PROJECT TYPES

Console

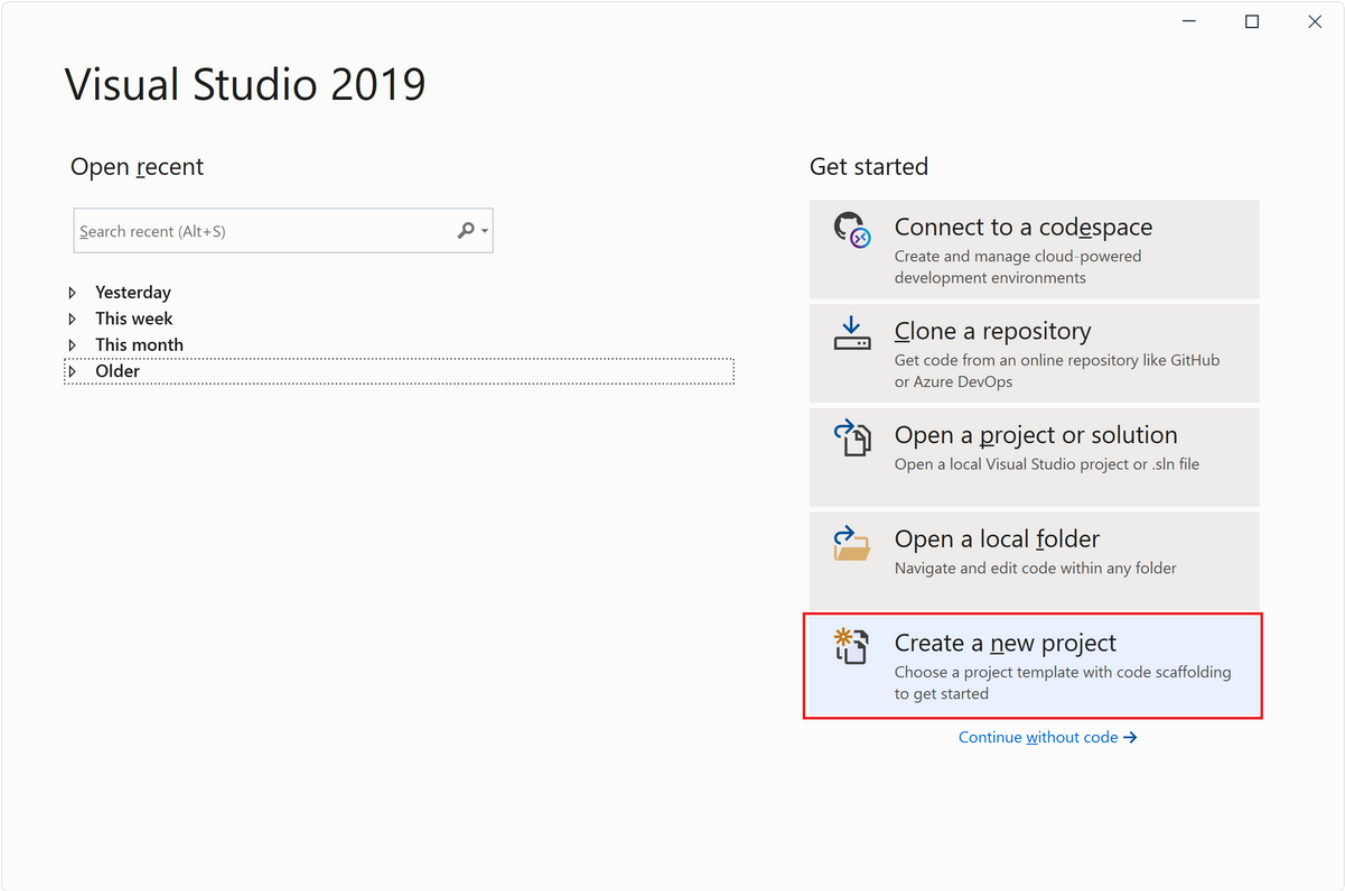
Console

A **Console application** is a program designed to be used via a text-only computer interface, such as a text terminal, the command line interface, etc.

