Visual Studio Template Solutions – Index

# Introduction

Visual studio contains a number of project templates that, when generated, produce helpful starting points for new projects.

Frequently, you will want to look the result of generating one of these project templates. However, it is a pain to generate these projects because:

* You need to select an output location.
  + And the default output location, [C:\Users\David\source\repos\](file:///C:\Users\David\source\repos\), is hard to remember.
* You need to remember where the prior output location was to look at the same generated output.
  + After generating the same console application template, you have ConsoleApp1, ConsoleApp2, ConsoleApp3, …
* When Visual Studio is updated, it updates the project templates! Thus you can regenerate the same template and get a different, confusing result.
* There are many, many templates, and remembering which one you chose can be complicated by the Visual Studio updates adding and removing entries from this list.
* There are multiple options for each template (ex: “Do not use top-level statements”, or “Enable native AOT publish”), and remembering, choosing, or selecting your desired options can be a real pain!
* There is also just the click-work clicking involved in opening a new Visual Studio instance, finding the right template name, choosing options, and then just clicking confirmations buttons.

So, instead of generating and re-generating these templates, wouldn’t it be better if there was a repository of already generated templates? That way the only work involved in seeing the template result is:

* Go to the repository.
* Select the template of interest.
* Double-click to open the solution file.

# Naming

The projects are numbered in the form P####, instead of descriptively named, since there are so many different options (implicit usings, .NET versions, etc.) that the resulting names would be gross, confusing, and difficult to find.

Instead, the simple project numbering allows looking up the projects by criteria in the lists below, and outright listed in the project list below.

Additionally, the projects themselves contain a file (a markdown file, Options.md, in the project directory) providing information on:

* The date of generation.
* The Visual Studio version of generation.
* The project template name.
* Options chosen during project template generation.
  + Including .NET version.

An example is {Project Directory Path}\Options.md:

|  |
| --- |
| # P0001  \* Template: Console App  ## Options  \* Framework: .NET 8.0  \* Do not use top-level statements: false (i.e. use top-level statements)  \* Enable native AOT publish: false    ## Name and Path Options  \* Project name: P0001  \* Solution name: P0001  \* Place solution and project in the same directory: False.    ## Visual Studio  \* Version: 17.8.1  As-Of Date: 20240107 |

Options are listed in terms of relevancy.

# Lists

## .NET 8

* P0001 – Console App, Visual Studio defaults
* P0002 – Console App, NOT using top-level statements
* P0003 – Class Library, .NET 8.0
* P0004 – Razor Class Library, Visual Studio defaults
* sP0005 – Razor Class Library (.NET 8.0, support pages and views)
* P0006 – Windows Forms App, Visual Studio defaults (.NET 8.0)
* P0007 – Windows Forms Class Library, Visual Studio defaults (.NET 8.0)

## .NET 8 – End

## .NET 7

* P0009 – Blazor WebAssembly App Empty

## .NET 7 – End

## Blazor Web App

* P0008 – Blazor Web App, WebAssembly (.NET 8.0, no authentication, configure for HTTPS, include sample pages)
* P0011 – Blazor Web App, WebAssembly global (.NET 8.0)

## Blazor Web App – End

## Console App

* P0001 – Console App, Visual Studio defaults
* P0002 – Console App, NOT using top-level statements
* P0011 – Blazor Web App, WebAssembly global (.NET 8.0)
* P0012 – Blazor Web App, WebAssembly global (.NET 8.0, without example pages)

## Console App – End

## Class Library

* P0003 – Class Library, .NET 8.0

## Class Library – End

## Razor Class Library

* P0004 – Razor Class Library, Visual Studio defaults.
* P0005 – Razor Class Library, support pages and views.

## Razor Class Library – End

## Web

* P0004 – Razor Class Library, Visual Studio defaults (.NET 8.0, do not support pages and views)
* P0005 – Razor Class Library (.NET 8.0, support pages and views)
* P0008 – Blazor Web App, WebAssembly (.NET 8.0, no authentication, configure for HTTPS, include sample pages)
* P0009 – Blazor WebAssembly App Empty (.NET 7.0, HTTPS, ASP.NET Core hosted)
* P0010 – ASP.NET Core Web API (.NET 8.0, )
* P0011 – Blazor Web App, WebAssembly global (.NET 8.0)
* P0012 – Blazor Web App, WebAssembly global (.NET 8.0, without example pages)

## Web – End

## Windows Forms App

* P0006 – Windows Forms App, Visual Studio defaults (.NET 8.0)

## Windows Forms App – End

## Windows Forms Class Library

* P0007 – Windows Forms Class Library, Visual Studio defaults (.NET 8.0)

## Windows Forms Class Library – End

## PXXXX

Here are the projects listed in order:

* P0001 – Console App, Visual Studio defaults (.NET 8, using top-level statements, no native AOT publish)
* P0002 – Console App (.NET 8, NOT using top-level statements)
* P0003 – Class Library, .NET 8.0
* P0004 – Razor Class Library, Visual Studio defaults (.NET 8.0, do not support pages and views)
* P0005 – Razor Class Library (.NET 8.0, support pages and views)
* P0006 – Windows Forms App, Visual Studio defaults (.NET 8.0)
* P0007 – Windows Forms Class Library, Visual Studio defaults (.NET 8.0)
* P0008 – Blazor Web App, WebAssembly (.NET 8.0, no authentication, configure for HTTPS, include sample pages)
* P0009 – Blazor WebAssembly App Empty (.NET 7.0, HTTPS, ASP.NET Core hosted)
* P0010 – ASP.NET Core Web API (.NET 8.0, )
* P0011 – Blazor Web App, WebAssembly global (.NET 8.0)
* P0012 – Blazor Web App, WebAssembly global (.NET 8.0, without example pages)

## PXXXX – End