```
            // Get members
            List<Member> currentMembers = WebAPICaller.CallGetMembers(uri, string.Empty);
            WriteMembers(currentMembers);

            // Get numboer of members
            int numOfCurrentMembers = WebAPICaller.CallGetGetNumsOfMembers(uri, string.Empty);
            WriteCurrentNumberOfMemers(numOfCurrentMembers);

            // Update Member
            createdMemberInDB.MemberName = "John and Jane Doe";
            createdMemberInDB.Address = "John and Jane's home";
            createdMemberInDB.Email = "johnnjane.doe@taemcorp.net";

            bool updateResult = WebAPICaller.CallSetMember(uri, createdMemberInDB);

            if (updateResult)
            {
                Member updatedMember = WebAPICaller.CallGetMember(uri,
createdMemberInDB.MemberID);
                WriteMember(updatedMember);
            }

            // Delete Member
            bool deleteResult = WebAPICaller.CallRemoveMember(uri, createdMemberInDB.MemberID);

            if (deleteResult)
            {
                List<Member> afterDeletedMembers = WebAPICaller.CallGetMembers(uri, string.Empty);
                WriteMembers(afterDeletedMembers);
            }
        }

        private static void WriteMember(Member member)
        {
            if (member != null)
            {
                System.Console.WriteLine(
                    @"This member has " +
                    @"[MemberID:{0}] [MemberName:{1}] [IsAvailable:{2}] " +
                    @"[Email:{3}] [PhoneNumber:{4}] [Address:{5}] " +
                    @"[InsertedDate:{6}] [UpdatedDate:{7}]",
                    member.MemberID, member.MemberName, member.IsAvailable,
                    member.Email, member.PhoneNumber, member.Address,
                    member.InsertedDate, member.UpdatedDate);
            }
        }

        private static void WriteMembers(List<Member> members)
        {
            foreach (Member member in members)
                WriteMember(member);
        }

        private static void WriteCurrentNumberOfMemers(int numOfMembers)
        {
            System.Console.WriteLine(
                @"Current number of members : {0}",
                numOfMembers
                );
        }
    }
}
```

Sample solution

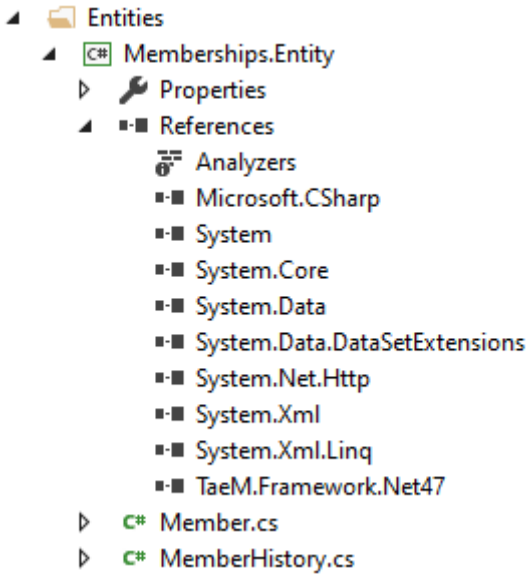
The overall solution that you have created and described so far is as follows. This configuration method is not perfect, but it is a structure we have created for readability and functional separation. So, it is good for you that you can selectively apply it to your own software development.



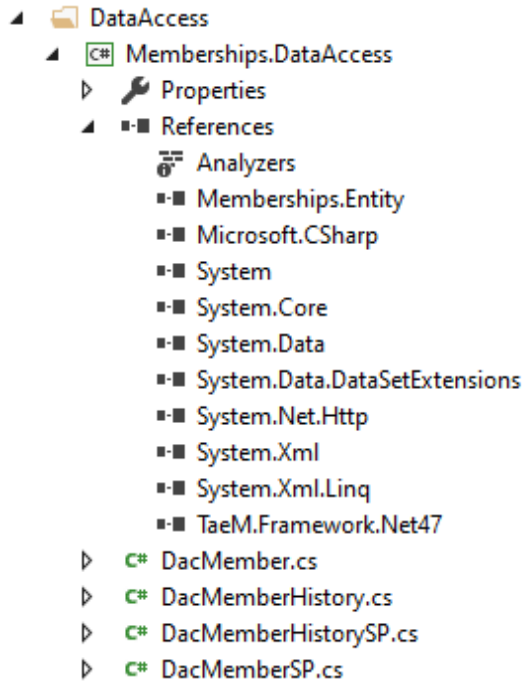
(1) Database solution folder: Database table and stored procedure are included.



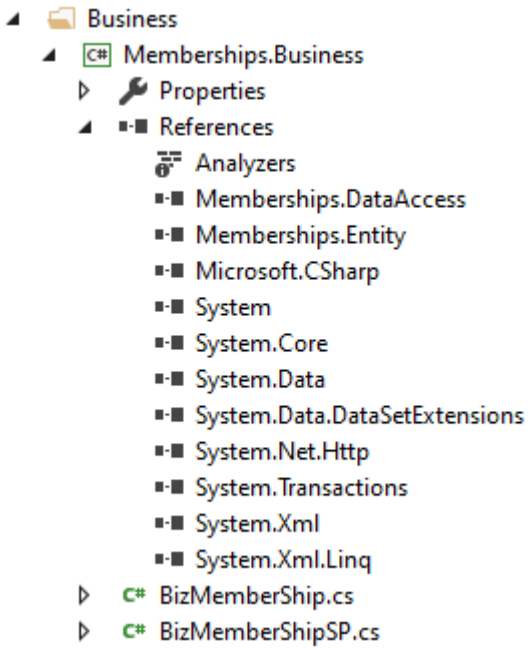
(2) Entities solution folder: Entity classes and projects are included.



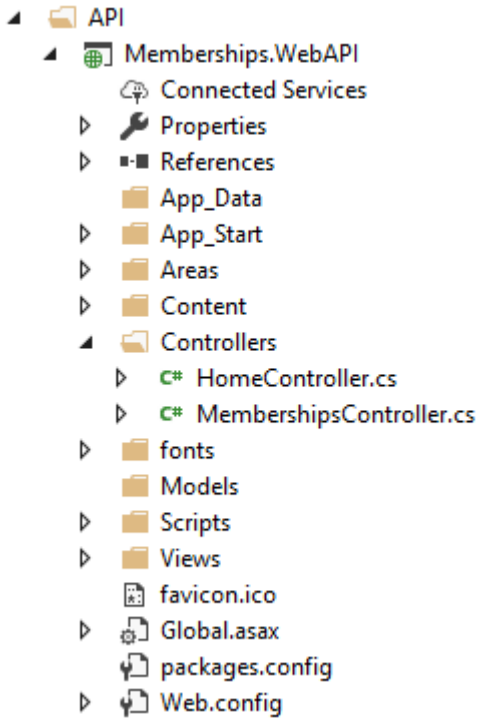
(3) DataAccess solution folder: Data access related classes and projects are included.



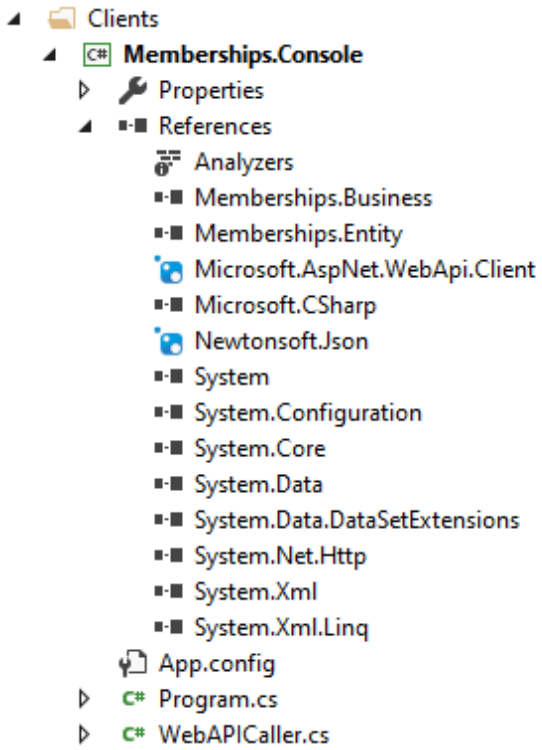
(4) Business solution folder: This includes classes and projects for transaction processing and data processing according to business.



(5) API solution folder: This includes classes and projects for processing API services such as WebAPI.



(6) Clients Solution folder: This includes test console application and other client applications.



How to use TaeM Framework with Oracle

Now, we will examine the process of applying TaeM Framework ORM function in Oracle. Since we have created it like the tutorial format, you can follow the steps in order to easily develop software based on Oracle Database by using TaeM Framework ORM function. Note that the below source codes and other samples are made in Microsoft Visual Studio 2017. And you can use other version of Visual Studio because Microsoft Visual Studio supports compatibility to another versions.

Database Table Schema & Entity Class

It's a chicken-and-egg problem. You can create an entity class and create a table of the database later or you can create a table of the database and create entity class later. So, we have created a database table first.

(1) Database Table Schema

We have defined Member and MemberHistory table used in common development. This Member table has member information and This MemberHistory tables keeps records of changes in Member table.

(A) Member table

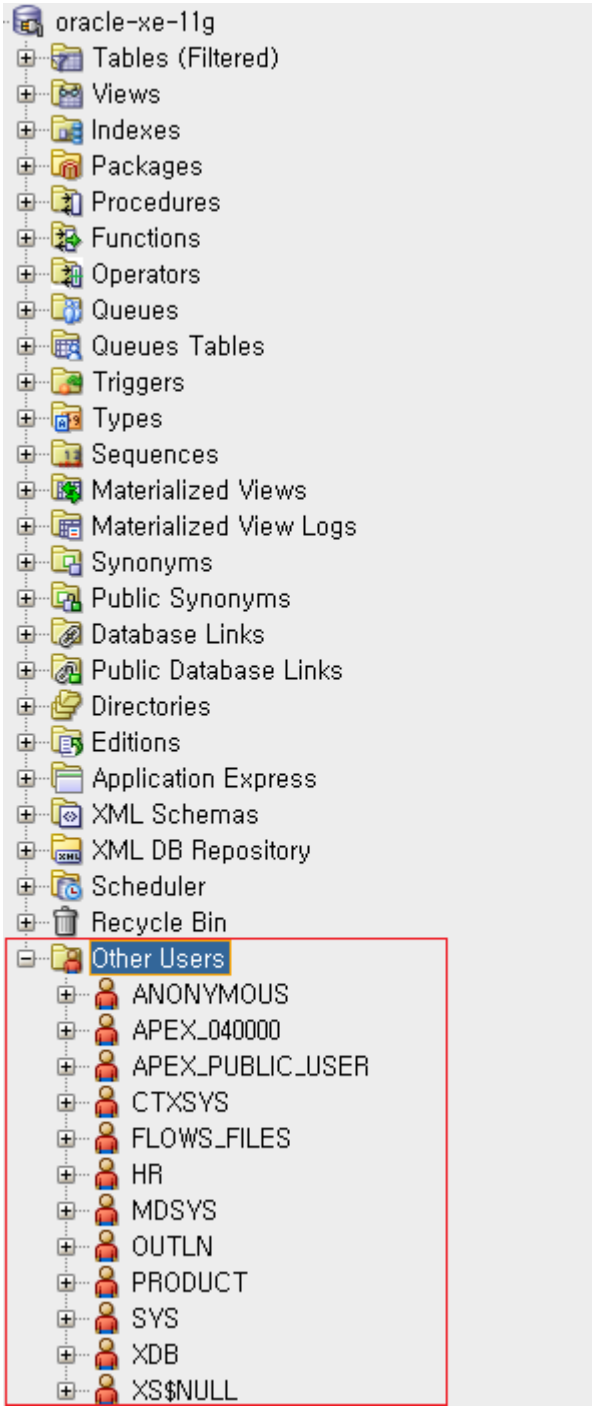
Table Name	Member				
This table has each member information in the system.					
Field Name	**Date type**	**Nullable**	**Default value**	**Primary key**	**Others**
MemberID	INTEGER	NOT NULL		Primary key	Using Oracle sequence (Product.SEQ_Member)
MemberName	NVARCHAR2(100)	NULL			
IsAvailable	SMALLINT	NULL			Boolean_IsAvailable_CK Constraint
Email	NVARCHAR2(100)	NULL			
PhoneNumber	NVARCHAR2(100)	NULL			
Address	NVARCHAR2(1024)	NULL			
InsertedDate	DATE	NULL			
UpdatedDate	DATE	NULL			

(B) MemberHistory table

Table Name	MemberHistory				
This table has member transaction history. It means that when the member table changes in each field the change history is stored in this table.					
Field Name	**Date type**	**Nullable**	**Default value**	**Primary key**	**Others**
Seq	INTEGER	NOT NULL		Primary key	Using Oracle sequence (Product.SEQ_MemberHistory)
MemberID	INTEGER	NOT NULL			
MemberName	NVARCHAR2(100)	NULL			
IsSuccess	SMALLINT	NULL			Boolean_IsSuccess_CK Constraint
Message	NVARCHAR2(1024)	NULL			
InsertedDate	DATE	NULL			

(C) Table creation query

In Oracle Database, you need to create the Product user as follows.



Then, the following table creation query is executed to create a Member and a MemberHistory table. Unlike MS-SQL Server, Oracle does not have an auto-increment field attribute. So, we need to create a sequence and get the value from this sequence in the query that inserts the data (Insert or Update query) and use it as the input value of the field that needs autogrow. Therefore, we need to create SEQ_Member as sequence of MemberID of Member table and SEQ_MemberHistory as sequence of Seq of MemberHistory table. You can see this table creation query in the sample project.

In TaeMFrameworkwithOracle [Database > Table creation > Create_Table_Queries.sql]

```
-- Member Table Creation Query
CREATE TABLE Product.Member (
	MemberID INTEGER NOT NULL CONSTRAINT Member_pk_meberid PRIMARY KEY,
	MemberName NVARCHAR2(100) NULL,
	IsAvailable SMALLINT CONSTRAINT Boolean_IsAvailable_CK CHECK(IsAvailable = 0 OR IsAvailable =
1) NULL,
	Email NVARCHAR2(100) NULL,
	PhoneNumber NVARCHAR2(100) NULL,
	Address NVARCHAR2(1024) NULL,
	InsertedDate DATE NULL,
	UpdatedDate DATE NULL
);

-- Drop table
-- DROP TABLE Product.Member;


-- MemberHistory Table Creation Query
CREATE TABLE Product.MemberHistory (
	Seq INTEGER NOT NULL CONSTRAINT MemberHistory_pk_seq PRIMARY KEY,
	MemberID INTEGER NOT NULL,
	MemberName NVARCHAR2(100) NULL,
	IsSuccess SMALLINT CONSTRAINT Boolean_IsSuccess_CK CHECK(IsSuccess = 0 OR IsSuccess = 1)
NULL,
	Message NVARCHAR2(1024) NULL,
	InsertedDate DATE NULL
);

-- Drop table
-- DROP TABLE Product.MemberHistory;

-- CREATE Member table Sequence
CREATE SEQUENCE Product.SEQ_Member;

-- SELECT Sequence
--SELECT Product.SEQ_Member.NEXTVAL FROM DUAL;

-- CREATE MemberHistory table Sequence
CREATE SEQUENCE Product.SEQ_MemberHistory;

-- SELECT Sequence
--SELECT Product.SEQ_MemberHistory.NEXTVAL FROM DUAL;
```

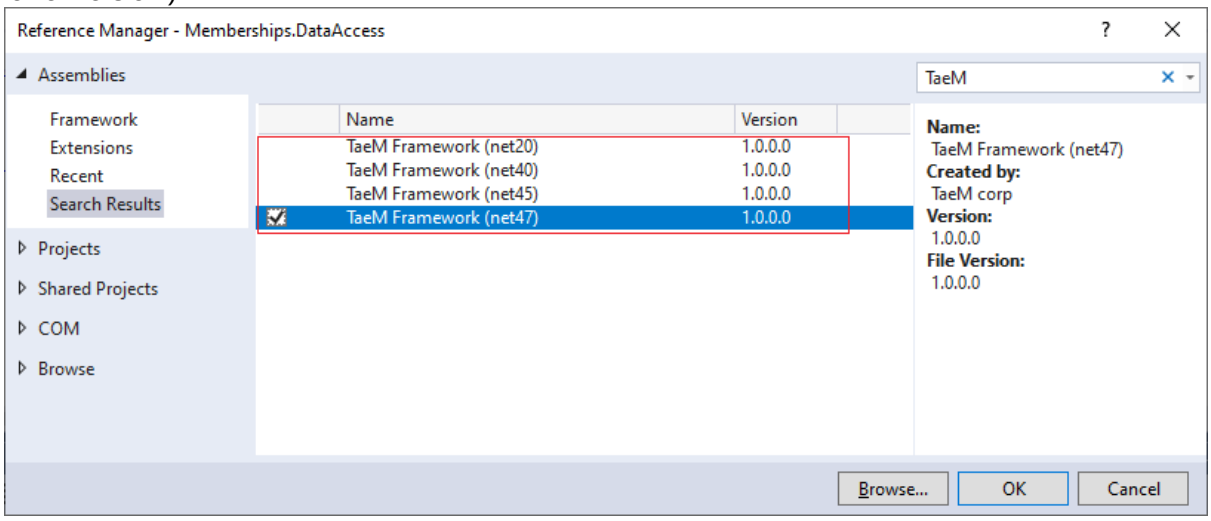
(2) Entity Class

Through the definition of the Member table and the MemberHistory table, each entity classes can be created as follows. In this case, we will map all the fields of the Member table and the MemberHistory table as they are, so we have defined that each classes have same properties with all the fields of the Member table and the MemberHistory table.

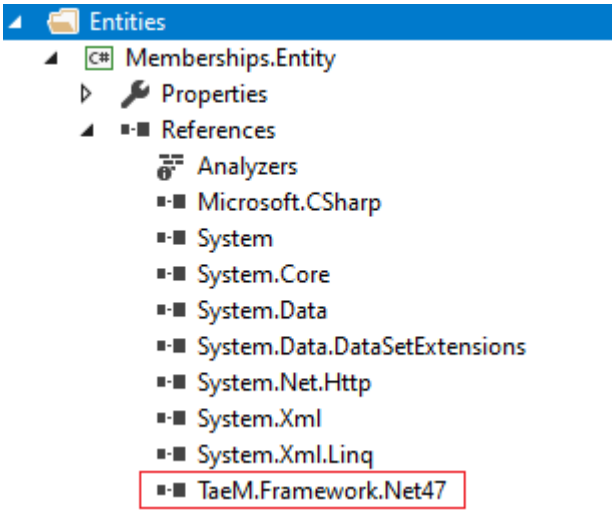
(A) Creation of project and addition of TaeM Framework dll as reference

Before you define the entity class, you must create a project to contain the entity class. You can make a project with Visual Studio that you are using. You can create a Memberships.Entity project through Visual Studio's [New> Project]. Then you can add TaeM.Framework.Net47.dll as a reference to the Memberships.Entity project. Because you installed the TaeM Framework in the previous chapter, here you will see the following "Reference Manager" when you click "Add Reference" in the Memberships.Entity project. When you search "Reference Manager" for "TaeM", you will see four .NET Framework versions of TaeM.Framework.dll. (Note that the target framework for the current project is the .NET Framework 4.7, so all four versions are displayed, including the

lower version.)



In this case, select "TaeM Framework (net47)". So, if you add a reference, you should add TaeM.Framework.Net47.dll to the Memberships.Entity project as shown below.



(B) Member entity class

You created a project and added TaeM.Framework.Net47.dll as a reference to the project. Then we will make the Member entity class to the class Memberships.Entity project. In Visual Studio, you can add the Member.cs class file through [New> File].

After you create the Member.cs class file, you need to add the TaeM.Framework.Data.Oracle code using the using statement as follows. TaeM.Framework.Data.Oracle has an attribute class defined as OracleDataBinder. This OracleDataBinder attribute class automatically binds the result of the DB execution to the field (or variable) or property of the specified class when this attribute is specified for the field (or variable) or property of the class.

```
using System;
using TaeM.Framework.Data.Oracle;
```

Then, we can define the Member entity class as shown below. The Member entity class has constructors and a properties. For each property, OracleDataBinder attribute can be specified as follows. The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Entities> Memberships.Entity> Member.cs]

```
using System;
using TaeM.Framework.Data.Oracle;

namespace Memberships.Entity
{
    public class Member
    {
        public Member() : this(string.Empty, false,
            string.Empty, string.Empty, string.Empty)

        {
        }

        public Member(string memberName, bool isAvailable,
            string email, string phoneNumber, string address)
            : this(-1, memberName, isAvailable, email, phoneNumber, address)
        {
        }

        public Member(int memberID, string memberName, bool isAvailable,
            string email, string phoneNumber, string address)
        {
            this.MemberID = memberID;
            this.MemberName = memberName;
            this.IsAvailable = isAvailable;

            this.Email = email;
            this.PhoneNumber = phoneNumber;
            this.Address = address;
        }

        [OracleDataBinder("MemberID")]
        public int MemberID { get; set; }

        [OracleDataBinder("MemberName")]
        public string MemberName { get; set; }

        [OracleDataBinder("IsAvailable")]
        public bool IsAvailable { get; set; }

        [OracleDataBinder("Email")]
        public string Email { get; set; }

        [OracleDataBinder("PhoneNumber")]
        public string PhoneNumber { get; set; }

        [OracleDataBinder("Address")]
        public string Address { get; set; }

        [OracleDataBinder("InsertedDate")]
        public DateTime InsertedDate { get; set; }

        [OracleDataBinder("UpdatedDate")]
        public DateTime UpdatedDate { get; set; }
    }
}
```

In other words, an important part of the above Member entity class definition is the OracleDataBinder attribute which specifies the field name of the database table (exactly the field name of the query result from the database) and this is specified in each property of the Member entity class for the mapping from each field in the database table. So, OracleDataBinder attribute is used to map a table field of Oracle Database to each field or property of Member entity class. The field names of

the table to be mapped are specified in the OracleDataBinder attribute parameter.

How to use OracleDataBinder attribute is as follows.

```
OracleDataBinder ("Field name of the Dababase table to be mapped")
```

In ORM of TaeM Framework, it is not necessary to create separate XML file differently from other ORM solution. Just when declaring entity class, you only need to declare Database data mapping information by declaring attribute. In this case, the OracleDataBinder attribute is used as the mapping declaration.

(C) MemberHistory Entity class

This time we define the MemberHistory Entity class. You need to create a MemberHistory class, add a using statement, and assign an OracleDataBinder attribute to the Property or Field. The defined MemberHistory.cs class file is as follows.

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Entities> Memberships.Entity> MemberHistory.cs]

```
using System;
using TaeM.Framework.Data.Oracle;

namespace Memberships.Entity
{
    public class MemberHistory
    {
        public MemberHistory()
            : this(-1, string.Empty, false, string.Empty)
        {
        }

        public MemberHistory(int memberID, string memberName, bool isSuccess, string message)
            : this(-1, memberID, memberName, isSuccess, message, DateTime.MinValue)
        {
        }

        public MemberHistory(int seq, int memberID, string memberName,
            bool isSuccess, string message, DateTime insertedDate)
        {
            this.Seq = seq;
            this.MemberID = memberID;
            this.MemberName = memberName;

            this.IsSuccess = isSuccess;
            this.Message = message;
            this.InsertedDate = insertedDate;
        }


        [OracleDataBinder("Seq")]
        public int Seq { get; set; }

        [OracleDataBinder("MemberID")]
        public int MemberID { get; set; }

        [OracleDataBinder("MemberName")]
        public string MemberName { get; set; }
```

```
            [OracleDataBinder("IsSuccess")]
            public bool IsSuccess { get; set; }

            [OracleDataBinder("Message")]
            public string Message { get; set; }

            [OracleDataBinder("InsertedDate")]
            public DateTime InsertedDate { get; set; }
        }
}
```

Data Access

Next, we will defines the class of the Data Access part. The classes in the Data Access part connect to the actual database, execute a query on the database, and bind the resultant value to the entity class. Of course, you might get the number of rows affected by the query type in the database, or you might want to import the first column of the first row.

At First, the functions of Oracle Database related helper classes provided by TaeM Framework 1.0.1.0 are as follows.

(1) OracleDataHelperFactory class

	Method name	Description
1	SelectSingleEntity<T>()	SelectSingleEntity<T> method executes query and return their single row result with a type of return object, an SQL command type, an SQL query, and SQL parameters.
2	SelectMultipleEntities<T>()	SelectMultipleEntities<T> method executes query and return their multiple rows result with a type of return object, an SQL command type, an SQL query, and SQL parameters.
3	SelectScalar()	SelectScalar method executes query and return the value of first row and first column with an SQL command type, an SQL query, and SQL parameters.
4	Execute()	Execute method executes query and return the number of affected row from the SQL query execution result with an SQL command type, an SQL query, and SQL parameters.

(2) OracleParameterHelperFactory class

	Property name	Description
1	ProviderName	ProviderName is a property used to set or get the DB provider name. For Orale DB, this value is usually "Oracle.ManagedDataAccess.Client" or “Oracle.DataAccess.Client” , and “Oracle.ManagedDataAccess.dll” or "Oracle.DataAccess.dll" is used.

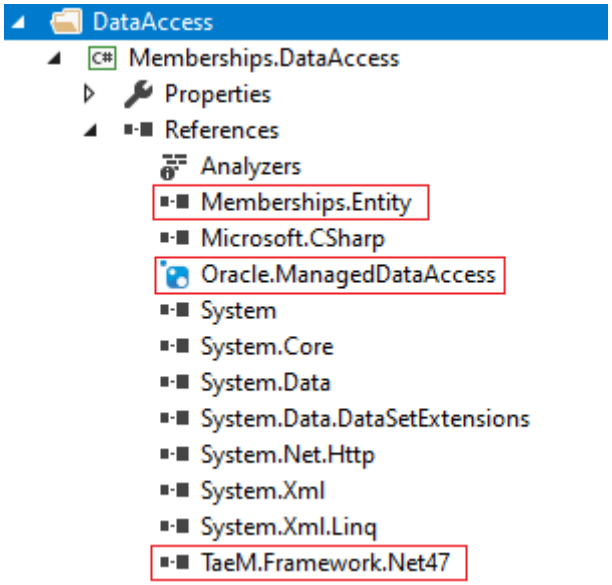
	Method name	Description
1	CreateParameter()	CreateParameter method creates and returns an object of Oracle DB Parameter with provider name, parameter name, parameter value, direction of parameter.
2	CreateParameterWOProviderName()	CreateParameter method creates and returns an object of Oracle DB Parameter with parameter name, parameter value, the direction of parameter. This should only be used if you have set a value for the ProviderName property.
3	CreateParameter<T>()	CreateParameter method creates and returns an object of Oracle DB Parameter with provider name, parameter name, parameter data type, parameter value, direction of parameter.
4	CreateParameterWOProviderName<T>()	CreateParameter method creates and returns an object of Oracle DB Parameter with parameter name, parameter value, the direction of parameter. This should only be used if you have set a value for the ProviderName property.

When using Oracle as a database, you can easily query the Oralce Database query by using the above two Helper classes OracleDataHelperFactory and OracleParameterHelper class and get the result. Using these two classes, we will code the CRUD ("Create: Create," "Retrieve or Read," "Update: Update," and "Delete or Destroy", A term that refers to a function that a user interface must have.) function with the database.

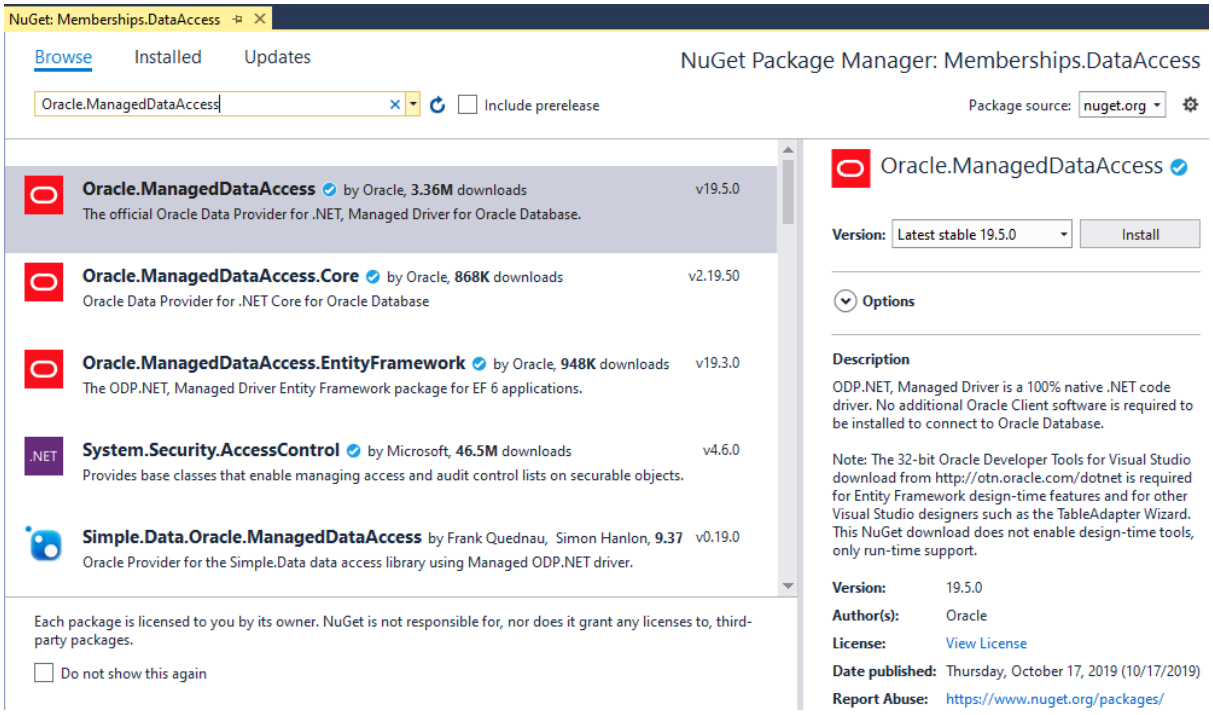
(3) Making the CRUD

Because it is responsible for Data Access, we will define the class with a name of type DacXXX. In this case, we will define a class name DacMember because it is responsible for the data access of the Member. And we will make a Membership.DataAccess project that is a collection of data access classes, and we will define the DacMember class in this project. We can make the Memberships.DataAccess project as shown below through [New> Project] in Visual Studio. In this case, for the readability and similar property is separated into separate projects, but it is possible to put them together in one project according to the coding rules you want or use.

