



Ground Up Series
ASP.NET CORE 3.X

Syed Awase Khirni

RESEARCHER | ENTREPRENEUR | TECHNOLOGY COACH

@sak008 | sak@sycliq.com/sak@territorialprescience.com | +91. 9035433124

Syed Awase earned his PhD from University of Zurich in GIS, supported by EU V Framework Scholarship from SPIRIT Project (www.geo-spirit.org). He currently provides consulting services through his startup www.territorialprescience.com and www.sycliq.com. He empowers the ecosystem by sharing his technical skills worldwide, since 2008. He provides training in Java Technology Stack, .Net Technology Stack, R, DataScience, Client Side frameworks (Angular, KnockOut, Aurelia, Vue, Ember, Backbone etc..), Node Stack, Machine Learning, Python Stack, Php Stack. Elain Technologies Canada holds copyrights of the material and codeplay source code.

.NET Core Release History

Please read terms of use for authorized access

Original Series

Version	Release Date	Support	Support End Date	
.Net Core 1.0	2016/06/27		2019/06/27	
.Net Core 1.1	2016/11/16		2019/06/27	
.Net Core 2.0	2017/08/14		2018/10/1	
.Net Core 2.1	2018/05/30		2019/08/21	
.Net Core 2.2	2018/12/04		2019/12/23	
.Net Core 3.0	2019/09/23		2020/03/03	
.Net Core 3.1	2019/12/03			
.Net Core 3.2				

Updated on May 2020

<https://github.com/awasekhirni/.NetCORE/tree/master/.NETCORE3.x>

.NET Framework 4.8 Setup

- <https://blogs.windows.com/windowsexperience/2019/05/21/how-to-get-the-windows-10-may-2019-update/#o6c6CPgiYoSk8rGK.97>
- <http://go.microsoft.com/fwlink/?LinkId=2085155>

OS Platform	.NET Framework 4.8 Redistributable	.NET Framework 4.8 Language Pack
Windows 7 SP1/Windows Server 2008 R2	KB4503548	KB4497410
Windows Server 2012	KB4486081	KB4087513
Windows 8.1/Windows Server 2012 R2	KB4486105	KB4087514
Windows 10 Version 1607	KB4486129 (Catalog Only)	KB4087515 (Catalog Only)
Windows 10 Version 1703	KB4486129	KB4087515
Windows Server 2016	KB4486129 (Catalog Only)	KB4087515 (Catalog Only)
Windows 10 Version 1709	KB4486153	KB4087642
Windows 10 Version 1803	KB4486153	KB4087642
Windows Server, version 1803	KB4486153	KB4087642
Windows 10 Version 1809	KB4486153	KB4087642
Windows Server, version 1809	KB4486153 (Catalog Only)	KB4087642 (Catalog Only)
Windows Server 2019	KB4486153 (Catalog Only)	KB4087642 (Catalog Only)

.NET Framework 4.8

Please read terms of use for authorized access

Original Series

- Included in Windows 10 May 2019 update
- Also available on
 - Windows 7+
 - Windows Server 2008 R2+.
- Runtime
 - JIT improvements
 - NGEN improvements
 - Antimalware Scanning for All Assemblies
- BCL
 - Updated Zlib
 - Reducing FIPS impact on Cryptography
- Windows Forms
 - Accessibility Enhancements
 - UIA LiveRegions Support in Labels and StatusStrips
 - UIA Notification Events
 - ToolTips on keyboard access
 - DataGridView control accessible hierarchy changes
- WCF
 - ServiceHealthBehaviour
- WPF
 - Screen narrators no longer announce collapse
 - High DPI enhancements
 - Tooltips on keyboard accessss
 - Added Support for SizeOfSet and PositionInSet Automation Properties

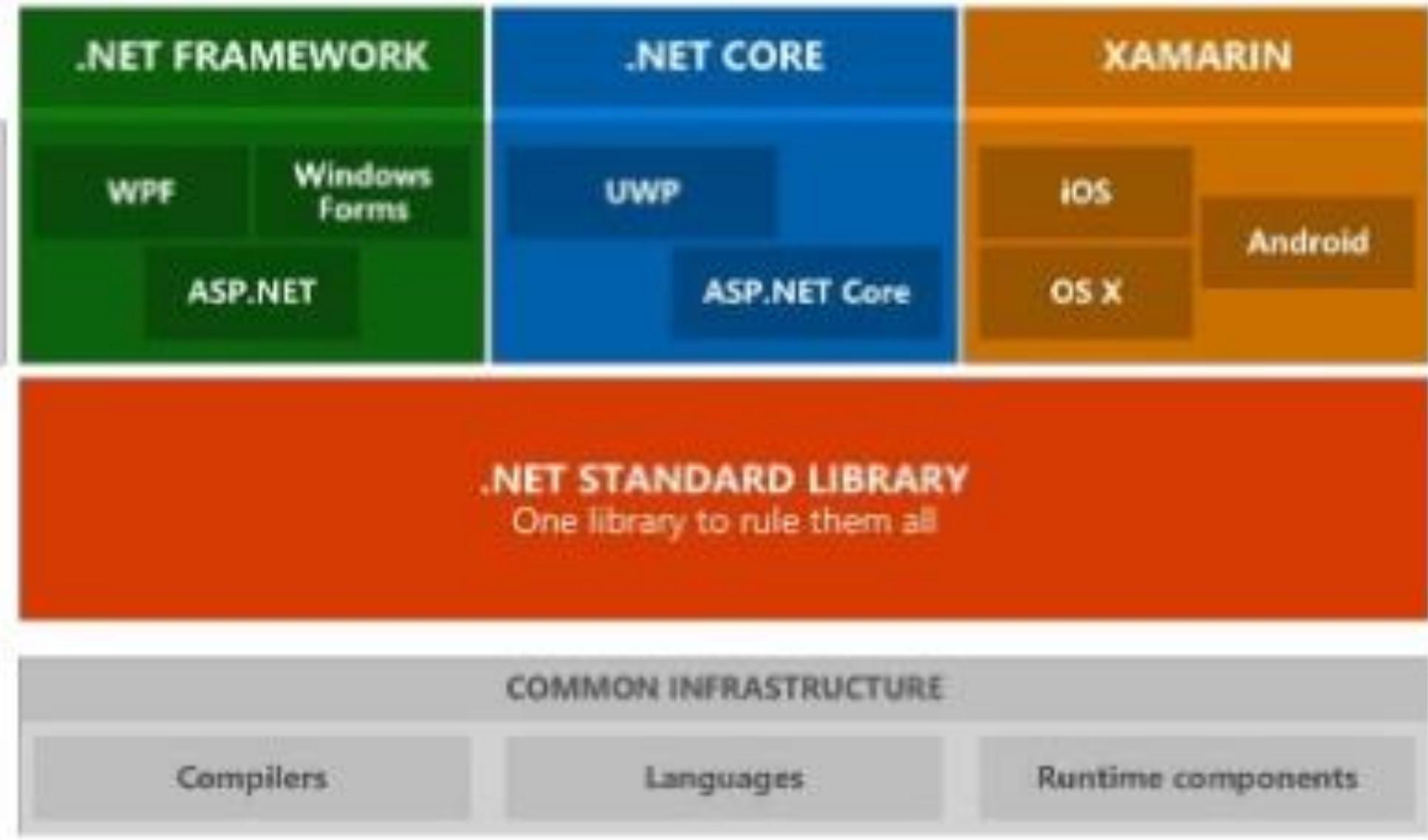
.NET Core 3.x

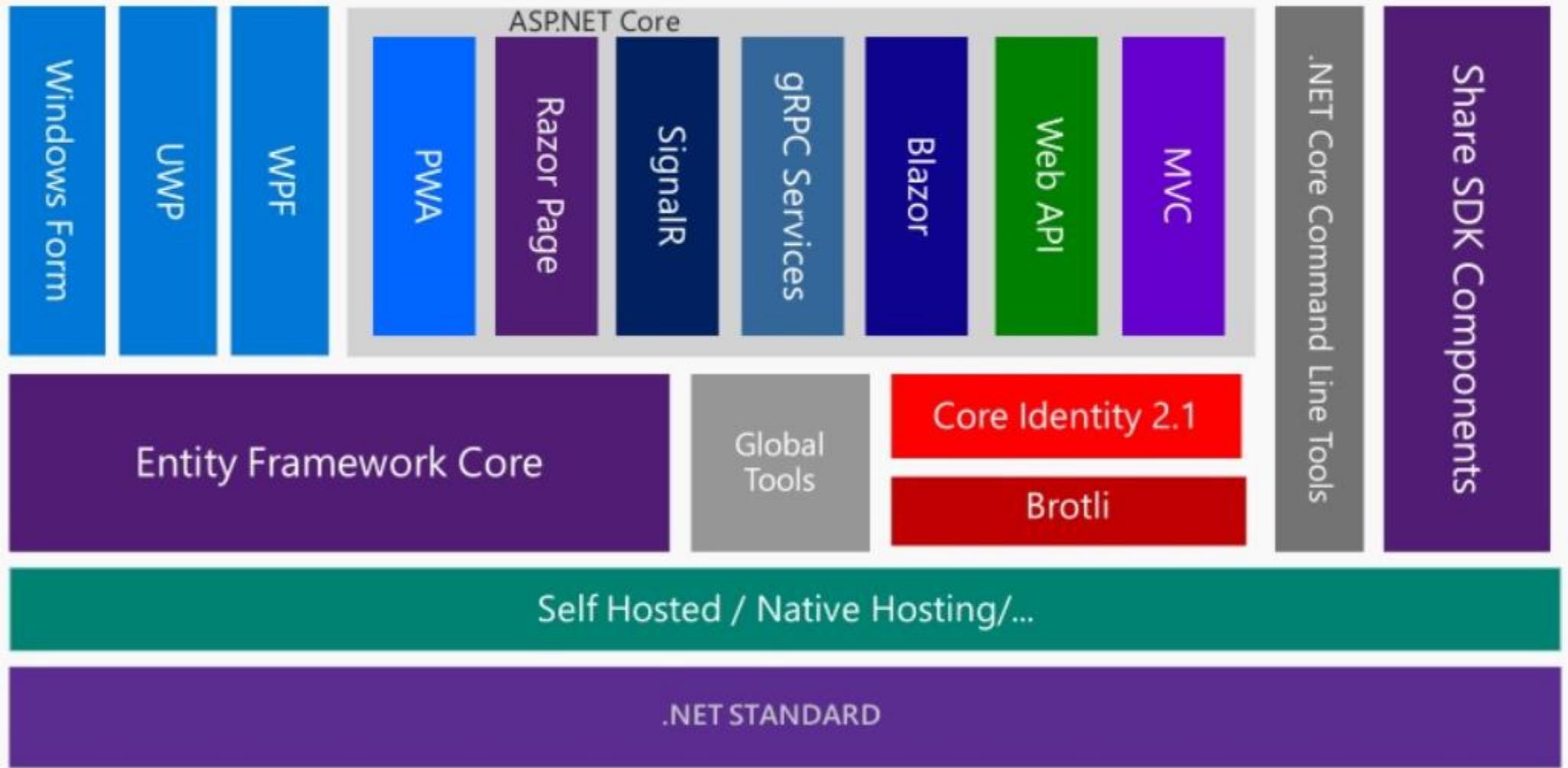
- Free and open source
- C#, F#
- Cross platform support
 - Windows
 - macOS
 - linux