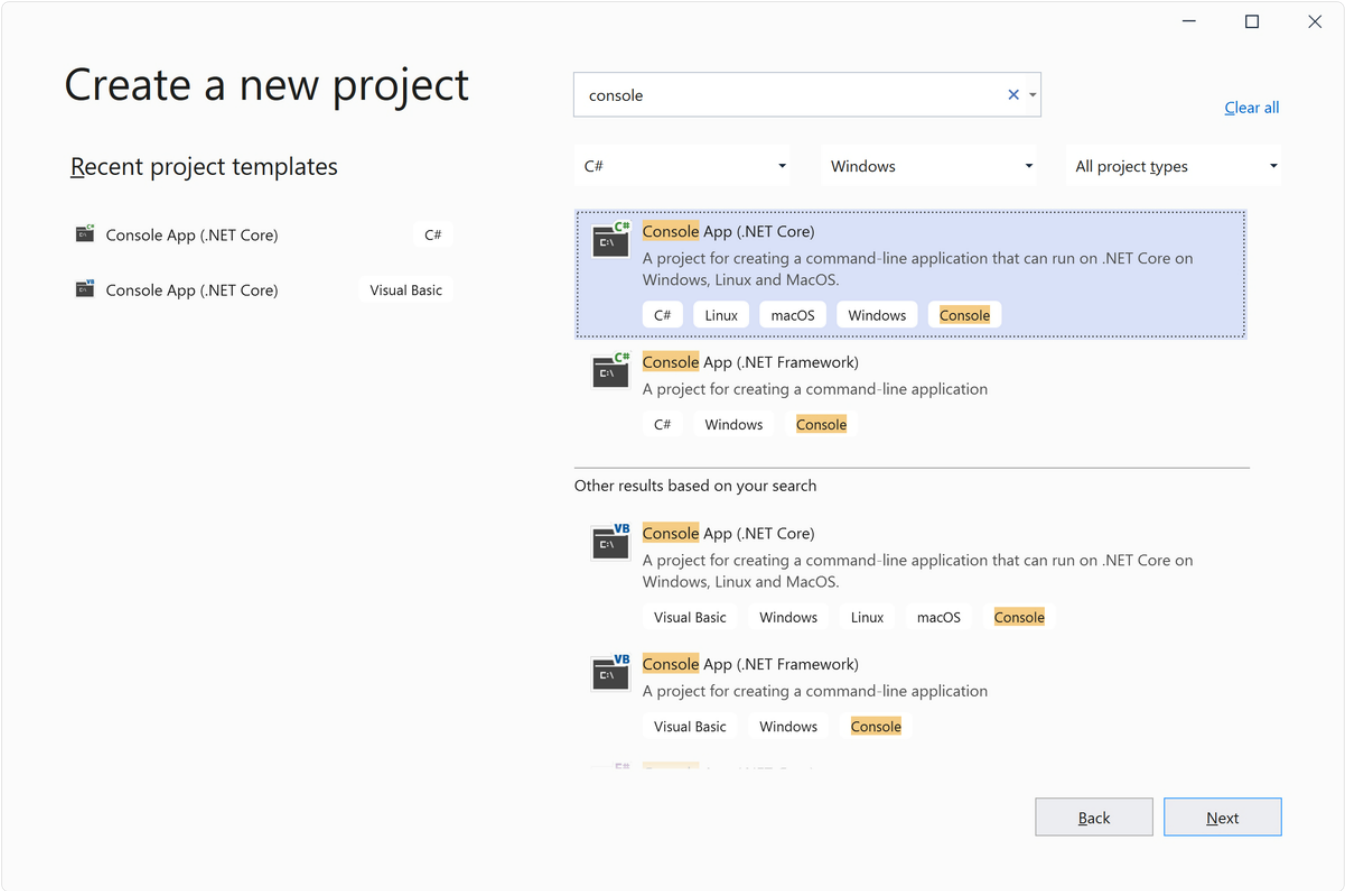
- A user typically interacts with a console application using only a keyboard and display screen, as opposed to GUI applications, which normally require the use of a mouse or other pointing device.
- Many console applications such as command-line interpreters are command-line tools, but numerous text-based user interface (TUI) programs also exist.