Since we created a new project, we need to add TaeM.Framework.Net47.dll as a reference to our Memberships.DataAccess project. Of course, if you use the same proejct before, you have already added TaeM.Framework.Net47.dll as a reference. Select "Add Reference" and add TaeM.Framework.Net47.dll in "Reference Manager". The following figure shows TaeM.Framework.Net47.dll and Memberships.Entity added to Memberships.DataAccess project. We also added Oracle.ManagedDataAccess.dll, a .NET Data Provider for Oracle Database, as a reference. This Oracle.ManagedDataAccess.dll is a database driver for Oracle Database.



For reference, Oracle.ManagedDataAccess.dll was added by NuGet.



We made Memberships.DataAccess project and added TaeM.Framework.Net47.dll and Oracle.ManagedDataAccess.dll to the project as references. Next, we will try adding the DacMember.cs class to the Memberships.DataAccess project. The Addition of the DacMember.cs class file through Visual Studio's [New> File]. After you added the DacMember.cs class file, you need to use the using statement to add the TaeM.Framework.Data code and the Memberships.Entity namespace of the entity class you created before as follows.

TaeM.Framework.Data has the OracleDataHelperFactory and OracleParameterHelper classes we discussed earlier. Therefore, we will add code using two helper classes in this class. And Memberships.Entity has a Member.cs class which is the entity class that we defined. We also defined the using statement to use System.Data and Oracle.ManagedDataAccess.Client to use the .NET Data Provider for Oracle Database.

```
using System;
using System.Collections.Generic;
using System.Data;

using Oracle.ManagedDataAccess.Client;

using TaeM.Framework.Data;
using Memberships.Entity;
```

Then, we can define the DacMember.cs class as shown below. The DacMember.cs class defines a PROVIDER_NAME that specifies the default DB provider, a CONNECTION_STRING that specifies the default Oracle Database connection string, a providerName with DB provider information, and a connection field with OracleConnection information, and three constructors and seven methods.

```
using System;
using System.Collections.Generic;
using System.Data;

using Oracle.ManagedDataAccess.Client;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMember
    {
        private static readonly string PROVIDER_NAME = "Oracle.ManagedDataAccess.Client";

        private static readonly string CONNECTION_STRING
            = "Data Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=59161))(CONNECT_DATA=(SERVICE_NAME=XE)));User Id=Product;Password=qwert12345!;";


        private string providerName;
        private OracleConnection connection;


        public DacMember() ...
        public DacMember(string providerName, string connectionString)...
        public DacMember(string providerName, OracleConnection connection)...


        /// <summary> InsertMember method - Insert Member table row from member informat ...
        public bool InsertMember(Member member)...

        /// <summary> SelectInsertedMemberID method - Select recently inserted MemberID
        public int SelectInsertedMemberID()...

        /// <summary> SelectMember method - Select Member table row by memberID
        public Member SelectMember(int memberID)...

        /// <summary> SelectMembers method - Select Member table rows by memberName
        public List<Member> SelectMembers(string memberName)...

        /// <summary> SelectNumsOfMembers method - Select number of Member table rows by ...
        public int SelectNumsOfMembers(string memberName)...

        /// <summary> UpdateMember method - Update Member table row by member informatio ...
        public bool UpdateMember(Member member)...

        /// <summary> DeleteMember() method - Delete Member table row by memberID
        public bool RemoveMember(int memberID)...
    }
}
```

The above seven methods need to execute the following kind of query on the Member table and return the result value.

	Query type	Method name
1	Retrieve a single data row	SelectMember(…)
2	Retrieve multiple data row	SelectMembers(…)
3	Creation of the data row	InsertMember(…)
4	Modification of the data row	UpdateMember(…)
5	Deletion of the data row	DeleteMember(…)
6	Retrieve the number of data row	SelectNumsOfMembers(…)
7	Retrieve recently inserted MemberID	SelectInsertedMemberID(…)

To implement the method according to the above query type, you can use the following method defined in the OracleDataHelperFactory class mentioned earlier in this section.

	Query type	Method name	The Method of OracleDataHelperFactory class
1	Retrieve a single data row	SelectMember(…)	Using the SelectSingleEntity <T> () method
2	Retrieve multiple data row	SelectMembers(…)	Using the SelectMultipleEntities <T> () method
3	Creation of the data row	InsertMember(…)	Using the Execute () method
4	Modification of the data row	UpdateMember(…)	Using the Execute () method
5	Deletion of the data row	DeleteMember(…)	Using the Execute () method
6	Retrieve the number of data row	SelectNumsOfMembers(…)	Using the SelectScalar () method
7	Retrieve recently	SelectInsertedMemberID(…)	Using the SelectScalar () method

	inserted MemberID		

The OracleParameterHelper class is a class that generalizes the parameters of a database and has one property and four methods.

So, we can make the DacMember.cs class which implements CRUD using OracleDataHelperFactory and OracleParameterHelper class as follows.

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [DataAccess> Memberships.DataAccess> DacMember.cs]

```
using System;
using System.Collections.Generic;
using System.Data;

using Oracle.ManagedDataAccess.Client;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMember
    {
        private static readonly string PROVIDER_NAME = "Oracle.ManagedDataAccess.Client";

        private static readonly string CONNECTION_STRING
            = "Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=59161))(CONNECT_DA
TA=(SERVICE_NAME=XE)));User Id=Product;Password=qwert12345!;";


        private string providerName;
        private OracleConnection connection;


        public DacMember() : this(PROVIDER_NAME, CONNECTION_STRING)
        {
        }
        public DacMember(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connection = new OracleConnection(connectionString);
        }
        public DacMember(string providerName, OracleConnection connection)
        {
            this.providerName = providerName;
            this.connection = connection;
        }


        /// <summary>
        /// InsertMember method
        /// - Insert Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public bool InsertMember(Member member)
        {
            try
            {
                using (connection)
                {
```

```
                        if (string.IsNullOrEmpty(OracleParameterHelperFactory.ProviderName))
                            OracleParameterHelperFactory.ProviderName = providerName;

                        connection.Open();

                        int ret = (int)OracleDataHelperFactory.Execute(connection,
                            CommandType.Text,
                            @"INSERT INTO Product.Member " +
                            @"( MemberID, MemberName, IsAvailable, Email, PhoneNumber, Address,
InsertedDate, UpdatedDate ) " +
                            @"VALUES " +

@"( Product.SEQ_Member.NEXTVAL, :MemberName, :IsAvailable, :Email, :PhoneNumber, :Address,
SYSDATE, NULL ) ",

OracleParameterHelperFactory.CreateParameterWOProviderName(":MemberName", member.MemberName,
ParameterDirection.Input),
                            OracleParameterHelperFactory.CreateParameterWOProviderName(":IsAvailable",
((member.IsAvailable) ? 1 : 0), ParameterDirection.Input),
                            OracleParameterHelperFactory.CreateParameterWOProviderName(":Email",
member.Email, ParameterDirection.Input),

OracleParameterHelperFactory.CreateParameterWOProviderName(":PhoneNumber",
member.PhoneNumber, ParameterDirection.Input),
                            OracleParameterHelperFactory.CreateParameterWOProviderName(":Address",
member.Address, ParameterDirection.Input)
                            );

                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// SelectInsertedMemberID method
            /// - Select recently inserted MemberID
            /// </summary>
            /// <returns></returns>
            public int SelectInsertedMemberID()
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        int ret = Convert.ToInt32(
                            OracleDataHelperFactory.SelectScalar(connection,
                                CommandType.Text,
                                @"SELECT Product.SEQ_Member.CURRVAL FROM DUAL"
                                )
                            );

                        return ret;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
```

```
        /// <summary>
        /// SelectMember method
        /// - Select Member table row by memberID
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public Member SelectMember(int memberID)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    Member ret =
(Member)OracleDataHelperFactory.SelectSingleEntity<Member>(connection,
                        typeof(Member),
                        CommandType.Text,
                        @"SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber,
Address, InsertedDate, UpdatedDate " +
                        @"FROM Product.Member " +
                        @"WHERE MemberID = :MemberID ",
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemberID",
memberID, ParameterDirection.Input)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// SelectMembers method
        /// - Select Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public List<Member> SelectMembers(string memberName)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    List<Member> ret =
(List<Member>)OracleDataHelperFactory.SelectMultipleEntities<Member>(connection,
                        typeof(Member),
                        CommandType.Text,
                        @"SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber,
Address, InsertedDate, UpdatedDate " +
                        @"FROM Product.Member " +
                        @"WHERE MemberName LIKE '%" + memberName + "%' "
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
```

```
        }

        /// <summary>
        /// SelectNumsOfMembers method
        /// - Select number of Member table rows by memberName
        /// </summary>
        /// <param name="memberName"></param>
        /// <returns></returns>
        public int SelectNumsOfMembers(string memberName)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    int ret = Convert.ToInt32(
                        OracleDataHelperFactory.SelectScalar(connection,
                            CommandType.Text,
                            @"SELECT COUNT(*) " +
                            @"FROM Product.Member " +
                            @"WHERE MemberName LIKE '%" + memberName + "%' "
                            )
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// UpdateMember method
        /// - Update Member table row by member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public bool UpdateMember(Member member)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    int ret = OracleDataHelperFactory.Execute(connection,
                        CommandType.Text,
                        @"UPDATE Product.Member " +
                        @"SET MemberName = :MemberName, IsAvailable = :IsAvailable, Email = :Email,
" +
                        @"    PhoneNumber = :PhoneNumber, Address = :Address, UpdatedDate =
SYSDATE " +
                        @"WHERE MemberID = :MemberID ",
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemberName",
member.MemberName, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":IsAvailable",
((member.IsAvailable) ? 1 : 0), ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":Email",
member.Email, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":PhoneNumber",
member.PhoneNumber, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":Address",
member.Address, ParameterDirection.Input),
```

```
                                OracleParameterHelperFactory.CreateParameter(providerName, ":MemberID",
member.MemberID, ParameterDirection.Input)
                                );

                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// DeleteMember() method
            /// - Delete Member table row by memberID
            /// </summary>
            /// <param name="memberID">Member ID</param>
            /// <returns></returns>
            public bool RemoveMember(int memberID)
            {
                try
                {
                    using (connection)
                    {
                        if (string.IsNullOrEmpty(OracleParameterHelperFactory.ProviderName))
                            OracleParameterHelperFactory.ProviderName = providerName;

                        connection.Open();

                        int ret = OracleDataHelperFactory.Execute(connection,
                            CommandType.Text,
                            @"DELETE FROM Product.Member " +
                            @"WHERE MemberID = :MemberID ",
                            OracleParameterHelperFactory.CreateParameterWOProviderName(":MemberID",
memberID, ParameterDirection.Input)
                            );

                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
        }
}
```

Here you can define the DacMemberHistory class. We have created the DacMemberHistory class, added the using statement, and defined methods in the DacMemberHistory.cs class as shown below.

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [DataAccess > Memberships.DataAccess > DacMemberHistory.cs]

```
using System;
using System.Collections.Generic;
using System.Data;

using Oracle.ManagedDataAccess.Client;

using TaeM.Framework.Data;
using Memberships.Entity;
```

```
namespace Memberships.DataAccess
{
    public class DacMemberHistory
    {
        private static readonly string PROVIDER_NAME = "Oracle.ManagedDataAccess.Client";

        private static readonly string CONNECTION_STRING
            = "Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=59161))(CONNECT_DA
TA=(SERVICE_NAME=XE)));User Id=Product;Password=qwert12345!;";


        private string providerName;
        private OracleConnection connection;


        public DacMemberHistory() : this(PROVIDER_NAME, CONNECTION_STRING)
        {
        }
        public DacMemberHistory(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connection = new OracleConnection(connectionString);
        }
        public DacMemberHistory(string providerName, OracleConnection connection)
        {
            this.providerName = providerName;
            this.connection = connection;
        }


        /// <summary>
        /// InsertMemberHistory method
        /// - Insert MemberHistory table row from member history information
        /// </summary>
        /// <param name="member">Member history information</param>
        /// <returns></returns>
        public bool InsertMemberHistory(MemberHistory memberHistory)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    int ret = (int)OracleDataHelperFactory.Execute(connection,
                        CommandType.Text,
                        @"INSERT INTO Product.MemberHistory " +
                        @"( Seq, MemberID, MemberName, IsSuccess, Message, InsertedDate ) " +
                        @"VALUES " +

@"( Product.SEQ_MemberHistory.NEXTVAL, :MemberID, :MemberName, :IsSuccess, :Message, SYSDATE )
",
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemberID",
memberHistory.MemberID, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemberName",
memberHistory.MemberName, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":IsSuccess",
((memberHistory.IsSuccess) ? 1 : 0), ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":Message",
memberHistory.Message, ParameterDirection.Input)
                        );

                    return (ret == 1) ? true : false;
                }
```

```
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        public int SelectInsertedSeq()
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    int ret = Convert.ToInt32(
                        OracleDataHelperFactory.SelectScalar(connection,
                            CommandType.Text,
                            @"SELECT Product.SEQ_MemberHistory.CURRVAL FROM DUAL"
                            )
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        public MemberHistory SelectMemberHistory(int seq)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    MemberHistory ret =
(MemberHistory)OracleDataHelperFactory.SelectSingleEntity<MemberHistory>(connection,
                        typeof(MemberHistory),
                        CommandType.Text,
                        @"SELECT Seq, MemberID, MemberName, IsSuccess, Message, InsertedDate "
+
                        @"FROM Product.MemberHistory " +
                        @"WHERE Seq = :Seq ",
                        OracleParameterHelperFactory.CreateParameter(providerName, ":Seq", seq,
ParameterDirection.Input)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// SelectMemberHistories() method
        /// - Select MemberHistory table row by fromDate and toDate
        /// </summary>
        /// <param name="fromDate">From date</param>
        /// <param name="toDate">To date</param>
```

```
        /// <returns></returns>
        public List<MemberHistory> SelectMemberHistories(DateTime fromDate, DateTime toDate)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    return
(List<MemberHistory>)OracleDataHelperFactory.SelectMultipleEntities<MemberHistory>(connection,
                        typeof(MemberHistory),
                        CommandType.Text,
                        @"SELECT Seq, MemberID, MemberName, IsSuccess, Message, InsertedDate "
+
                        @"FROM Product.MemberHistory " +
                        @"WHERE InsertedDate >= :FromDate AND InsertedDate <= :ToDate " +
                        @"ORDER BY InsertedDate DESC ",
                        OracleParameterHelperFactory.CreateParameter(providerName, ":FromDate",
fromDate, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":ToDate",
toDate, ParameterDirection.Input)
                        );
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
    }
}
```

(4) Using stored procedure

In the DacMember.cs and DacMemberHistory.cs class definitions above, we created a pure query in text format to execute the query directly to Oracle Database. However, you can use the stored procedure as shown below for performance.

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [DataAccess > Memberships.DataAccess > DacMemberSP.cs]

```
using System;
using System.Collections.Generic;
using System.Data;

using Oracle.ManagedDataAccess.Client;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMemberSP
    {
        private static readonly string PROVIDER_NAME = "Oracle.ManagedDataAccess.Client";

        private static readonly string CONNECTION_STRING
            =
"Server=localhost;Port=32785;Database=Product;Uid=root;Pwd=qwert12345!;ConnectionLifeTime=60;AllowU
serVariables=true;";
```

```
        private string providerName;
        private OracleConnection connection;


        public DacMemberSP() : this(PROVIDER_NAME, CONNECTION_STRING)
        {
        }
        public DacMemberSP(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connection = new OracleConnection(connectionString);
        }
        public DacMemberSP(string providerName, OracleConnection connection)
        {
            this.providerName = providerName;
            this.connection = connection;
        }


        /// <summary>
        /// InsertMember method
        /// - Insert Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member InsertMember(Member member)
        {
            try
            {
                using (connection)
                {
                    if (string.IsNullOrEmpty(OracleParameterHelperFactory.ProviderName))
                        OracleParameterHelperFactory.ProviderName = providerName;

                    connection.Open();

                    Member ret =
(Member)OracleDataHelperFactory.SelectSingleEntity<Member>(connection,
                        typeof(Member),
                        CommandType.StoredProcedure,
                        "Product.sp_Insert_Member",
                        OracleParameterHelperFactory.CreateParameterWOProviderName(":MemName",
member.MemberName, ParameterDirection.Input),

OracleParameterHelperFactory.CreateParameterWOProviderName(":MemIsAvailable",
((member.IsAvailable) ? 1 : 0), ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameterWOProviderName(":MemEmail",
member.Email, ParameterDirection.Input),

OracleParameterHelperFactory.CreateParameterWOProviderName(":MemPhoneNumber",
member.PhoneNumber, ParameterDirection.Input),

OracleParameterHelperFactory.CreateParameterWOProviderName(":MemAddress", member.Address,
ParameterDirection.Input),

OracleParameterHelperFactory.CreateParameterWOProviderName<OracleDbType>(":OutputData", null,
OracleDbType.RefCursor, ParameterDirection.Output)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
```

```
        }

        /// <summary>
        /// SelectMember method
        /// - Select Member table row by memberID
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public Member SelectMember(int memberID)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    Member ret =
(Member)OracleDataHelperFactory.SelectSingleEntity<Member>(connection,
                        typeof(Member),
                        CommandType.StoredProcedure,
                        "Product.sp_Select_Member_By_MemID",
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemID",
memberID, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter<OracleDbType>(providerName,
":OutputData", null, OracleDbType.RefCursor, ParameterDirection.Output)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// SelectMembers method
        /// - Select Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public List<Member> SelectMembers(string memberName)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    List<Member> ret =
(List<Member>)OracleDataHelperFactory.SelectMultipleEntities<Member>(connection,
                        typeof(Member),
                        CommandType.StoredProcedure,
                        "Product.sp_Select_Members_By_MemName",
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemName",
String.Format("%{0}%", memberName), ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter<OracleDbType>(providerName,
":OutputData", null, OracleDbType.RefCursor, ParameterDirection.Output)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
```

```
                throw ex;
            }
        }

        /// <summary>
        /// SelectNumsOfMembers method
        /// - Select number of Member table rows by memberName
        /// </summary>
        /// <param name="memberName"></param>
        /// <returns></returns>
        public int SelectNumsOfMembers(string memberName)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    OracleParameter resultPrameter =
(OracleParameter)OracleParameterHelperFactory.CreateParameter<OracleDbType>(providerName,
":OutputData", null, OracleDbType.Int32, ParameterDirection.Output);

                    int notResult = Convert.ToInt32(OracleDataHelperFactory.SelectScalar(connection,
                        CommandType.StoredProcedure,
                        "Product.sp_Num_Of_Member_By_MemName",
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemName",
String.Format("%{0}%", memberName), ParameterDirection.Input),
                        resultPrameter
                        ));

                    int ret = Convert.ToInt32(Convert.ToString(resultPrameter.Value));
                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// UpdateMember method
        /// - Update Member table row by member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public bool UpdateMember(Member member)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    OracleParameter resultPrameter =
(OracleParameter)OracleParameterHelperFactory.CreateParameter<OracleDbType>(providerName,
":OutputData", null, OracleDbType.Int32, ParameterDirection.Output);

                    int notResult = OracleDataHelperFactory.Execute(connection,
                        CommandType.StoredProcedure,
                        "Product.sp_Update_Member",
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemName",
member.MemberName, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName,
":MemIsAvailable", ((member.IsAvailable) ? 1 : 0), ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemEmail",
```

```
member.Email, ParameterDirection.Input),
                                OracleParameterHelperFactory.CreateParameter(providerName,
":MemPhoneNumber", member.PhoneNumber, ParameterDirection.Input),
                                OracleParameterHelperFactory.CreateParameter(providerName, ":MemAddress",
member.Address, ParameterDirection.Input),
                                OracleParameterHelperFactory.CreateParameter(providerName, ":MemID",
member.MemberID, ParameterDirection.Input),
                                resultPrameter
                                );

                        int ret = Convert.ToInt32(Convert.ToString(resultPrameter.Value));
                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// DeleteMember() method
            /// - Delete Member table row by memberID
            /// </summary>
            /// <param name="memberID">Member ID</param>
            /// <returns></returns>
            public bool RemoveMember(int memberID)
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        OracleParameter resultPrameter =
(OracleParameter)OracleParameterHelperFactory.CreateParameter<OracleDbType>(providerName,
":OutputData", null, OracleDbType.Int32, ParameterDirection.Output);

                        int notResult = OracleDataHelperFactory.Execute(connection,
                            CommandType.StoredProcedure,
                            "Product.sp_Delete_Member_By_MemID",
                            OracleParameterHelperFactory.CreateParameter(providerName, ":MemID",
memberID, ParameterDirection.Input),
                            resultPrameter
                            );

                        int ret = Convert.ToInt32(Convert.ToString(resultPrameter.Value));
                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
        }
}
```

The stored procedures used in DacMemberSP.cs are as follows.

- sp_Insert_Member.sql

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Database > Stored procedure > Member > sp_ Insert_Member.sql]

```
CREATE OR REPLACE PROCEDURE sp_Insert_Member
(
	MemName IN NVARCHAR2,
	MemIsAvailable IN SMALLINT,
	MemEmail IN NVARCHAR2,
	MemPhoneNumber IN NVARCHAR2,
	MemAddress IN NVARCHAR2,
	OutputData OUT SYS_REFCURSOR
)
IS
	InsertedMemberID INTEGER;
BEGIN
	INSERT INTO Product.Member
	( MemberID, MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate, UpdatedDate )
	VALUES
	( Product.SEQ_Member.NEXTVAL, MemName, MemIsAvailable, MemEmail, MemPhoneNumber,
MemAddress, SYSDATE, NULL );

	SELECT Product.SEQ_Member.CURRVAL
	INTO InsertedMemberID
	FROM DUAL;

	OPEN OutputData FOR
	SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate,
UpdatedDate
	FROM Product.Member
	WHERE MemberID = InsertedMemberID;
END;
```

- sp_Select_Member_By_MemID.sql

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Database > Stored procedure > Member > sp_Select_Member_By_MemID.sql]

```
CREATE OR REPLACE PROCEDURE sp_Select_Member_By_MemID
(
	MemID IN INTEGER,
	OutputData OUT SYS_REFCURSOR
)
IS
BEGIN
	OPEN OutputData FOR
		SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate,
UpdatedDate
	FROM Product.Member
	WHERE MemberID = MemID;
END;
```

- sp_Select_Members_By_MemName.sql

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Database > Stored procedure > Member > sp_Select_Members_By_MemName.sql]

```
CREATE OR REPLACE PROCEDURE sp_Select_Members_By_MemName
(
	MemName IN NVARCHAR2,
	OutputData OUT SYS_REFCURSOR
```

```
)
IS
BEGIN
    OPEN OutputData FOR
        SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate,
UpdatedDate
    FROM Product.Member
    WHERE MemberName like MemName;
END;
```

- sp_Num_Of_Member_By_MemName.sql

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Database > Stored procedure > Member > sp_Num_Of_Member_By_MemName.sql]

```
CREATE OR REPLACE PROCEDURE sp_Num_Of_Member_By_MemName
(
    MemName IN NVARCHAR2,
    OutputData OUT INTEGER
)
IS
BEGIN
    SELECT COUNT(*)
    INTO OutputData
    FROM Product.Member
    WHERE MemberName like MemName;
END;
```

- sp_Update_Member.sql

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Database > Stored procedure > Member > sp_Update_Member.sql]

```
CREATE OR REPLACE PROCEDURE sp_Update_Member
(
    MemName IN NVARCHAR2,
    MemIsAvailable IN SMALLINT,
    MemEmail IN NVARCHAR2,
    MemPhoneNumber IN NVARCHAR2,
    MemAddress IN NVARCHAR2,
    MemID IN INTEGER,
    OutputData OUT INTEGER
)
IS
BEGIN
    OutputData := -1;

    UPDATE Product.Member
    SET MemberName = MemName, IsAvailable = MemIsAvailable, Email = MemEmail,
        PhoneNumber = MemPhoneNumber, Address = MemAddress, UpdatedDate = SYSDATE()
    WHERE MemberID = MemID;

    IF SQL%ROWCOUNT = 1 THEN
        OutputData := 1;
    ELSE
        OutputData := 0;
    END IF;
END;
```

- sp_Delete_Member_By_MemID.sql

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Database > Stored procedure > Member > sp_Delete_Member_By_MemID.sql]

```
CREATE OR REPLACE PROCEDURE sp_Delete_Member_By_MemID
(
	MemID IN INTEGER,
	OutputData OUT INTEGER
)
IS
BEGIN
	OutputData := -1;

	DELETE FROM Product.Member
	WHERE MemberID = MemID;

	IF SQL%ROWCOUNT = 1 THEN
		OutputData := 1;
	ELSE
		OutputData := 0;
	END IF;
END;
```

DacMemberHistory.cs has also changed the DacMemberHistorySP.cs file to use the stored procedure as follows.

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [DataAccess > Memberships.DataAccess > DacMemberHistorySP.cs]

```
using System;
using System.Collections.Generic;
using System.Data;

using Oracle.ManagedDataAccess.Client;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
	public class DacMemberHistorySP
	{
		private static readonly string PROVIDER_NAME = "Oracle.ManagedDataAccess.Client";

		private static readonly string CONNECTION_STRING
			= "Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=59161))(CONNECT_DA
TA=(SERVICE_NAME=XE)));User Id=Product;Password=qwert12345!;";


		private string providerName;
		private OracleConnection connection;


		public DacMemberHistorySP() : this(PROVIDER_NAME, CONNECTION_STRING)
		{
		}
		public DacMemberHistorySP(string providerName, string connectionString)
		{
			this.providerName = providerName;
```

```
            this.connection = new OracleConnection(connectionString);
        }
        public DacMemberHistorySP(string providerName, OracleConnection connection)
        {
            this.providerName = providerName;
            this.connection = connection;
        }


        /// <summary>
        /// InsertMemberHistory method
        /// - Insert MemberHistory table row from member history information
        /// </summary>
        /// <param name="member">Member history information</param>
        /// <returns></returns>
        public MemberHistory InsertMemberHistory(MemberHistory memberHistory)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    MemberHistory ret =
(MemberHistory)OracleDataHelperFactory.SelectSingleEntity<MemberHistory>(connection,
                        typeof(MemberHistory),
                        CommandType.StoredProcedure,
                        "Product.sp_Insert_MemberHistory",
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemID",
memberHistory.MemberID, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemName",
memberHistory.MemberName, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName,
":MemIsSuccess", ((memberHistory.IsSuccess) ? 1 : 0), ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter(providerName, ":MemMessage",
memberHistory.Message, ParameterDirection.Input),
                        OracleParameterHelperFactory.CreateParameter<OracleDbType>(providerName,
":OutputData", null, OracleDbType.RefCursor, ParameterDirection.Output)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// SelectMemberHistories() method
        /// - Select MemberHistory table row by fromDate and toDate
        /// </summary>
        /// <param name="fromDate">From date</param>
        /// <param name="toDate">To date</param>
        /// <returns></returns>
        public List<MemberHistory> SelectMemberHistories(DateTime fromDate, DateTime toDate)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    return
(List<MemberHistory>)OracleDataHelperFactory.SelectMultipleEntities<MemberHistory>(connection,
```

```
                                typeof(MemberHistory),
                                CommandType.StoredProcedure,
                                "Product.sp_Select_MemberHistories_Date",
                                OracleParameterHelperFactory.CreateParameter(providerName, ":FromDate",
fromDate, ParameterDirection.Input),
                                OracleParameterHelperFactory.CreateParameter(providerName, ":ToDate",
toDate, ParameterDirection.Input),
                                OracleParameterHelperFactory.CreateParameter<OracleDbType>(providerName,
":OutputData", null, OracleDbType.RefCursor, ParameterDirection.Output)
                                );
                        }
                    }
                    catch (Exception ex)
                    {
                        throw ex;
                    }
                }
            }
        }
```

The stored procedure in DacMemberHistorySP.cs above is as follows.

- sp_Insert_MemberHistory.sql

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Database > Stored procedure > Member > sp_Insert_MemberHistory.sql]

```
CREATE OR REPLACE PROCEDURE sp_Insert_MemberHistory
(
    MemID IN INTEGER,
    MemName IN NVARCHAR2,
    MemIsSuccess IN SMALLINT,
    MemMessage IN NVARCHAR2,
    OutputData OUT SYS_REFCURSOR
)
IS
    InsertedSeq INTEGER;
BEGIN
    INSERT INTO Product.MemberHistory
    ( Seq, MemberID, MemberName, IsSuccess, Message, InsertedDate )
    VALUES
    ( Product.SEQ_MemberHistory.NEXTVAL, MemID, MemName, MemIsSuccess, MemMessage,
SYSDATE() );

    SELECT Product.SEQ_MemberHistory.CURRVAL
    INTO InsertedSeq
    FROM DUAL;

    OPEN OutputData FOR
    SELECT Seq, MemberID, MemberName, IsSuccess, Message, InsertedDate
    FROM Product.MemberHistory
    WHERE Seq = InsertedSeq;
END;
```

- sp_Select_MemberHistory_Date.sql

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Database > Stored procedure > Member > sp_Select_MemberHistory_Date.sql]

```
CREATE OR REPLACE PROCEDURE sp_Select_MemberHistories_Date
(
	FromDate IN DATE,
	ToDate IN DATE,
	OutputData OUT SYS_REFCURSOR
)
IS
BEGIN
	OPEN OutputData FOR
	SELECT Seq, MemberID, MemberName, IsSuccess, Message, InsertedDate
	FROM Product.MemberHistory
	WHERE InsertedDate >= FromDate AND InsertedDate <= ToDate
	ORDER BY InsertedDate DESC;
END;
```

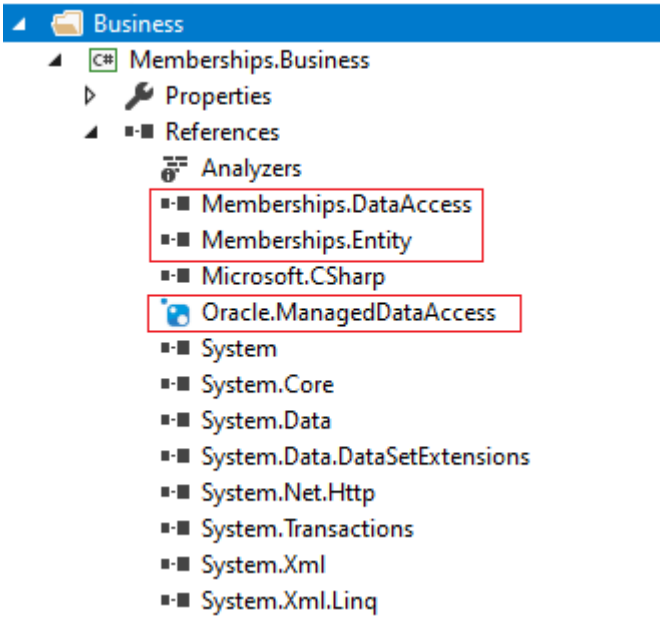
We know that it is better to use the stored procedure in the performance. However, if the stored procedure is used, the DB query part is not included in the source code that we manage, and in case of the actual stored procedure, it needs to be stored in the database server beforehand. This is so uncomfortable. In this case, you can select the method you need. Of course, if you develop a pure query and there is a performance issue, switching to a stored procedure can be a good choice for your development. So we can always choose a step-by-step approach because we have to finish our work at a set time.

