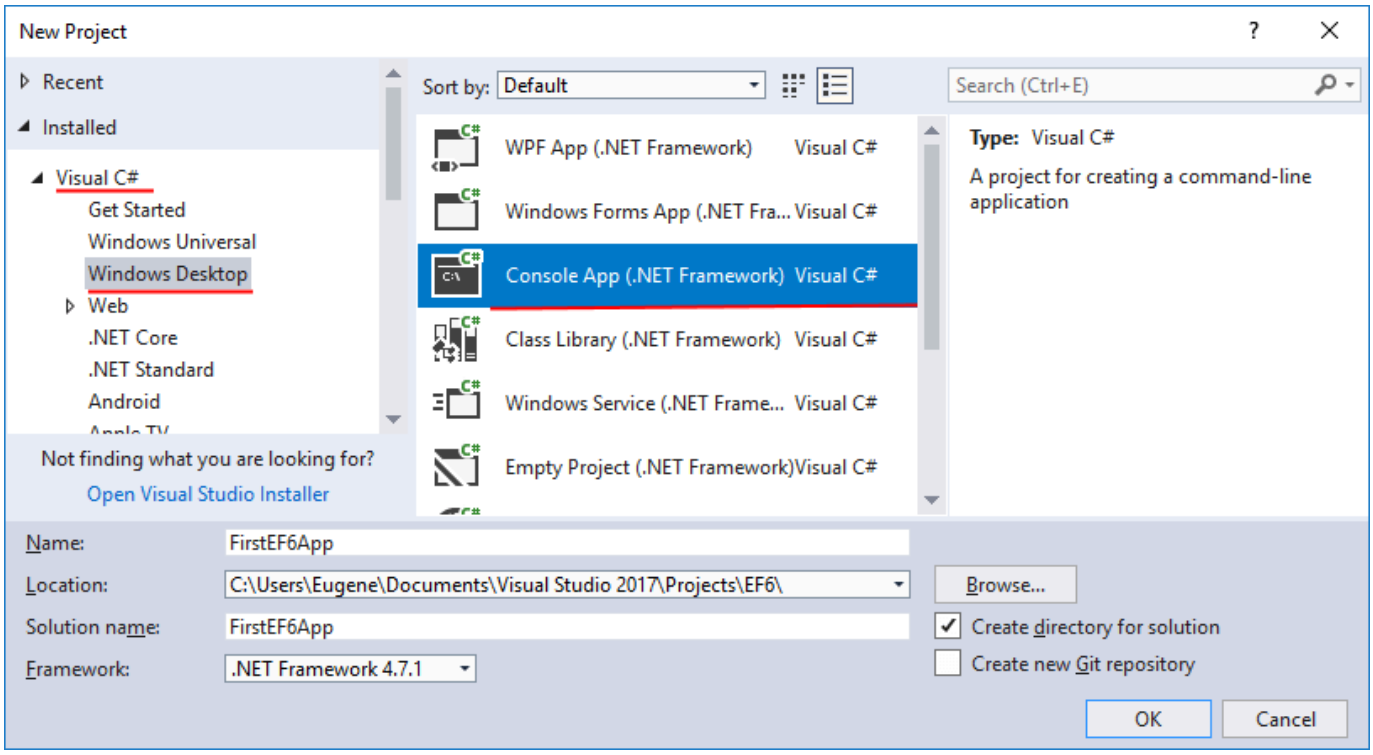
Первое приложение с Entity Framework. Подход Code First

Данное руководство устарело. Актуальное руководство: Руководство по Entity Framework Core 7

Последнее обновление: 24.10.2018

Чтобы непосредственно начать работать с Entity Framework, создадим первое приложение. Для этого нам нужна будет, во-первых, среда разработки. В качестве среды разработки выберем Visual Studio 2017.

В окне создания проекта в левой части выберем секцию **Visual C#->Windows Desktop** и в центральной части окна в качестве типа проекта выберем **Console App (.NET Framework)**.