ASP.NET	API, MVC, Razor Pages, Blazor Server, Blazor client, gRPC, SignalR
Windows Forms	
WPF	
Entity Framework	.Net Standard 2.1, x-plat lightweight, C#8 (Async streams, nullable reference types) CosmosDB
WCF	Core WCF, Rebuild to gRPC

Current Platform Requirements

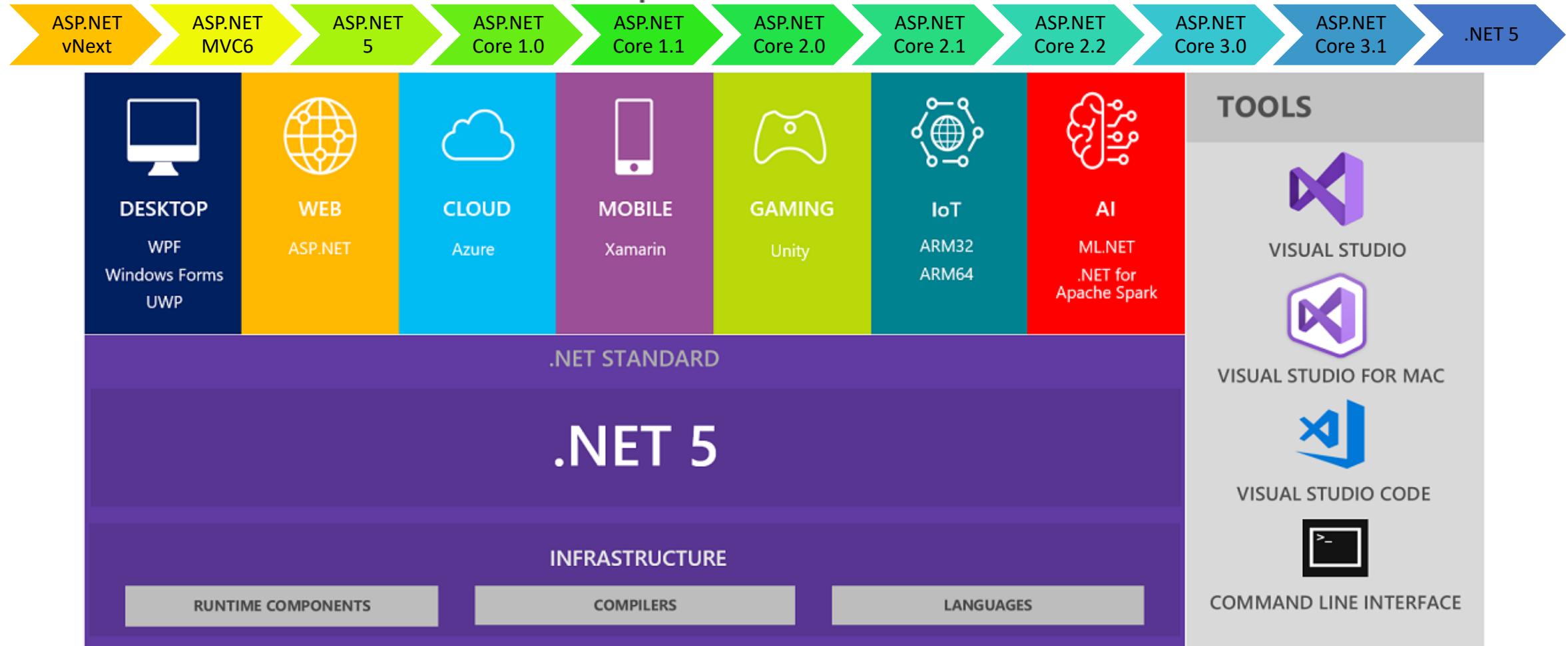
- High Performance Framework/Platform, Highly Secure and Easily Scalable platform.
- Efficient memory management.
- Cross-platform support with CICD – develop once deploy everywhere.
- Support for Machine Learning and AI
- Server Support for HTTP/S + ALL FEATURES
- Encrypted HTTPS connection through SSL
- Browser Support for HTTP/2
- Easy Caching and Identity Management
- High performance RPC services
- Support for WebAssembly.





(source: Microsoft)

.NET – A unified platform



(source: Microsoft)

.NET Schedule



- Please read terms of use for authorized access
- Original Series
- .NET Core 3.0 release in September
 - .NET Core 3.1 = Long Term Support (LTS)
 - .NET 5.0 release in November 2020
 - Major releases every year, LTS for even numbered releases
 - Predictable schedule, minor releases if needed

(source: Microsoft)



ⓘ Not sure what to download? [See recommended downloads for the latest version of .NET](#).

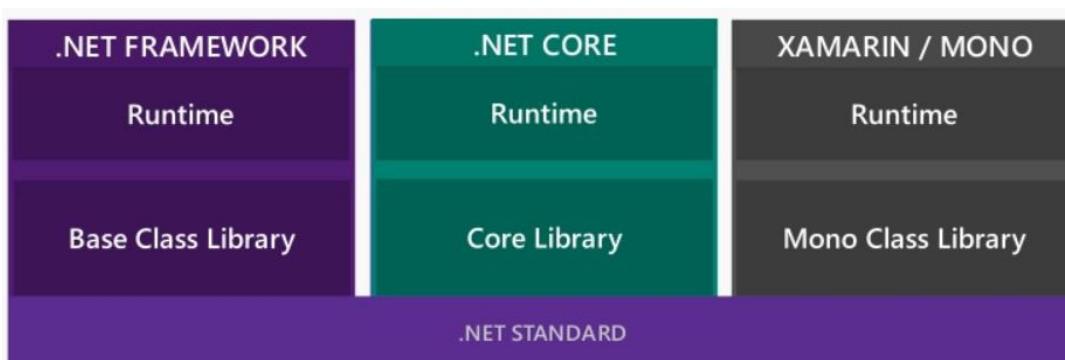
Version	Status	Latest release	Latest release date	End of support
.NET 5.0	Preview ⓘ	5.0.0-preview.7	2020-07-21	
.NET Core 3.1 (recommended)	LTS ⓘ	3.1.7	2020-08-11	2022-12-03
.NET Core 3.0	End of life ⓘ	3.0.3	2020-02-18	2020-03-03
.NET Core 2.2	End of life ⓘ	2.2.8	2019-11-19	2019-12-23
.NET Core 2.1	LTS ⓘ	2.1.21	2020-08-11	2021-08-21
.NET Core 2.0	End of life ⓘ	2.0.9	2018-07-10	2018-10-01
.NET Core 1.1	End of life ⓘ	1.1.13	2019-05-14	2019-06-27
.NET Core 1.0	End of life ⓘ	1.0.16	2019-05-14	2019-06-27