Transaction & Business

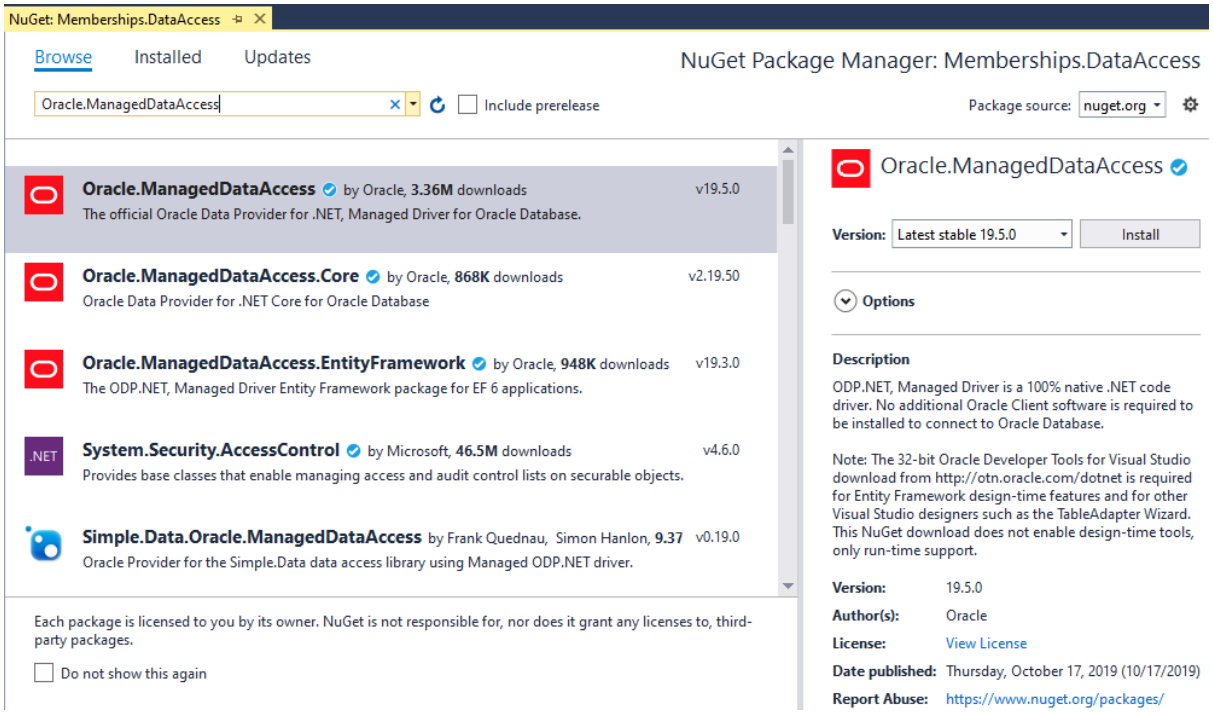
There are many ways to handle transactions in the database. This can be handled either on the DB side, in the .NET code side, or in the COM transaction. Transaction processing on the DB side has seen the handling of transactions in the stored procedure as in "sp_Insert_Member.sql" or "sp_Insert_MemberHistory.sql", and here is how to deal with transactions using TransactionScope in .NET code and you can see as the following code.

We are going to create a new project to separate the transaction and business processing parts from other code. So, we can create the Memberships.Business project through [New> Project] in Visual Studio as below.

And we have added a reference to the Memberships.Entity and Memberships.DataAccess project and Oracle.ManagedDataAccess.dll. Of course, if you have defined Entity classes or Data Access classes in one project, you do not need to add references.



For reference, Oracle.ManagedDataAccess.dll was added by NuGet.



And we will create the BizMemberShip.cs class file and added the code below.

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Business > Memberships.Business > BizMembership.cs]

```
using System;
using System.Collections.Generic;
using System.Transactions;

using Memberships.Entity;
using Memberships.DataAccess;

namespace Memberships.Business
{
    public class BizMemberShip
    {
        private string providerName;
        private string connectionString;


        public BizMemberShip() : this(string.Empty, string.Empty)
        {
        }
        public BizMemberShip(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connectionString = connectionString;
        }


        /// <summary>
        /// CreateMember method
        /// - Create Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member CreateMember(Member member)
        {
```

```
            Member newMember = null;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    bool isInserted = new DacMember(providerName,
connectionString).InsertMember(member);

                    if (isInserted)
                    {
                        int insertedMemberID = new DacMember(providerName,
connectionString).SelectInsertedMemberID();

                        newMember = new DacMember(providerName,
connectionString).SelectMember(insertedMemberID);

                        if (newMember != null)
                        {
                            // Success
                            bool isInsertedHistory = new DacMemberHistory(providerName,
connectionString).InsertMemberHistory(
                                new MemberHistory(newMember.MemberID, newMember.MemberName,
true,
                                    string.Format("Create new member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        newMember.MemberID, newMember.MemberName,
newMember.IsAvailable,
                                        newMember.Email, newMember.PhoneNumber,
newMember.Address)
                                )
                            );

                            if (isInsertedHistory)
                            {
                                int insertedHistorySeq = new DacMemberHistory(providerName,
connectionString).SelectInsertedSeq();

                                MemberHistory mh = new DacMemberHistory(providerName,
connectionString).SelectMemberHistory(insertedHistorySeq);
                            }
                        }
                        else
                        {
                            // Fail
                            bool isInsertedHistory = new DacMemberHistory(providerName,
connectionString).InsertMemberHistory(
                                new MemberHistory(member.MemberID, member.MemberName, false,
                                    string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                        member.MemberName, member.IsAvailable,
                                        member.Email, member.PhoneNumber, member.Address)
                                )
                            );

                            if (isInsertedHistory)
                            {
                                int insertedHistorySeq = new DacMemberHistory(providerName,
connectionString).SelectInsertedSeq();

                                MemberHistory mh = new DacMemberHistory(providerName,
connectionString).SelectMemberHistory(insertedHistorySeq);
                            }
                        }
                    }
                    else
                    {
                        // Fail
```

```
                                bool isInsertedHistory = new DacMemberHistory(providerName,
connectionString).InsertMemberHistory(
                                    new MemberHistory(member.MemberID, member.MemberName, false,
                                        string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                            member.MemberName, member.IsAvailable,
                                            member.Email, member.PhoneNumber, member.Address)
                                    )
                                );

                                if (isInsertedHistory)
                                {
                                    int insertedHistorySeq = new DacMemberHistory(providerName,
connectionString).SelectInsertedSeq();

                                    MemberHistory mh = new DacMemberHistory(providerName,
connectionString).SelectMemberHistory(insertedHistorySeq);
                                }
                            }
                        }
                        catch (Exception ex)
                        {
                            // Fail
                            bool isInsertedHistory = new DacMemberHistory(providerName,
connectionString).InsertMemberHistory(
                                new MemberHistory(member.MemberID, member.MemberName, false,
                                    string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                        member.MemberName, member.IsAvailable,
                                        member.Email, member.PhoneNumber, member.Address)
                                )
                            );

                            if (isInsertedHistory)
                            {
                                int insertedHistorySeq = new DacMemberHistory(providerName,
connectionString).SelectInsertedSeq();

                                MemberHistory mh = new DacMemberHistory(providerName,
connectionString).SelectMemberHistory(insertedHistorySeq);
                            }

                            throw ex;
                        }
                        finally
                        {
                            scope.Complete();
                        }
                    }

                    return newMember;
                }

                /// <summary>
                /// GetMember method
                /// - Get Member table row by memberID
                /// </summary>
                /// <param name="memberID">Member ID</param>
                /// <returns></returns>
                public Member GetMember(int memberID)
                {
                    Member ret = null;

                    try
                    {
                        ret = new DacMember(providerName, connectionString).SelectMember(memberID);
                    }
                    catch (Exception ex)
```

```
                {
                    throw ex;
                }

                return ret;
            }

            /// <summary>
            /// GetMembers method
            /// - Get Member table rows by memberName
            /// </summary>
            /// <param name="memberName">Member name</param>
            /// <returns></returns>
            public List<Member> GetMembers(string memberName)
            {
                List<Member> ret = null;

                try
                {
                    ret = new DacMember(providerName, connectionString).SelectMembers(memberName);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            /// <summary>
            /// GetNumsOfMembers method
            /// - Get number of Member table rows by memberName
            /// </summary>
            /// <param name="memberName">Member name</param>
            /// <returns></returns>
            public int GetNumsOfMembers(string memberName)
            {
                int ret = -1;

                try
                {
                    ret = new DacMember(providerName,
connectionString).SelectNumsOfMembers(memberName);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            /// <summary>
            /// SetMember method
            /// - Set Member table row by member information
            /// </summary>
            /// <param name="member">Member information</param>
            /// <returns></returns>
            public bool SetMember(Member member)
            {
                bool? ret;

                using (TransactionScope scope = new TransactionScope())
                {
                    try
                    {
```

```
                        ret = new DacMember(providerName, connectionString).UpdateMember(member);

                        if (ret != null)
                        {
                            // Success
                            new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                                new MemberHistory(member.MemberID, member.MemberName, true,
                                    string.Format("Update member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        member.MemberID, member.MemberName, member.IsAvailable,
                                        member.Email, member.PhoneNumber, member.Address)
                                )
                            );
                        }
                        else
                        {
                            // Fail
                            new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                                new MemberHistory(member.MemberID, member.MemberName, false,
                                    string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        member.MemberID, member.MemberName, member.IsAvailable,
                                        member.Email, member.PhoneNumber, member.Address)
                                )
                            );
                        }
                    }
                    catch (Exception ex)
                    {
                        // Fail
                        new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    member.MemberID, member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );

                        throw ex;
                    }
                    finally
                    {
                        scope.Complete();
                    }

                    return (ret == true) ? true : false;
                }
            }

            /// <summary>
            /// RemoveMember() method
            /// - Remove Member table row by memberID
            /// </summary>
            /// <param name="memberID">Member ID</param>
            /// <returns></returns>
            public bool RemoveMember(int memberID)
            {
                bool? ret;
                Member removedMember = null;

                using (TransactionScope scope = new TransactionScope())
                {
                    try
                    {
                        removedMember = new DacMember(providerName,
connectionString).SelectMember(memberID);

                        ret = new DacMember(providerName, connectionString).RemoveMember(memberID);
```

```
                    if (ret != null && ret == true)
                    {
                        // Success
                        new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(removedMember.MemberID,
removedMember.MemberName, true,
                                string.Format("Remove member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                    removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(removedMember.MemberID,
removedMember.MemberName, false,
                                string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                    removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                            )
                        );
                    }
                }
                catch (Exception ex)
                {
                    // Fail
                    new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                        new MemberHistory(removedMember.MemberID, removedMember.MemberName,
false,
                            string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                        )
                    );

                    throw ex;
                }
                finally
                {
                    scope.Complete();
                }

                return (ret == true) ? true : false;
            }
        }
    }
}
```

In BizMemberShip.cs, we call pure query data access classes and BizMemberShipSP.cs which calls stored procedure as below.

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Business > Memberships.Business > BizMembershipSP.cs]

```
using System;
using System.Collections.Generic;
using System.Transactions;

using Memberships.Entity;
using Memberships.DataAccess;

namespace Memberships.Business
{
    public class BizMemberShipSP
    {
        private string providerName;
        private string connectionString;


        public BizMemberShipSP() : this(string.Empty, string.Empty)
        {
        }
        public BizMemberShipSP(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connectionString = connectionString;
        }


        /// <summary>
        /// CreateMember method
        /// - Create Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member CreateMember(Member member)
        {
            Member newMember = null;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    newMember = new DacMemberSP(providerName,
connectionString).InsertMember(member);

                    if (newMember != null)
                    {
                        // Success
                        MemberHistory mh = new DacMemberHistorySP(providerName,
connectionString).InsertMemberHistory(
                            new MemberHistory(newMember.MemberID, newMember.MemberName,
true,
                                string.Format("Create new member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    newMember.MemberID, newMember.MemberName,
newMember.IsAvailable,
                                    newMember.Email, newMember.PhoneNumber,
newMember.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        MemberHistory mh = new DacMemberHistorySP(providerName,
connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                    member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
```

```
                                    )
                                );
                            }
                        }
                        catch (Exception ex)
                        {
                            // Fail
                            MemberHistory mh = new DacMemberHistorySP(providerName,
connectionString).InsertMemberHistory(
                                new MemberHistory(member.MemberID, member.MemberName, false,
                                    string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                        member.MemberName, member.IsAvailable,
                                        member.Email, member.PhoneNumber, member.Address)
                                )
                            );

                            throw ex;
                        }
                        finally
                        {
                            scope.Complete();
                        }
                    }

                    return newMember;
                }

                /// <summary>
                /// GetMember method
                /// - Get Member table row by memberID
                /// </summary>
                /// <param name="memberID">Member ID</param>
                /// <returns></returns>
                public Member GetMember(int memberID)
                {
                    Member ret = null;

                    try
                    {
                        ret = new DacMemberSP(providerName, connectionString).SelectMember(memberID);
                    }
                    catch (Exception ex)
                    {
                        throw ex;
                    }

                    return ret;
                }

                /// <summary>
                /// GetMembers method
                /// - Get Member table rows by memberName
                /// </summary>
                /// <param name="memberName">Member name</param>
                /// <returns></returns>
                public List<Member> GetMembers(string memberName)
                {
                    List<Member> ret = null;

                    try
                    {
                        ret = new DacMemberSP(providerName, connectionString).SelectMembers(memberName);
                    }
                    catch (Exception ex)
                    {
                        throw ex;
```

```
            }

            return ret;
        }

        /// <summary>
        /// GetNumsOfMembers method
        /// - Get number of Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public int GetNumsOfMembers(string memberName)
        {
            int ret = -1;

            try
            {
                ret = new DacMemberSP(providerName,
connectionString).SelectNumsOfMembers(memberName);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        /// <summary>
        /// SetMember method
        /// - Set Member table row by member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public bool SetMember(Member member)
        {
            bool? ret;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    ret = new DacMemberSP(providerName, connectionString).UpdateMember(member);

                    if (ret != null)
                    {
                        // Success
                        MemberHistory mh = new DacMemberHistorySP(providerName,
connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, true,
                                string.Format("Update member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    member.MemberID, member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        MemberHistory mh = new DacMemberHistorySP(providerName,
connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    member.MemberID, member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
```

```
                        );
                    }
                }
                catch (Exception ex)
                {
                    // Fail
                    MemberHistory mh = new DacMemberHistorySP(providerName,
connectionString).InsertMemberHistory(
                        new MemberHistory(member.MemberID, member.MemberName, false,
                            string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                member.MemberID, member.MemberName, member.IsAvailable,
                                member.Email, member.PhoneNumber, member.Address)
                        )
                    );

                    throw ex;
                }
                finally
                {
                    scope.Complete();
                }

                return (ret == true) ? true : false;
            }
        }

        /// <summary>
        /// RemoveMember() method
        /// - Remove Member table row by memberID
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public bool RemoveMember(int memberID)
        {
            bool? ret;
            Member removedMember = null;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    removedMember = new DacMemberSP(providerName,
connectionString).SelectMember(memberID);

                    ret = new DacMemberSP(providerName,
connectionString).RemoveMember(memberID);

                    if (ret != null && ret == true)
                    {
                        // Success
                        MemberHistory mh = new DacMemberHistorySP(providerName,
connectionString).InsertMemberHistory(
                            new MemberHistory(removedMember.MemberID,
removedMember.MemberName, true,
                                string.Format("Remove member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                    removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        MemberHistory mh = new DacMemberHistorySP(providerName,
```

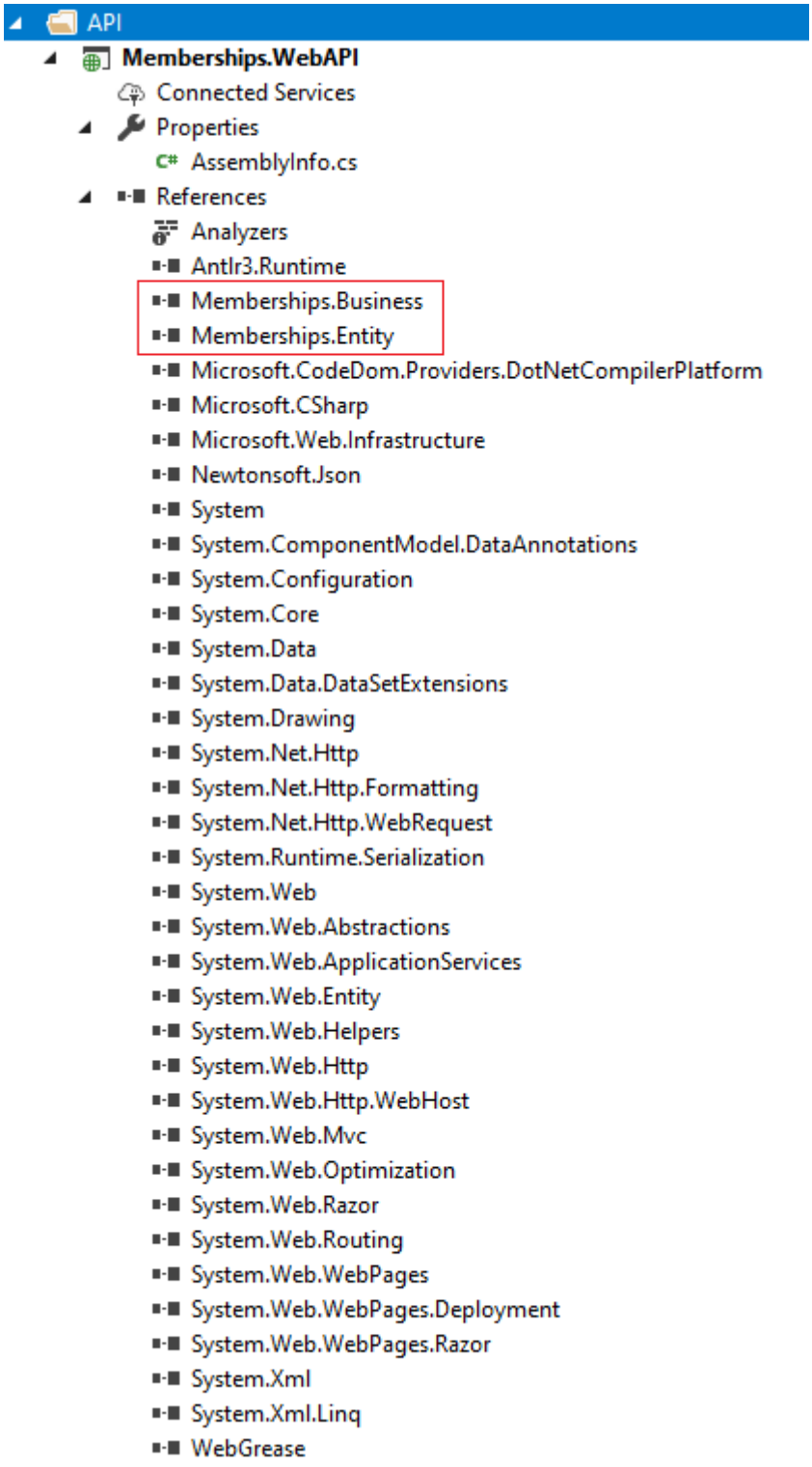
```
connectionString).InsertMemberHistory(
                                    new MemberHistory(removedMember.MemberID,
removedMember.MemberName, false,
                                        string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                            removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                            removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                                    )
                                );
                            }
                        }
                        catch (Exception ex)
                        {
                            // Fail
                            MemberHistory mh = new DacMemberHistorySP(providerName,
connectionString).InsertMemberHistory(
                                new MemberHistory(removedMember.MemberID, removedMember.MemberName,
false,
                                    string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                        removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                                )
                            );

                            throw ex;
                        }
                        finally
                        {
                            scope.Complete();
                        }

                        return (ret == true) ? true : false;
                    }
                }
            }
}
```

API

We will make a web service API by directly utilizing Entity classes, Data Access classes and Business classes defined above. This is similar to the real web service API associated with membership information.

We will create an ASP.NET WebAPI project and check the actual IIS services. We can create a Memberships.WebAPI ASP.NET Web Application project through Visual Studio's [New> Project]. Note that when you create a project, you must create a project by selecting the Web API project template, and in the addition folder and core reference, you need to select the MVC and Web API. This will create a Memberships.WebAPI project as shown below.

And you need to add Memberships.Entity and Memberships.Business as references to the Memberships.WebAPI project. Of course, you can use Memberships.DataAccess as a reference instead of Memberships.Business. We will use Memberships.Business instead of Memberships.DataAccess because the transaction was implemented in Memberships.Business.



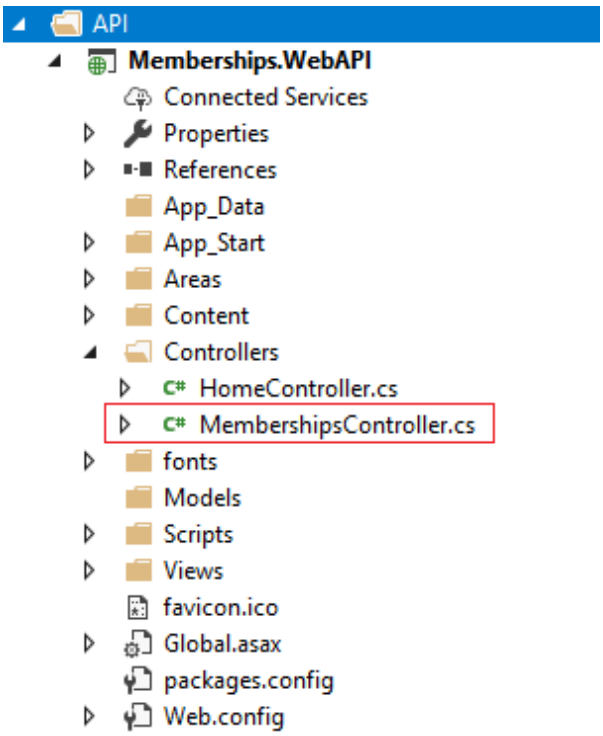
Then, you need to open the Web.config file and add the connection string to connect Oracle Database as follows. Note that the default connection string is defined in the DacMember.cs or DacMemberHistory.cs, DacMemberSP.cs, and DacMemberHistorySP.cs files of the DataAccess class, but we will define and use the DB connection string that is commonly used by the WebAPI service.

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [API> Memberships.API> Web.config]

```
<configuration>
  <connectionStrings>
    <add name="ORACLEDB"
      connectionString="Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=59161))(CONNECT_DA
TA=(SERVICE_NAME=XE)));User Id=Product;Password=qwert12345!;"
      providerName="System.Data.OracleClient" />
  </connectionStrings>
  …
  …
</configuration>
```

Next, you need to add the MembershipsController.cs file to the Controllers folder of the Memberships.WebAPI project. As you can see, you can create an API just by implementing a Controller in a WebAPI project.



Then you need to add the following code to the MembershipsController.cs.

The code below can be found in the sample project.

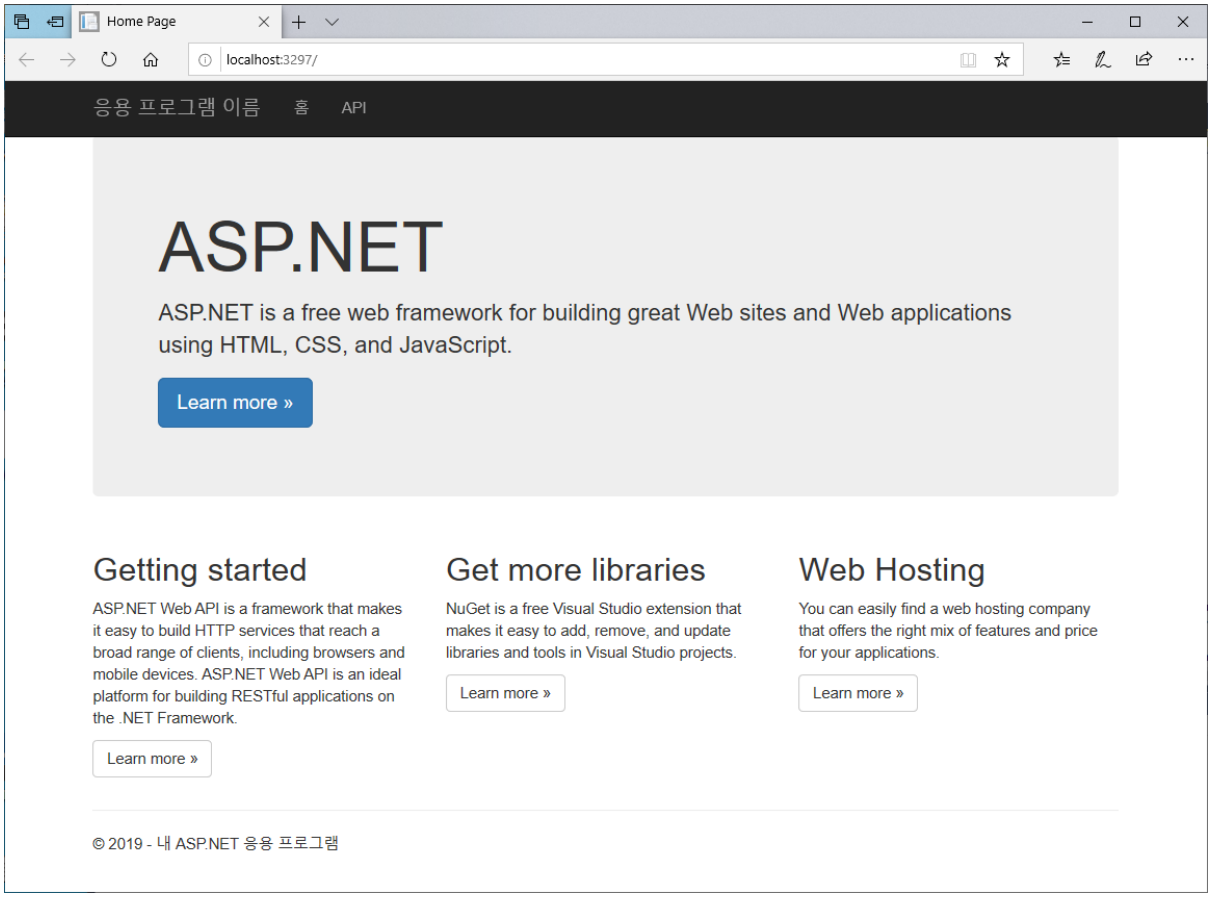
In TaeMFrameworkwithOracle [API> Memberships.WebAPI> Controllers> MembershipsController.cs]

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Web.Http;

using Memberships.Business;
using Memberships.Entity;

namespace Memberships.WebAPI.Controllers
{
    public class MembershipsController : ApiController
    {
        private static string ORACLE_PROVIDER_NAME =
ConfigurationManager.ConnectionStrings["OracleDB"].ProviderName;
        private static string ORACLE_CONN_STR =
```

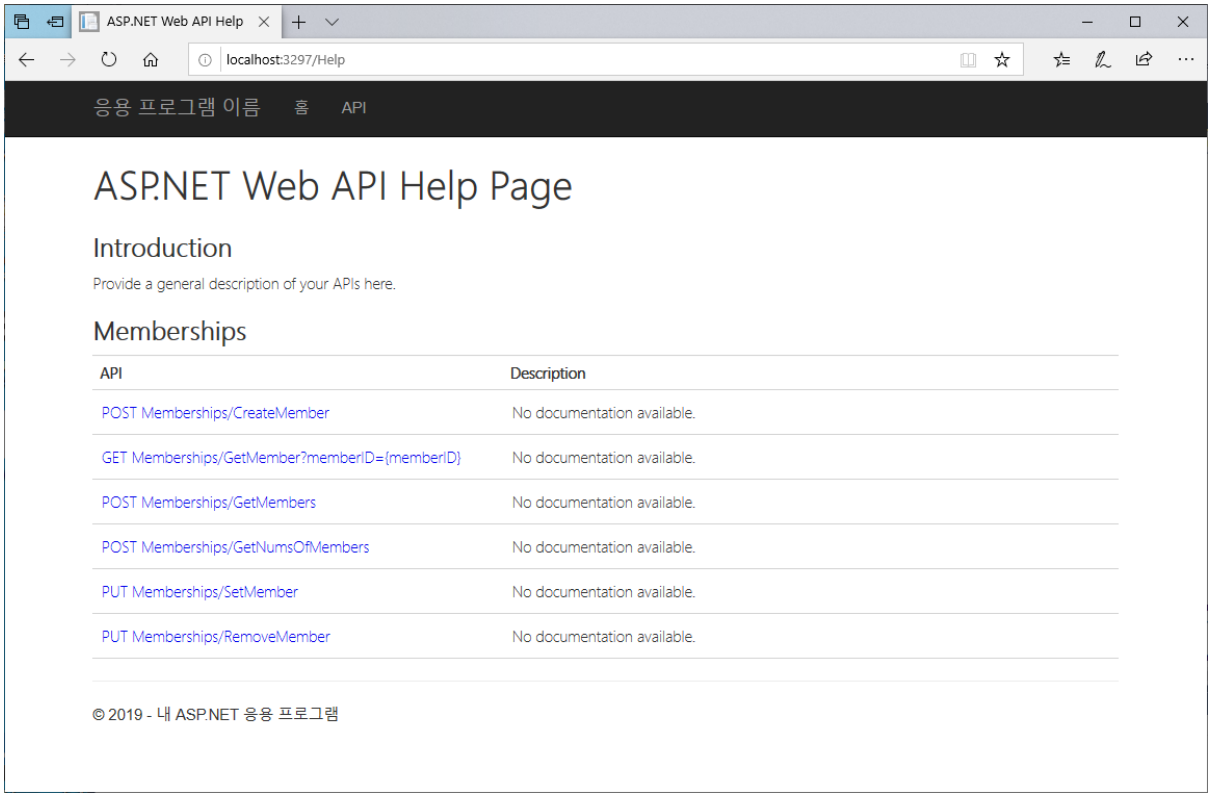
```
ConfigurationManager.ConnectionStrings["OracleDB"].ConnectionString;