Create a Console App

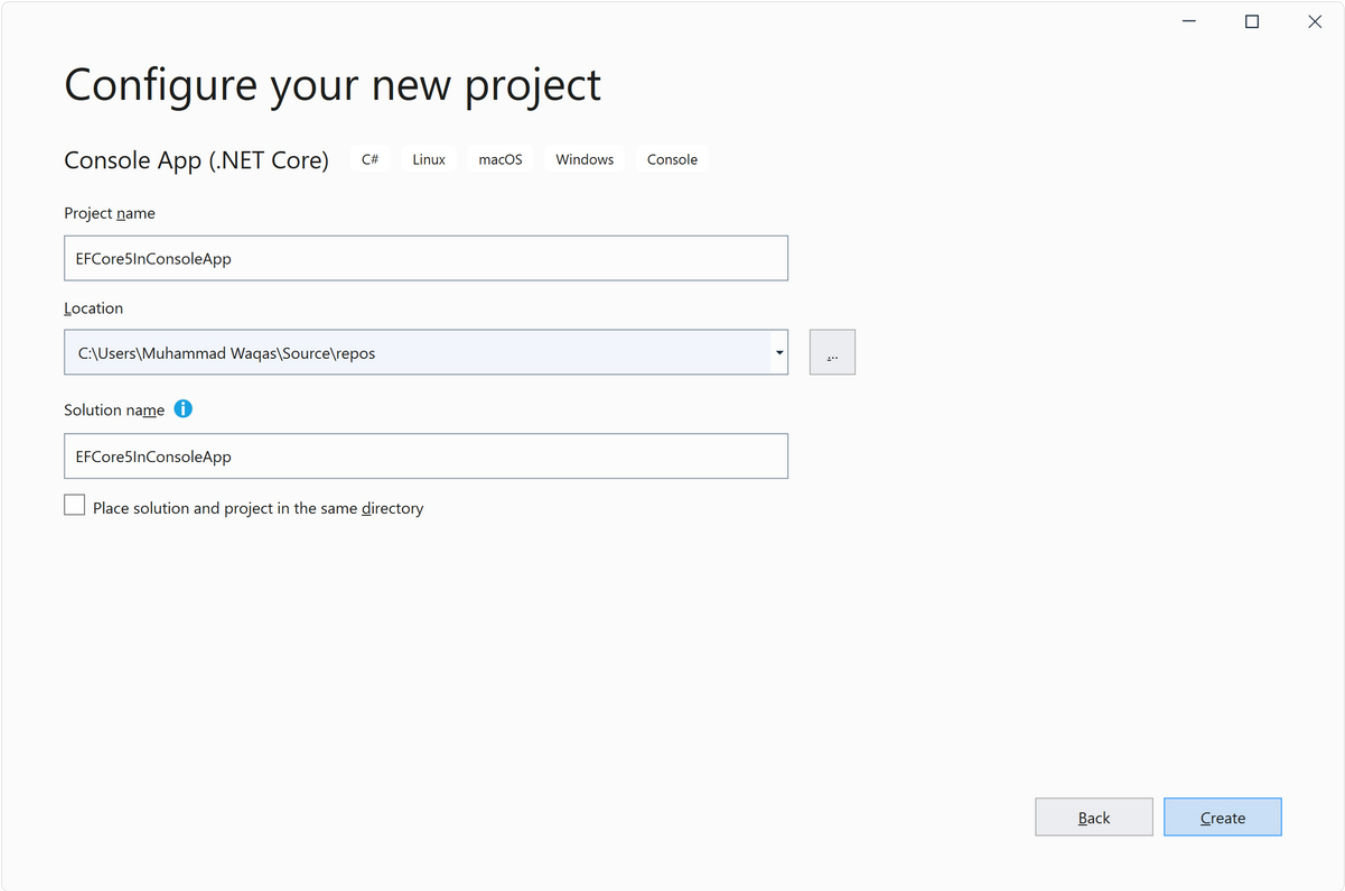
To start, we will create a Console application project. The project type comes with all the template files you will need before adding anything. Let's open Visual Studio 2019. If you haven't already installed Visual Studio, go to the Visual Studio downloads page to install it for free.