Source: <https://dotnet.microsoft.com/download/dotnet-core>

.NET 5

Please read terms of use for authorized access

Original Series

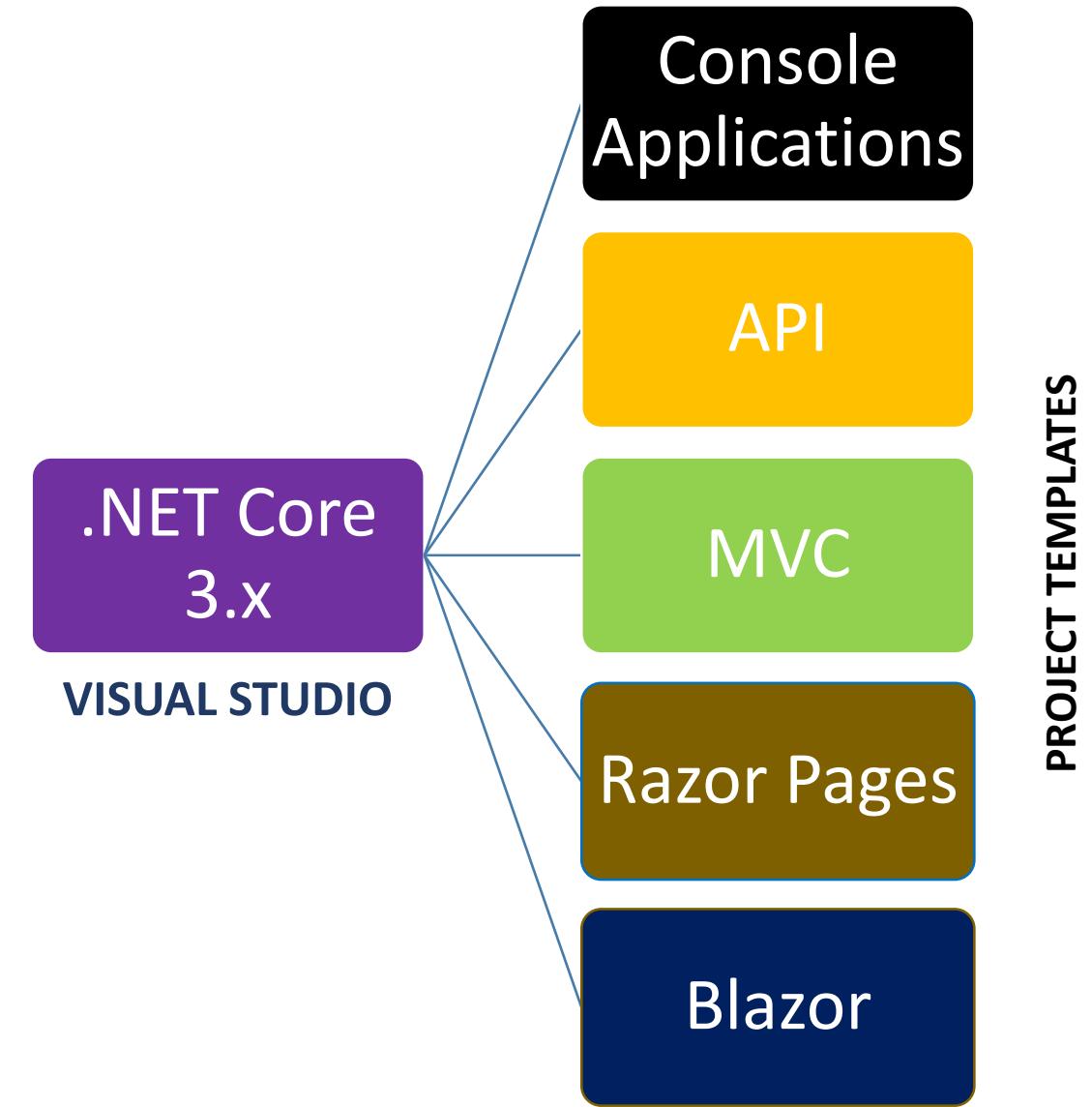


.NET5



ASP.NET CORE 3.0

- .NET Core 3.0 implements .NET standard 2.1
- .NET Core 3.0 adds support for C# 8.0
 - Requires Visual Studio 2019 version 16.3 or newer version.
 - VSCode with latest C# Extension
- New JSON Serializer
 - Uses System.Text.Json by default for JSON serialization
 - Reads and writes JSON asynchronously.
 - Optimized for UTF-8 text.
 - Higher performance than Newtonsoft.JSON
- Worker Service : used for long running tasks such as Windows services or linux Systemd daemons.
- Single EXEs publish
 - Dotnet publish –r win10-x64 /p:PublishSingleFile =true.
- Angular template updated to Angular 8.
- Performance improvements
 - Less memory consumption for using the built-in dependency injection container for scoped services
 - Less memory consumption for websocket connections.
 - Memory reduction and throughput improvements for HTTPS connections
- A high performance open source universal RPC framework – gRPC
- Blazor- a framework for building interactive client-side web UI with .NET
- SignalR now uses System.Text.Json to serialize and deserialize JSON messages
- New SqlClient- System.Data.SqlClient



Setting up ASP.NET Core 3.x

SYED AWASE KHIRNI

<https://dotnet.microsoft.com/download/dotnet/current>

Download .NET Core 3.1

- <https://download.visualstudio.microsoft.com/download/pr/e24a5908-d199-4575-b94b-5828bbbc956e/bc1b7a8c4e06a18780eeb15925aaee9b/dotnet-sdk-3.1.107-win-x64.exe>

SDK 3.1.107

Visual Studio support

Visual Studio 2019 (v16.4)

Included in

Visual Studio 16.4.12

Included runtimes

.NET Runtime 3.1.7

ASP.NET Core Runtime 3.1.7

Desktop Runtime 3.1.7

Language support

C# 8.0

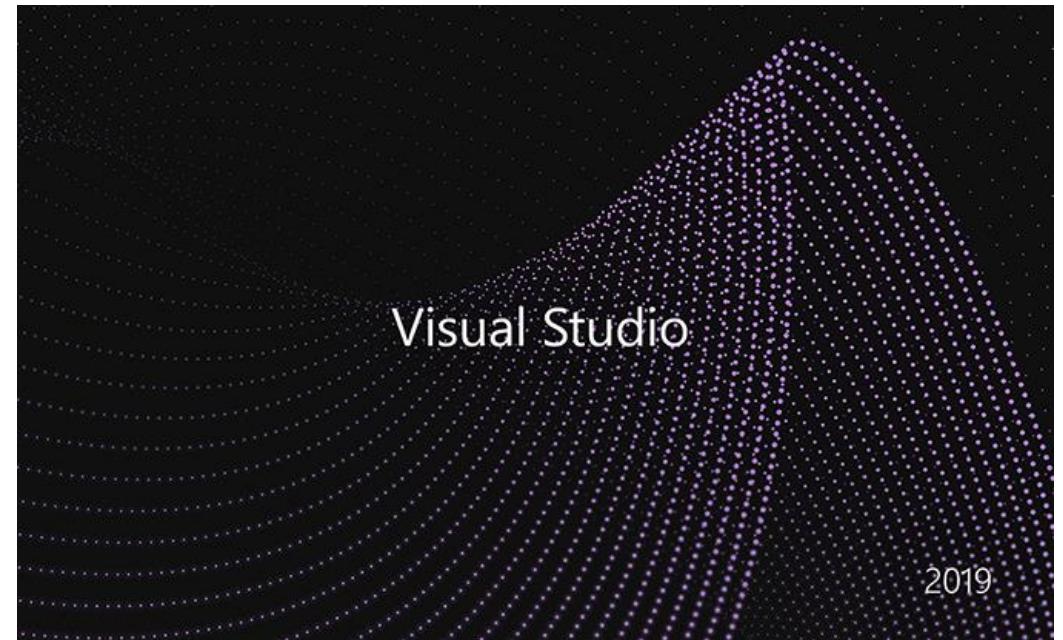
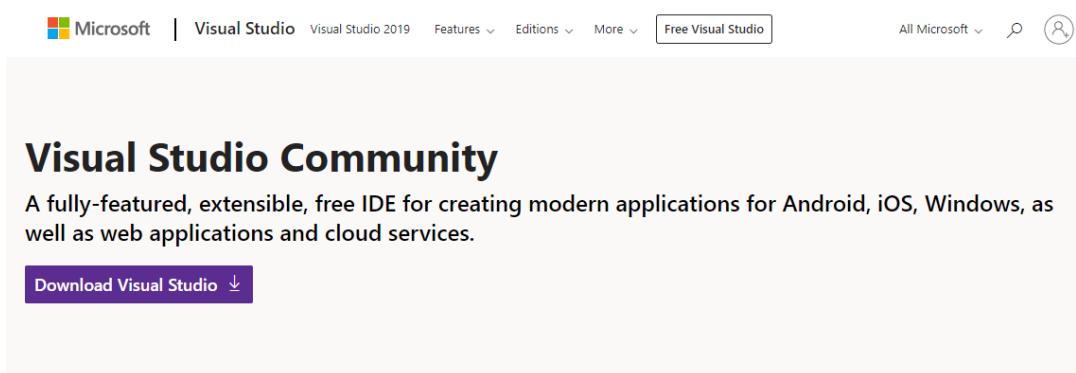
F# 4.7

OS	Installers	Binaries
Linux	Package manager instructions	ARM32 ARM64 x64 Alpine x64 RHEL 6 x64
macOS	x64	x64
Windows	x64 x86	ARM32 x64 x86
All	dotnet-install scripts	

<https://dotnet.microsoft.com/download/dotnet/current>

Visual Studio Community IDE 2019

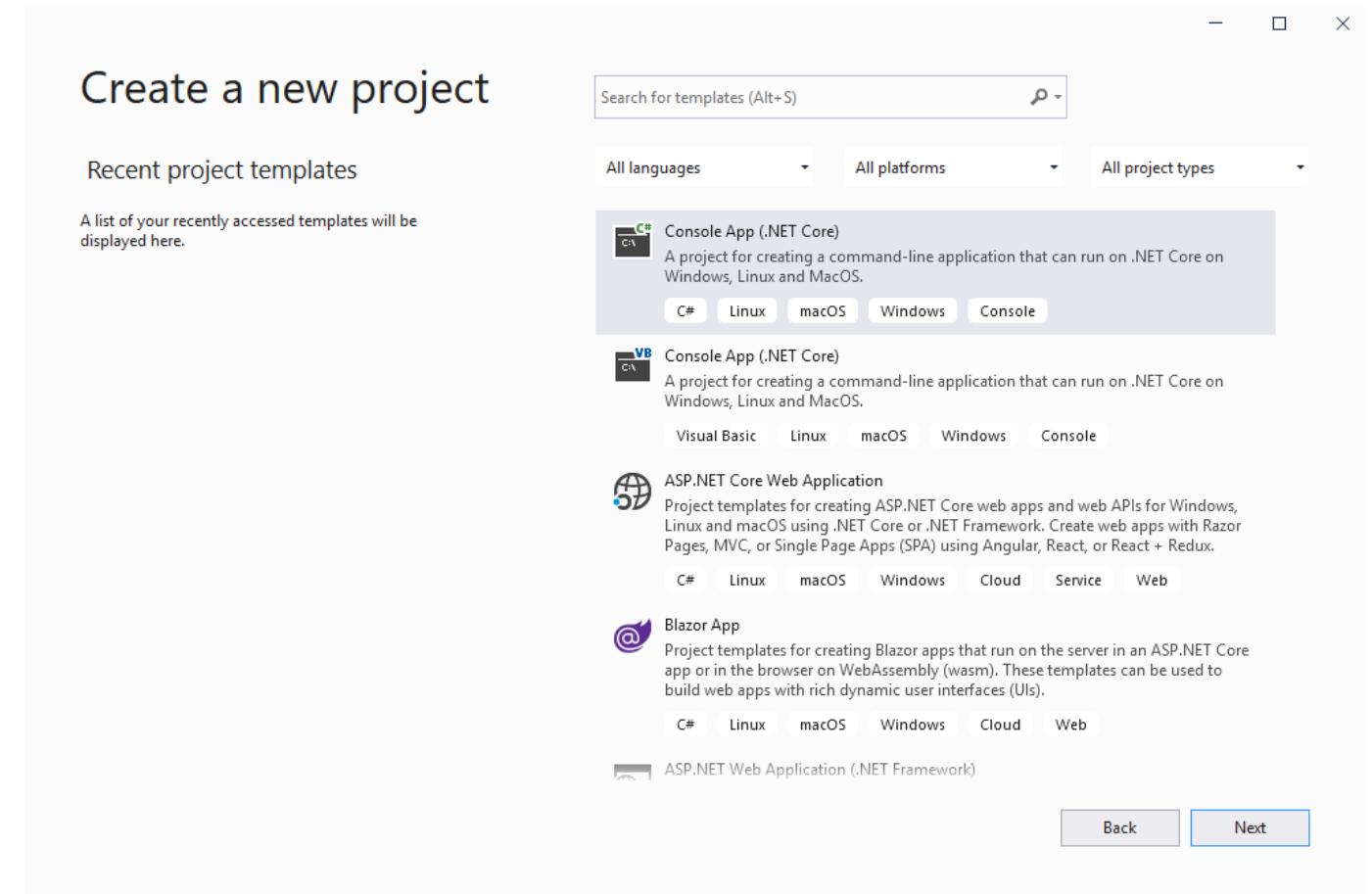
- <https://visualstudio.microsoft.com/vs/community/>
- .NET Core 3.1 works with Visual Studio Community IDE 2019 and above only.