        [HttpPost]
        [Route("Memberships/CreateMember")]
        public Member CreateMember(Member member)
        {
            Member ret = null;

            try
            {
                ret = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).CreateMember(member);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        [HttpGet]
        [Route("Memberships/GetMember")]
        public Member GetMember(int memberID)
        {
            Member ret = null;

            try
            {
                ret = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMember(memberID);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        [HttpPost]
        [Route("Memberships/GetMembers")]
        public List<Member> GetMembers([FromBody]string memberName)
        {
            List<Member> ret = null;

            try
            {
                ret = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMembers(memberName);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        [HttpPost]
        [Route("Memberships/GetNumsOfMembers")]
        public int GetNumsOfMembers([FromBody]string memberName)
        {
            int ret = -1;

            try
```

```
                {
                    ret = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetNumsOfMembers(memberName);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            [HttpPut]
            [Route("Memberships/SetMember")]
            public bool SetMember(Member member)
            {
                bool ret = false;

                try
                {
                    ret = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).SetMember(member);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            [HttpPut]
            [Route("Memberships/RemoveMember")]
            public bool RemoveMember([FromBody]int memberID)
            {
                bool ret = false;

                try
                {
                    ret = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).RemoveMember(memberID);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }
        }
}
```

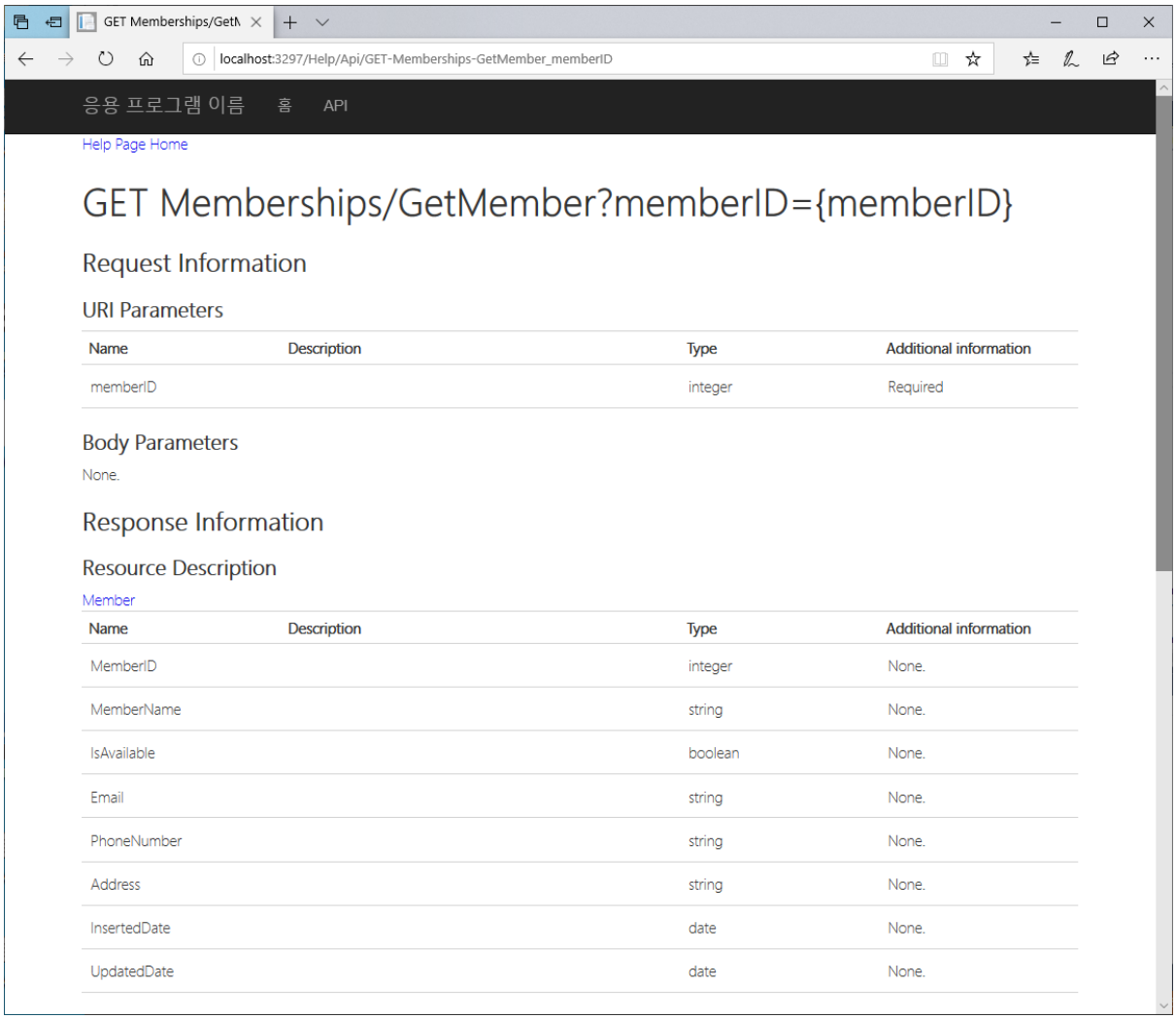
After adding the Controller, build and run the Memberships.WebAPI project. Execution of the Memberships.WebAPI project can be performed on IIS Express, which is the Visual Studio's default WebAPI project execution environment, or on a separate IIS after setting the WebAPI in IIS. The first screen you have run is shown below.

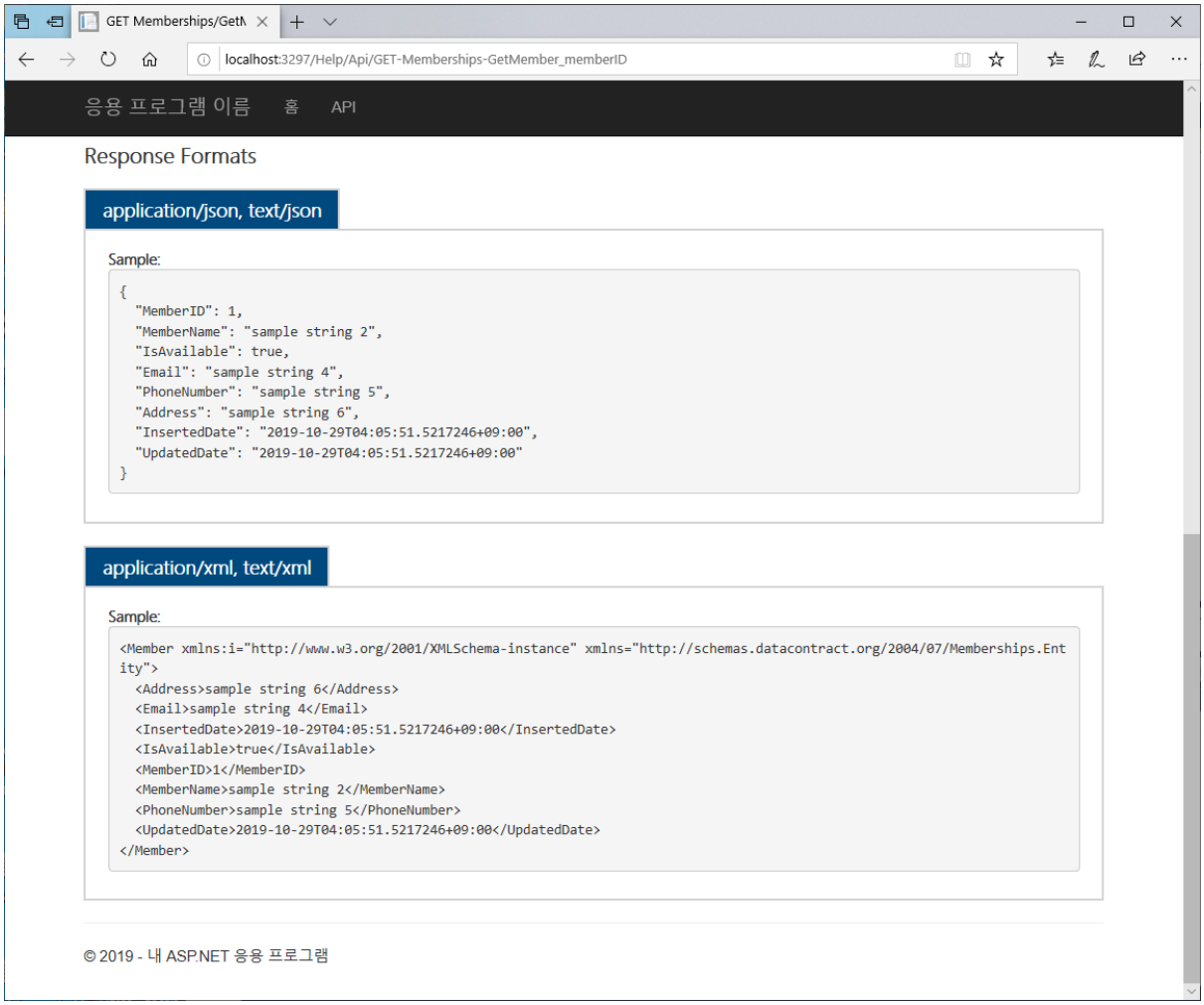


When this screen is displayed, click the "API" item. So you will see a list of the individual methods of the API implemented in the Memberships Controller, as shown below.



Here we will try to click on the GetMember method as GET format. So you can get a detailed description of the GetMember method request format and response format.

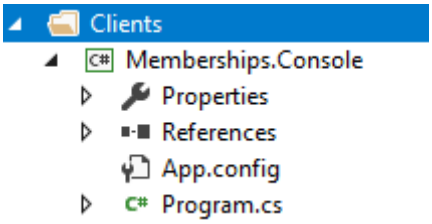



GET Memberships/GetM
localhost:3297/Help/Api/GET-Memberships-GetMember_memberID
응용 프로그램 이름
홈
API
Response Formats
application/json, text/json
Sample:
{
"MemberID": 1,
"MemberName": "sample string 2",
"IsAvailable": true,
"Email": "sample string 4",
"PhoneNumber": "sample string 5",
"Address": "sample string 6",
"InsertedDate": "2019-10-29T04:05:51.5217246+09:00",
"UpdatedDate": "2019-10-29T04:05:51.5217246+09:00"
}
application/xml, text/xml
Sample:
<Member xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/Memberships.Entity">
<Address>sample string 6</Address>
<Email>sample string 4</Email>
<InsertedDate>2019-10-29T04:05:51.5217246+09:00</InsertedDate>
<IsAvailable>true</IsAvailable>
<MemberID>1</MemberID>
<MemberName>sample string 2</MemberName>
<PhoneNumber>sample string 5</PhoneNumber>
<UpdatedDate>2019-10-29T04:05:51.5217246+09:00</UpdatedDate>
</Member>
© 2019 - 내 ASP.NET 응용 프로그램

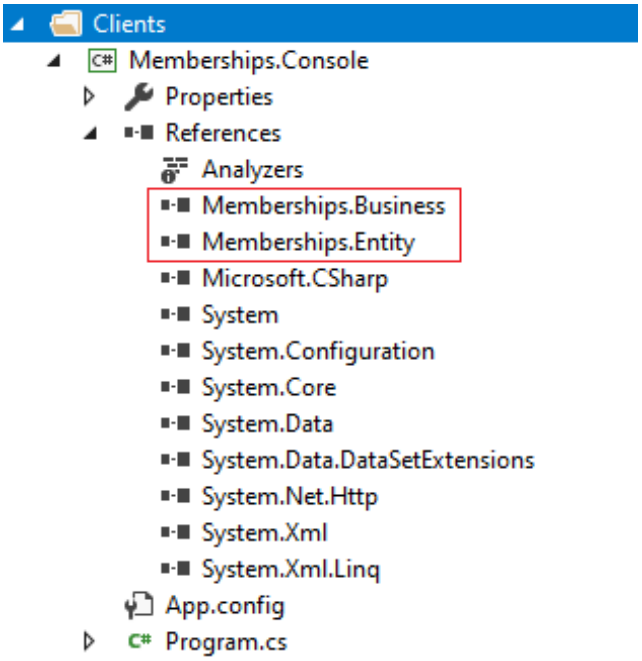
Test client

We made the design of database table, Entity class, Data Access classes, Business classes, and WebAPI. So we will create a client console application to check that the classes we created are working properly.

You can create a Memberships.Console project in Console Application (.NET Framework) project type through [New> Project] in Visual Studio. This will be created the Memberships.Console project as shown below.



You need to add Memberships.Entity, Memberships.Business as references in the Memberships.Console project as follows.



You need to add the DB connection string and WebAPI connection information to the App.config file as follows.

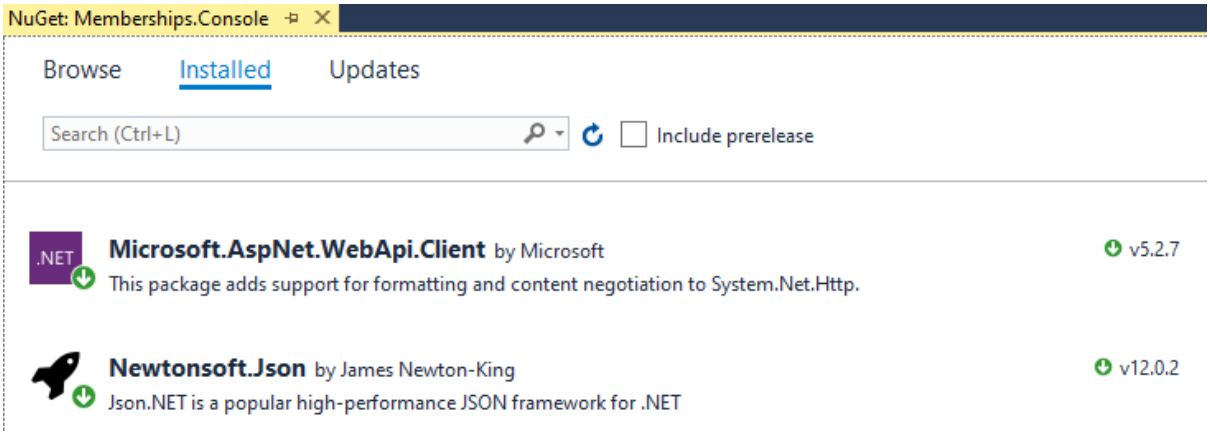
The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Clients> Memberships.Console> App.config]

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7" />
  </startup>
  <connectionStrings>
    <add name="ORACLEDB"
         connectionString="Data
Source=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=59161))(CONNECT_DA
TA=(SERVICE_NAME=XE)));User Id=Product;Password=qwert12345!;"
         providerName="System.Data.OracleClient" />
```

```
  </connectionStrings>
  <appSettings>
    <!-- VS IIS Express Uri -->
    <add key="WebAPI_IIS_Express_Uri" value="http://localhost:3297/"/>
    <!-- End -->

    <!-- IIS Uri -->
    <add key="WebAPI_IIS" value="http://localhost/Memberships.WebAPI.3297/"/>
    <!-- End -->

    <add key="HeaderType" value="application/json" />
  </appSettings>
</configuration>
```

And you need to add the following WebAPICaller.cs file to be used when calling to the WebAPI client. The WebAPICaller.cs uses several extension methods to facilitate WebAPI calls. So you need to install two NuGet packages: Microsoft.AspNet.WebApi.Client and Newtonsoft.Json.



We installed two NuGet packages and defined WebAPICaller.cs as shown below.

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Clients> Memberships.Console> WebAPICaller.cs]

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Net.Http.Formatting;
using System.Threading;

using Memberships.Entity;

namespace Memberships.Console
{
    public class WebAPICaller
    {
        public static Member CallCreateMember(string uri, Member member)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PostAsJsonAsync(
                "Memberships/CreateMember",
                member).Result;
```

```
            if (response.IsSuccessStatusCode)
            {
                Member ret = response.Content.ReadAsAsync<Member>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call CreateMember API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return null;
        }

        public static Member CallGetMember(string uri, int memberID)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.GetAsync(
                string.Format("Memberships/GetMember?memberID={0}", memberID)
                ).Result;

            if (response.IsSuccessStatusCode)
            {
                Member ret = response.Content.ReadAsAsync<Member>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call GetMember API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return null;
        }

        public static List<Member> CallGetMembers(string uri, string memberName)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PostAsJsonAsync(
                "Memberships/GetMembers",
                memberName).Result;

            if (response.IsSuccessStatusCode)
            {
                List<Member> ret = response.Content.ReadAsAsync<List<Member>>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call GetMembers API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return null;
        }

        public static int CallGetGetNumsOfMembers(string uri, string memberName)
```

```
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PostAsJsonAsync(
                "Memberships/GetNumsOfMembers",
                memberName).Result;

            if (response.IsSuccessStatusCode)
            {
                int ret = response.Content.ReadAsAsync<int>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call GetNumsOfMembers API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return -1;
        }

        public static bool CallSetMember(string uri, Member member)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PutAsJsonAsync(
                "Memberships/SetMember",
                member).Result;

            if (response.IsSuccessStatusCode)
            {
                bool ret = response.Content.ReadAsAsync<bool>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call SetMember API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return false;
        }

        public static bool CallRemoveMember(string uri, int memberID)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PutAsJsonAsync(
                "Memberships/RemoveMember",
                memberID).Result;

            if (response.IsSuccessStatusCode)
            {
                bool ret = response.Content.ReadAsAsync<bool>().Result;
                return ret;
            }
            else
```

```
                {
                    System.Console.WriteLine("{0} : {1} ({2})", "Call SetMember API method",
(int)response.StatusCode, response.ReasonPhrase);
                }

                return false;
            }
        }
}
```

Finally, you need to add the following code to the Program.cs to put the code that calls the Business class directly and the WebAPI directly.

The code below can be found in the sample project.

In TaeMFrameworkwithOracle [Clients > Memberships.Console > Program.cs]

```
using System.Collections.Generic;
using System.Configuration;

using Memberships.Business;
using Memberships.Entity;

namespace Memberships.Console
{
    class Program
    {
        private static string ORACLE_PROVIDER_NAME =
ConfigurationManager.ConnectionStrings["ORACLEDB"].ProviderName;
        private static string ORACLE_CONN_STR =
ConfigurationManager.ConnectionStrings["ORACLEDB"].ConnectionString;

        static void Main(string[] args)
        {
            // Call Business class - Pure query
            CallBusiness();

            // Call Business class - Stored procedure
            CallBusinessSP();

            // Call WebAPI
            CallWebAPI();
        }

        private static void CallBusiness()
        {
            // Create new member
            Member newMember = new Member("John Doe", true, "john@taemcorp.net", "080-00-1234-
5678", "anywhere");
            Member createdMemberInDB = new BizMemberShip(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).CreateMember(newMember);
            WriteMember(createdMemberInDB);

            // Select member
            Member selectedMemberInDB = new BizMemberShip(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMember(createdMemberInDB.MemberID);
            WriteMember(selectedMemberInDB);


            // Create new member
            Member newMember2 = new Member("Jane Doe", true, "jane@taemcorp.net", "080-00-0234-
5378", "anywhere");
            Member createdMemberInDB2 = new BizMemberShip(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).CreateMember(newMember2);
```

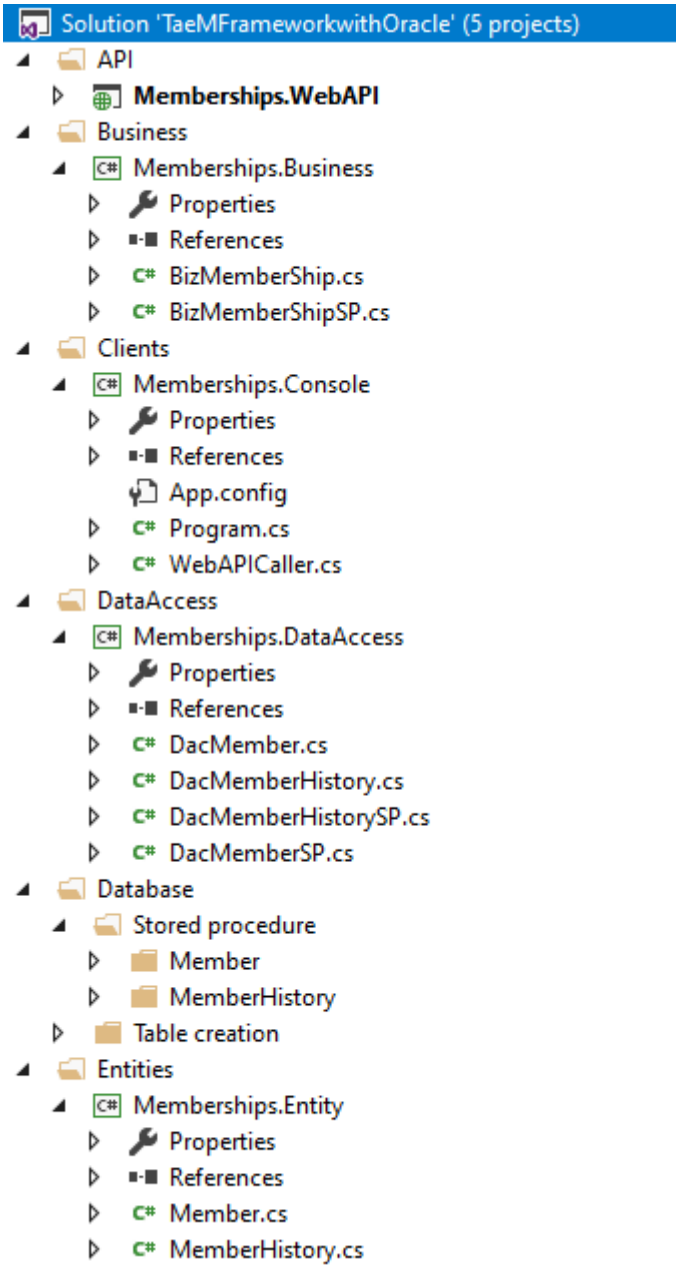
```
            WriteMember(createdMemberInDB2);

            // Get member
            Member selectedMemberInDB2 = new BizMemberShip(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMember(createdMemberInDB2.MemberID);
            WriteMember(selectedMemberInDB2);


            // Get members
            List<Member> currentMembers = new BizMemberShip(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMembers(string.Empty);
            WriteMembers(currentMembers);

            // Get numboer of members
            int numOfCurrentMembers = new BizMemberShip(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetNumsOfMembers(string.Empty);
            WriteCurrentNumberOfMemers(numOfCurrentMembers);

            // Update Member
            createdMemberInDB.MemberName = "John and Jane Doe";
            createdMemberInDB.Address = "John and Jane's home";
            createdMemberInDB.Email = "johnnjane.doe@taemcorp.net";

            bool updateResult = new BizMemberShip(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).SetMember(createdMemberInDB);

            if (updateResult)
            {
                Member updatedMember = new BizMemberShip(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMember(createdMemberInDB.MemberID);
                WriteMember(updatedMember);
            }

            // Delete Member
            bool deleteResult = new BizMemberShip(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).RemoveMember(createdMemberInDB.MemberID);

            List<Member> afterDeletedMembers = new BizMemberShip(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMembers(string.Empty);
            WriteMembers(afterDeletedMembers);
        }

        private static void CallBusinessSP()
        {
            // Create new member
            Member newMember = new Member("John Doe", true, "john@taemcorp.net", "080-00-1234-
5678", "anywhere");
            Member createdMemberInDB = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).CreateMember(newMember);
            WriteMember(createdMemberInDB);

            // Select member
            Member selectedMemberInDB = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMember(createdMemberInDB.MemberID);
            WriteMember(selectedMemberInDB);


            // Create new member
            Member newMember2 = new Member("Jane Doe", true, "jane@taemcorp.net", "080-00-0234-
5378", "anywhere");
            Member createdMemberInDB2 = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).CreateMember(newMember2);
            WriteMember(createdMemberInDB2);

            // Get member
            Member selectedMemberInDB2 = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
```

```
ORACLE_CONN_STR).GetMember(createdMemberInDB2.MemberID);
                WriteMember(selectedMemberInDB2);


                // Get members
                List<Member> currentMembers = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMembers(string.Empty);
                WriteMembers(currentMembers);

                // Get numboer of members
                int numOfCurrentMembers = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetNumsOfMembers(string.Empty);
                WriteCurrentNumberOfMemers(numOfCurrentMembers);

                // Update Member
                createdMemberInDB.MemberName = "John and Jane Doe";
                createdMemberInDB.Address = "John and Jane's home";
                createdMemberInDB.Email = "johnnjane.doe@taemcorp.net";

                bool updateResult = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).SetMember(createdMemberInDB);

                if (updateResult)
                {
                    Member updatedMember = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMember(createdMemberInDB.MemberID);
                    WriteMember(updatedMember);
                }

                // Delete Member
                bool deleteResult = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).RemoveMember(createdMemberInDB.MemberID);

                List<Member> afterDeletedMembers = new BizMemberShipSP(ORACLE_PROVIDER_NAME,
ORACLE_CONN_STR).GetMembers(string.Empty);
                WriteMembers(afterDeletedMembers);
            }

            private static void CallWebAPI()
            {
                string uri = ConfigurationManager.AppSettings["WebAPI_IIS_Express_Uri"];

                // Create new member
                Member newMember = new Member("John Doe", true, "john@taemcorp.net", "080-00-1234-
5678", "anywhere");
                Member createdMemberInDB = WebAPICaller.CallCreateMember(uri, newMember);
                WriteMember(createdMemberInDB);

                // Select member
                Member selectedMemberInDB = WebAPICaller.CallGetMember(uri,
createdMemberInDB.MemberID);
                WriteMember(selectedMemberInDB);


                // Create new member
                Member newMember2 = new Member("Jane Doe", true, "jane@taemcorp.net", "080-00-0234-
5378", "anywhere");
                Member createdMemberInDB2 = WebAPICaller.CallCreateMember(uri, newMember2);
                WriteMember(createdMemberInDB2);

                // Get member
                Member selectedMemberInDB2 = WebAPICaller.CallGetMember(uri,
createdMemberInDB2.MemberID);
                WriteMember(selectedMemberInDB2);
```

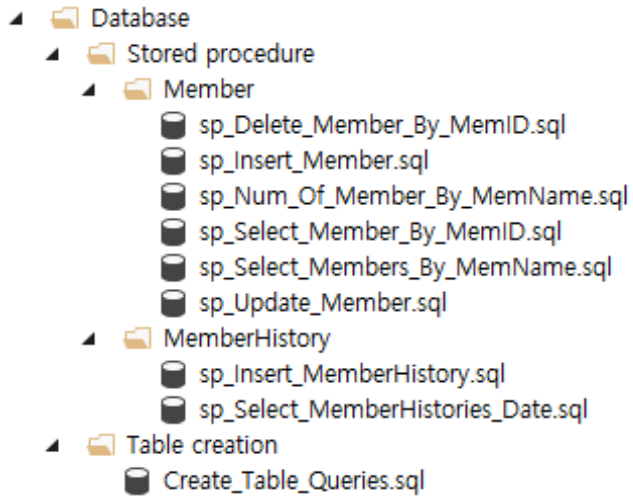
```
            // Get members
            List<Member> currentMembers = WebAPICaller.CallGetMembers(uri, string.Empty);
            WriteMembers(currentMembers);

            // Get numboer of members
            int numOfCurrentMembers = WebAPICaller.CallGetGetNumsOfMembers(uri, string.Empty);
            WriteCurrentNumberOfMemers(numOfCurrentMembers);

            // Update Member
            createdMemberInDB.MemberName = "John and Jane Doe";
            createdMemberInDB.Address = "John and Jane's home";
            createdMemberInDB.Email = "johnnjane.doe@taemcorp.net";

            bool updateResult = WebAPICaller.CallSetMember(uri, createdMemberInDB);

            if (updateResult)
            {
                Member updatedMember = WebAPICaller.CallGetMember(uri,
createdMemberInDB.MemberID);
                WriteMember(updatedMember);
            }