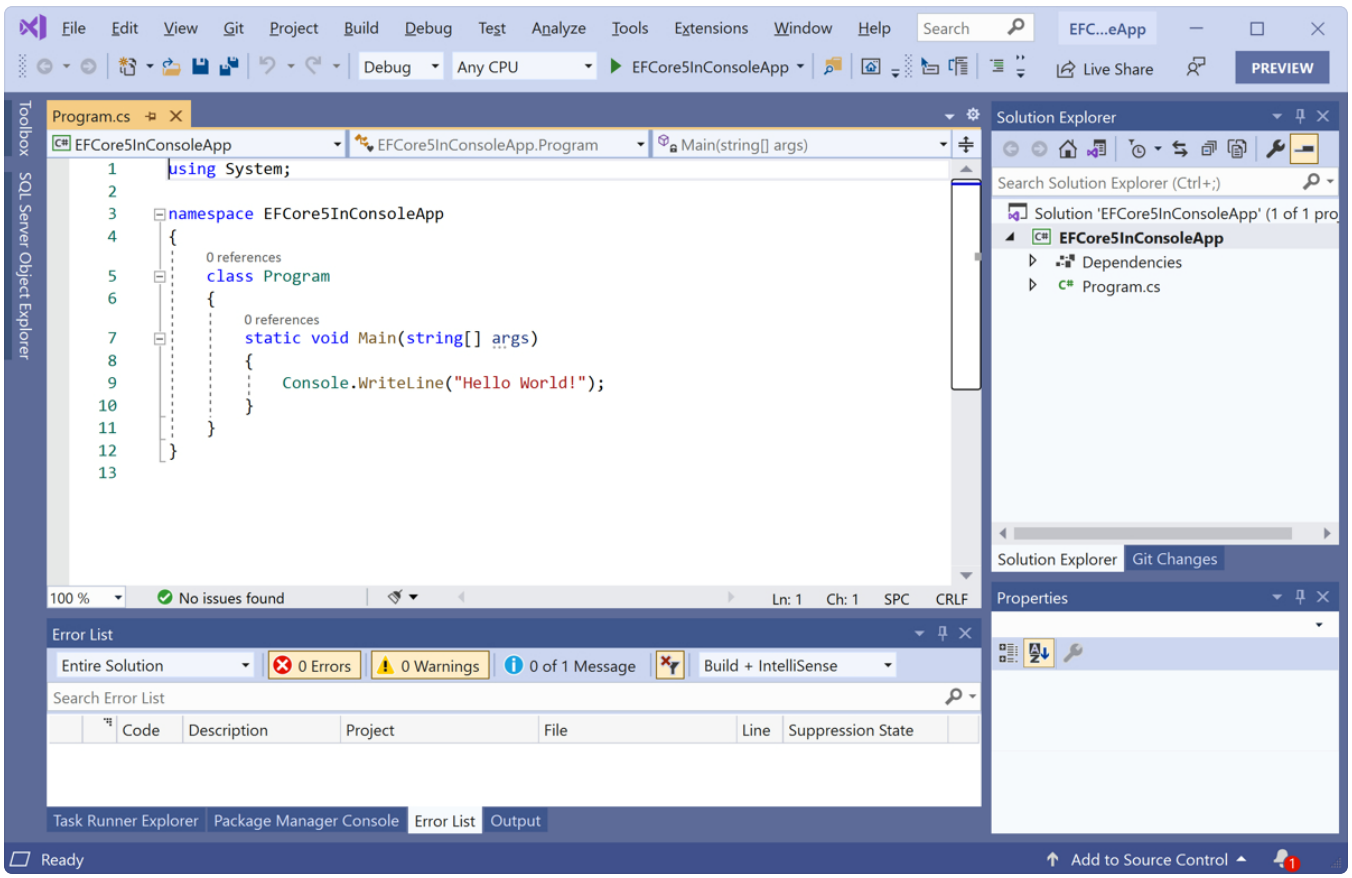
On the start window, choose **Create a new project**.



On the **Create a new project** window, enter or type *console* in the search box. Next, choose **C#** from the Language list, and then choose **Windows** from the Platform list. Select the **Console App (.NET Core)** template, and then choose **Next**.



In the **Configure your new project** window, type or enter ***EFCore5InConsoleApp*** in the **Project name** box and click on the **Create** button.



Visual Studio opens your new project and includes the default "Hello World" code in your project.

Install Entity Framework Core

To use Entity Framework Core we need to install Microsoft.EntityFrameworkCore library. It is available as a nuget package and you can install it using **Nuget Package Manager**.

In the **Package Manager Console** window, enter the following command.

```
PM> Install-Package Microsoft.EntityFrameworkCore
```

For SQL Server LocalDB, which is installed with Visual Studio, we need to install Microsoft.EntityFrameworkCore.SqlServer and will get all the packages required for EF Core.

```
PM> Install-Package Microsoft.EntityFrameworkCore.SqlServer
```

Create a Data Model and Database Context

To create a data model for our application, we will start with the following two entities.

```
public class Author
{
    public int AuthorId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime BirthDate { get; set; }
    public List<Book> Books { get; set; }
}

public class Book
{
    public int BookId { get; set; }
    public string Title { get; set; }
    public Author Author { get; set; }
}
```

The database context class provides the main functionality to coordinate Entity Framework with a given data model. So, let's add a new `BookStore` class which will inherit the `DbContext` class.