Visual Studio 2019 Community Edition

Please read terms of use for authorized access

Original Series



The screenshot shows the 'Create a new project' dialog in Visual Studio 2019. At the top, there is a search bar labeled 'Search for templates (Alt+S)' and three dropdown menus: 'All languages', 'All platforms', and 'All project types'. Below these are several project template cards:

- Console App (.NET Core)**: A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
Languages: C#, VB
Platforms: Linux, macOS, Windows, Console
- 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.
Languages: C#
Platforms: Linux, macOS, Windows, Cloud, Service, Web
- Blazor App**: Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).
Languages: C#
Platforms: Linux, macOS, Windows, Cloud, Web
- ASP.NET Web Application (.NET Framework)**: This template is partially visible at the bottom.

At the bottom right of the dialog are 'Back' and 'Next' buttons.

SYED AWASE KHIRNI

Example 1: ConsoleApp .NET CORE

Learning Outcomes:

1. Creating a .Net Core 3.x console application using visual studio 2019.

Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

SELECT Console App =>

Search for templates (Alt+S)

All languages All platforms All project types

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Console

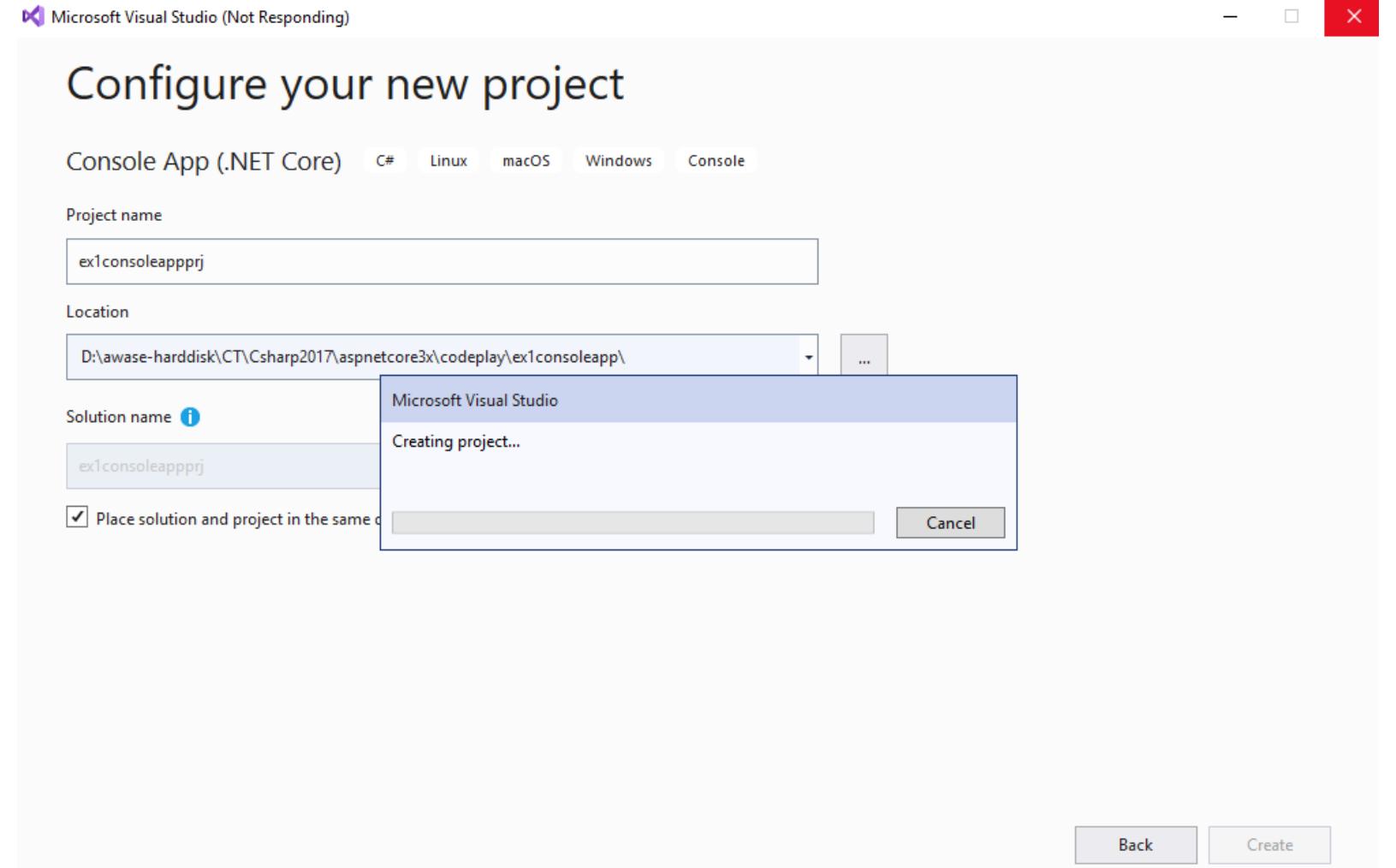
Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
Visual Basic Linux macOS Windows Console

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

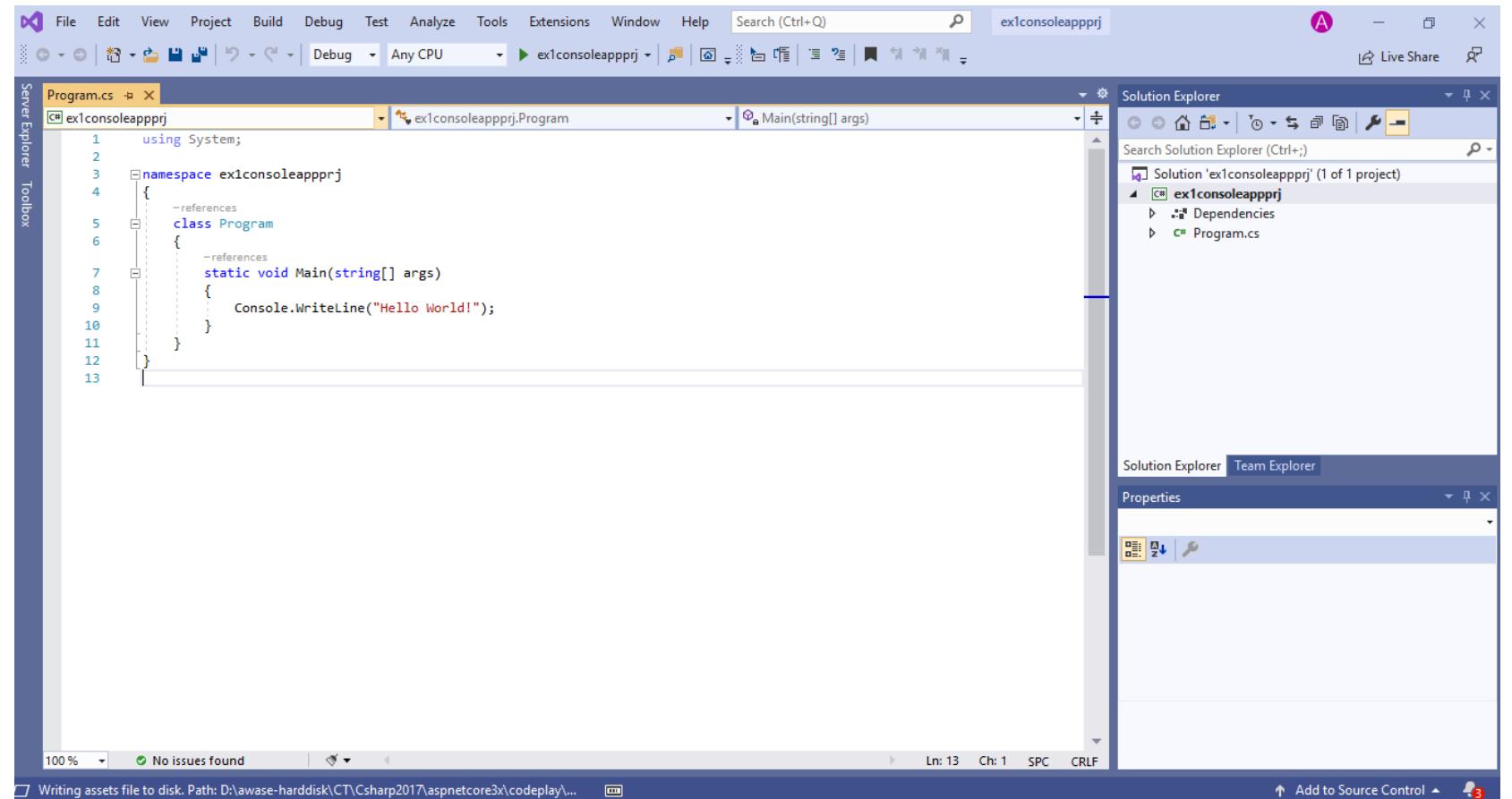
Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web

ASP.NET Web Application (.NET Framework)

Back Next



Example: 1

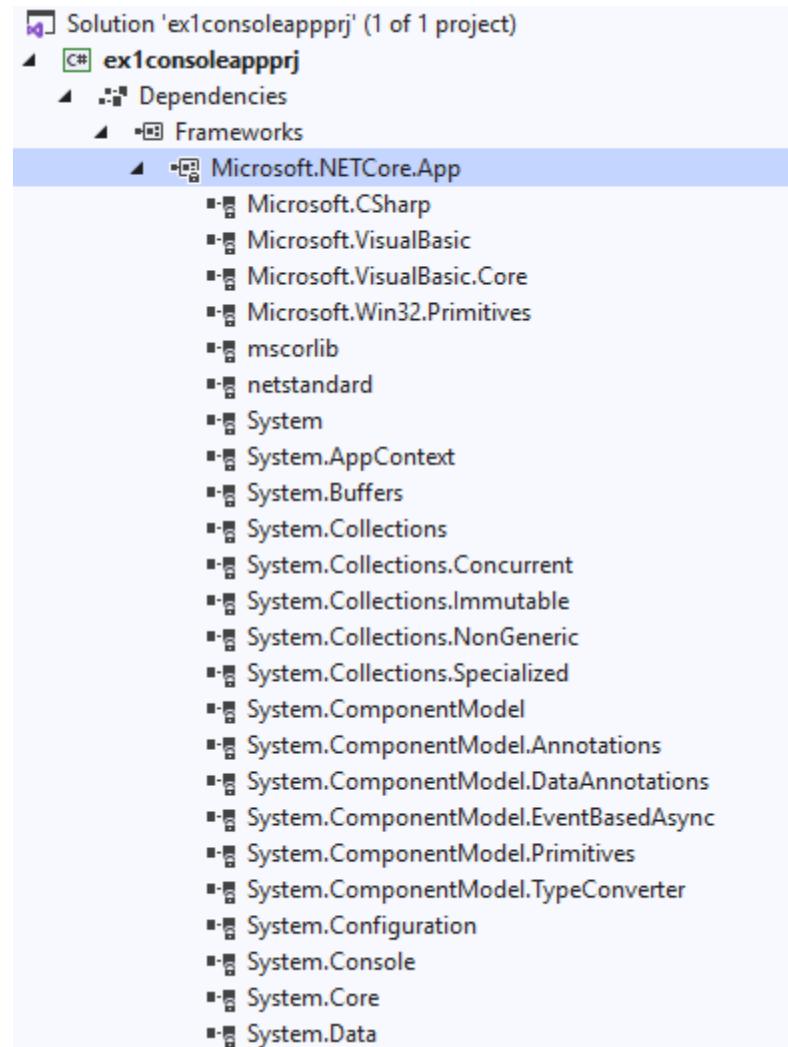


The screenshot shows the Microsoft Visual Studio 2019 interface with a C# console application project named "ex1consoleappprj". The "Program.cs" file contains the following code:

```
1  using System;
2
3  namespace ex1consoleappprj
4  {
5      //references
6      class Program
7      {
8          //references
9          static void Main(string[] args)
10         {
11             Console.WriteLine("Hello World!");
12         }
13     }
}
```

The Solution Explorer shows the project structure with files "Dependencies" and "Program.cs". The Properties window is also visible.