            // Delete Member
            bool deleteResult = WebAPICaller.CallRemoveMember(uri, createdMemberInDB.MemberID);

            if (deleteResult)
            {
                List<Member> afterDeletedMembers = WebAPICaller.CallGetMembers(uri, string.Empty);
                WriteMembers(afterDeletedMembers);
            }
        }

        private static void WriteMember(Member member)
        {
            if (member != null)
            {
                System.Console.WriteLine(
                    @"This member has " +
                    @"[MemberID:{0}] [MemberName:{1}] [IsAvailable:{2}] " +
                    @"[Email:{3}] [PhoneNumber:{4}] [Address:{5}] " +
                    @"[InsertedDate:{6}] [UpdatedDate:{7}]",
                    member.MemberID, member.MemberName, member.IsAvailable,
                    member.Email, member.PhoneNumber, member.Address,
                    member.InsertedDate, member.UpdatedDate);
            }
        }

        private static void WriteMembers(List<Member> members)
        {
            foreach (Member member in members)
                WriteMember(member);
        }

        private static void WriteCurrentNumberOfMemers(int numOfMembers)
        {
            System.Console.WriteLine(
                @"Current number of members : {0}",
                numOfMembers
                );
        }
    }
}
```

Sample solution

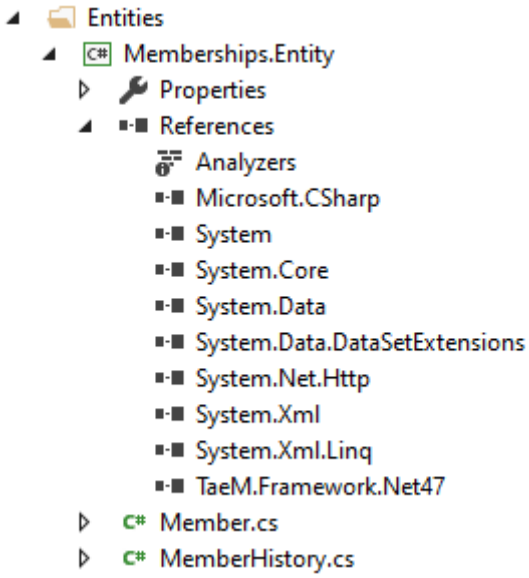
The overall solution that you have created and described so far is as follows. This configuration method is not perfect, but it is a structure we have created for readability and functional separation. So, it is good for you that you can selectively apply it to your own software development.



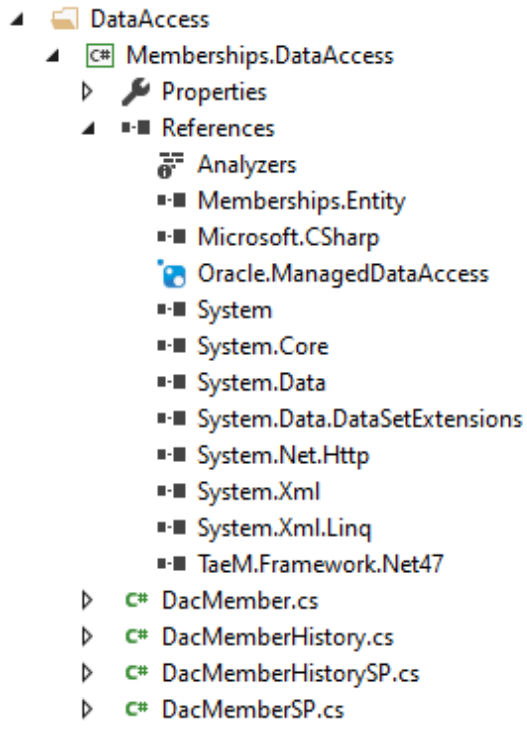
(1) Database solution folder: Database table and stored procedure are included.



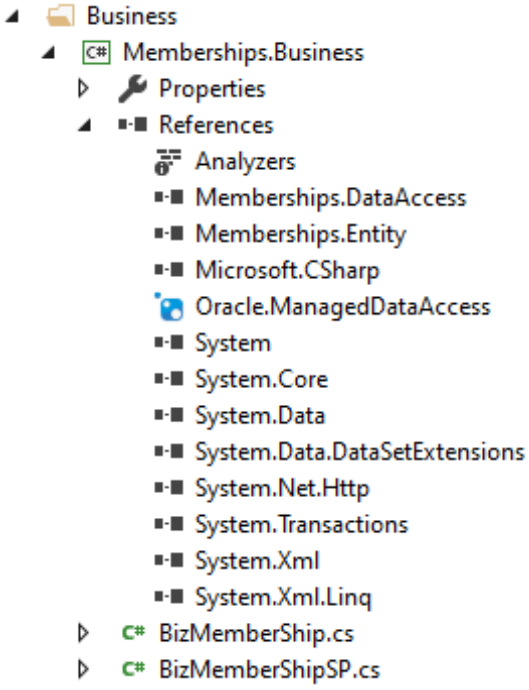
(2) Entities solution folder: Entity classes and projects are included.



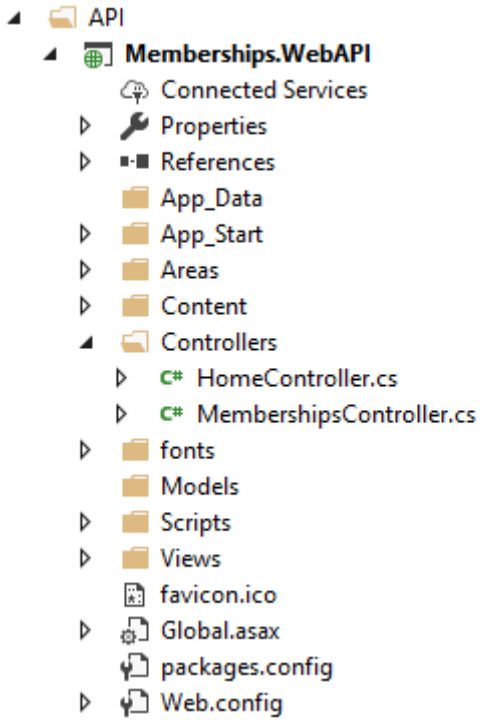
(3) DataAccess solution folder: Data access related classes and projects are included.



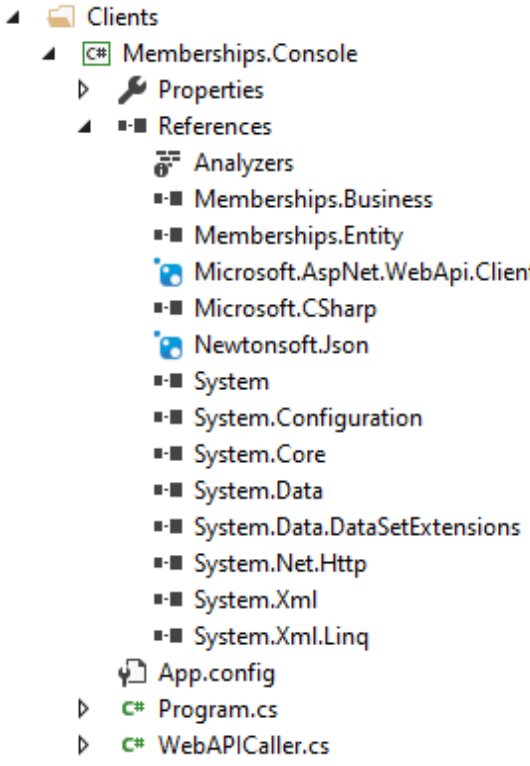
(4) Business solution folder: This includes classes and projects for transaction processing and data processing according to business.



(5) API solution folder: This includes classes and projects for processing API services such as WebAPI.



(6) Clients Solution folder: This includes test console application and other client applications.



How to use TaeM Framework with MySQL

Now, we will examine the process of applying TaeM Framework ORM function in MySQL Database. Since we have created it like the tutorial format, you can follow the steps in order to easily develop software based on MySQL Database by using TaeM Framework ORM function. Note that the below source codes and other samples are made in Microsoft Visual Studio 2017. And you can use other version of Visual Studio because Microsoft Visual Studio supports compatibility to another versions.

Database Table Schema & Entity Class

It's a chicken-and-egg problem. You can create an entity class and create a table of the database later or you can create a table of the database and create entity class later. So, we have created a database table first.

(1) Database Table Schema

We have defined Member and MemberHistory table used in common development. This Member table has member information and This MemberHistory tables keeps records of changes in Member table.

(A) Member table

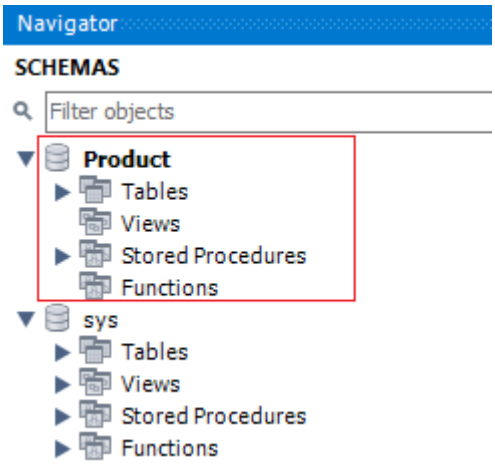
Table Name	Member				
This table has each member information in the system.					
Field Name	**Date type**	**Nullable**	**Default value**	**Primary key**	**Others**
MemberID	INTEGER	NOT NULL		Primary key	AUTO_INCREMENT
MemberName	NVARCHAR(100)	NULL			
IsAvailable	TINYINT(1)	NULL			
Email	NVARCHAR(100)	NULL			
PhoneNumber	NVARCHAR(100)	NULL			
Address	NVARCHAR(1024)	NULL			
InsertedDate	DATETIME	NULL			
UpdatedDate	DATETIME	NULL			

(B) MemberHistory table

Table Name	MemberHistory				
This table has member transaction history. It means that when the member table changes in each field the change history is stored in this table.					
Field Name	**Date type**	**Nullable**	**Default value**	**Primary key**	**Others**
Sequence	INTEGER	NOT NULL		Primary key	AUTO_INCREMENT
MemberID	INTEGER	NOT NULL			
MemberName	NVARCHAR(100)	NULL			
Successful	TINYINT(1)	NULL	0		
Message	NVARCHAR(1024)	NULL			
InsertedDate	DATETIME	NULL			

(C) Table creation query

In MySQL Database, you need to create the Product schema as follows.



Then, the following table creation query is executed to create a Member and a MemberHistory table. You can see this table creation query in the sample project.

In TaeMFrameworkwithMySQL [Database > Table creation > Create_Table_Queries.sql]

```
use Product;

-- Member Table Creation Query
CREATE TABLE Member (
	MemberID INTEGER NOT NULL AUTO_INCREMENT,
	MemberName NVARCHAR(100) NULL,
	IsAvailable TINYINT(1) NULL,
	Email NVARCHAR(100) NULL,
	PhoneNumber NVARCHAR(100) NULL,
	Address NVARCHAR(1024) NULL,
	InsertedDate DATETIME NULL,
	UpdatedDate DATETIME NULL,
	PRIMARY KEY (MemberID)
);

-- Drop table
-- DROP TABLE Member;

CREATE TABLE MemberHistory (
	Sequence INTEGER NOT NULL AUTO_INCREMENT,
	MemberID INTEGER NOT NULL,
	MemberName NVARCHAR(100) NULL,
	Successful TINYINT(1) NULL DEFAULT 0,
	Message NVARCHAR(1024) NULL,
	InsertedDate DATETIME NULL,
	PRIMARY KEY(Sequence)
);

-- Drop table
-- DROP TABLE MemberHistory;
```

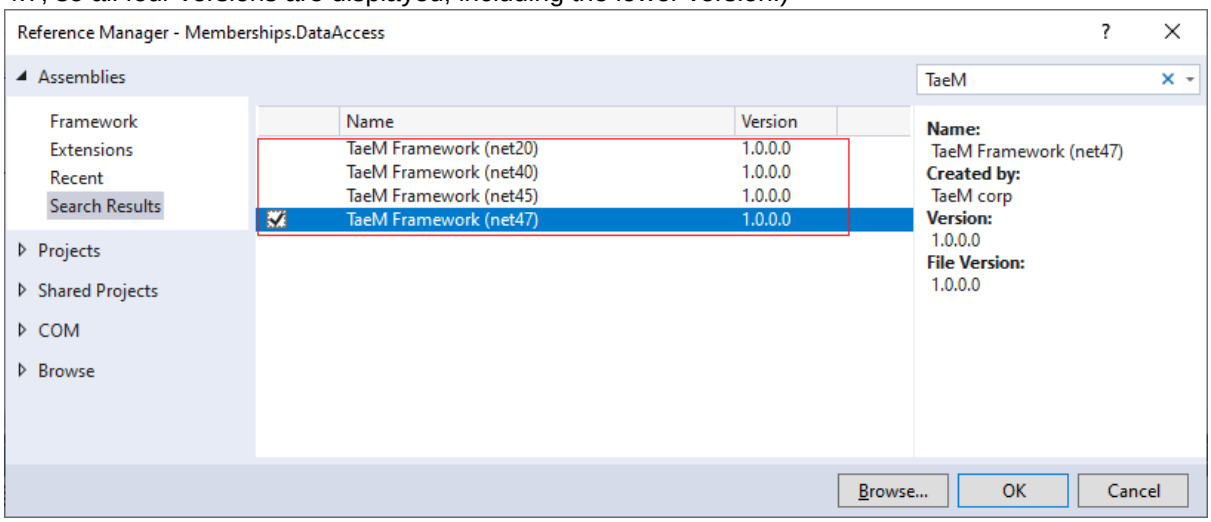
(2) Entity Class

Through the definition of the Member table and the MemberHistory table, each entity classes can be created as follows. In this case, we will map all the fields of the Member table and the MemberHistory table as they are, so we have defined that each classes have same properties with all the fields of the Member table and the MemberHistory table.

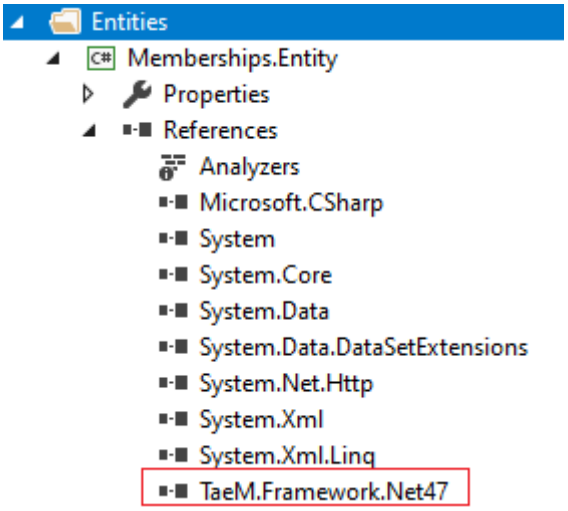
(A) Creation of project and addition of TaeM Framework dll as reference

Before you define the entity class, you must create a project to contain the entity class. You can make a project with Visual Studio that you are using. You can create a Memberships.Entity project through Visual Studio's [New> Project]. The figure below shows the project information after creating the Memberships.Entity project.

Then you can add TaeM.Framework.Net47.dll as a reference to the Memberships.Entity project. Because you installed the TaeM Framework in the previous chapter, here you will see the following "Reference Manager" when you click "Add Reference" in the Memberships.Entity project. When you search "Reference Manager" for "TaeM", you will see four .NET Framework versions of TaeM.Framework.dll. (Note that the target framework for the current project is the .NET Framework 4.7, so all four versions are displayed, including the lower version.)



In this case, select "TaeM Framework (net47)". So, if you add a reference, you should add TaeM.Framework.Net47.dll to the Memberships.Entity project as shown below.



(B) Member entity class

You created a project and added TaeM.Framework.Net47.dll as a reference to the project. Then we will make the Member entity class to the class Memberships.Entity project. In Visual Studio, you can add the Member.cs class file through [New> File].

After you create the Member.cs class file, you need to add the TaeM.Framework.Data.MySql code using the using statement as follows. TaeM.Framework.Data.MySql has an attribute class defined as MySqlDataBinder. This MySqlDataBinder attribute class automatically binds the result of the DB

execution to the field (or variable) or property of the specified class when this attribute is specified for the field (or variable) or property of the class.

```
using System;
using TaeM.Framework.Data.MySql;
```

Then, we can define the Member entity class as shown below. The Member entity class has constructors and a properties. For each property, MySqlDataBinder attribute can be specified as follows. The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Entities> Memberships.Entity> Member.cs]

```
using System;
using TaeM.Framework.Data.MySql;

namespace Memberships.Entity
{
    public class Member
    {
        public Member() : this(string.Empty, false,
            string.Empty, string.Empty, string.Empty)
        {
        }

        public Member(string memberName, bool isAvailable,
            string email, string phoneNumber, string address)
            : this(-1, memberName, isAvailable, email, phoneNumber, address, DateTime.MinValue,
DateTime.MinValue)
        {
        }

        public Member(int memberID, string memberName, bool isAvailable,
            string email, string phoneNumber, string address,
            DateTime insertedDate, DateTime updatedDate)
        {
            this.MemberID = memberID;
            this.MemberName = memberName;
            this.IsAvailable = isAvailable;

            this.Email = email;
            this.PhoneNumber = phoneNumber;
            this.Address = address;

            this.InsertedDate = insertedDate;
            this.UpdatedDate = updatedDate;
        }

        [MySqlDataBinder("MemberID")]
        public int MemberID { get; set; }

        [MySqlDataBinder("MemberName")]
        public string MemberName { get; set; }

        [MySqlDataBinder("IsAvailable")]
        public bool IsAvailable { get; set; }

        [MySqlDataBinder("Email")]
        public string Email { get; set; }

        [MySqlDataBinder("PhoneNumber")]
        public string PhoneNumber { get; set; }

        [MySqlDataBinder("Address")]
        public string Address { get; set; }
```

```
            [MySqlDataBinder("InsertedDate")]
            public DateTime InsertedDate { get; set; }

            [MySqlDataBinder("UpdatedDate")]
            public DateTime UpdatedDate { get; set; }
        }
}
```

In other words, an important part of the above Member entity class definition is the MySqlDataBinder attribute which specifies the field name of the database table (exactly the field name of the query result from the database) and this is specified in each property of the Member entity class for the mapping from each field in the database table. So, MySqlDataBinder attribute is used to map a table field of MySQL Database to each field or property of Member entity class. The field names of the table to be mapped are specified in the MySqlDataBinder attribute parameter.

How to use MySqlDataBinder attribute is as follows.

```
MySqlDataBinder("Field name of the Dababase table to be mapped")
```

In ORM of TaeM Framework, it is not necessary to create separate XML file differently from other ORM solution. Just when declaring entity class, you only need to declare Database data mapping information by declaring attribute. In this case, the MySqlDataBinder attribute is used as the mapping declaration.

(C) MemberHistory Entity class

This time we define the MemberHistory Entity class. You need to create a MemberHistory class, add a using statement, and assign an MySqlDataBinder attribute to the Property or Field. The defined MemberHistory.cs class file is as follows.

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Entities> Memberships.Entity> MemberHistory.cs]

```
using System;
using TaeM.Framework.Data.MySql;

namespace Memberships.Entity
{
    public class MemberHistory
    {
        public MemberHistory()
            : this(-1, string.Empty, false, string.Empty)
        {
        }

        public MemberHistory(int memberID, string memberName, bool isSuccess, string message)
            : this(-1, memberID, memberName, isSuccess, message, DateTime.MinValue)
        {
        }

        public MemberHistory(int seq, int memberID, string memberName,
            bool isSuccess, string message, DateTime insertedDate)
        {
            this.Seq = seq;
            this.MemberID = memberID;
            this.MemberName = memberName;
```

```
                this.IsSuccess = isSuccess;
                this.Message = message;
                this.InsertedDate = insertedDate;
            }

            [MySqlDataBinder("Sequence")]
            public int Seq { get; set; }

            [MySqlDataBinder("MemberID")]
            public int MemberID { get; set; }

            [MySqlDataBinder("MemberName")]
            public string MemberName { get; set; }

            [MySqlDataBinder("Successful")]
            public bool IsSuccess { get; set; }

            [MySqlDataBinder("Message")]
            public string Message { get; set; }

            [MySqlDataBinder("InsertedDate")]
            public DateTime InsertedDate { get; set; }
        }
    }
```

Data Access

Next, we will defines the class of the Data Access part. The classes in the Data Access part connect to the actual database, execute a query on the database, and bind the resultant value to the entity class. Of course, you might get the number of rows affected by the query type in the database, or you might want to import the first column of the first row.

At First, the functions of MySQL Database related helper classes provided by TaeM Framework 1.0.1.0 are as follows.

(1) MySqlDataHelperFactory class

	Method name	Description
1	SelectSingleEntity<T>()	SelectSingleEntity<T> method executes query and return their single row result with a type of return object, an SQL command type, an SQL query, and SQL parameters.
2	SelectMultipleEntities<T>()	SelectMultipleEntities<T> method executes query and return their multiple rows result with a type of return object, an SQL command type, an SQL query, and SQL parameters.
3	SelectScalar()	SelectScalar method executes query and return the value of first row and first column with an SQL command type, an SQL query, and SQL parameters.
4	Execute()	Execute method executes query and return the number of affected row from the SQL query execution result with an SQL command type, an SQL query, and SQL parameters.

(2) MySqlParameterHelperFactory class

	Property name	Description
1	ProviderName	ProviderName is a property used to set or get the DB provider name. For MySQL DB, this value is usually "MySql.Data.MySqlClient", and “MySql.Data.dll” is used.

	Method name	Description
1	CreateParameter()	CreateParameter method creates and returns an object of MySQL DB Parameter with provider name, parameter name, parameter value, direction of parameter.
2	CreateParameterWOProviderName()	CreateParameter method creates and returns an object of MySQL DB Parameter with parameter name, parameter value, the direction of parameter. This should only be used if you have set a value for the ProviderName property.
	CreateParameter<T>()	CreateParameter method creates and returns an object of MySQL DB Parameter with provider name, parameter name, parameter data type, parameter value, direction of parameter.
	CreateParameterWOProviderName<T>()	CreateParameter method creates and returns an object of MySQL DB Parameter with parameter name, parameter value, the direction of parameter. This should only be used if you have set a value for the ProviderName property.

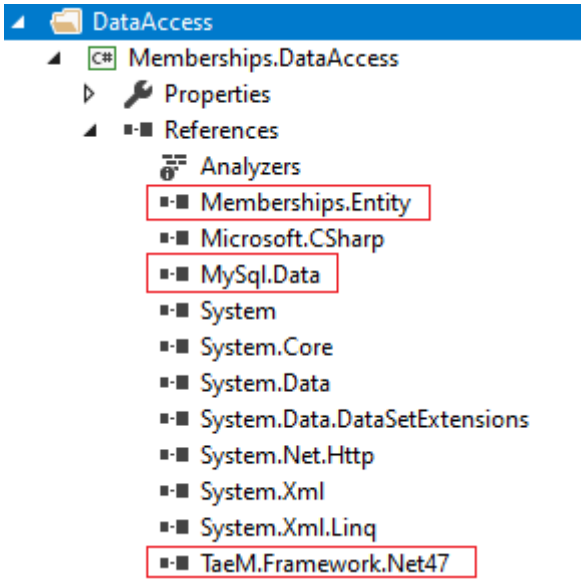
When using MySQL as a database, you can easily query the MySQL Database query by using the above two Helper classes MySqlDataHelperFactory and MySqlParameterHelper class and get the

result. Using these two classes, we will code the CRUD ("Create: Create," "Retrieve or Read," "Update: Update," and "Delete or Destroy", A term that refers to a function that a user interface must have.) function with the database.

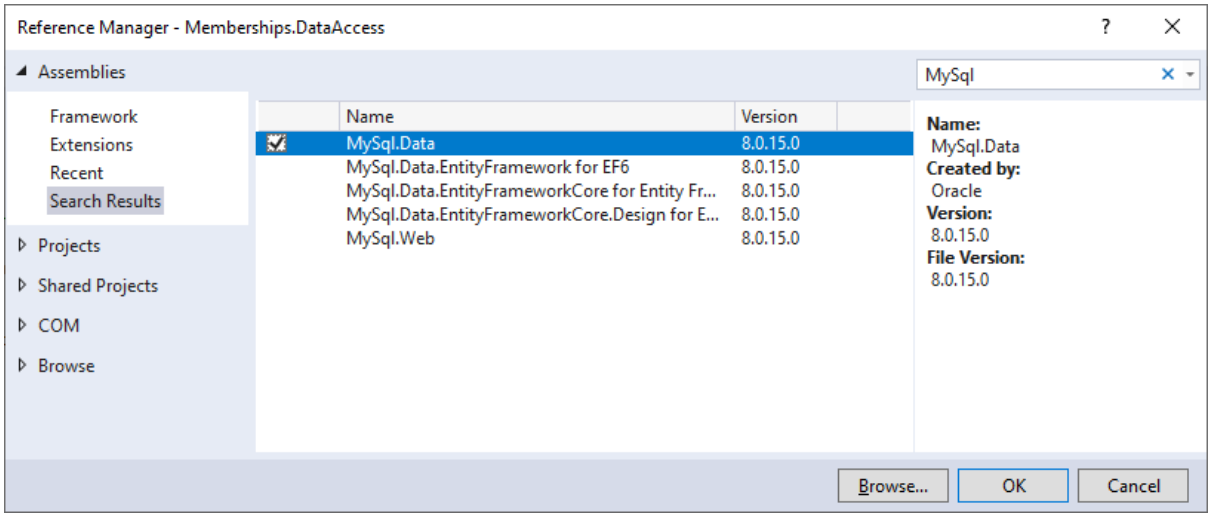
(3) Making the CRUD

Because it is responsible for Data Access, we will define the class with a name of type DacXXX. In this case, we will define a class name DacMember because it is responsible for the data access of the Member. And we will make a Membership.DataAccess project that is a collection of data access classes, and we will define the DacMember class in this project. We can make the Memberships.DataAccess project as shown below through [New> Project] in Visual Studio. In this case, for the readability and similar property is separated into separate projects, but it is possible to put them together in one project according to the coding rules you want or use.

Since we created a new project, we need to add TaeM.Framework.Net47.dll as a reference to our Memberships.DataAccess project. Of course, if you use the same proejct before, you have already added TaeM.Framework.Net47.dll as a reference. Select "Add Reference" and add TaeM.Framework.Net47.dll in "Reference Manager". The following figure shows TaeM.Framework.Net47.dll added to Memberships.DataAccess project. We also added MySql.Data.dll, a .NET Data Provider for MySQL Database, as a reference. This MySql.Data.dll is a database driver for MySQL Database.



For reference, After MySql .NET Connector 8.0.15 installation, we added MySql.Data.dll version 8.0.15.



We made Memberships.DataAccess project and added TaeM.Framework.Net47.dll and MySql.Data.dll to the project as references. Next, we will try adding the DacMember.cs class to the Memberships.DataAccess project. The Addition of the DacMember.cs class file through Visual Studio's [New> File]. After you added the DacMember.cs class file, you need to use the using statement to add the TaeM.Framework.Data code and the Memberships.Entity namespace of the entity class you created before as follows.

TaeM.Framework.Data has the MySqlDataHelperFactory and MySqlParameterHelper classes we discussed earlier. Therefore, we will add code using two helper classes in this class. And Memberships.Entity has a Member.cs class which is the entity class that we defined. We also defined the using statement to use System.Data, MySql.Data and MySql.Data.MySqlClient to use the .NET Data Provider for MySQL Database.

```
using System;
using System.Collections.Generic;
using System.Data;

using MySql.Data;
using MySql.Data.MySqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;
```

Then, we can define the DacMember.cs class as shown below. The DacMember.cs class defines a PROVIDER_NAME that specifies the default DB provider, a CONNECTION_STRING that specifies the default MySQL Database connection string, a providerName with DB provider information, and a connection field with MySqlConnection information, and three constructors and six methods.

```
using System;
using System.Collections.Generic;
using System.Data;

using MySql.Data;
using MySql.Data.MySqlClient;

using Mango.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMember
    {
        private static readonly string CONNECTION_STRING
            = "Server=localhost;Port=32785;Database=Product;Uid=root;Pwd=qwert12345!;ConnectionLifeTime=60;AllowUserVariables=true;";

        private MySqlConnection _connection;

        public DacMember() : this(CONNECTION_STRING)
        {
        }
        public DacMember(string connectionString)
        {
            this._connection = new MySqlConnection(connectionString);
        }
        public DacMember(MySqlConnection connection)
        {
            this._connection = connection;
        }


        /// <summary> InsertMember method - Insert Member table row from member informat ...
        public Member InsertMember(Member member)...

        /// <summary> SelectMember method - Select Member table row by memberID
        public Member SelectMember(int memberID)...

        /// <summary> SelectMembers method - Select Member table rows by memberName
        public List<Member> SelectMembers(string memberName)...

        /// <summary> SelectNumsOfMembers method - Select number of Member table rows by ...
        public int SelectNumsOfMembers(string memberName)...

        /// <summary> UpdateMember method - Update Member table row by member informatio ...
        public bool UpdateMember(Member member)...

