Tooling in .Net Core

**Abstract**

You can now manage all aspects of working with solutions and projects using the new command-line tooling. The principle command is dotnet. Using this command you can create solutions, create projects and add them to the solution, add project dependencies, add NuGet dependencies—the works!

All this is done so that you no longer need to rely on VS for all your building needs. We now have full integration with MSBuild.

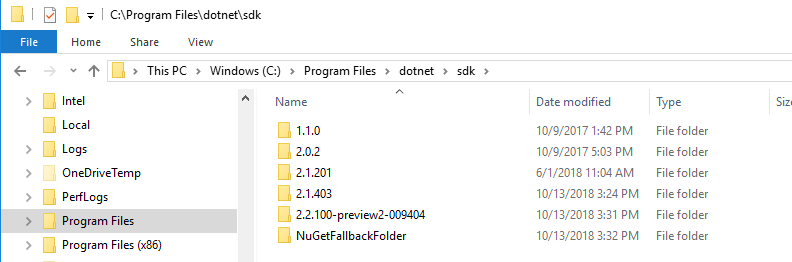
This document covers only the very basics. You’re going to have to google the rest.

# Setup

How to get the tooling:

* All the tooling you’ll need comes when you install VS 2017 (you can probably pick it up with the community edition). You CANNOT use VS 2015!
* You can download it manually:
  + <https://dot.net> 🡺 <https://microsoft.com/net>
  + Look for the sdk.

On Windows, the dot-net sdk gets installed under program files/dotnet/sdk.



The main tooling command is dotnet. (think of it like npm, where you typed npm start, npm test, etc.)

To see what version you’re on, you can use --version.

C:\Git\Projects\DemoProjects>dotnet --version

2.2.100-preview2-009404

You can get help:

dotnet -h

This will give you a quick summary of commands.

# Creating a new project

So, to create a new project, you shell out, and use the dotnet command.

You’ll need to initialize the global package cache, so type this:

dotnet new -h

If it hasn’t been initialized yet, it will do its thing, then it will show you a list of project templates that you can use:

Templates Short Name Language Tags

----------------------------------------------------------------------------------------------------------------------------

Console Application console [C#], F#, VB Common/Console

Class library classlib [C#], F#, VB Common/Library

Unit Test Project mstest [C#], F#, VB Test/MSTest

NUnit 3 Test Project nunit [C#], F#, VB Test/NUnit

NUnit 3 Test Item nunit-test [C#], F#, VB Test/NUnit

xUnit Test Project xunit [C#], F#, VB Test/xUnit

Razor Page page [C#] Web/ASP.NET

MVC ViewImports viewimports [C#] Web/ASP.NET

MVC ViewStart viewstart [C#] Web/ASP.NET

ASP.NET Core Empty web [C#], F# Web/Empty

ASP.NET Core Web App (Model-View-Controller) mvc [C#], F# Web/MVC

ASP.NET Core Web App razor [C#] Web/MVC/Razor Pages

ASP.NET Core with Angular angular [C#] Web/MVC/SPA

ASP.NET Core with React.js react [C#] Web/MVC/SPA

ASP.NET Core with React.js and Redux reactredux [C#] Web/MVC/SPA

Razor Class Library razorclasslib [C#] Web/Razor/Library/Razor Class Library

ASP.NET Core Web API webapi [C#], F# Web/WebAPI

global.json file globaljson Config

NuGet Config nugetconfig Config

Web Config webconfig Config

Solution File sln Solution

Examples:

dotnet new mvc --auth Individual

dotnet new nunit-test

dotnet new --help

When creating a template, you can use the long name (enclosed with quotes), or you can use the short name.

so, pick a template name and a folder location, and away you go

dotnet new reactredux -o AspReactDemo

Before you can build anything, you’ll need to fetch all the dependencies. Use the following command:

dotnet restore

And then you can run it:

dotnet run

Or alternatively you can straight-up build the project:

dotnet build

## Adding packages to your project

Adding packages is pretty simple. Start by going into the same folder as your .csproj file:

dotnet add package <package-name>

## Getting help