Example:



The screenshot shows a Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output is as follows:

```
Hello World!
D:\awase-harddisk\CT\Csharp2017\aspnetcore3x\codeplay\ex1consoleapp\ex1consoleappprj\bin\Debug\netcoreapp3.1\ex1consoleappprj.exe (process 40068) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

SYED AWASE KHIRNI

Example 2: ASP.NET Core Web Application

Learning Outcomes:

1. Understand the steps to create ASP.NET Core 3.x empty web application
2. Understand the web application structure
3. Understand the key dependencies for ASP.NET Core 3.x empty web application
4. Executing ASP.NET Core 3.x empty web application and rendering it on `http://localhost:44346`

Create a new project

Recent project templates

Search for templates (Alt+S)

All languages ▾ All platforms ▾ All project types ▾

 Console App (.NET Core) C#
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Console

 Console App (.NET Core) C#
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
Visual Basic Linux macOS Windows Console

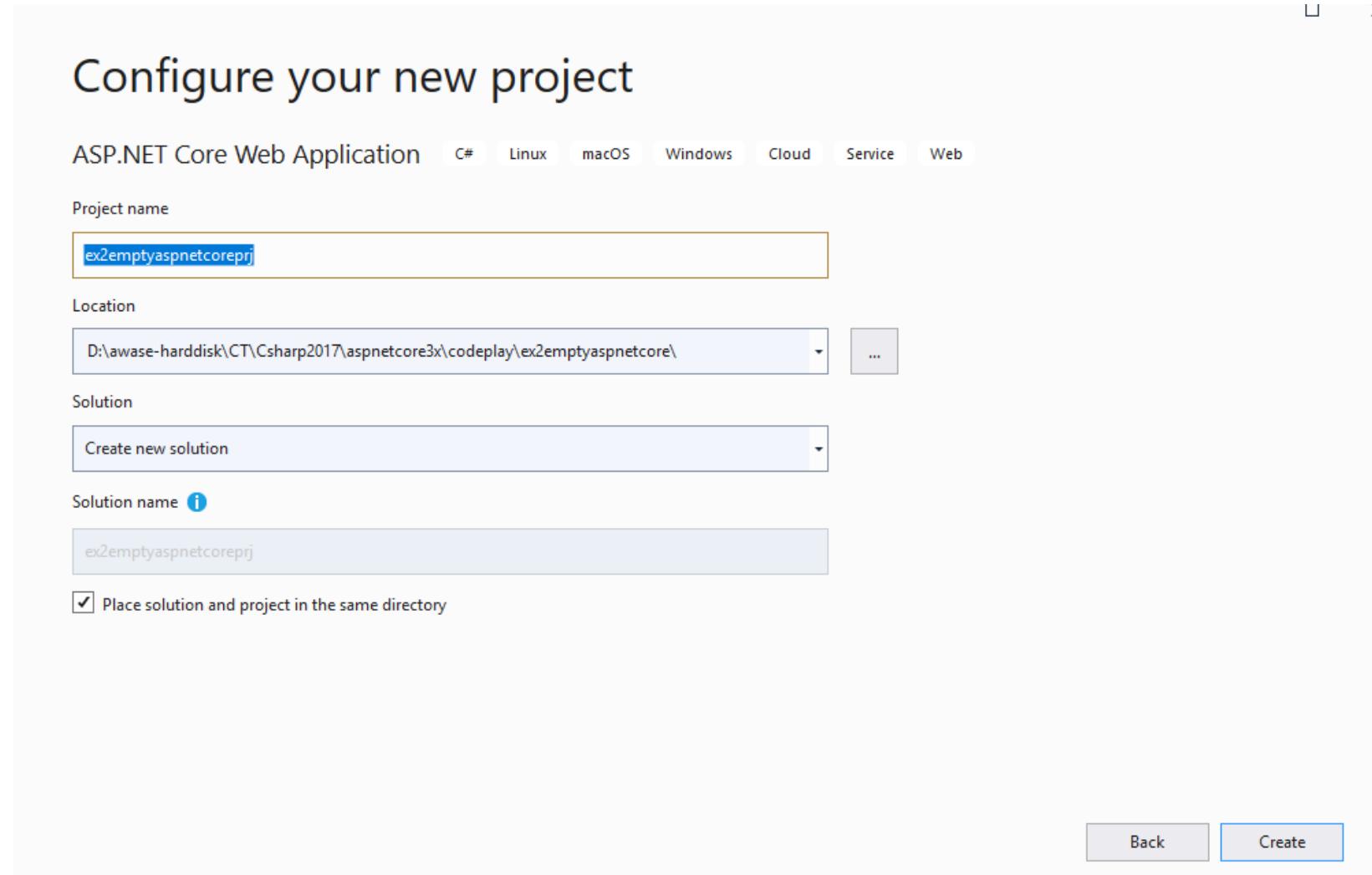
 ASP.NET Core Web Application C#
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.
C# Linux macOS Windows Cloud Service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

 Blazor App C#
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).
Linux macOS Windows Cloud Web

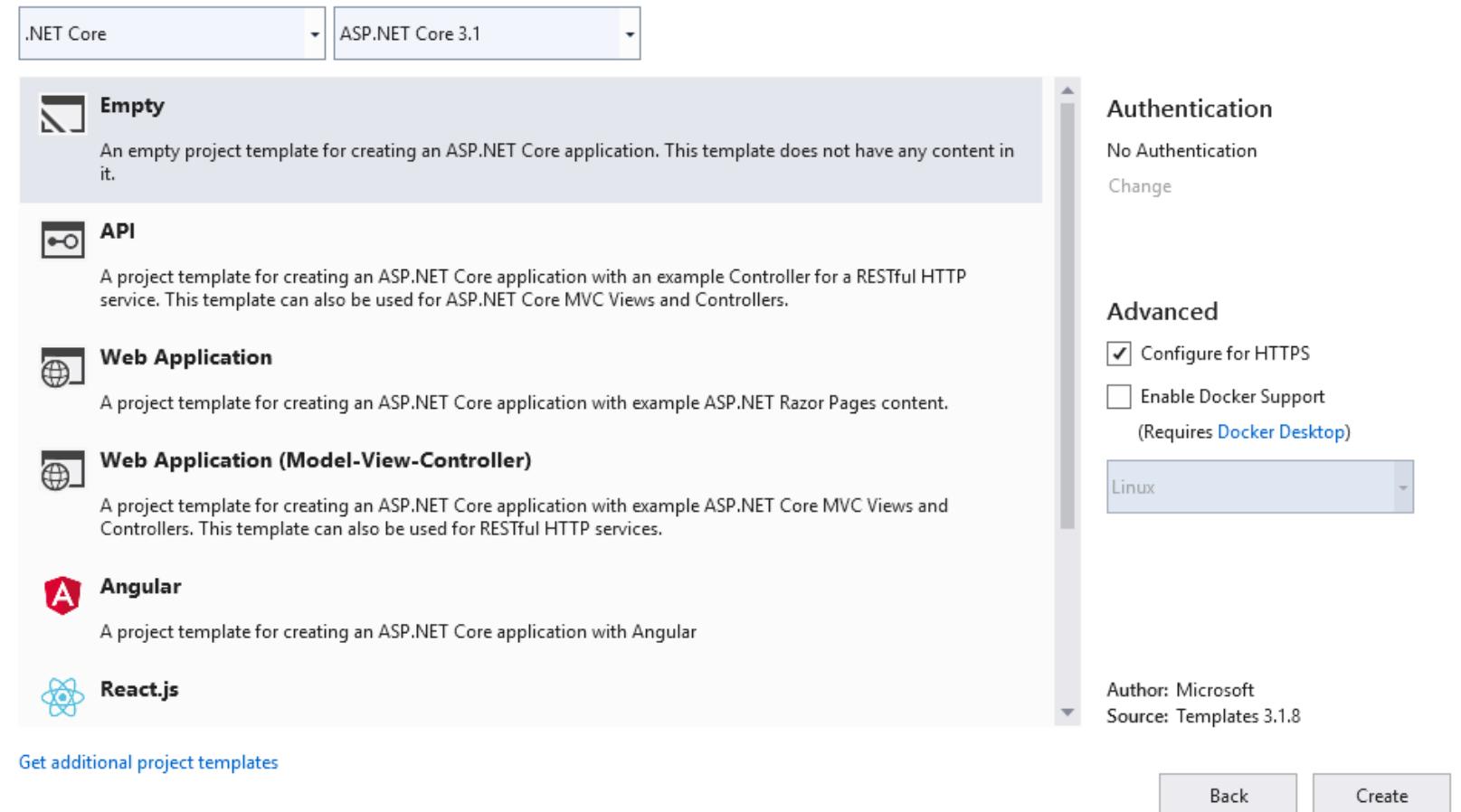
 ASP.NET Web Application (.NET Framework)

[Next](#)

SELECT TO CREATE



Create a new ASP.NET Core web application



The screenshot shows the 'Create a new ASP.NET Core web application' dialog box. At the top, there are two dropdown menus: '.NET Core' (selected) and 'ASP.NET Core 3.1'. Below these are several project template options:

- Empty**: An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
- API**: A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
- Web Application**: A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
- Web Application (Model-View-Controller)**: A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
- Angular**: A project template for creating an ASP.NET Core application with Angular.
- React.js**: A project template for creating an ASP.NET Core application with React.js.

On the right side of the dialog, there are sections for 'Authentication' (No Authentication, Change), 'Advanced' (Configure for HTTPS checked, Enable Docker Support unchecked, Requires Docker Desktop), and a 'Linux' dropdown menu. At the bottom, it says 'Author: Microsoft' and 'Source: Templates 3.1.8'. There are 'Back' and 'Create' buttons at the bottom right.

Get additional project templates

Back Create

Authentication

No Authentication

Change

Advanced

Configure for HTTPS

Enable Docker Support
(Requires Docker Desktop)

Linux

Author: Microsoft

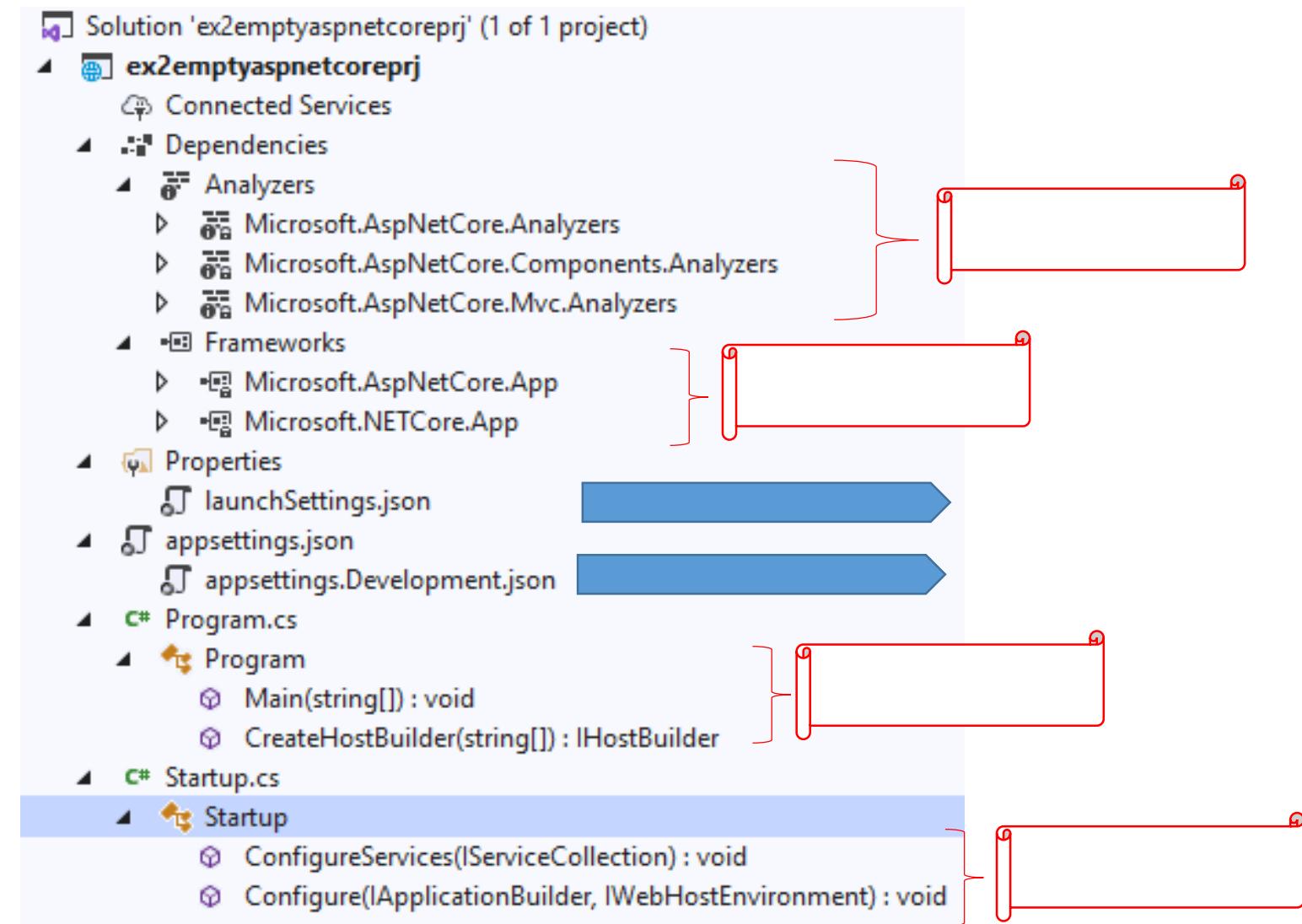
Source: Templates 3.1.8

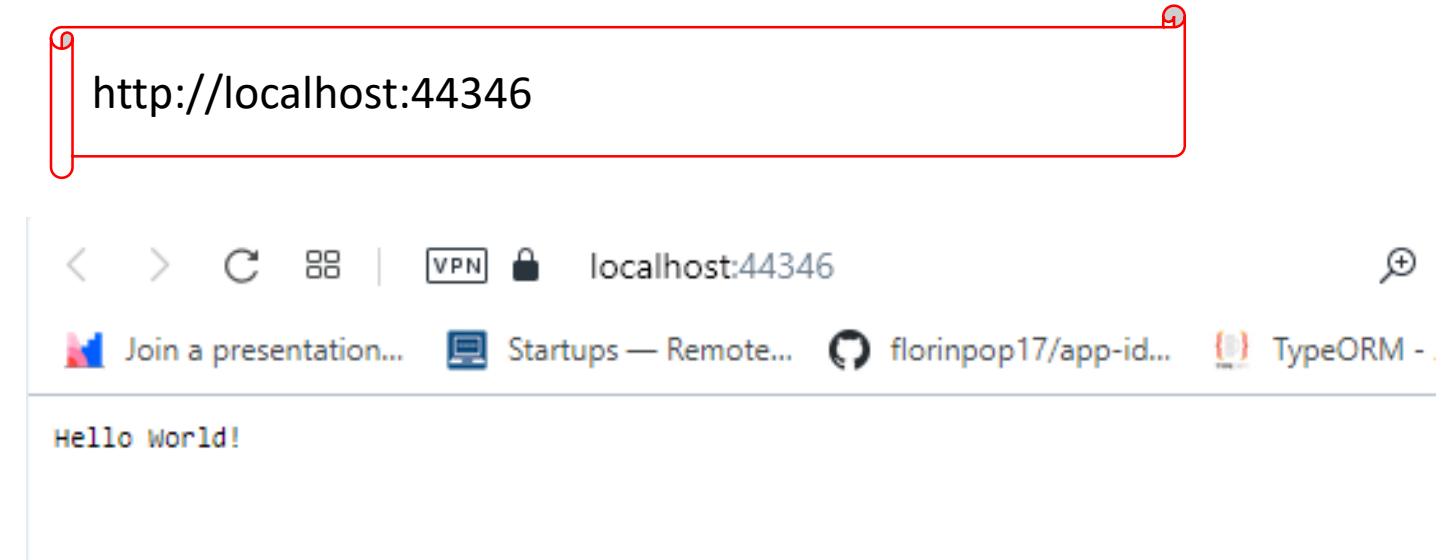
Example: 2

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar indicates the project name is "ex2emptyaspnetcorepj". The main code editor displays the "Startup.cs" file, which contains the following C# code:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Builder;
6  using Microsoft.AspNetCore.Hosting;
7  using Microsoft.AspNetCore.Http;
8  using Microsoft.Extensions.DependencyInjection;
9  using Microsoft.Extensions.Hosting;
10 
11 namespace ex2emptyaspnetcorepj
12 {
13     public class Startup
14     {
15         // This method gets called by the runtime. Use this method to add services to the container.
16         // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?linkid=864539
17         public void ConfigureServices(IServiceCollection services)
18         {
19         }
20 
21         // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
22     }
}
```