        /// <summary> DeleteMember() method - Delete Member table row by memberID
        public bool RemoveMember(int memberID)...
    }
}
```

The above six methods need to execute the following kind of query on the Member table and return the result value.

	Query type	Method name
1	Retrieve a single data row	SelectMember(…)
2	Retrieve multiple data row	SelectMembers(…)
3	Creation of the data row	InsertMember(…)
4	Modification of the data row	UpdateMember(…)
5	Deletion of the data row	DeleteMember(…)
6	Retrieve the number of data row	SelectNumsOfMembers(…)

To implement the method according to the above query type, you can use the following method defined in the MySqlDataHelperFactory class mentioned earlier in this section.

	Query type	Method name	The Method of MySqlDataHelperFactory class
1	Retrieve a single data row	SelectMember(…)	Using the SelectSingleEntity <T> () method
2	Retrieve multiple data row	SelectMembers(…)	Using the SelectMultipleEntities <T> () method
3	Creation of the data row	InsertMember(…)	Using the SelectScalar () method and the SelectSingleEntity <T> () method
4	Modification of the data row	UpdateMember(…)	Using the Execute () method

5	Deletion of the data row	DeleteMember(…)	Using the Execute () method
6	Retrieve the number of data row	SelectNumsOfMembers(…)	Using the SelectScalar () method

Particularly here is the InsertMember (...) method, we used SelectScalar () method and the SelectSingleEngity <T> () method. For the InsertMember (...) method, you must use the Execute () method to execute a query that only generates a row in the Member table. But, if you want to create a row in the Member table and get the row information of the generated table, you need to use the SelectScalar () method and SelectSingleEngity <T> () method to get the data row type as the result.

The MySqlParameterHelper class is a class that generalizes the parameters of a database and has one property and four methods.

So, we can make the DacMember.cs class which implements CRUD using MySqlDataHelperFactory and MySqlParameterHelper class as follows.

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [DataAccess> Memberships.DataAccess> DacMember.cs]

```
using System;
using System.Collections.Generic;
using System.Data;

using MySql.Data;
using MySql.Data.MySqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMember
    {
        private static readonly string PROVIDER_NAME = "MySql.Data.MySqlClient";

        private static readonly string CONNECTION_STRING
            =
"Server=localhost;Port=32785;Database=Product;Uid=root;Pwd=qwert12345!;ConnectionLifeTime=60;AllowUserVariables=true;";


        private string providerName;
        private MySqlConnection connection;


        public DacMember() : this(PROVIDER_NAME, CONNECTION_STRING)
        {
        }
        public DacMember(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connection = new MySqlConnection(connectionString);
        }
        public DacMember(string providerName, MySqlConnection connection)
        {
            this.providerName = providerName;
            this.connection = connection;
        }


        /// <summary>
```

```
        /// InsertMember method
        /// - Insert Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member InsertMember(Member member)
        {
            try
            {
                using (connection)
                {
                    if (string.IsNullOrEmpty(MySqlParameterHelperFactory.ProviderName))
                        MySqlParameterHelperFactory.ProviderName = providerName;

                    connection.Open();

                    int insertedMemberID = Convert.ToInt32(
                        MySqlDataHelperFactory.SelectScalar(connection,
                            CommandType.Text,
                            @"INSERT INTO Product.Member " +
                            @"( MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate,
UpdatedDate ) " +
                            @"VALUES " +
                            @"( @MemberName, @IsAvailable, @Email, @PhoneNumber, @Address,
SYSDATE(), NULL ); " +
                            @" " +
                            @"SELECT LAST_INSERT_ID(); ",

MySqlParameterHelperFactory.CreateParameterWOProviderName("@MemberName",
member.MemberName, ParameterDirection.Input),

MySqlParameterHelperFactory.CreateParameterWOProviderName("@IsAvailable", member.IsAvailable,
ParameterDirection.Input),
                            MySqlParameterHelperFactory.CreateParameterWOProviderName("@Email",
member.Email, ParameterDirection.Input),

MySqlParameterHelperFactory.CreateParameterWOProviderName("@PhoneNumber",
member.PhoneNumber, ParameterDirection.Input),

MySqlParameterHelperFactory.CreateParameterWOProviderName("@Address", member.Address,
ParameterDirection.Input)
                            )
                        );

                    Member ret =
(Member)MySqlDataHelperFactory.SelectSingleEntity<Member>(connection,
                        typeof(Member),
                        CommandType.Text,
                        @"SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber,
Address, InsertedDate, UpdatedDate " +
                        @"FROM Product.Member " +
                        @"WHERE MemberID = @MemberID; ",

MySqlParameterHelperFactory.CreateParameterWOProviderName<MySqlDbType>("@MemberID",
insertedMemberID, MySqlDbType.Int32, ParameterDirection.Input)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
```

```
        /// <summary>
        /// SelectMember method
        /// - Select Member table row by memberID
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public Member SelectMember(int memberID)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    Member ret =
(Member)MySqlDataHelperFactory.SelectSingleEntity<Member>(connection,
                        typeof(Member),
                        CommandType.Text,
                        @"SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber,
Address, InsertedDate, UpdatedDate " +
                        @"FROM Product.Member " +
                        @"WHERE MemberID = @MemberID; ",
                        MySqlParameterHelperFactory.CreateParameter<MySqlDbType>(providerName,
"@MemberID", memberID, MySqlDbType.Int32, ParameterDirection.Input)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// SelectMembers method
        /// - Select Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public List<Member> SelectMembers(string memberName)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    List<Member> ret =
(List<Member>)MySqlDataHelperFactory.SelectMultipleEntities<Member>(connection,
                        typeof(Member),
                        CommandType.Text,
                        @"SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber,
Address, InsertedDate, UpdatedDate " +
                        @"FROM Product.Member " +
                        @"WHERE MemberName like @MemberName; ",
                        MySqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", String.Format("%{0}%", memberName), ParameterDirection.Input)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
```

```
                throw ex;
            }
        }

        /// <summary>
        /// SelectNumsOfMembers method
        /// - Select number of Member table rows by memberName
        /// </summary>
        /// <param name="memberName"></param>
        /// <returns></returns>
        public int SelectNumsOfMembers(string memberName)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    int ret = Convert.ToInt32(MySqlDataHelperFactory.SelectScalar(connection,
                        CommandType.Text,
                        @"SELECT COUNT(*) " +
                        @"FROM Product.Member " +
                        @"WHERE MemberName like @MemberName; ",
                        MySqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", String.Format("%{0}%", memberName), ParameterDirection.Input)
                        ));

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// UpdateMember method
        /// - Update Member table row by member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public bool UpdateMember(Member member)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    int ret = MySqlDataHelperFactory.Execute(connection,
                        CommandType.Text,
                        @"UPDATE Product.Member " +
                        @"SET MemberName = @MemberName, IsAvailable = @IsAvailable, Email =
@Email, " +
                        @"    PhoneNumber = @PhoneNumber, Address = @Address, UpdatedDate =
SYSDATE() " +
                        @"WHERE MemberID = @MemberID; ",
                        MySqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", member.MemberName, ParameterDirection.Input),
                        MySqlParameterHelperFactory.CreateParameter(providerName, "@IsAvailable",
member.IsAvailable, ParameterDirection.Input),
                        MySqlParameterHelperFactory.CreateParameter(providerName, "@Email",
member.Email, ParameterDirection.Input),
                        MySqlParameterHelperFactory.CreateParameter(providerName,
"@PhoneNumber", member.PhoneNumber, ParameterDirection.Input),
```

```
                                MySqlParameterHelperFactory.CreateParameter(providerName, "@Address",
member.Address, ParameterDirection.Input),
                                MySqlParameterHelperFactory.CreateParameter(providerName, "@MemberID",
member.MemberID, ParameterDirection.Input)
                                );

                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// DeleteMember() method
            /// - Delete Member table row by memberID
            /// </summary>
            /// <param name="memberID">Member ID</param>
            /// <returns></returns>
            public bool RemoveMember(int memberID)
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        int ret = MySqlDataHelperFactory.Execute(connection,
                            CommandType.Text,
                            @"DELETE FROM Product.Member " +
                            @"WHERE MemberID = @MemberID; ",
                            MySqlParameterHelperFactory.CreateParameter<MySqlDbType>(providerName,
"@MemberID", memberID, MySqlDbType.Int32, ParameterDirection.Input)
                            );

                        return (ret == 1) ? true : false;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
        }
}
```

Here you can define the DacMemberHistory class. We have created the DacMemberHistory class, added the using statement, and defined methods in the DacMemberHistory.cs class as shown below.

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [DataAccess > Memberships.DataAccess > DacMemberHistory.cs]

```
using System;
using System.Collections.Generic;
using System.Data;

using MySql.Data;
using MySql.Data.MySqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;
```

```
namespace Memberships.DataAccess
{
	public class DacMemberHistory
	{
		private static readonly string PROVIDER_NAME = "MySql.Data.MySqlClient";

		private static readonly string CONNECTION_STRING
			=
"Server=localhost;Port=32785;Database=Product;Uid=root;Pwd=qwert12345!;ConnectionLifeTime=60;AllowUserVariables=true;";


		private string providerName;
		private MySqlConnection connection;


		public DacMemberHistory() : this(PROVIDER_NAME, CONNECTION_STRING)
		{
		}
		public DacMemberHistory(string providerName, string connectionString)
		{
			this.providerName = providerName;
			this.connection = new MySqlConnection(connectionString);
		}
		public DacMemberHistory(string providerName, MySqlConnection connection)
		{
			this.providerName = providerName;
			this.connection = connection;
		}


		/// <summary>
		/// InsertMemberHistory method
		/// - Insert MemberHistory table row from member history information
		/// </summary>
		/// <param name="member">Member history information</param>
		/// <returns></returns>
		public MemberHistory InsertMemberHistory(MemberHistory memberHistory)
		{
			try
			{
				using (connection)
				{
					connection.Open();

					int insertedSequence = Convert.ToInt32(
						MySqlDataHelperFactory.SelectScalar(connection,
							CommandType.Text,
							@"INSERT INTO Product.MemberHistory " +
							@"( MemberID, MemberName, Successful, Message, InsertedDate ) " +
							@"VALUES " +
							@"( @MemberID, @MemberName, @Successful, @Message, SYSDATE() );
" +
							@" " +
							@"SELECT LAST_INSERT_ID(); ",

MySqlParameterHelperFactory.CreateParameter<MySqlDbType>(providerName, "@MemberID",
memberHistory.MemberID, MySqlDbType.Int32, ParameterDirection.Input),
							MySqlParameterHelperFactory.CreateParameter(providerName,
"@MemberName", memberHistory.MemberName, ParameterDirection.Input),
							MySqlParameterHelperFactory.CreateParameter(providerName,
"@Successful", memberHistory.IsSuccess, ParameterDirection.Input),
							MySqlParameterHelperFactory.CreateParameter(providerName,
"@Message", memberHistory.Message, ParameterDirection.Input)
							)
```

```
                                );

                        MemberHistory ret =
(MemberHistory)MySqlDataHelperFactory.SelectSingleEntity<MemberHistory>(connection,
                                typeof(MemberHistory),
                                CommandType.Text,
                                @"SELECT Sequence, MemberID, MemberName, Successful, Message,
InsertedDate " +
                                @"FROM Product.MemberHistory " +
                                @"WHERE Sequence = @Sequence; ",
                                MySqlParameterHelperFactory.CreateParameter<MySqlDbType>(providerName,
"@Sequence", insertedSequence, MySqlDbType.Int32, ParameterDirection.Input)
                                );

                        return ret;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// SelectMemberHistories() method
            /// - Select MemberHistory table row by fromDate and toDate
            /// </summary>
            /// <param name="fromDate">From date</param>
            /// <param name="toDate">To date</param>
            /// <returns></returns>
            public List<MemberHistory> SelectMemberHistories(DateTime fromDate, DateTime toDate)
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        return
(List<MemberHistory>)MySqlDataHelperFactory.SelectMultipleEntities<MemberHistory>(connection,
                                typeof(MemberHistory),
                                CommandType.Text,
                                @"SELECT Sequence, MemberID, MemberName, Successful, Message,
InsertedDate " +
                                @"FROM Product.MemberHistory " +
                                @"WHERE InsertedDate >= @FromDate AND InsertedDate <= @ToDate " +
                                @"ORDER BY InsertedDate DESC; ",
                                MySqlParameterHelperFactory.CreateParameter(providerName, "@FromDate",
fromDate, ParameterDirection.Input),
                                MySqlParameterHelperFactory.CreateParameter(providerName, "@ToDate",
toDate, ParameterDirection.Input)
                                );
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
        }
}
```

(4) Using stored procedure

In the DacMember.cs and DacMemberHistory.cs class definitions above, we created a pure query in text format to execute the query directly to MySQL Database. However, you can use the stored procedure as shown below for performance.

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [DataAccess > Memberships.DataAccess > DacMemberSP.cs]

```
using System;
using System.Collections.Generic;
using System.Data;

using MySql.Data;
using MySql.Data.MySqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMemberSP
    {
        private static readonly string PROVIDER_NAME = "MySql.Data.MySqlClient";

        private static readonly string CONNECTION_STRING
            =
"Server=localhost;Port=32785;Database=Product;Uid=root;Pwd=qwert12345!;ConnectionLifeTime=60;AllowUserVariables=true;";


        private string providerName;
        private MySqlConnection connection;


        public DacMemberSP() : this(PROVIDER_NAME, CONNECTION_STRING)
        {
        }
        public DacMemberSP(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connection = new MySqlConnection(connectionString);
        }
        public DacMemberSP(string providerName, MySqlConnection connection)
        {
            this.providerName = providerName;
            this.connection = connection;
        }


        /// <summary>
        /// InsertMember method
        /// - Insert Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member InsertMember(Member member)
        {
            try
            {
                using (connection)
                {
                    if (string.IsNullOrEmpty(MySqlParameterHelperFactory.ProviderName))
                        MySqlParameterHelperFactory.ProviderName = providerName;

                    connection.Open();
```

```
                        int insertedMemberID = Convert.ToInt32(
                            MySqlDataHelperFactory.SelectScalar(connection,
                                CommandType.StoredProcedure,
                                "Product.sp_Memberships_Insert_Member",

MySqlParameterHelperFactory.CreateParameterWOProviderName("@MemName", member.MemberName,
ParameterDirection.Input),

MySqlParameterHelperFactory.CreateParameterWOProviderName("@MemIsAvailable", member.IsAvailable,
ParameterDirection.Input),

MySqlParameterHelperFactory.CreateParameterWOProviderName("@MemEmail", member.Email,
ParameterDirection.Input),

MySqlParameterHelperFactory.CreateParameterWOProviderName("@MemPhoneNumber",
member.PhoneNumber, ParameterDirection.Input),

MySqlParameterHelperFactory.CreateParameterWOProviderName("@MemAddress", member.Address,
ParameterDirection.Input)
                                )
                            );

                        Member ret =
(Member)MySqlDataHelperFactory.SelectSingleEntity<Member>(connection,
                            typeof(Member),
                            CommandType.StoredProcedure,
                            "Product.sp_Memberships_Select_Member_By_MemberID",

MySqlParameterHelperFactory.CreateParameterWOProviderName<MySqlDbType>("@MemID",
insertedMemberID, MySqlDbType.Int32, ParameterDirection.Input)
                            );

                        return ret;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// SelectMember method
            /// - Select Member table row by memberID
            /// </summary>
            /// <param name="memberID">Member ID</param>
            /// <returns></returns>
            public Member SelectMember(int memberID)
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        Member ret =
(Member)MySqlDataHelperFactory.SelectSingleEntity<Member>(connection,
                            typeof(Member),
                            CommandType.StoredProcedure,
                            "Product.sp_Memberships_Select_Member_By_MemberID",
                            MySqlParameterHelperFactory.CreateParameter<MySqlDbType>(providerName,
"@MemID", memberID, MySqlDbType.Int32, ParameterDirection.Input)
                            );

                        return ret;
```

```
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// SelectMembers method
        /// - Select Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public List<Member> SelectMembers(string memberName)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    List<Member> ret =
(List<Member>)MySqlDataHelperFactory.SelectMultipleEntities<Member>(connection,
                        typeof(Member),
                        CommandType.StoredProcedure,
                        "Product.sp_Memberships_Select_Members_By_MemberName",
                        MySqlParameterHelperFactory.CreateParameter(providerName, "@MemName",
String.Format("%{0}%", memberName), ParameterDirection.Input)
                        );

                    return ret;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// SelectNumsOfMembers method
        /// - Select number of Member table rows by memberName
        /// </summary>
        /// <param name="memberName"></param>
        /// <returns></returns>
        public int SelectNumsOfMembers(string memberName)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    int ret = Convert.ToInt32(MySqlDataHelperFactory.SelectScalar(connection,
                        CommandType.StoredProcedure,
                        "Product.sp_Memberships_Select_Num_Of_Members_By_MemberName",
                        MySqlParameterHelperFactory.CreateParameter(providerName, "@MemName",
String.Format("%{0}%", memberName), ParameterDirection.Input)
                        ));

                    return ret;
                }
            }
            catch (Exception ex)
            {
```

```
                throw ex;
            }
        }

        /// <summary>
        /// UpdateMember method
        /// - Update Member table row by member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public bool UpdateMember(Member member)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    int ret = MySqlDataHelperFactory.Execute(connection,
                        CommandType.StoredProcedure,
                        "Product.sp_Memberships_Update_Member",
                        MySqlParameterHelperFactory.CreateParameter(providerName, "@MemName",
member.MemberName, ParameterDirection.Input),
                        MySqlParameterHelperFactory.CreateParameter(providerName,
"@MemIsAvailable", member.IsAvailable, ParameterDirection.Input),
                        MySqlParameterHelperFactory.CreateParameter(providerName, "@MemEmail",
member.Email, ParameterDirection.Input),
                        MySqlParameterHelperFactory.CreateParameter(providerName,
"@MemPhoneNumber", member.PhoneNumber, ParameterDirection.Input),
                        MySqlParameterHelperFactory.CreateParameter(providerName, "@MemAddress",
member.Address, ParameterDirection.Input),
                        MySqlParameterHelperFactory.CreateParameter(providerName, "@MemID",
member.MemberID, ParameterDirection.Input)
                        );

                    return (ret == 1) ? true : false;
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// DeleteMember() method
        /// - Delete Member table row by memberID
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public bool RemoveMember(int memberID)
        {
            try
            {
                using (connection)
                {
                    connection.Open();

                    int ret = MySqlDataHelperFactory.Execute(connection,
                        CommandType.StoredProcedure,
                        "Product.sp_Memberships_Delete_Member_By_MemberID",
                        MySqlParameterHelperFactory.CreateParameter<MySqlDbType>(providerName,
"@MemID", memberID, MySqlDbType.Int32, ParameterDirection.Input)
                        );

                    return (ret == 1) ? true : false;
```

```
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
        }
    }
```

The stored procedures used in DacMemberSP.cs are as follows.

- sp_Memberships_Insert_Member.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Database > Stored procedure > Member > sp_Memberships_Insert_Member.sql]

```
delimiter ;
delimiter //
CREATE PROCEDURE sp_Memberships_Insert_Member(
    MemName NVARCHAR(100),
    MemIsAvailable TINYINT(1),
    MemEmail NVARCHAR(100),
    MemPhoneNumber NVARCHAR(100),
    MemAddress NVARCHAR(1024))

    BEGIN
        INSERT INTO Product.Member
        ( MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate, UpdatedDate )
        VALUES
        ( MemName, MemIsAvailable, MemEmail, MemPhoneNumber, MemAddress, SYSDATE(), NULL );

        SELECT @InsertedMemberID := LAST_INSERT_ID();
    END;
```

- sp_Memberships_Select_Member_By_MemberID.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Database > Stored procedure > Member > sp_Memberships_Select_Member_By_MemberID.sql]

```
delimiter ;
delimiter //
CREATE PROCEDURE sp_Memberships_Select_Member_By_MemberID(MemID INTEGER)
    BEGIN
        SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate,
UpdatedDate
        FROM Product.Member
        WHERE MemberID = MemID;
    END;
```

- sp_Memberships_Select_Members_By_MemberName.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Database > Stored procedure > Member > sp_Memberships_Select_Members_By_MemberName.sql]

```
delimiter ;
delimiter //
CREATE PROCEDURE sp_Memberships_Select_Members_By_MemberName(MemName
NVARCHAR(100))
    BEGIN
        SELECT MemberID, MemberName, IsAvailable, Email, PhoneNumber, Address, InsertedDate,
UpdatedDate
        FROM Product.Member
        WHERE MemberName like MemName;
    END;
```

- sp_Memberships_Select_Num_Of_Members_By_MemberName.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Database > Stored procedure > Member > sp_Memberships_Select_Num_Of_Members_By_MemberName.sql]

```
delimiter ;
delimiter //
CREATE PROCEDURE sp_Memberships_Select_Num_Of_Members_By_MemberName(MemName
NVARCHAR(100))
    BEGIN
        SELECT COUNT(*)
        FROM Product.Member
        WHERE MemberName like MemName;
    END;
```

- sp_Memberships_Update_Member.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Database > Stored procedure > Member > sp_Memberships_Update_Member.sql]

```
delimiter ;
delimiter //
CREATE PROCEDURE sp_Memberships_Update_Member(
    MemName NVARCHAR(100),
    MemIsAvailable TINYINT(1),
    MemEmail NVARCHAR(100),
    MemPhoneNumber NVARCHAR(100),
    MemAddress NVARCHAR(1024),
    MemID INTEGER)
    BEGIN
        UPDATE Product.Member
        SET MemberName = MemName, IsAvailable = MemIsAvailable, Email = MemEmail,
            PhoneNumber = MemPhoneNumber, Address = MemAddress, UpdatedDate = SYSDATE()
        WHERE MemberID = MemID;
    END;
```

- sp_Memberships_Delete_Member_By_MemberID.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Database > Stored procedure > Member > sp_Memberships_Delete_Member_By_MemberID.sql]

```
delimiter ;
delimiter //
CREATE PROCEDURE sp_Memberships_Delete_Member_By_MemberID(MemID INTEGER)
    BEGIN
```

```
        DELETE FROM Product.Member
        WHERE MemberID = MemID;
    END;
```

DacMemberHistory.cs has also changed the DacMemberHistorySP.cs file to use the stored procedure as follows.

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [DataAccess > Memberships.DataAccess > DacMemberHistorySP.cs]

```
using System;
using System.Collections.Generic;
using System.Data;

using MySql.Data;
using MySql.Data.MySqlClient;

using TaeM.Framework.Data;
using Memberships.Entity;

namespace Memberships.DataAccess
{
    public class DacMemberHistorySP
    {
        private static readonly string PROVIDER_NAME = "MySql.Data.MySqlClient";

        private static readonly string CONNECTION_STRING
            =
"Server=localhost;Port=32785;Database=Product;Uid=root;Pwd=qwert12345!;ConnectionLifeTime=60;AllowUserVariables=true;";


        private string providerName;
        private MySqlConnection connection;


        public DacMemberHistorySP() : this(PROVIDER_NAME, CONNECTION_STRING)
        {
        }
        public DacMemberHistorySP(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connection = new MySqlConnection(connectionString);
        }
        public DacMemberHistorySP(string providerName, MySqlConnection connection)
        {
            this.providerName = providerName;
            this.connection = connection;
        }

        /// <summary>
        /// InsertMemberHistory method
        /// - Insert MemberHistory table row from member history information
        /// </summary>
        /// <param name="member">Member history information</param>
        /// <returns></returns>
        public MemberHistory InsertMemberHistory(MemberHistory memberHistory)
        {
            try
            {
                using (connection)
                {
```

```
                        connection.Open();

                        int insertedSequence = Convert.ToInt32(
                            MySqlDataHelperFactory.SelectScalar(connection,
                                CommandType.StoredProcedure,
                                "Product.sp_Memberships_Insert_MemberHistory",
                                MySqlParameterHelperFactory.CreateParameter(providerName, "@MemID",
memberHistory.MemberID, ParameterDirection.Input),
                                MySqlParameterHelperFactory.CreateParameter(providerName,
"@MemName", memberHistory.MemberName, ParameterDirection.Input),
                                MySqlParameterHelperFactory.CreateParameter(providerName,
"@MemSuccessful", memberHistory.IsSuccess, ParameterDirection.Input),
                                MySqlParameterHelperFactory.CreateParameter(providerName,
"@MemMessage", memberHistory.Message, ParameterDirection.Input)
                                )
                            );

                        MemberHistory ret =
(MemberHistory)MySqlDataHelperFactory.SelectSingleEntity<MemberHistory>(connection,
                            typeof(MemberHistory),
                            CommandType.StoredProcedure,
                            "Product.sp_Memberships_Select_MemberHistory_By_Sequence",
                            MySqlParameterHelperFactory.CreateParameter<MySqlDbType>(providerName,
"@Seq", insertedSequence, MySqlDbType.Int32, ParameterDirection.Input));

                        return ret;
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }

            /// <summary>
            /// SelectMemberHistories() method
            /// - Select MemberHistory table row by fromDate and toDate
            /// </summary>
            /// <param name="fromDate">From date</param>
            /// <param name="toDate">To date</param>
            /// <returns></returns>
            public List<MemberHistory> SelectMemberHistories(DateTime fromDate, DateTime toDate)
            {
                try
                {
                    using (connection)
                    {
                        connection.Open();

                        return
(List<MemberHistory>)MySqlDataHelperFactory.SelectMultipleEntities<MemberHistory>(connection,
                            typeof(MemberHistory),
                            CommandType.StoredProcedure,
                            "Product.sp_Memberships_Select_MemberHistories_By_FromToDate",
                            MySqlParameterHelperFactory.CreateParameter(providerName, "@FromDate",
fromDate, ParameterDirection.Input),
                            MySqlParameterHelperFactory.CreateParameter(providerName, "@ToDate",
toDate, ParameterDirection.Input)
                            );
                    }
                }
                catch (Exception ex)
                {
                    throw ex;
                }
            }
```

```
    }
}
```

The stored procedure in DacMemberHistorySP.cs above is as follows.

- sp_Memberships_Insert_MemberHistory.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Database > Stored procedure > Member > sp_Memberships_Insert_MemberHistory.sql]

```
delimiter ;
delimiter //
CREATE PROCEDURE sp_Memberships_Insert_MemberHistory(
    MemID INTEGER,
    MemName NVARCHAR(100),
    MemSuccessful TINYINT(1),
    MemMessage NVARCHAR(100))
    BEGIN
        INSERT INTO Product.MemberHistory
        ( MemberID, MemberName, Successful, Message, InsertedDate )
        VALUES
        ( MemID, MemName, MemSuccessful, MemMessage, SYSDATE() );

        SELECT @InsertedSequence := LAST_INSERT_ID();
    END;
```

- sp_Memberships_Select_MemberHistory_By_Sequence.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Database > Stored procedure > Member > sp_Memberships_Select_MemberHistory_By_Sequence.sql]

```
delimiter ;
delimiter //
CREATE PROCEDURE sp_Memberships_Select_MemberHistory_By_Sequence(Seq INTEGER)
    BEGIN
        SELECT Sequence, MemberID, MemberName, Successful, Message, InsertedDate
        FROM Product.MemberHistory
        WHERE Sequence = Seq;
    END;
```

- sp_Memberships_Select_MemberHistory_By_FromToDate.sql

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Database > Stored procedure > Member > sp_Memberships_Select_MemberHistory_By_FromToDate.sql]

```
delimiter ;
delimiter //
CREATE PROCEDURE sp_Memberships_Select_MemberHistories_By_FromToDate(
    FromDate DATETIME,
    ToDate DATETIME)
    BEGIN
        SELECT Sequence, MemberID, MemberName, Successful, Message, InsertedDate
        FROM Product.MemberHistory
        WHERE InsertedDate >= FromDate AND InsertedDate <= ToDate
        ORDER BY InsertedDate DESC;
    END;
```

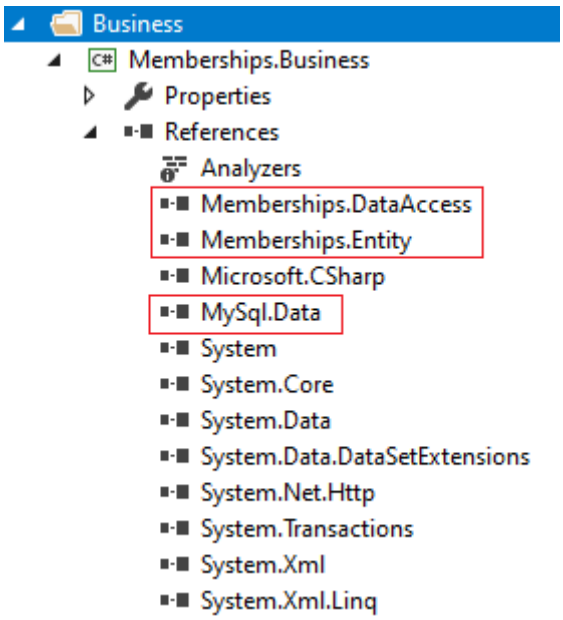
We know that it is better to use the stored procedure in the performance. However, if the stored procedure is used, the DB query part is not included in the source code that we manage, and in case of the actual stored procedure, it needs to be stored in the database server beforehand. This is so uncomfortable. In this case, you can select the method you need. Of course, if you develop a pure query and there is a performance issue, switching to a stored procedure can be a good choice for your development. So we can always choose a step-by-step approach because we have to finish our work at a set time.

Transaction & Business

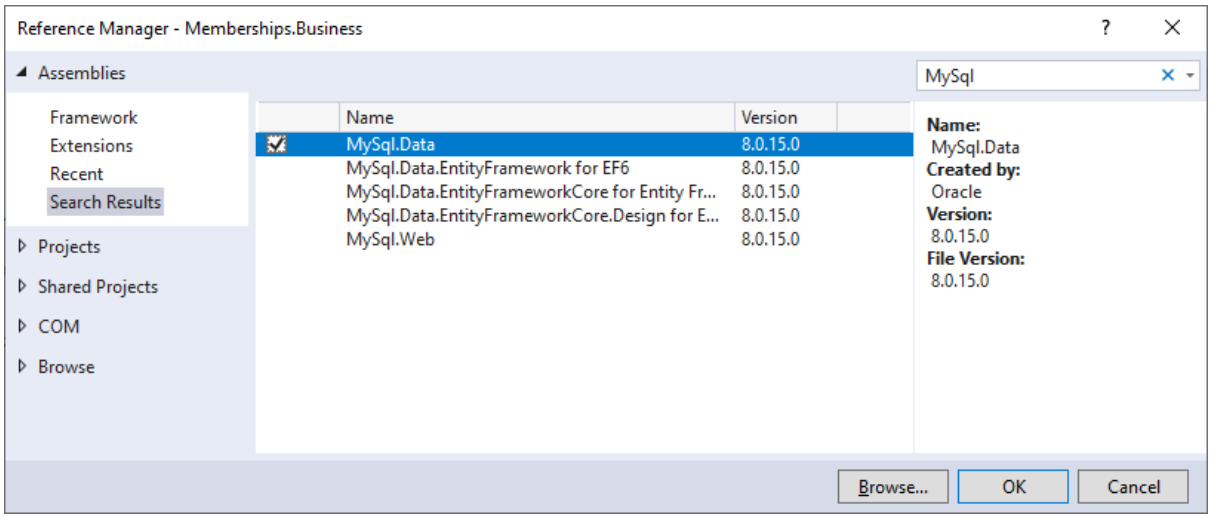
There are many ways to handle transactions in the database. This can be handled either on the DB side, in the .NET code side, or in the COM transaction. Transaction processing on the DB side has seen the handling of transactions in the stored procedure as in the "sp_Memberships_Insert_Member.sql" file or the "sp_Memberships_Insert_MemberHistory.sql" file, and here is how to deal with transactions using TransactionScope in .NET code and you can see as the following code.

We are going to create a new project to separate the transaction and business processing parts from other code. So, we can create the Memberships.Business project through [New> Project] in Visual Studio as below.

We have added a reference to the Memberships.Entity and Memberships.DataAccess and MySql.Data.dll as shown below. Of course, if you have defined Entity classes or Data Access classes in one project, you do not need to add references.



For reference, After MySql .NET Connector 8.0.15 installation, we added MySql.Data.dll version 8.0.15.



And we will create the BizMemberShip.cs class file and added the code below.

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Business > Memberships.Business > BizMembership.cs]

```
using System;
using System.Collections.Generic;
using System.Transactions;

using Memberships.Entity;
using Memberships.DataAccess;

namespace Memberships.Business
{
    public class BizMemberShip
    {
        private string providerName;
        private string connectionString;


        public BizMemberShip() : this(string.Empty, string.Empty)
        {
        }
        public BizMemberShip(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connectionString = connectionString;
        }


        /// <summary>
        /// CreateMember method
        /// - Create Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member CreateMember(Member member)
        {
            Member newMember = null;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    newMember = new DacMember(providerName,
connectionString).InsertMember(member);

                    if (newMember != null)
                    {
                        // Success
                        MemberHistory mh = new DacMemberHistory(providerName,
connectionString).InsertMemberHistory(
                            new MemberHistory(newMember.MemberID, newMember.MemberName,
true,
                                string.Format("Create new member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    newMember.MemberID, newMember.MemberName,
newMember.IsAvailable,
                                    newMember.Email, newMember.PhoneNumber,
newMember.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        MemberHistory mh = new DacMemberHistory(providerName,
connectionString).InsertMemberHistory(
```

```
                                new MemberHistory(member.MemberID, member.MemberName, false,
                                    string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                        member.MemberName, member.IsAvailable,
                                        member.Email, member.PhoneNumber, member.Address)
                                )
                            );
                        }
                    }
                    catch (Exception ex)
                    {
                        // Fail
                        new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                    member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );

                        throw ex;
                    }
                    finally
                    {
                        scope.Complete();
                    }
                }

                return newMember;
            }

            /// <summary>
            /// GetMember method
            /// - Get Member table row by memberID
            /// </summary>
            /// <param name="memberID">Member ID</param>
            /// <returns></returns>
            public Member GetMember(int memberID)
            {
                Member ret = null;

                try
                {
                    ret = new DacMember(providerName, connectionString).SelectMember(memberID);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            /// <summary>
            /// GetMembers method
            /// - Get Member table rows by memberName
            /// </summary>
            /// <param name="memberName">Member name</param>
            /// <returns></returns>
            public List<Member> GetMembers(string memberName)
            {
                List<Member> ret = null;

                try
                {
                    ret = new DacMember(providerName, connectionString).SelectMembers(memberName);
                }
```

```
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            /// <summary>
            /// GetNumsOfMembers method
            /// - Get number of Member table rows by memberName
            /// </summary>
            /// <param name="memberName">Member name</param>
            /// <returns></returns>
            public int GetNumsOfMembers(string memberName)
            {
                int ret = -1;

                try
                {
                    ret = new DacMember(providerName,
connectionString).SelectNumsOfMembers(memberName);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            /// <summary>
            /// SetMember method
            /// - Set Member table row by member information
            /// </summary>
            /// <param name="member">Member information</param>
            /// <returns></returns>
            public bool SetMember(Member member)
            {
                bool? ret;

                using (TransactionScope scope = new TransactionScope())
                {
                    try
                    {
                        ret = new DacMember(providerName, connectionString).UpdateMember(member);

                        if (ret != null)
                        {
                            // Success
                            new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                                new MemberHistory(member.MemberID, member.MemberName, true,
                                    string.Format("Update member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        member.MemberID, member.MemberName, member.IsAvailable,
                                        member.Email, member.PhoneNumber, member.Address)
                                )
                            );
                        }
                        else
                        {
                            // Fail
                            new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                                new MemberHistory(member.MemberID, member.MemberName, false,
                                    string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        member.MemberID, member.MemberName, member.IsAvailable,
                                        member.Email, member.PhoneNumber, member.Address)
```

```
                                    )
                                );
                        }
                    }
                    catch (Exception ex)
                    {
                        // Fail
                        new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    member.MemberID, member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );

                        throw ex;
                    }
                    finally
                    {
                        scope.Complete();
                    }

                    return (ret == true) ? true : false;
                }
            }

            /// <summary>
            /// RemoveMember() method
            /// - Remove Member table row by memberID
            /// </summary>
            /// <param name="memberID">Member ID</param>
            /// <returns></returns>
            public bool RemoveMember(int memberID)
            {
                bool? ret;
                Member removedMember = null;

                using (TransactionScope scope = new TransactionScope())
                {
                    try
                    {
                        removedMember = new DacMember(providerName,
connectionString).SelectMember(memberID);

                        ret = new DacMember(providerName, connectionString).RemoveMember(memberID);

                        if (ret != null && ret == true)
                        {
                            // Success
                            new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                                new MemberHistory(removedMember.MemberID,
removedMember.MemberName, true,
                                    string.Format("Remove member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                        removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                                )
                            );
                        }
                        else
                        {
                            // Fail
                            new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                                new MemberHistory(removedMember.MemberID,
removedMember.MemberName, false,
```

```
                                        string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                            removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                            removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                                    )
                                );
                            }
                        }
                        catch (Exception ex)
                        {
                            // Fail
                            new DacMemberHistory(providerName, connectionString).InsertMemberHistory(
                                new MemberHistory(removedMember.MemberID, removedMember.MemberName,
false,
                                    string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                        removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                        removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                                )
                            );

                            throw ex;
                        }
                        finally
                        {
                            scope.Complete();
                        }

                        return (ret == true) ? true : false;
                    }
                }
            }
        }
```

In BizMemberShip.cs, we call pure query data access classes and BizMemberShipSP.cs which calls stored procedure as below.

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Business > Memberships.Business > BizMembershipSP.cs]

```
using System;
using System.Collections.Generic;
using System.Transactions;

using Memberships.Entity;
using Memberships.DataAccess;

namespace Memberships.Business
{
    public class BizMemberShipSP
    {
        private string providerName;
        private string connectionString;

        public BizMemberShipSP() : this(string.Empty, string.Empty)
        {
        }
        public BizMemberShipSP(string providerName, string connectionString)
        {
            this.providerName = providerName;
            this.connectionString = connectionString;
```

```
        }

        /// <summary>
        /// CreateMember method
        /// - Create Member table row from member information
        /// </summary>
        /// <param name="member">Member information</param>
        /// <returns></returns>
        public Member CreateMember(Member member)
        {
            Member newMember = null;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    newMember = new DacMemberSP(providerName,
connectionString).InsertMember(member);

                    if (newMember != null)
                    {
                        // Success
                        MemberHistory mh = new DacMemberHistorySP(providerName,
connectionString).InsertMemberHistory(
                            new MemberHistory(newMember.MemberID, newMember.MemberName,
true,
                                string.Format("Create new member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    newMember.MemberID, newMember.MemberName,
newMember.IsAvailable,
                                    newMember.Email, newMember.PhoneNumber,
newMember.Address)
                            )
                        );
                    }
                    else
                    {
                        // Fail
                        MemberHistory mh = new DacMemberHistorySP(providerName,
connectionString).InsertMemberHistory(
                            new MemberHistory(member.MemberID, member.MemberName, false,
                                string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                    member.MemberName, member.IsAvailable,
                                    member.Email, member.PhoneNumber, member.Address)
                            )
                        );
                    }
                }
                catch (Exception ex)
                {
                    // Fail
                    new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                        new MemberHistory(member.MemberID, member.MemberName, false,
                            string.Format("Fail creation of new member [{0}, {1}, {2}, {3}, {4}]",
                                member.MemberName, member.IsAvailable,
                                member.Email, member.PhoneNumber, member.Address)
                        )
                    );

                    throw ex;
                }
                finally
                {
                    scope.Complete();
                }
            }
```

```
            return newMember;
        }

        /// <summary>
        /// GetMember method
        /// - Get Member table row by memberID
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public Member GetMember(int memberID)
        {
            Member ret = null;

            try
            {
                ret = new DacMemberSP(providerName, connectionString).SelectMember(memberID);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        /// <summary>
        /// GetMembers method
        /// - Get Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public List<Member> GetMembers(string memberName)
        {
            List<Member> ret = null;

            try
            {
                ret = new DacMemberSP(providerName, connectionString).SelectMembers(memberName);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        /// <summary>
        /// GetNumsOfMembers method
        /// - Get number of Member table rows by memberName
        /// </summary>
        /// <param name="memberName">Member name</param>
        /// <returns></returns>
        public int GetNumsOfMembers(string memberName)
        {
            int ret = -1;

            try
            {
                ret = new DacMemberSP(providerName,
connectionString).SelectNumsOfMembers(memberName);
            }
            catch (Exception ex)
            {
                throw ex;
            }
```

```
        return ret;
    }

    /// <summary>
    /// SetMember method
    /// - Set Member table row by member information
    /// </summary>
    /// <param name="member">Member information</param>
    /// <returns></returns>
    public bool SetMember(Member member)
    {
        bool? ret;

        using (TransactionScope scope = new TransactionScope())
        {
            try
            {
                ret = new DacMemberSP(providerName, connectionString).UpdateMember(member);

                if (ret != null)
                {
                    // Success
                    new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                        new MemberHistory(member.MemberID, member.MemberName, true,
                            string.Format("Update member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                member.MemberID, member.MemberName, member.IsAvailable,
                                member.Email, member.PhoneNumber, member.Address)
                        )
                    );
                }
                else
                {
                    // Fail
                    new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                        new MemberHistory(member.MemberID, member.MemberName, false,
                            string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                member.MemberID, member.MemberName, member.IsAvailable,
                                member.Email, member.PhoneNumber, member.Address)
                        )
                    );
                }
            }
            catch (Exception ex)
            {
                // Fail
                new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                    new MemberHistory(member.MemberID, member.MemberName, false,
                        string.Format("Fail update of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                            member.MemberID, member.MemberName, member.IsAvailable,
                            member.Email, member.PhoneNumber, member.Address)
                    )
                );

                throw ex;
            }
            finally
            {
                scope.Complete();
            }

            return (ret == true) ? true : false;
        }
    }

    /// <summary>
```

```
        /// RemoveMember() method
        /// - Remove Member table row by memberID
        /// </summary>
        /// <param name="memberID">Member ID</param>
        /// <returns></returns>
        public bool RemoveMember(int memberID)
        {
            bool? ret;
            Member removedMember = null;

            using (TransactionScope scope = new TransactionScope())
            {
                try
                {
                    removedMember = new DacMemberSP(providerName,
connectionString).SelectMember(memberID);

                    ret = new DacMemberSP(providerName,
connectionString).RemoveMember(memberID);

                    if (ret != null && ret == true)
                    {
                        // Success
                        new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(removedMember.MemberID,
removedMember.MemberName, true,
                                string.Format("Remove member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                    removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                                )
                        );
                    }
                    else
                    {
                        // Fail
                        new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                            new MemberHistory(removedMember.MemberID,
removedMember.MemberName, false,
                                string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                    removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                    removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                                )
                        );
                    }
                }
                catch (Exception ex)
                {
                    // Fail
                    new DacMemberHistorySP(providerName, connectionString).InsertMemberHistory(
                        new MemberHistory(removedMember.MemberID, removedMember.MemberName,
false,
                            string.Format("Fail remove of member [{0}, {1}, {2}, {3}, {4}, {5}]",
                                removedMember.MemberID, removedMember.MemberName,
removedMember.IsAvailable,
                                removedMember.Email, removedMember.PhoneNumber,
removedMember.Address)
                            )
                    );

                    throw ex;
                }
                finally
```

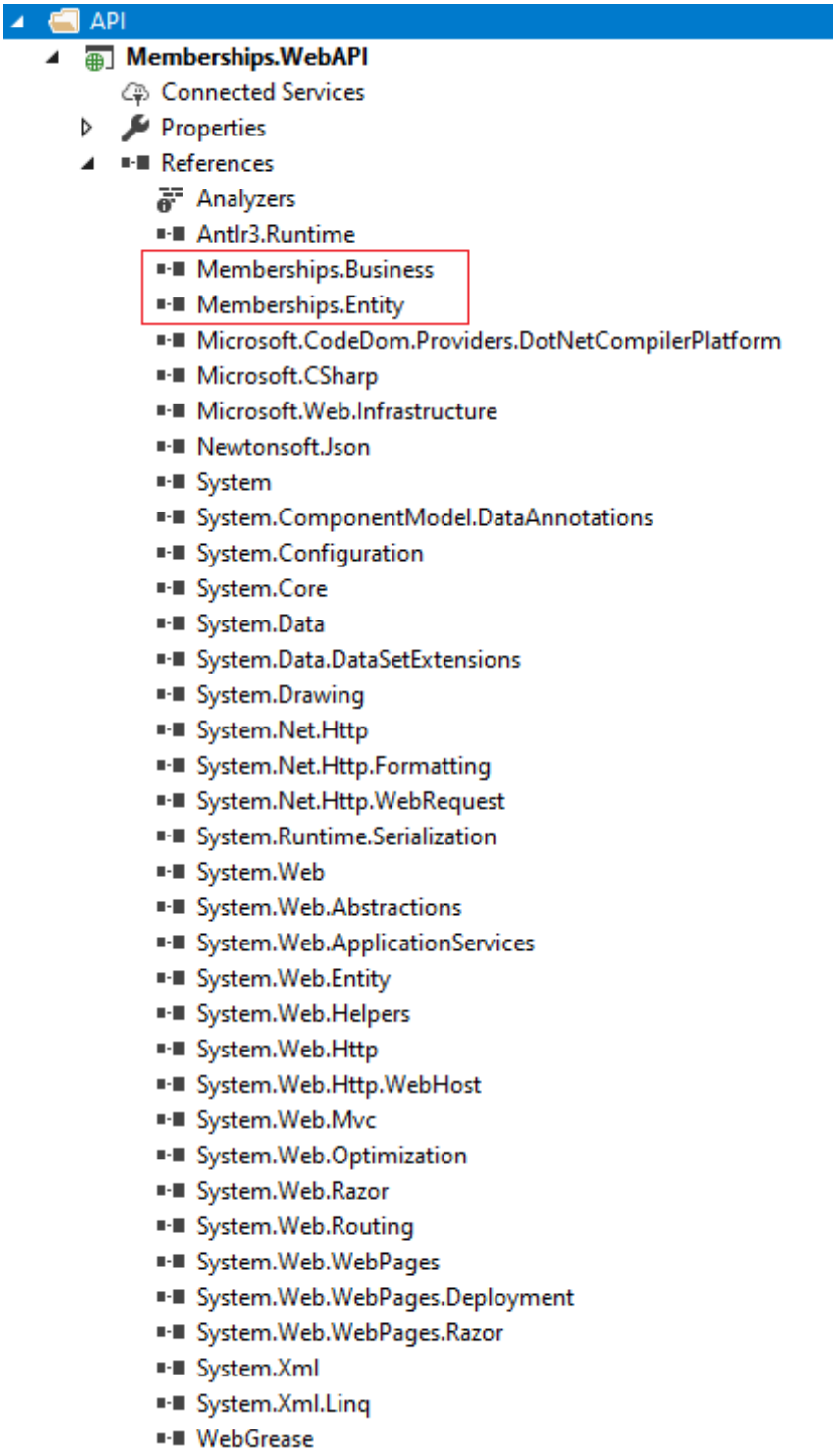
```
                    {
                        scope.Complete();
                    }

                    return (ret == true) ? true : false;
                }
            }
        }
}
```

API

We will make a web service API by directly utilizing Entity classes, Data Access classes and Business classes defined above. This is similar to the real web service API associated with membership information.

We will create an ASP.NET WebAPI project and check the actual IIS services. We can create a Memberships.WebAPI ASP.NET Web Application project through Visual Studio's [New> Project]. Note that when you create a project, you must create a project by selecting the Web API project template, and in the addition folder and core reference, you need to select the MVC and Web API. This will create a Memberships.WebAPI project as shown below.

And you need to add Memberships.Entity and Memberships.Business as references to the Memberships.WebAPI project. Of course, you can use Memberships.DataAccess as a reference instead of Memberships.Business. We will use Memberships.Business instead of Memberships.DataAccess because the transaction was implemented in Memberships.Business.



Then, you need to open the Web.config file and add the connection string to connect MySQL Database as follows. Note that the default connection string is defined in the DacMember.cs or DacMemberHistory.cs, DacMemberSP.cs, and DacMemberHistorySP.cs files of the DataAccess class, but we will define and use the DB connection string that is commonly used by the WebAPI service.

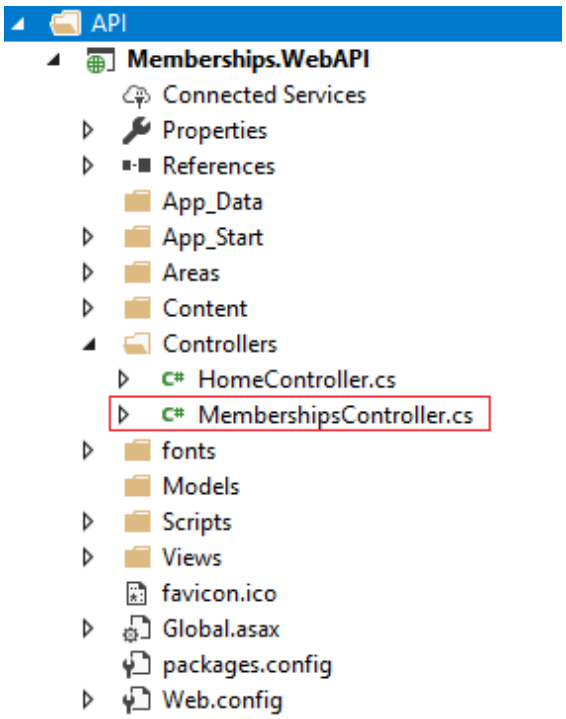
The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [API> Memberships.API> Web.config]

```
<configuration>
  <connectionStrings>
```

```
      <add name="MYSQLDB"
         connectionString="AllowUserVariables=true;Server=localhost;Port=32785;Database=Product;Uid=root;
         Pwd=qwert12345!;ConnectionLifeTime=60;"
         providerName="MySql.Data.MySqlClient" />
   </connectionStrings>
   …
   …
</configuration>
```

Next, you need to add the MembershipsController.cs file to the Controllers folder of the Memberships.WebAPI project. As you can see, you can create an API just by implementing a Controller in a WebAPI project.



Then you need to add the following code to the MembershipsController.cs.

The code below can be found in the sample project.

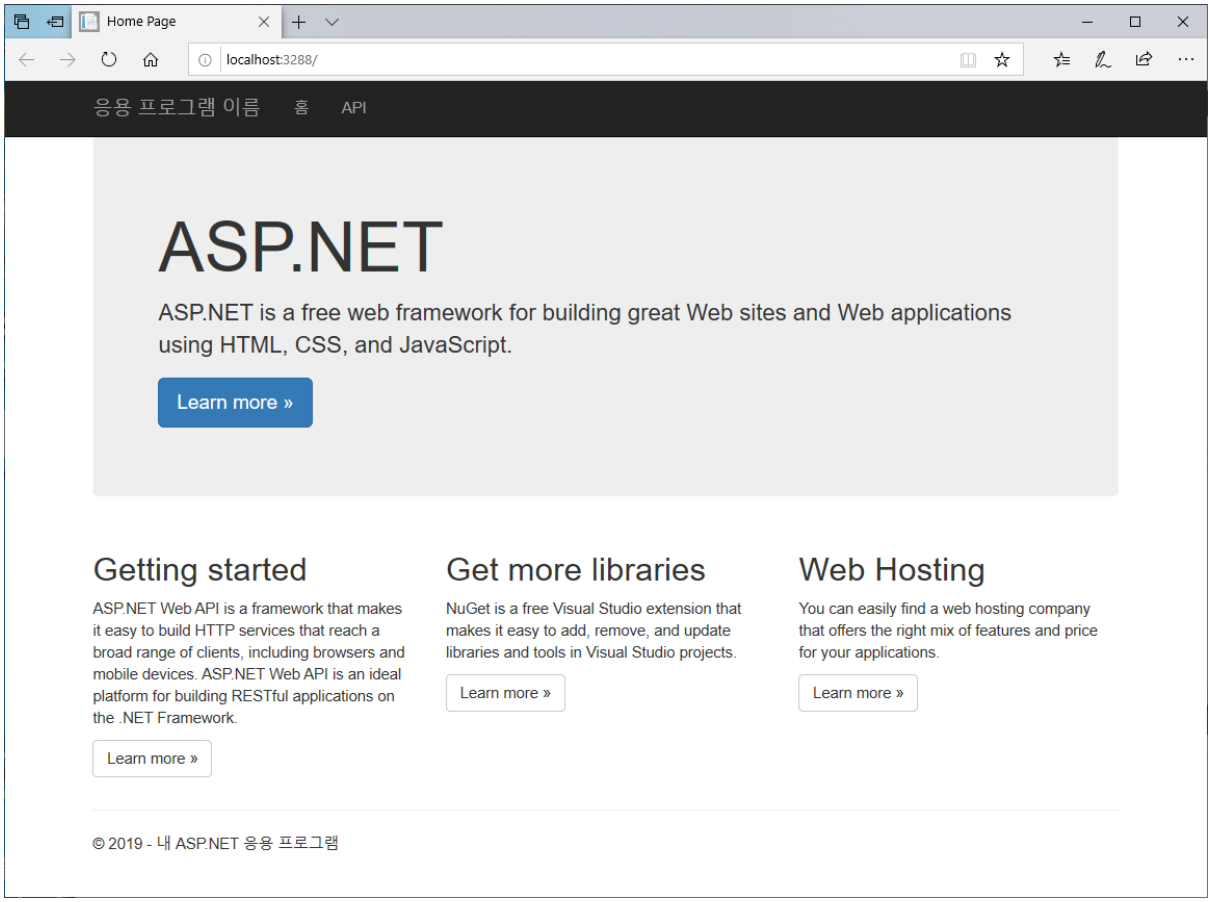
In TaeMFrameworkwithMySQL [API> Memberships.WebAPI> Controllers> MembershipsController.cs]

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Web.Http;

using Memberships.Business;
using Memberships.Entity;

namespace Memberships.WebAPI.Controllers
{
    public class MembershipsController : ApiController
    {
        private static string MY_SQL_PROVIDER_NAME =
ConfigurationManager.ConnectionStrings["MYSQLDB"].ProviderName;
        private static string MY_SQL_CONN_STR =
ConfigurationManager.ConnectionStrings["MYSQLDB"].ConnectionString;
```

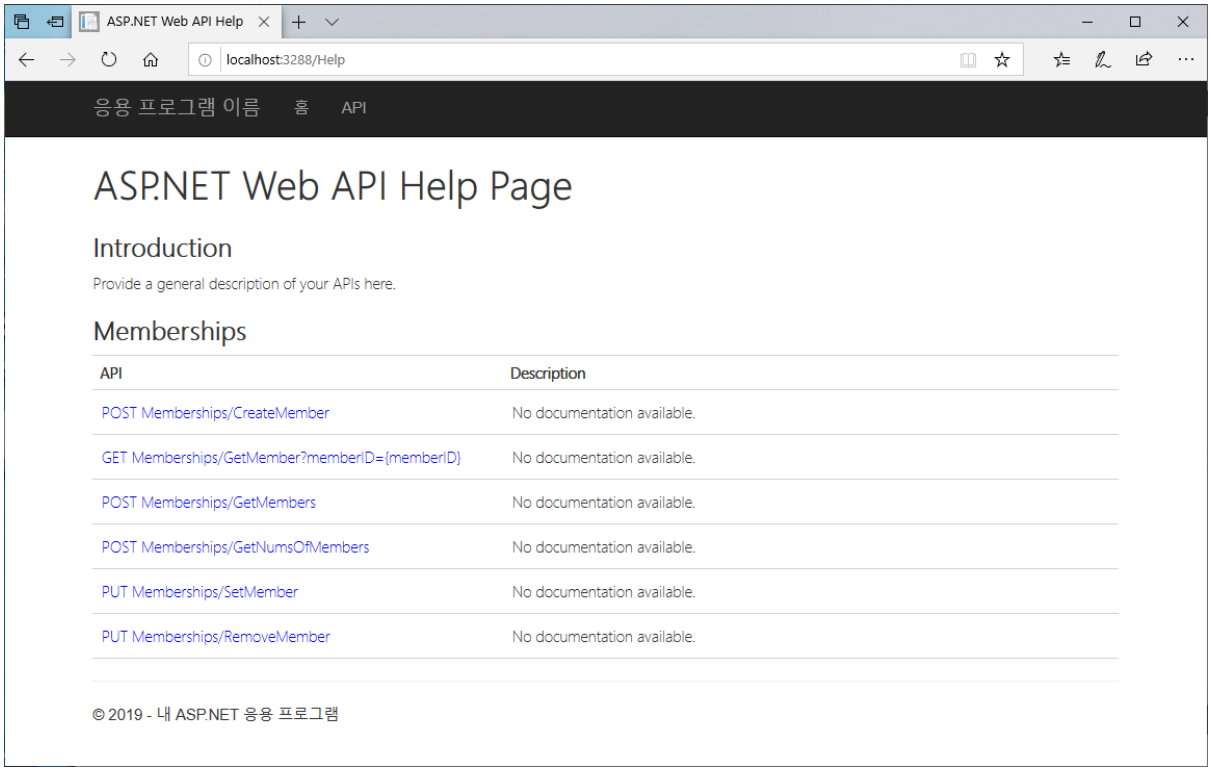
```
        [HttpPost]
        [Route("Memberships/CreateMember")]
        public Member CreateMember(Member member)
        {
            Member ret = null;

            try
            {
                ret = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).CreateMember(member);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        [HttpGet]
        [Route("Memberships/GetMember")]
        public Member GetMember(int memberID)
        {
            Member ret = null;

            try
            {
                ret = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMember(memberID);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        [HttpPost]
        [Route("Memberships/GetMembers")]
        public List<Member> GetMembers([FromBody]string memberName)
        {
            List<Member> ret = null;

            try
            {
                ret = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMembers(memberName);
            }
            catch (Exception ex)
            {
                throw ex;
            }

            return ret;
        }

        [HttpPost]
        [Route("Memberships/GetNumsOfMembers")]
        public int GetNumsOfMembers([FromBody]string memberName)
        {
            int ret = -1;

            try
            {
                ret = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
```

```
MY_SQL_CONN_STR).GetNumsOfMembers(memberName);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            [HttpPut]
            [Route("Memberships/SetMember")]
            public bool SetMember(Member member)
            {
                bool ret = false;

                try
                {
                    ret = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).SetMember(member);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }

            [HttpPut]
            [Route("Memberships/RemoveMember")]
            public bool RemoveMember([FromBody]int memberID)
            {
                bool ret = false;

                try
                {
                    ret = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).RemoveMember(memberID);
                }
                catch (Exception ex)
                {
                    throw ex;
                }

                return ret;
            }
        }
}
```

After adding the Controller, build and run the Memberships.WebAPI project. Execution of the Memberships.WebAPI project can be performed on IIS Express, which is the Visual Studio's default WebAPI project execution environment, or on a separate IIS after setting the WebAPI in IIS. The first screen you have run is shown below.

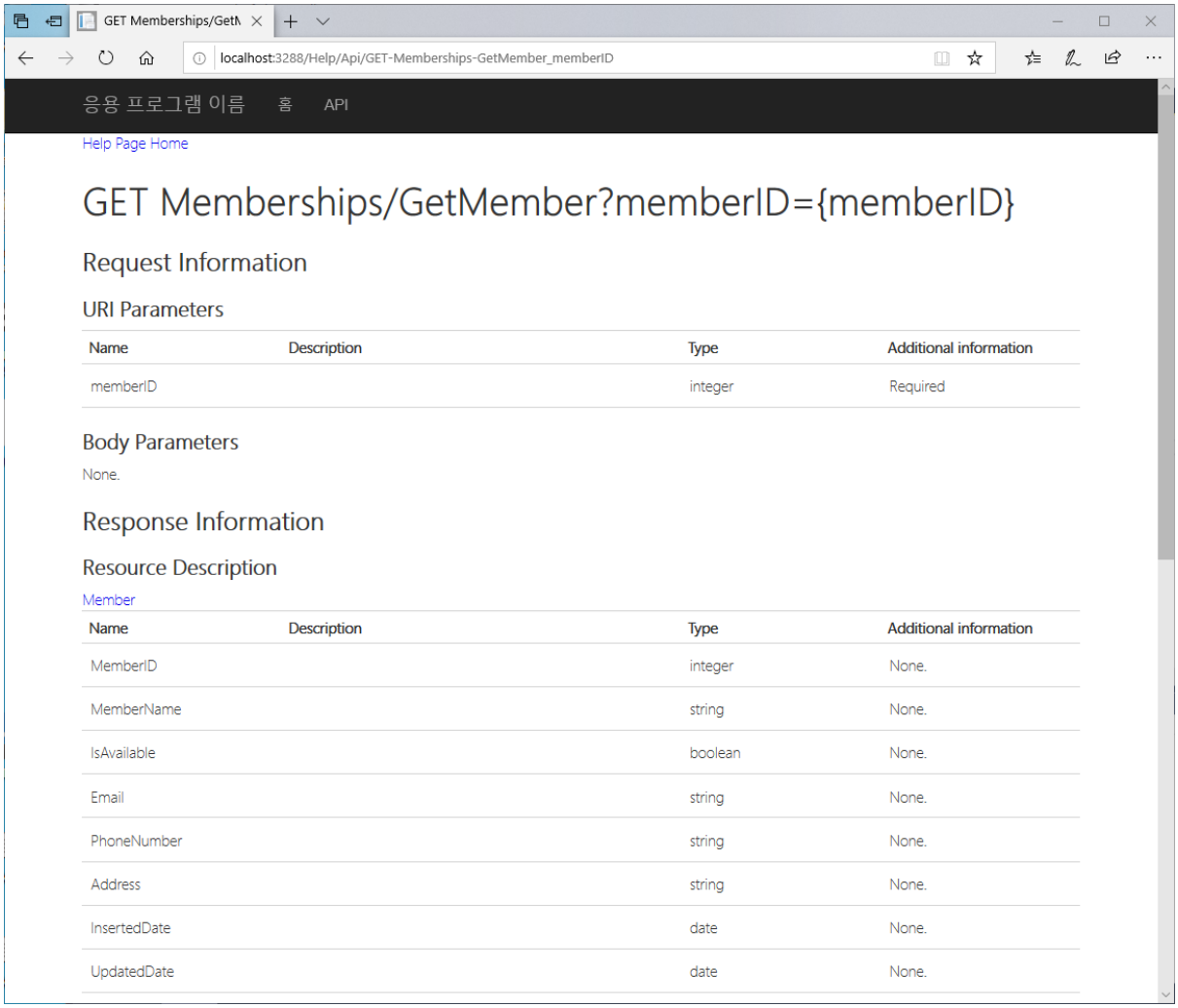


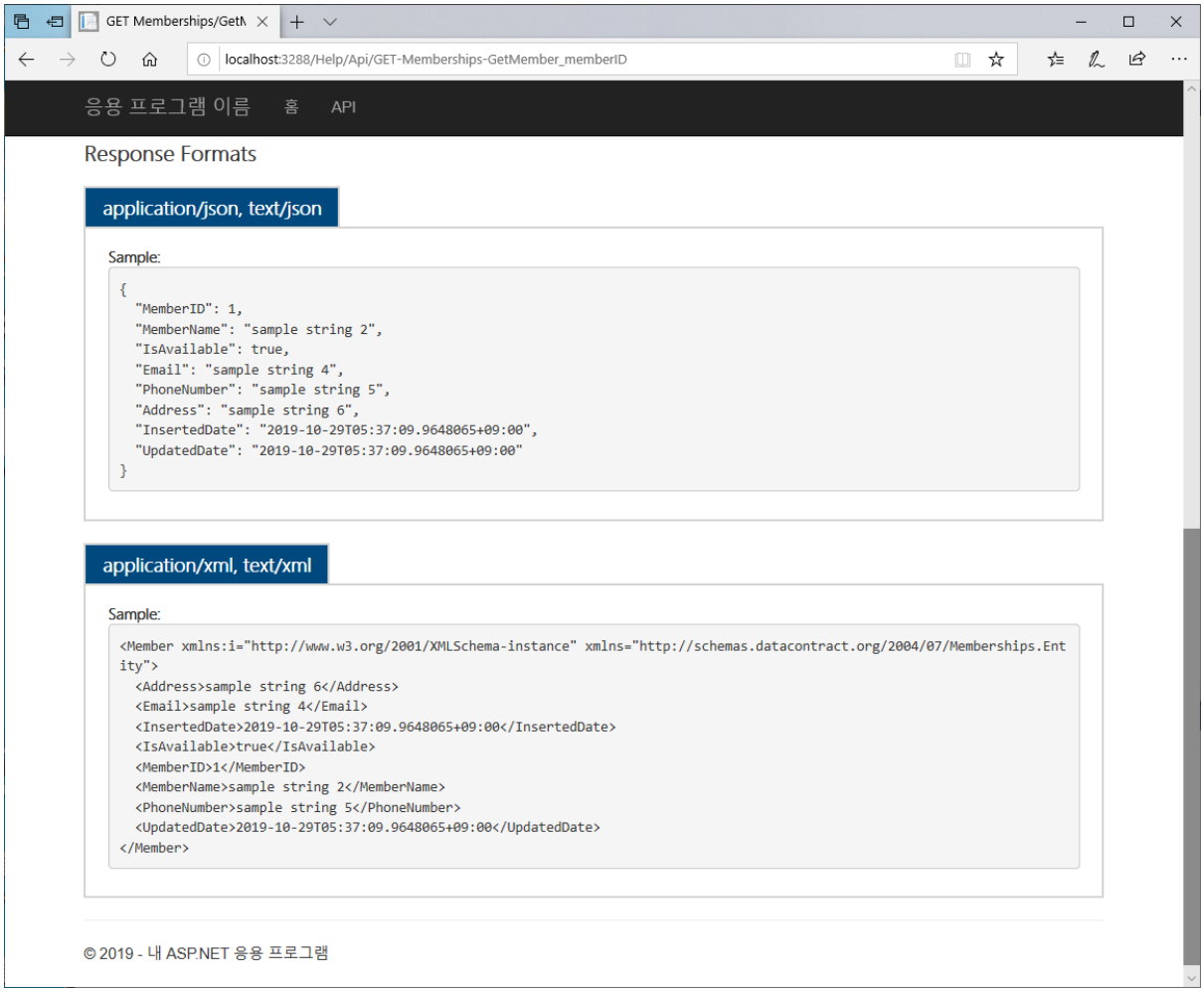
When this screen is displayed, click the "API" item. So you will see a list of the individual methods of the API implemented in the Memberships Controller, as shown below.