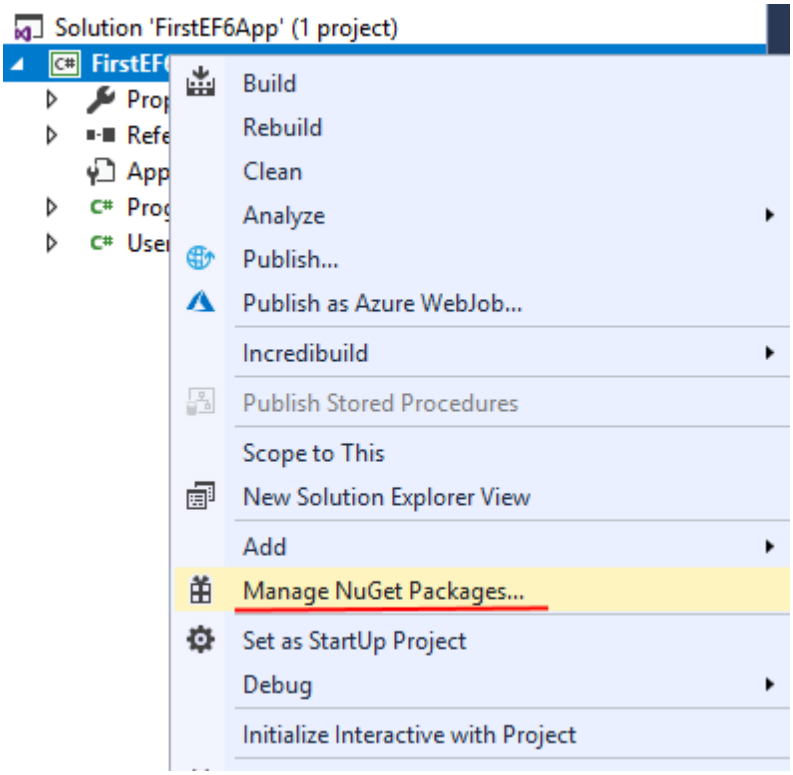
Теперь первым делом добавим новый класс, который будет описывать данные. Пусть наше приложение будет посвящено работе с пользователями. Поэтому добавим в проект новый класс User:

```
public class User
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Age { get; set; }
}
```