The Solution Explorer on the right shows the project structure with files like Program.cs and Startup.cs. The Output window at the bottom is empty.





SYED AWASE KHIRNI

Example 3: ASP.NET Core Razor Web Application

Learning Outcomes:

1. Understand the steps to create ASP.NET Core 3.x web application with razor pages
2. Understand the web application structure
3. Understand the key dependencies for ASP.NET Core 3.x web application with razor pages
4. Executing ASP.NET Core 3.x web application and rendering it on `http://localhost:44360`

Configure your new project

ASP.NET Core Web Application C# Linux macOS Windows Cloud Service Web

Project name

Location

...

Solution

Solution name i

Place solution and project in the same directory

Back

Create

Create a new project

Recent project templates

	All languages	All platforms	All project types
Console App (.NET Core)	C#	C# Linux macOS Windows Console	Console App (.NET Core) A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
			Visual Basic Linux macOS Windows Console
			Console App (.NET Core) A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
			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
			Blazor App Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).
			C# Linux macOS Windows Cloud Web
			ASP.NET Web Application (.NET Framework)

SELECT TO CREATE

[Next](#)

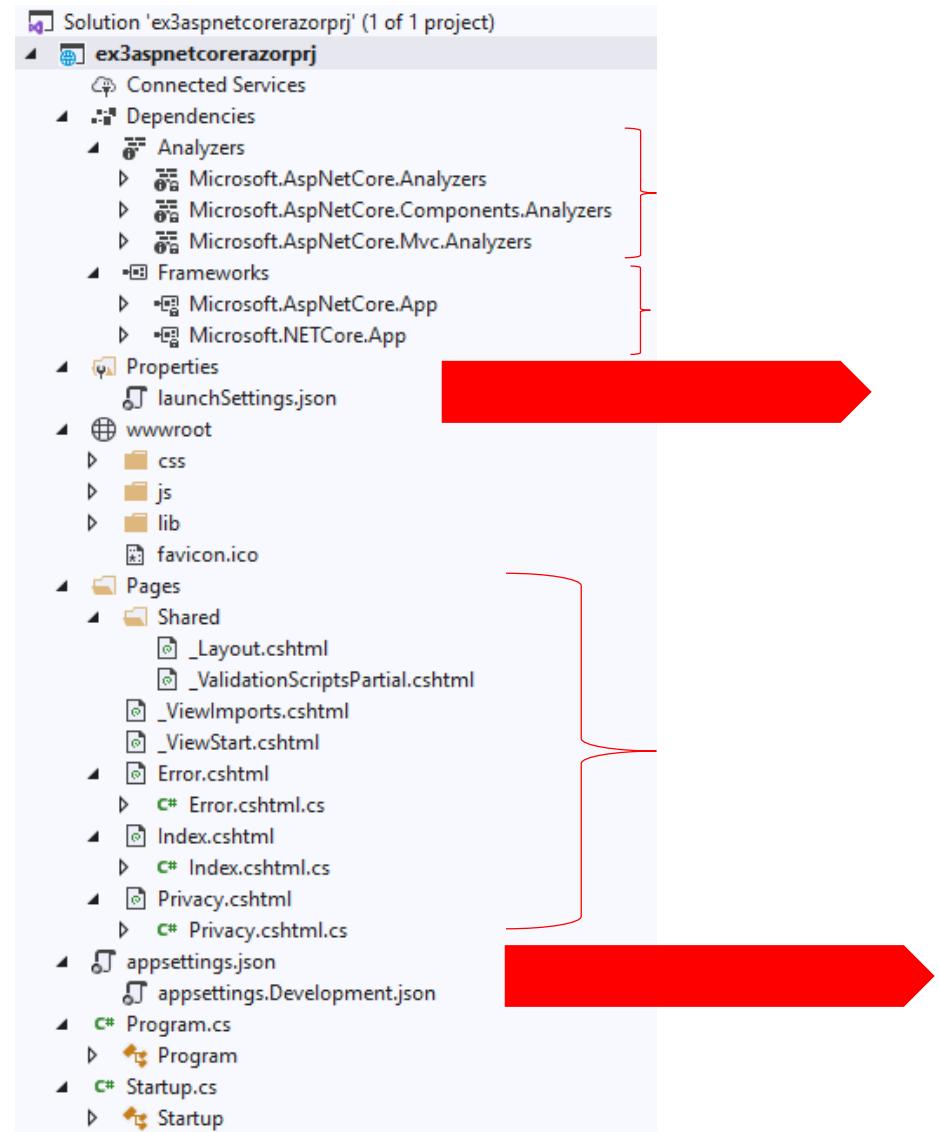
Create a new ASP.NET Core web application