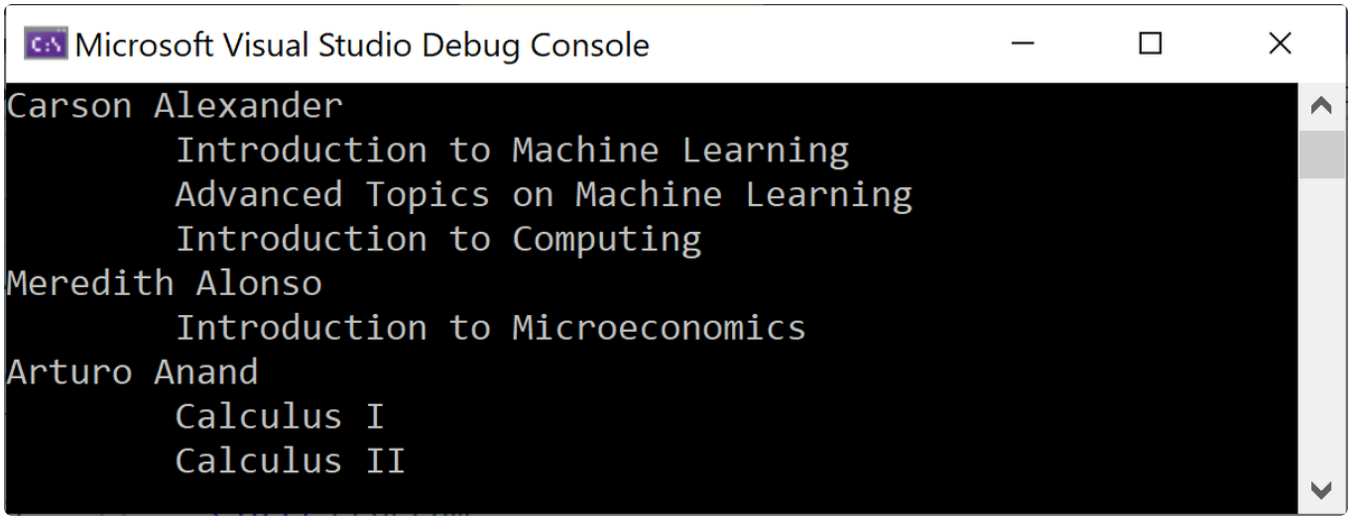
```
public class BookStore : DbContext
{
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilc
    {
        optionsBuilder.UseSqlServer(@"Data Source=(localdb)\ProjectsV13;Initia
    }

    public DbSet<Author> Authors { get; set; }
    public DbSet<Book> Books { get; set; }
}
```

Now, we are done with the required classes and database creation, let's add some authors and book records to the database and then retrieve them as shown below.

```
static void Main(string[] args)
{
    using (var context = new BookStore())
    {
        context.Database.EnsureCreated();

        var authors = new List<Author>
        {
            new Author
            {
                FirstName ="Carson",
                LastName ="Alexander",
                BirthDate = DateTime.Parse("1985-09-01"),
                Books = new List<Book>()
                {
                    new Book { Title = "Introduction to Machine Learning"},
                    new Book { Title = "Advanced Topics on Machine Learning"},
                    new Book { Title = "Introduction to Computing"}
                }
            },
            new Author
            {
                FirstName ="Meredith",
                LastName ="Alonso",
                BirthDate = DateTime.Parse("1970-09-01"),
                Books = new List<Book>()
                {
                    new Book { Title = "Introduction to Microeconomics"}
                }
            },
            new Author
            {
                FirstName ="Arturo",
                LastName ="Anand",
                BirthDate = DateTime.Parse("1963-09-01"),
                Books = new List<Book>()
                {
                    new Book { Title = "Calculus I"},
                    new Book { Title = "Calculus II"}
                }
            }
        };

        context.Authors.AddRange(authors);
        context.SaveChanges();
    }
```

```
    using (var context = new BookStore())
    {
        var list = context.Authors
            .Include(a => a.Books)
            .ToList();

        foreach (var author in list)
        {
            Console.WriteLine(author.FirstName + " " + author.LastName);

            foreach (var book in author.Books)
            {
                Console.WriteLine("\t" + book.Title);
            }
        }
    }
}
```

If you run the application, you will see that authors and books are successfully inserted into the database and also print on the console window.



References

- EF Core Console Application

Previous
Project Types

Next
MVC

Last updated 1 year ago