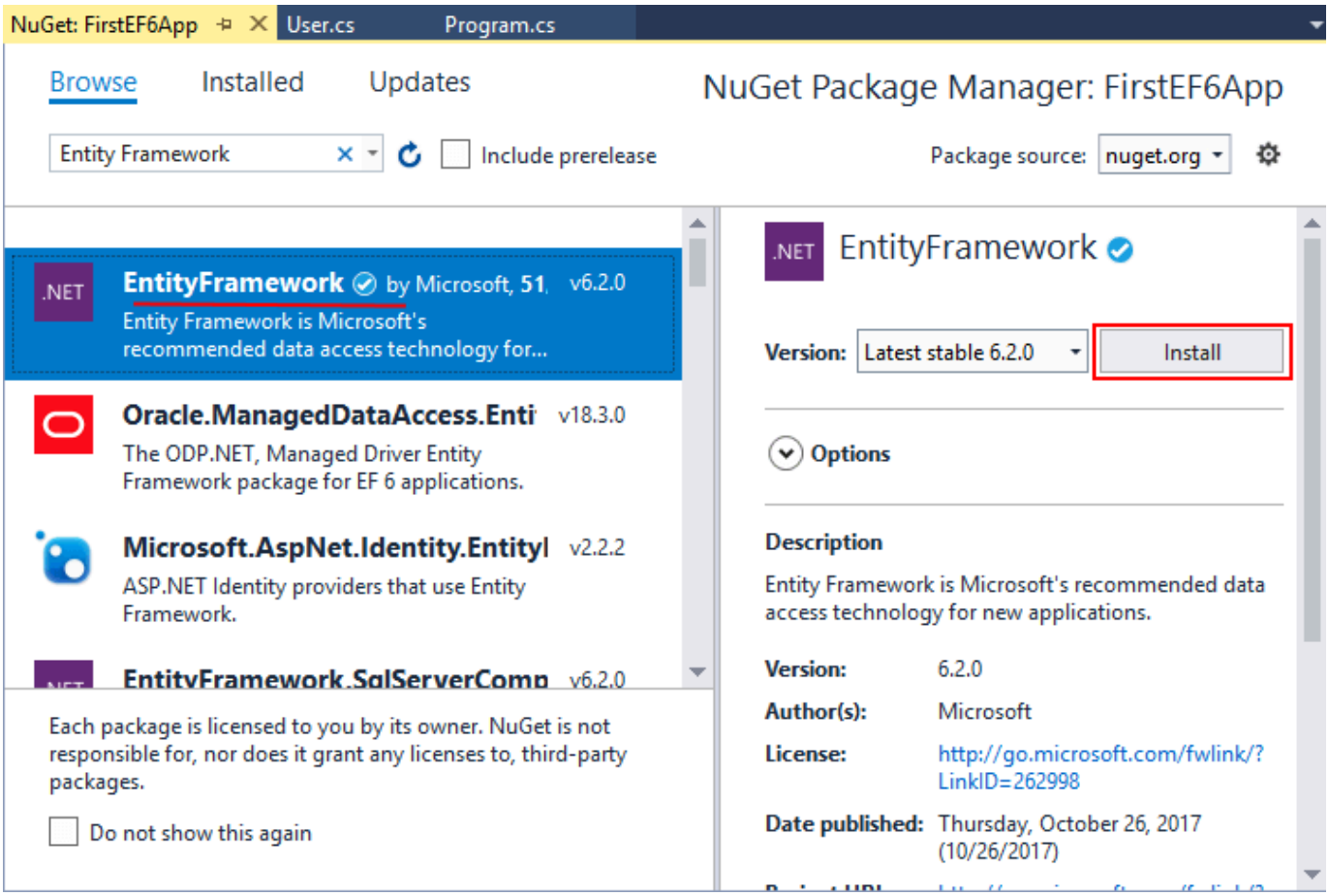
Это обычный класс, который содержит некоторое количество автосвойств. Каждое свойство будет сопоставляться с отдельным столбцом в таблице из бд.

Надо отметить, что Entity Framework при работе с Code First требует определения ключа элемента для создания первичного ключа в таблице в бд. По умолчанию при генерации бд EF в качестве первичных ключей будет рассматривать свойства с именами Id или [Имя_класса]Id (то есть UserId). Если же мы хотим назвать ключевое свойство иначе, то нам нужно будет внести дополнительную логику на c#.

Теперь для взаимодействия с бд нам нужен контекст данных. Это своего рода посредник между бд и классами, описывающими данные. Но, у нас по умолчанию еще не добавлена библиотека для EF. Чтобы ее добавить, нажмем на проект правой кнопкой мыши и выберем в контекстном меню Manage NuGet Packages...:



Затем в появившемся окне управления NuGet-пакетами в окне поиска введем слово "Entity" и выберем пакет собственно Entity Framework и установим его:



После установки пакета добавим в проект новый класс UserContext:

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

namespace FirstEF6App
{
    class UserContext : DbContext
    {
        public UserContext()
            :base("DbConnection")
        { }