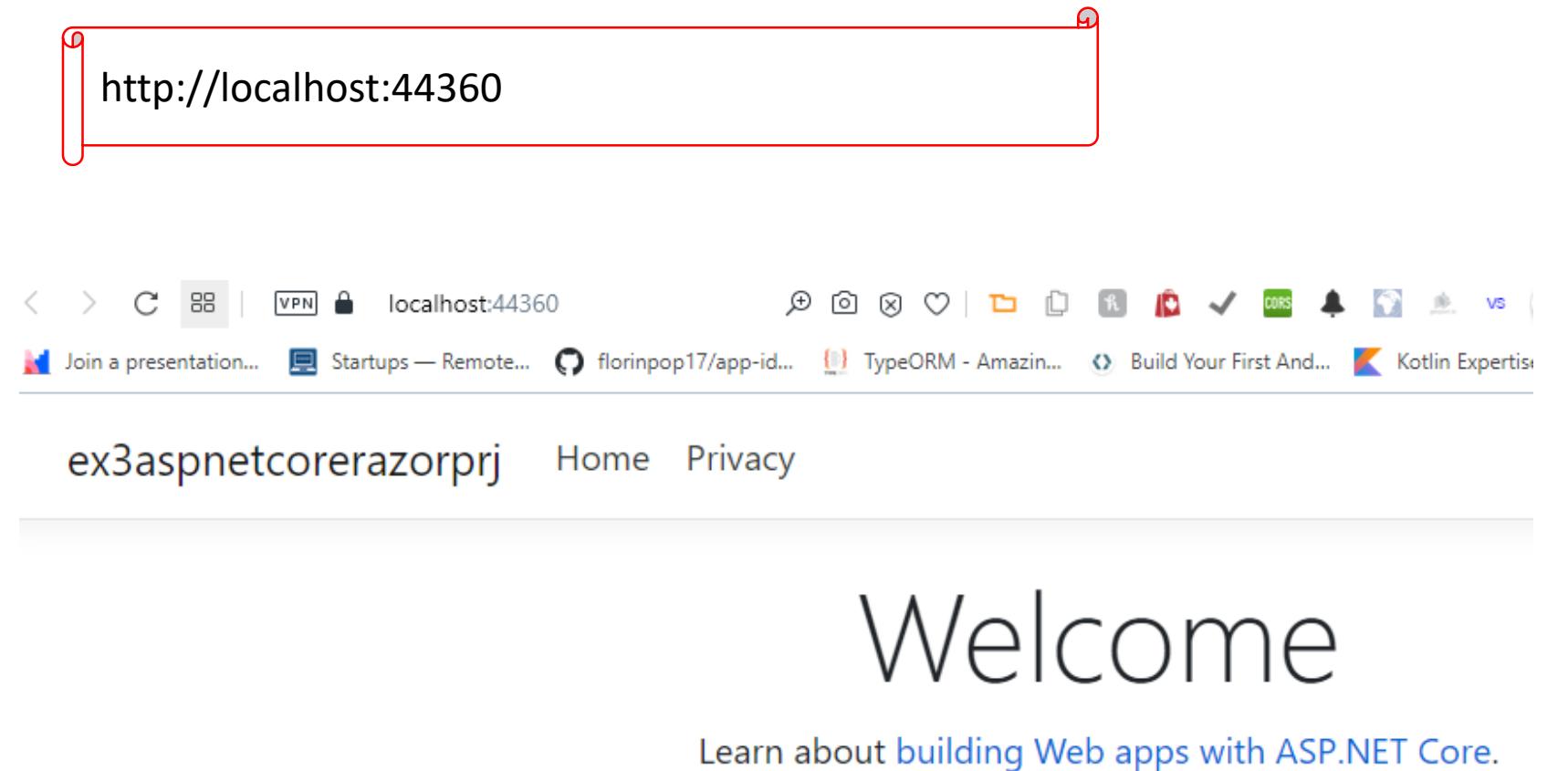
The screenshot shows the 'Create a new ASP.NET Core web application' dialog. At the top, there are two dropdown menus: '.NET Core' (set to 'ASP.NET Core') and 'ASP.NET Core 3.1'. Below these are several project template options:

- Empty**: An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
- API**: A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
- Web Application** (selected): A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
- Web Application (Model-View-Controller)**: A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
- Angular**: A project template for creating an ASP.NET Core application with Angular.
- React.js**: A project template for creating an ASP.NET Core application with React.js.

On the right side of the dialog, there are sections for 'Authentication' (set to 'No Authentication' with a 'Change' link) and 'Advanced' settings (checkboxes for 'Configure for HTTPS' (checked), 'Enable Docker Support' (unchecked), 'Linux' (selected), and 'Enable Razor runtime compilation' (unchecked)). At the bottom, it shows 'Author: Microsoft' and 'Source: Templates 3.1.8'. There are 'Back' and 'Create' buttons at the bottom right.

Example: 3





SYED AWASE KHIRNI

Example 4:ASP.NET Core Web Api Project

Learning Outcomes:

1. Understand the steps to create ASP.NET Core 3.x web api project
2. Understand the web application structure
3. Understand the key dependencies for ASP.NET Core 3.x web api project
4. Executing ASP.NET Core 3.x web api and rendering it on <http://localhost:44360>

Create a new project

Recent project templates

 ASP.NET Core Web Application

C#

 Console App (.NET Core)

C#

SELECT TO CREATE 

All languages

All platforms

All project types

  C# Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

C# Linux macOS Windows Console

  VB Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

Visual Basic Linux macOS Windows Console

 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

 Blazor App

Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).

C# Linux macOS Windows Cloud Web

 ASP.NET Web Application (.NET Framework)[Next](#)

Configure your new project

ASP.NET Core Web Application

C#

Linux

macOS

Windows

Cloud

Service

Web

Project name

ex4aspnetwebapiprj

Location

D:\awase-harddisk\CT\Csharp2017\aspnetcore3x\codeplay\ex4aspnetwebapi\



Solution

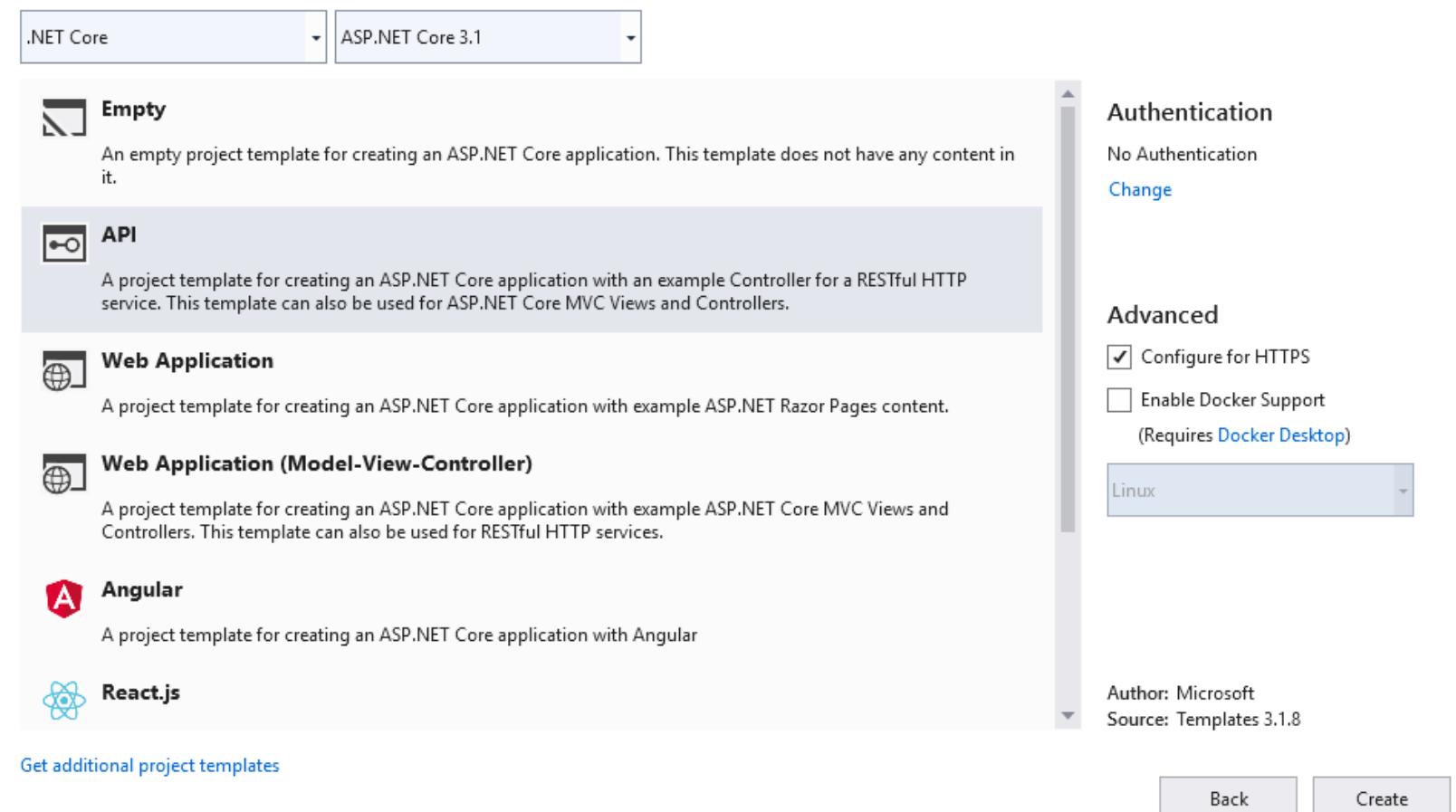
Create new solution

Solution name 

ex4aspnetwebapiprj

 Place solution and project in the same directory[Back](#)[Create](#)