Here we will try to click on the GetMember method as GET format. So you can get a detailed

description of the GetMember method request format and response format.

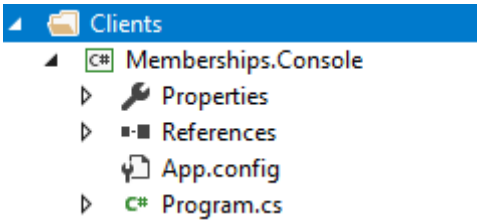



GET Memberships/GetM
localhost:3288/Help/Api/GET-Memberships-GetMember_memberID
응용 프로그램 이름
홈
API
Response Formats
application/json, text/json
Sample:
{
"MemberID": 1,
"MemberName": "sample string 2",
"IsAvailable": true,
"Email": "sample string 4",
"PhoneNumber": "sample string 5",
"Address": "sample string 6",
"InsertedDate": "2019-10-29T05:37:09.9648065+09:00",
"UpdatedDate": "2019-10-29T05:37:09.9648065+09:00"
}
application/xml, text/xml
Sample:
<Member xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2004/07/Memberships.Entity">
<Address>sample string 6</Address>
<Email>sample string 4</Email>
<InsertedDate>2019-10-29T05:37:09.9648065+09:00</InsertedDate>
<IsAvailable>true</IsAvailable>
<MemberID>1</MemberID>
<MemberName>sample string 2</MemberName>
<PhoneNumber>sample string 5</PhoneNumber>
<UpdatedDate>2019-10-29T05:37:09.9648065+09:00</UpdatedDate>
</Member>
© 2019 - 내 ASP.NET 응용 프로그램

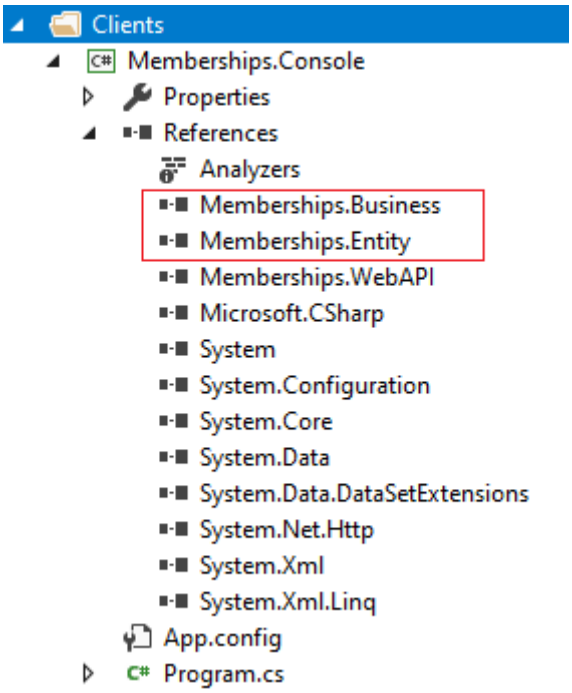
Test client

We made the design of database table, Entity class, Data Access classes, Business classes, and WebAPI. So we will create a client console application to check that the classes we created are working properly.

You can create a Memberships.Console project in Console Application (.NET Framework) project type through [New> Project] in Visual Studio. This will be created the Memberships.Console project as shown below.



You need to add Memberships.Entity, Memberships.Business as references in the Memberships.Console project as follows.



You need to add the DB connection string and WebAPI connection information to the App.config file as follows.

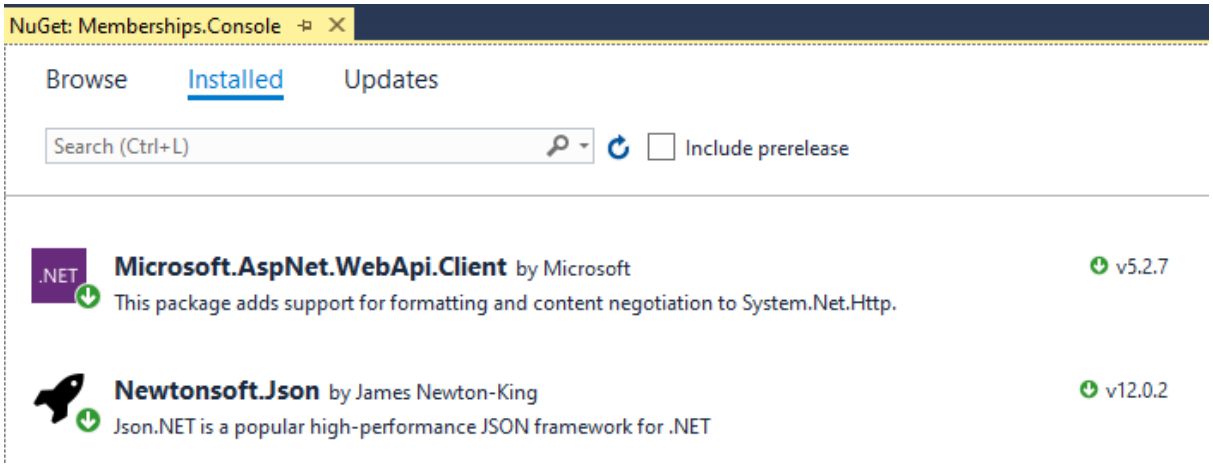
The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Clients> Memberships.Console> App.config]

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7" />
  </startup>
  <connectionStrings>
    <add name="MYSQLDB"
      connectionString="Server=localhost;Port=32785;Database=Product;Uid=root;Pwd=qwert12345!;Conne
      ctionLifeTime=60;AllowUserVariables=true;"
      providerName="MySql.Data.MySqlClient" />
```

```
  </connectionStrings>
  <appSettings>
    <!-- VS IIS Express Uri -->
    <add key="WebAPI_IIS_Express_Uri" value="http://localhost:3288/"/>
    <!-- End -->

    <!-- IIS Uri -->
    <add key="WebAPI_IIS" value="http://localhost/Memberships.WebAPI.3288/"/>
    <!-- End -->

    <add key="HeaderType" value="application/json" />
  </appSettings>
</configuration>
```

And you need to add the following WebAPICaller.cs file to be used when calling to the WebAPI client. The WebAPICaller.cs uses several extension methods to facilitate WebAPI calls. So you need to install two NuGet packages: Microsoft.AspNet.WebApi.Client and Newtonsoft.Json.



We installed two NuGet packages and defined WebAPICaller.cs as shown below.

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Clients> Memberships.Console> WebAPICaller.cs]

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Net.Http.Formatting;
using System.Threading;

using Memberships.Entity;

namespace Memberships.Console
{
    public class WebAPICaller
    {
        public static Member CallCreateMember(string uri, Member member)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PostAsJsonAsync(
                "Memberships/CreateMember",
```

```
                member).Result;

            if (response.IsSuccessStatusCode)
            {
                Member ret = response.Content.ReadAsAsync<Member>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call CreateMember API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return null;
        }

        public static Member CallGetMember(string uri, int memberID)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.GetAsync(
                string.Format("Memberships/GetMember?memberID={0}", memberID)
                ).Result;

            if (response.IsSuccessStatusCode)
            {
                Member ret = response.Content.ReadAsAsync<Member>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call GetMember API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return null;
        }

        public static List<Member> CallGetMembers(string uri, string memberName)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PostAsJsonAsync(
                "Memberships/GetMembers",
                memberName).Result;

            if (response.IsSuccessStatusCode)
            {
                List<Member> ret = response.Content.ReadAsAsync<List<Member>>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call GetMembers API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return null;
        }
```

```
        public static int CallGetGetNumsOfMembers(string uri, string memberName)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PostAsJsonAsync(
                "Memberships/GetNumsOfMembers",
                memberName).Result;

            if (response.IsSuccessStatusCode)
            {
                int ret = response.Content.ReadAsAsync<int>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call GetNumsOfMembers API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return -1;
        }

        public static bool CallSetMember(string uri, Member member)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PutAsJsonAsync(
                "Memberships/SetMember",
                member).Result;

            if (response.IsSuccessStatusCode)
            {
                bool ret = response.Content.ReadAsAsync<bool>().Result;
                return ret;
            }
            else
            {
                System.Console.WriteLine("{0} : {1} ({2})", "Call SetMember API method",
(int)response.StatusCode, response.ReasonPhrase);
            }

            return false;
        }

        public static bool CallRemoveMember(string uri, int memberID)
        {
            HttpClient client = new HttpClient();
            client.BaseAddress = new Uri(uri);
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue(ConfigurationManager.AppSettings["HeaderType"]));

            HttpResponseMessage response = client.PutAsJsonAsync(
                "Memberships/RemoveMember",
                memberID).Result;

            if (response.IsSuccessStatusCode)
            {
                bool ret = response.Content.ReadAsAsync<bool>().Result;
                return ret;
            }
```

```
                else
                {
                    System.Console.WriteLine("{0} : {1} ({2})", "Call SetMember API method",
(int)response.StatusCode, response.ReasonPhrase);
                }

                return false;
            }
        }
}
```

Finally, you need to add the following code to the Program.cs to put the code that calls the Business class directly and the WebAPI directly.

The code below can be found in the sample project.

In TaeMFrameworkwithMySQL [Clients > Memberships.Console > Program.cs]

```
using System.Collections.Generic;
using System.Configuration;

using Memberships.Business;
using Memberships.Entity;

namespace Memberships.Console
{
    class Program
    {
        private static string MY_SQL_PROVIDER_NAME =
ConfigurationManager.ConnectionStrings["MYSQLDB"].ProviderName;
        private static string MY_SQL_CONN_STR =
ConfigurationManager.ConnectionStrings["MYSQLDB"].ConnectionString;

        static void Main(string[] args)
        {
            // Call Business class - Pure query
            CallBusiness();

            // Call Business class - Stored procedure
            CallBusinessSP();

            // Call WebAPI
            CallWebAPI();
        }

        private static void CallBusiness()
        {
            // Create new member
            Member newMember = new Member("John Doe", true, "john@taemcorp.net", "080-00-1234-
5678", "anywhere");
            Member createdMemberInDB = new BizMemberShip(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).CreateMember(newMember);
            WriteMember(createdMemberInDB);

            // Select member
            Member selectedMemberInDB = new BizMemberShip(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMember(createdMemberInDB.MemberID);
            WriteMember(selectedMemberInDB);


            // Create new member
            Member newMember2 = new Member("Jane Doe", true, "jane@taemcorp.net", "080-00-0234-
5378", "anywhere");
            Member createdMemberInDB2 = new BizMemberShip(MY_SQL_PROVIDER_NAME,
```

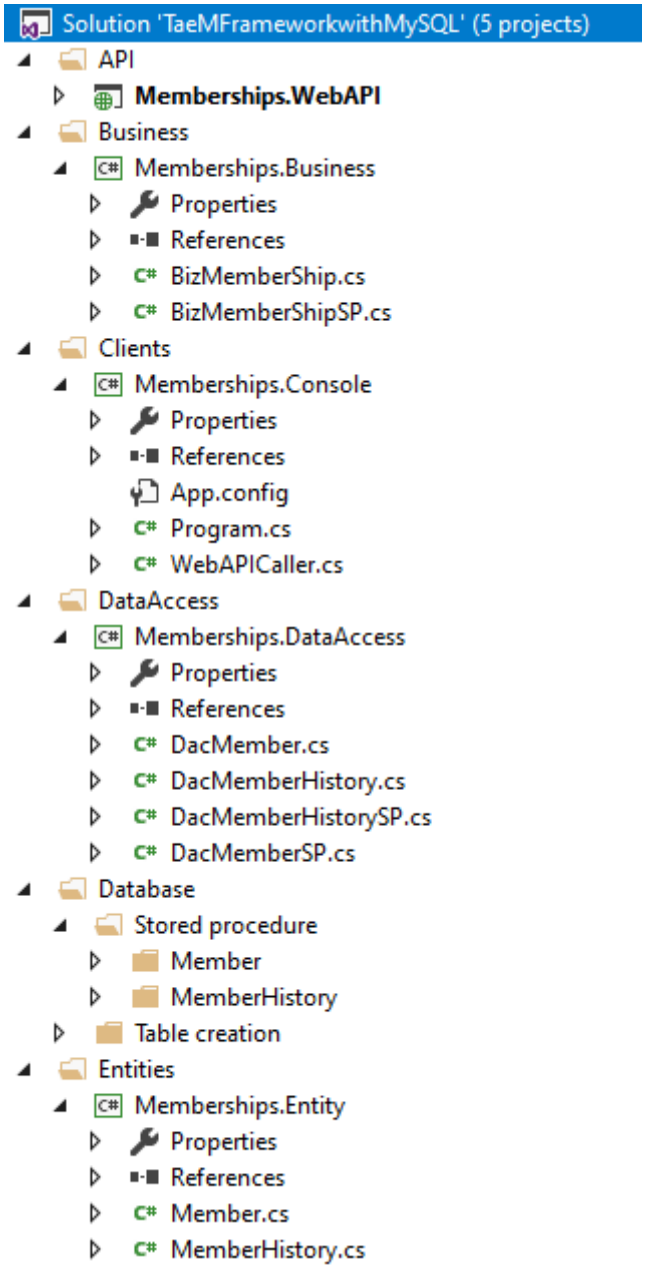
```
MY_SQL_CONN_STR).CreateMember(newMember2);
                WriteMember(createdMemberInDB2);

                // Get member
                Member selectedMemberInDB2 = new BizMemberShip(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMember(createdMemberInDB2.MemberID);
                WriteMember(selectedMemberInDB2);


                // Get members
                List<Member> currentMembers = new BizMemberShip(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMembers(string.Empty);
                WriteMembers(currentMembers);

                // Get numboer of members
                int numOfCurrentMembers = new BizMemberShip(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetNumsOfMembers(string.Empty);
                WriteCurrentNumberOfMemers(numOfCurrentMembers);

                // Update Member
                createdMemberInDB.MemberName = "John and Jane Doe";
                createdMemberInDB.Address = "John N Jane";
                createdMemberInDB.Email = "johnnjane@taemcorp.net";

                bool updateResult = new BizMemberShip(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).SetMember(createdMemberInDB);

                if (updateResult)
                {
                    Member updatedMember = new BizMemberShip(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMember(createdMemberInDB.MemberID);
                    WriteMember(updatedMember);
                }

                // Delete Member
                bool deleteResult = new BizMemberShip(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).RemoveMember(createdMemberInDB.MemberID);

                List<Member> afterDeletedMembers = new BizMemberShip(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMembers(string.Empty);
                WriteMembers(afterDeletedMembers);
            }

            private static void CallBusinessSP()
            {
                // Create new member
                Member newMember = new Member("John Doe", true, "john@taemcorp.net", "080-00-1234-
5678", "anywhere");
                Member createdMemberInDB = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).CreateMember(newMember);
                WriteMember(createdMemberInDB);

                // Select member
                Member selectedMemberInDB = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMember(createdMemberInDB.MemberID);
                WriteMember(selectedMemberInDB);


                // Create new member
                Member newMember2 = new Member("Jane Doe", true, "jane@taemcorp.net", "080-00-0234-
5378", "anywhere");
                Member createdMemberInDB2 = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).CreateMember(newMember2);
                WriteMember(createdMemberInDB2);

                // Get member
```

```
            Member selectedMemberInDB2 = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMember(createdMemberInDB2.MemberID);
            WriteMember(selectedMemberInDB2);


            // Get members
            List<Member> currentMembers = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMembers(string.Empty);
            WriteMembers(currentMembers);

            // Get numboer of members
            int numOfCurrentMembers = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetNumsOfMembers(string.Empty);
            WriteCurrentNumberOfMemers(numOfCurrentMembers);

            // Update Member
            createdMemberInDB.MemberName = "John and Jane Doe";
            createdMemberInDB.Address = "John N Jane";
            createdMemberInDB.Email = "johnnjane@taemcorp.net";

            bool updateResult = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).SetMember(createdMemberInDB);

            if (updateResult)
            {
                Member updatedMember = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMember(createdMemberInDB.MemberID);
                WriteMember(updatedMember);
            }

            // Delete Member
            bool deleteResult = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).RemoveMember(createdMemberInDB.MemberID);

            List<Member> afterDeletedMembers = new BizMemberShipSP(MY_SQL_PROVIDER_NAME,
MY_SQL_CONN_STR).GetMembers(string.Empty);
            WriteMembers(afterDeletedMembers);
        }

        private static void CallWebAPI()
        {
            string uri = ConfigurationManager.AppSettings["WebAPI_IIS_Express_Uri"];

            // Create new member
            Member newMember = new Member("John Doe", true, "john@taemcorp.net", "080-00-1234-
5678", "anywhere");
            Member createdMemberInDB = WebAPICaller.CallCreateMember(uri, newMember);
            WriteMember(createdMemberInDB);

            // Select member
            Member selectedMemberInDB = WebAPICaller.CallGetMember(uri,
createdMemberInDB.MemberID);
            WriteMember(selectedMemberInDB);


            // Create new member
            Member newMember2 = new Member("Jane Doe", true, "jane@taemcorp.net", "080-00-0234-
5378", "anywhere");
            Member createdMemberInDB2 = WebAPICaller.CallCreateMember(uri, newMember2);
            WriteMember(createdMemberInDB2);

            // Get member
            Member selectedMemberInDB2 = WebAPICaller.CallGetMember(uri,
createdMemberInDB2.MemberID);
            WriteMember(selectedMemberInDB2);
```

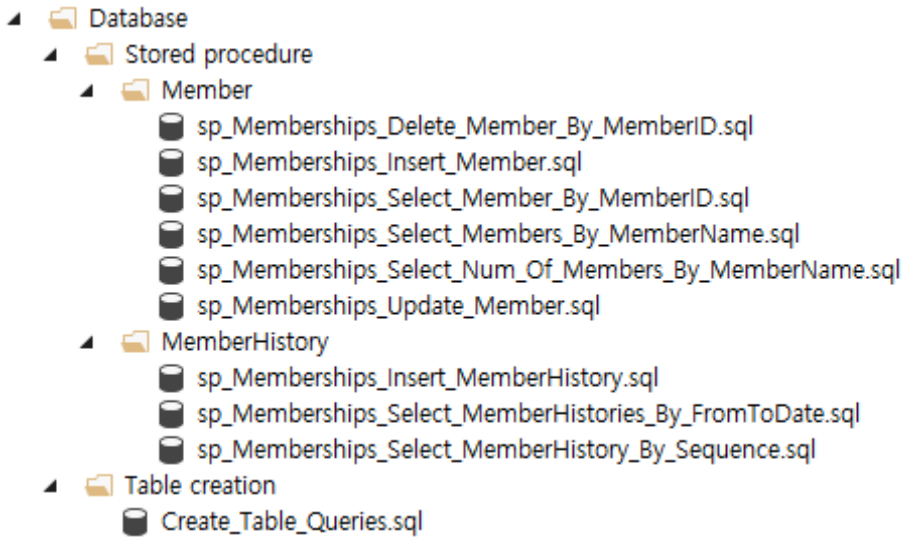
```
            // Get members
            List<Member> currentMembers = WebAPICaller.CallGetMembers(uri, string.Empty);
            WriteMembers(currentMembers);

            // Get numboer of members
            int numOfCurrentMembers = WebAPICaller.CallGetGetNumsOfMembers(uri, string.Empty);
            WriteCurrentNumberOfMemers(numOfCurrentMembers);

            // Update Member
            createdMemberInDB.MemberName = "John Jane Doe";
            createdMemberInDB.Address = "John N Jane";
            createdMemberInDB.Email = "johnnjane@taemcorp.net";

            bool updateResult = WebAPICaller.CallSetMember(uri, createdMemberInDB);

            if (updateResult)
            {
                Member updatedMember = WebAPICaller.CallGetMember(uri,
createdMemberInDB.MemberID);
                WriteMember(updatedMember);
            }

            // Delete Member
            bool deleteResult = WebAPICaller.CallRemoveMember(uri, createdMemberInDB.MemberID);

            if (deleteResult)
            {
                List<Member> afterDeletedMembers = WebAPICaller.CallGetMembers(uri, string.Empty);
                WriteMembers(afterDeletedMembers);
            }
        }

        private static void WriteMember(Member member)
        {
            if (member != null)
            {
                System.Console.WriteLine(
                    @"This member has " +
                    @"[MemberID:{0}] [MemberName:{1}] [IsAvailable:{2}] " +
                    @"[Email:{3}] [PhoneNumber:{4}] [Address:{5}] " +
                    @"[InsertedDate:{6}] [UpdatedDate:{7}]",
                    member.MemberID, member.MemberName, member.IsAvailable,
                    member.Email, member.PhoneNumber, member.Address,
                    member.InsertedDate, member.UpdatedDate);
            }
        }

        private static void WriteMembers(List<Member> members)
        {
            foreach (Member member in members)
                WriteMember(member);
        }

        private static void WriteCurrentNumberOfMemers(int numOfMembers)
        {
            System.Console.WriteLine(
                @"Current number of members : {0}",
                numOfMembers
                );
        }
    }
}
```

Sample solution

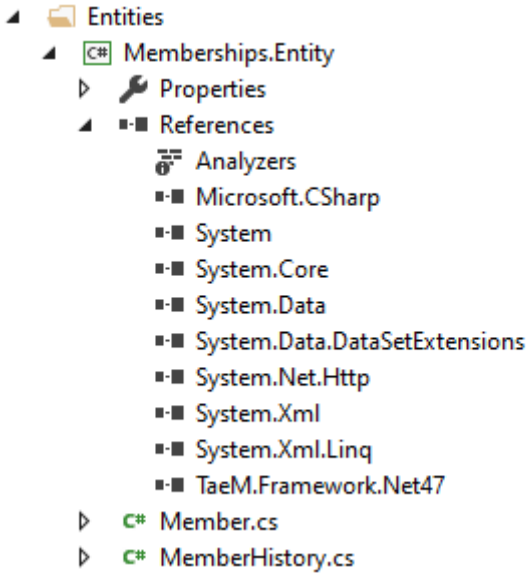
The overall solution that you have created and described so far is as follows. This configuration method is not perfect, but it is a structure we have created for readability and functional separation. So, it is good for you that you can selectively apply it to your own software development.



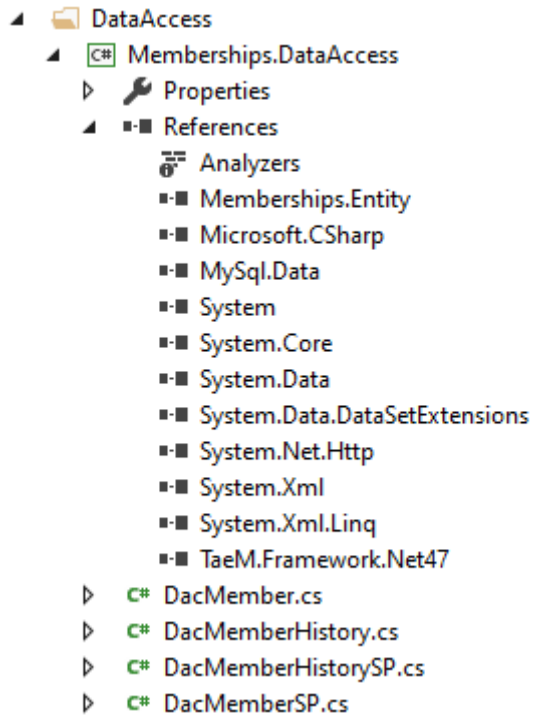
(1) Database solution folder: Database table and stored procedure are included.



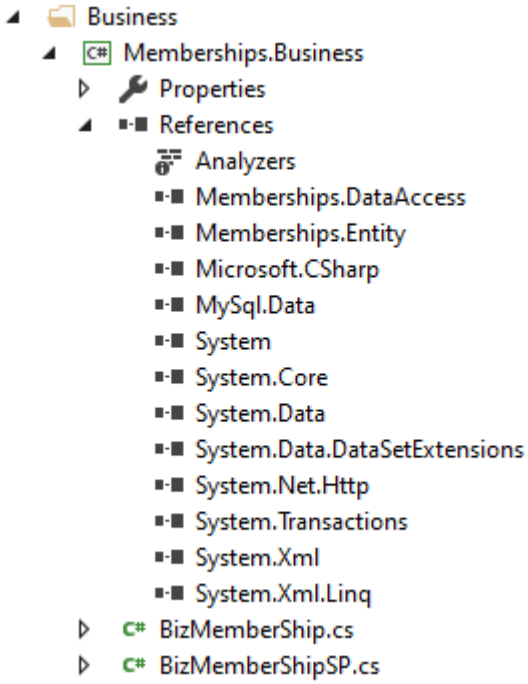
(2) Entities solution folder: Entity classes and projects are included.



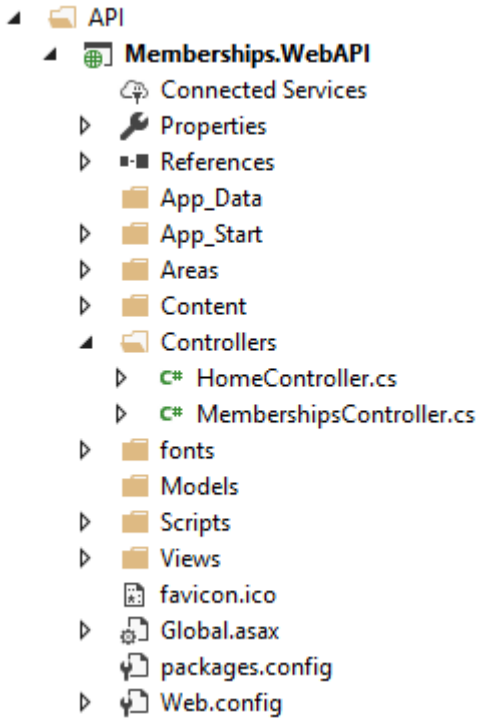
(3) DataAccess solution folder: Data access related classes and projects are included.



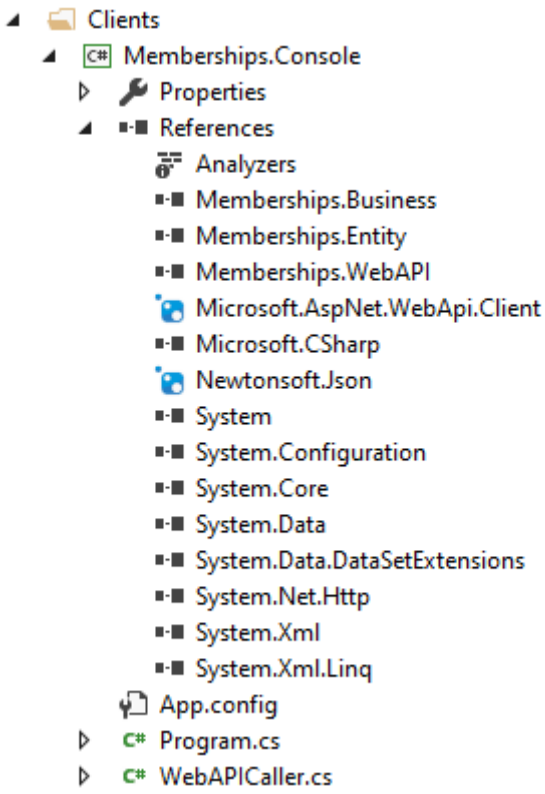
(4) Business solution folder: This includes classes and projects for transaction processing and data processing according to business.



(5) API solution folder: This includes classes and projects for processing API services such as WebAPI.



(6) Clients Solution folder: This includes test console application and other client applications.



TaeM Framework API document & Samples

API document

If you are purchasing the TaeM Framework, you will get the TaeM Framework API and the TaeM Framework API document. The TaeM Framework API document (chm file, Windows help file format) can also be downloaded from the TaeM Framework installation file or from the following Internet site:

http://www.taemcorp.net/html/downloads/taem_downloads.html

The online version of the TaeM Framework API documentation is available on the Internet at:

http://www.taemcorp.net/html/downloads/taem_api_reference.html

Samples

If you purchase the TaeM Framework, samples with the TaeM Framework will also be included. It can also be downloaded from the following Internet site.TaeM Framework samples download at :

http://www.taemcorp.net/html/downloads/taem_downloads.html