        public DbSet<User> Users { get; set; }
    }
}
```

Основу функциональности Entity Framework составляют классы, находящиеся в пространстве имен *System.Data.Entity*. Среди всего набора классов этого пространства имен следует выделить следующие:

- **DbContext**: определяет контекст данных, используемый для взаимодействия с базой данных.
- **DbModelBuilder**: сопоставляет классы на языке C# с сущностями в базе данных.

- **DbSet/DbSet<TEntity>**: представляет набор сущностей, хранящихся в базе данных

В любом приложении, работающим с БД через Entity Framework, нам нужен будет контекст (класс производный от DbContext) и набор данных DbSet, через который мы сможем взаимодействовать с таблицами из БД. В данном случае таким контекстом является класс UserContext.

В конструкторе этого класса вызывается конструктор базового класса, в который передается строка "DbConnection" - это имя будущей строки подключения к базе данных. В принципе мы можем не использовать конструктор, тогда в этом случае строка подключения носила бы имя самого класса контекста данных.

И также в классе определено одно свойство Users, которое будет хранить набор объектов User. В классе контекста данных набор объектов представляет класс `DbSet<T>`. Через это свойство будет осуществляться связь с таблицей объектов User в бд.

И теперь нам надо установить подключение к базе данных. Для установки подключения обычно используется файл конфигурации приложения. В проектах для десктопных приложений файл конфигурации называется *App.config* (как в нашем случае), в проектах веб-приложений - *web.config*. В нашем случае, поскольку у нас консольное приложение, это файл **App.config**. После добавления Entity Framework он выглядит примерно следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>

    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.En
  <!-- For more information on Entity Framework configuration, visit http://go.micr
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.1" />
  </startup>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnec
      <parameters>
        <parameter value="mssqllocaldb" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlS
    </providers>
  </entityFramework>
</configuration>
```

Содержимое файла в каждом конкретном случае может отличаться. Но в любом случае после добавления EntityFramework в проект в нем будет содержаться элемент configSections. И **после** закрывающего тега `</configSections>` добавим следующий элемент:

```
<connectionStrings>
   <add name="DBConnection" connectionString="data source=(localdb)\MSSQLLocalDB;Ini
providerName="System.Data.SqlClient"/>
 </connectionStrings>
```

Все подключения к источникам данных устанавливаются в секции `connectionStrings`, а каждое отдельное подключение представляет элемент `add`. В конструкторе класса контекста UserContext мы передаем в качестве названия подключения строку "DbConnection", поэтому данное название указывается в атрибуте `name="DBConnection"`.

Настройку строки подключения задает атрибут `connectionString`. В данном случае мы устанавливаем название базы данных, с которой будем взаимодействовать - userstore.

Теперь перейдем к файлу **Program.cs** и изменим его содержание следующим образом:

```
using System;

namespace FirstEF6App
{
    class Program
    {
        static void Main(string[] args)
        {
            using(UserContext db = new UserContext())
            {
                // создаем два объекта User
                User user1 = new User { Name = "Tom", Age = 33 };
                User user2 = new User { Name = "Sam", Age = 26 };

                // добавляем их в бд
                db.Users.Add(user1);
                db.Users.Add(user2);
                db.SaveChanges();
                Console.WriteLine("Объекты успешно сохранены");

                // получаем объекты из бд и выводим на консоль
                var users = db.Users;
                Console.WriteLine("Список объектов:");
                foreach(User u in users)
                {
                    Console.WriteLine("{0}.{1} - {2}", u.Id, u.Name, u.Age);
                }
            }
```

```
            Console.Read();
        }
    }
}
```

Так как класс UserContext через родительский класс DbContext реализует интерфейс `IDisposable`, то для работы с UserContext с автоматическим закрытием данного объекта мы можем использовать конструкцию `using`.

В конструкции `using` создаются два объекта User и добавляются в базу данных. Для их сохранения нам достаточно использовать метод `Add`: `db.Users.Add(user1)`

Чтобы получить список данных из бд, достаточно воспользоваться свойством Users контекста данных: `db.Users`

