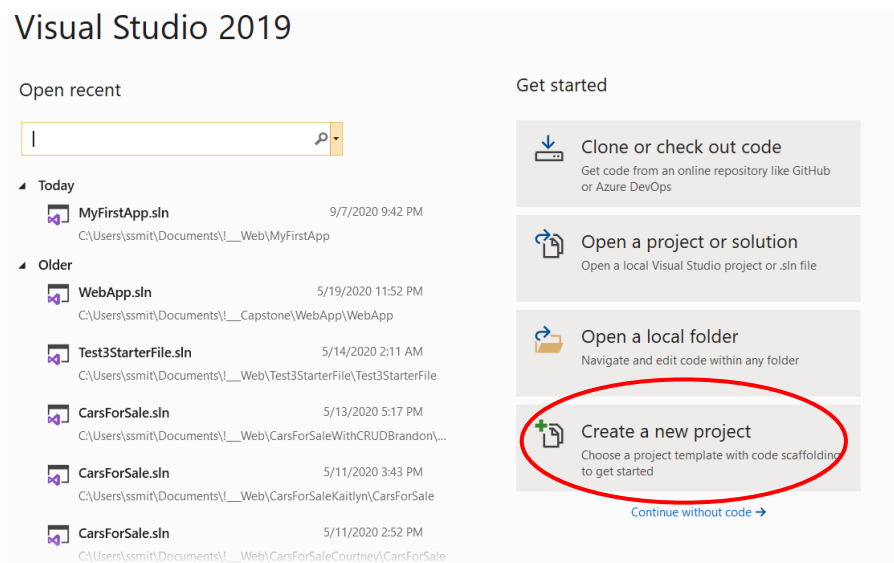


# Creating a Model in Visual Studio

## Demo

Let's create a Model class within an MVC ASP.NET application...

Start by opening Visual Studio 2019.



## Create a new project

Recent project templates

- ASP.NET Web Application (.NET Framework) C#
- Unit Test Project (.NET Framework) C#

Be careful not to choose ASP.NET Core, and be careful to choose C# and not Visual Basic.

Search: asp.net

All Languages All Platforms All Project Types

- ASP.NET Core Web Application
 

Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.

C# Linux macOS Windows Cloud Service Web
- ASP.NET Web Application (.NET Framework)
 

Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

Visual Basic Windows Cloud Web
- ASP.NET Web Application (.NET Framework)
 

Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

C# Windows Cloud Web
- ASP.NET Core Web Application
 

Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.

F# Linux macOS Windows Web
- Blazor App
 

Default framework for creating Blazor apps that run on the browser or on .NET Core

Back Next

## Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Cloud Web

Project name

MyFirstApp

Location

C:\Users\ssmit\Documents\...\\_Web\

Solution name ⓘ

MyFirstApp

☐ Place solution and project in the same directory

Framework

.NET Framework 4.7.2

Back Create

This will be set to the same name as the project and that is usually what we want. We'll talk later about what a solution is.

## Create a new ASP.NET Web Application

**Empty**  
An empty project template for creating ASP.NET applications. This template does not have any content in it.

**Web Forms**  
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

**MVC**  
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

**Web API**  
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

**Single Page Application**  
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

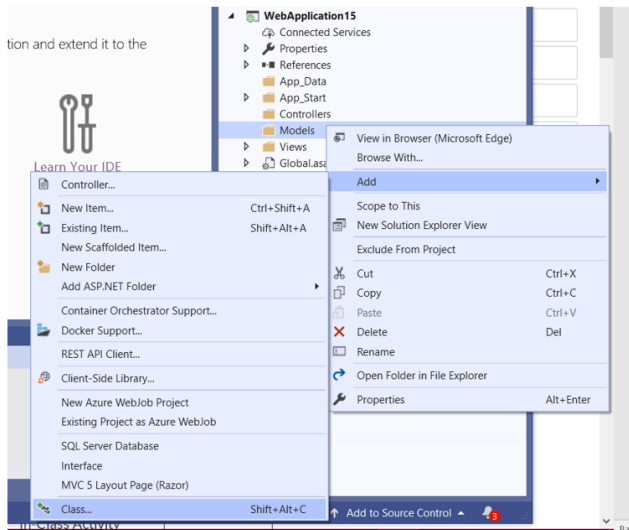
**Authentication**  
No Authentication  
[Change](#)

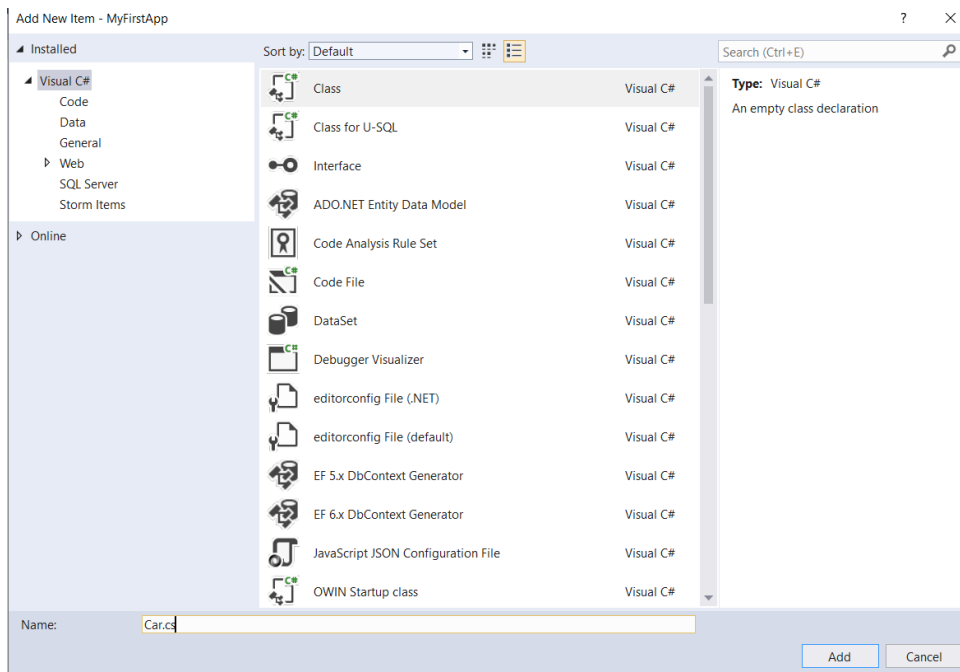
**Add folders & core references**  
☐ Web Forms  
☒ **MVC**  
☐ Web API

**Advanced**  
☐ Configure for HTTPS **← Uncheck**  
☐ Docker support  
(Requires [Docker Desktop](#))  
☐ Also create a project for unit tests

[Back](#)
[Create](#)

Highlight the Model folder at the right of your screen, right-click, choose Add > Class.





## Car.cs **Note: This class is a model for a car object so it is named Car.cs not Cars.cs.**

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

← **using** statements are similar to import and include in Java and C++. They won't be needed for our model class but they are put there by Visual Studio.

```
namespace MyFirstApp.Models
{
    public class Car
    {
        0 references
        public int MyProperty { get; set; }
    }
}
```

← While inside the Car class, you can **type prop and then press TAB twice** and you will get this generic line that includes the syntax needed to create an auto-generated property. You can then go in and change the type if it is not an integer and change the name.

## Car.cs

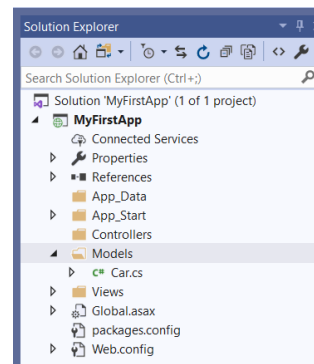
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MyFirstApp.Models
{
    public class Car
    {
        public int CarID { get; set; }
        public string Make { get; set; }
        public string Model { get; set; }
        public decimal Price { get; set; }
        public int Year { get; set; }
        public int Miles { get; set; }
    }
}
```

We'll type in a line for each property that we want in our Car class.

We usually use the *decimal* type in C# for currency.

Now we see Car.cs in the Solution Explorer under the Models folder.



Since this is just a simple demo and we will only create one model class, we won't have to worry about navigation properties.