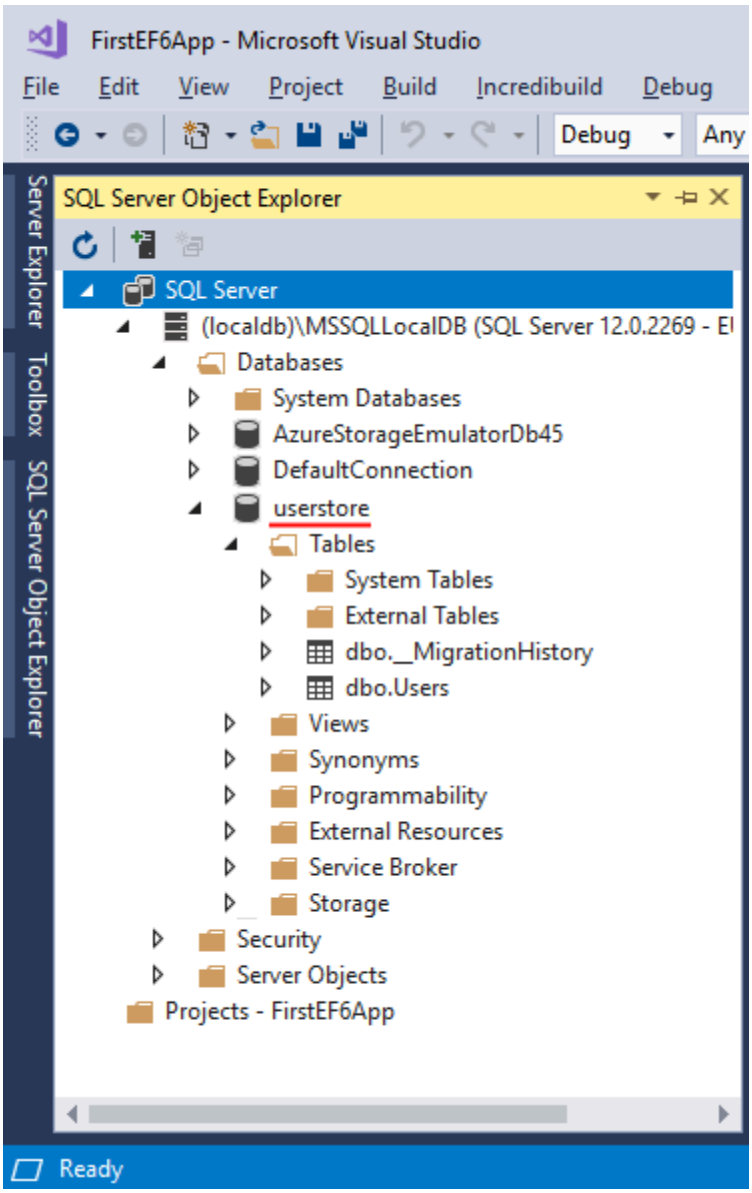
В результате после запуска программа выведет на консоль:

```
Объекты успешно сохранены
Список объектов:
1.Tom - 33
2.Sam - 26
```

Таким образом, Entity Framework обеспечивает простое и удобное управление объектами из базы данных. При том в данном случае нам не надо даже создавать базу данных и определять в ней таблицы. Entity Framework все сделает за нас на основе определения класса контекста данных и классов моделей. И если база данных уже имеется, то EF не будет повторно создавать ее.

Наша задача - только определить модель, которая будет храниться в базе данных, и класс контекста. Поэтому данный подход называется **Code First** - сначала пишется код, а потом по нему создается база данных и ее таблицы.

Возникает вопрос, а где же находится БД? Чтобы физически увидеть базу данных, мы можем подключиться к ней из Visual Studio через окно **View->SQL Server Object Explorer**. После этого мы можем увидеть в SQL Server Object Explorer созданную базу данных, посмотреть ее строение, таблицы, открыть и даже изменить данные в таблицах:



Физически база данных по умолчанию будет располагаться в папке пользователя, в частности, у меня она размещена в каталоге *C:\Users\Eugene*, только к ее названию буде добавляться стандартное расширение mdf - userstore.mdf.

Назад **Содержание** **Вперед**

Телеграмм

Вконтакте | Телеграм | Донаты/Помощь сайту

Contacts: metanit22@mail.ru