Create a new ASP.NET Core web application



The screenshot shows the 'Create a new ASP.NET Core web application' dialog. At the top, there are two dropdown menus: '.NET Core' and 'ASP.NET Core 3.1'. Below these are several project template options:

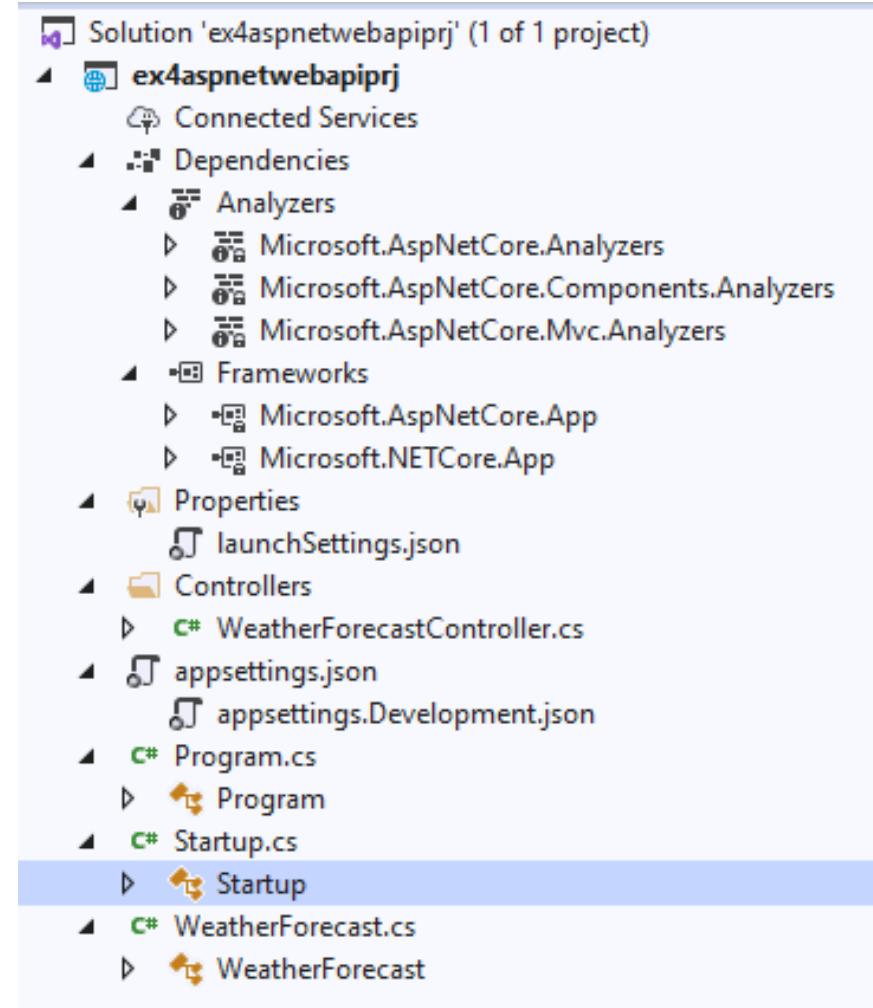
- Empty**: An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
- API**: A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
- Web Application**: A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
- Web Application (Model-View-Controller)**: A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
- Angular**: A project template for creating an ASP.NET Core application with Angular.
- React.js**: A project template for creating an ASP.NET Core application with React.js.

On the right side of the dialog, there are sections for 'Authentication' (set to 'No Authentication'), 'Change', 'Advanced' (with checkboxes for 'Configure for HTTPS' (checked), 'Enable Docker Support' (unchecked), and 'Requires Docker Desktop'), and a dropdown for 'Linux'. At the bottom, it says 'Author: Microsoft' and 'Source: Templates 3.1.8'. There are 'Back' and 'Create' buttons at the bottom right.

Example:

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the `Startup.cs` file for a project named `ex4aspnetwebapiprj`. The code implements the `IConfiguration` interface and uses the `ConfigureServices` and `Configure` methods to set up services and configure the HTTP request pipeline. The Solution Explorer on the right shows the project structure, including files like `appsettings.json`, `Program.cs`, and `WeatherForecast.cs`.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.AspNetCore.Builder;
6  using Microsoft.AspNetCore.Hosting;
7  using Microsoft.AspNetCore.HttpsPolicy;
8  using Microsoft.AspNetCore.Mvc;
9  using Microsoft.Extensions.Configuration;
10 using Microsoft.Extensions.DependencyInjection;
11 using Microsoft.Extensions.Hosting;
12 using Microsoft.Extensions.Logging;
13
14 namespace ex4aspnetwebapiprj
15 {
16     public class Startup
17     {
18         public Startup(IConfiguration configuration)
19         {
20             Configuration = configuration;
21         }
22
23         public IConfiguration Configuration { get; }
24
25         // This method gets called by the runtime. Use this method to add services to the container.
26         public void ConfigureServices(IServiceCollection services)
27         {
28             services.AddControllers();
29         }
30
31         // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
32     }
33 }
```



```
[{"date": "2020-08-26T11:33:46.7193998-04:00", "temperatureC": 18, "temperatureF": 64, "summary": "Freezing"}, {"date": "2020-08-27T11:33:46.7430728-04:00", "temperatureC": 15, "temperatureF": 58, "summary": "Chilly"}, {"date": "2020-08-28T11:33:46.7430996-04:00", "temperatureC": 43, "temperatureF": 109, "summary": "Scorching"}, {"date": "2020-08-29T11:33:46.743103-04:00", "temperatureC": 34, "temperatureF": 93, "summary": "Bracing"}, {"date": "2020-08-30T11:33:46.7431048-04:00", "temperatureC": -12, "temperatureF": 11, "summary": "Warm"}]
```

SYED AWASE KHIRNI

Example 5:ASP.NET Core MVC Web Application

Learning Outcomes:

1. Understand the steps to create ASP.NET Core 3.x mvc web application
2. Understand the web application structure
3. Understand the key dependencies for ASP.NET Core 3.x mvc web application
4. Executing ASP.NET Core 3.x mvc web application and rendering it on <http://localhost:44360>

Create a new project

Recent project templates

- ASP.NET Core Web Application C#
- Console App (.NET Core) C#

Search for templates (Alt+S)

All languages ▾ All platforms ▾ All project types ▾

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Console

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
Visual Basic Linux macOS Windows Console

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

Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web

ASP.NET Web Application (.NET Framework)

Next

SELECT TO CREATE

Configure your new project

ASP.NET Core Web Application C# Linux macOS Windows Cloud Service Web

Project name

Location

Solution

Solution name i

Place solution and project in the same directory

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

Create a new ASP.NET Core web application

.NET Core ASP.NET Core 3.1

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

 **API**
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core Views and Controllers.

 **Web Application**
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

 **Web Application (Model-View-Controller)**
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

 **Angular**
A project template for creating an ASP.NET Core application with Angular

 **React.js**

[Get additional project templates](#)

Authentication
No Authentication
[Change](#)

Advanced

Configure for HTTPS

Enable Docker Support
(Requires [Docker Desktop](#))

Linux

Enable Razor runtime compilation

Author: Microsoft
Source: Templates 3.1.8

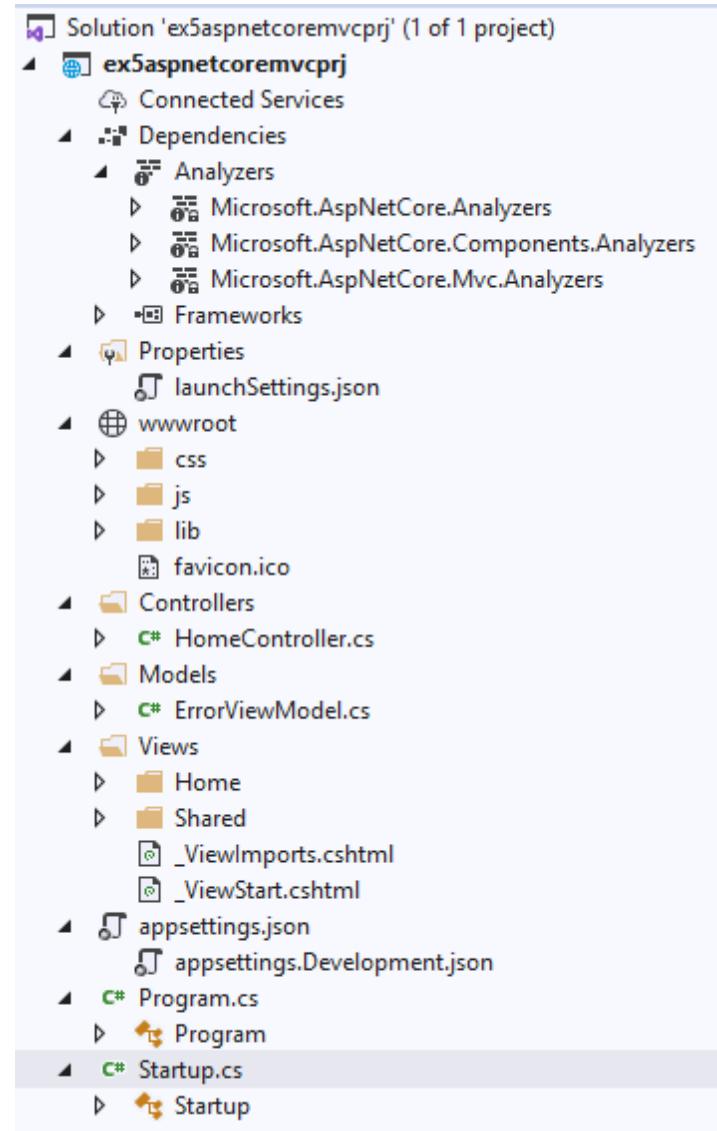
Back Create

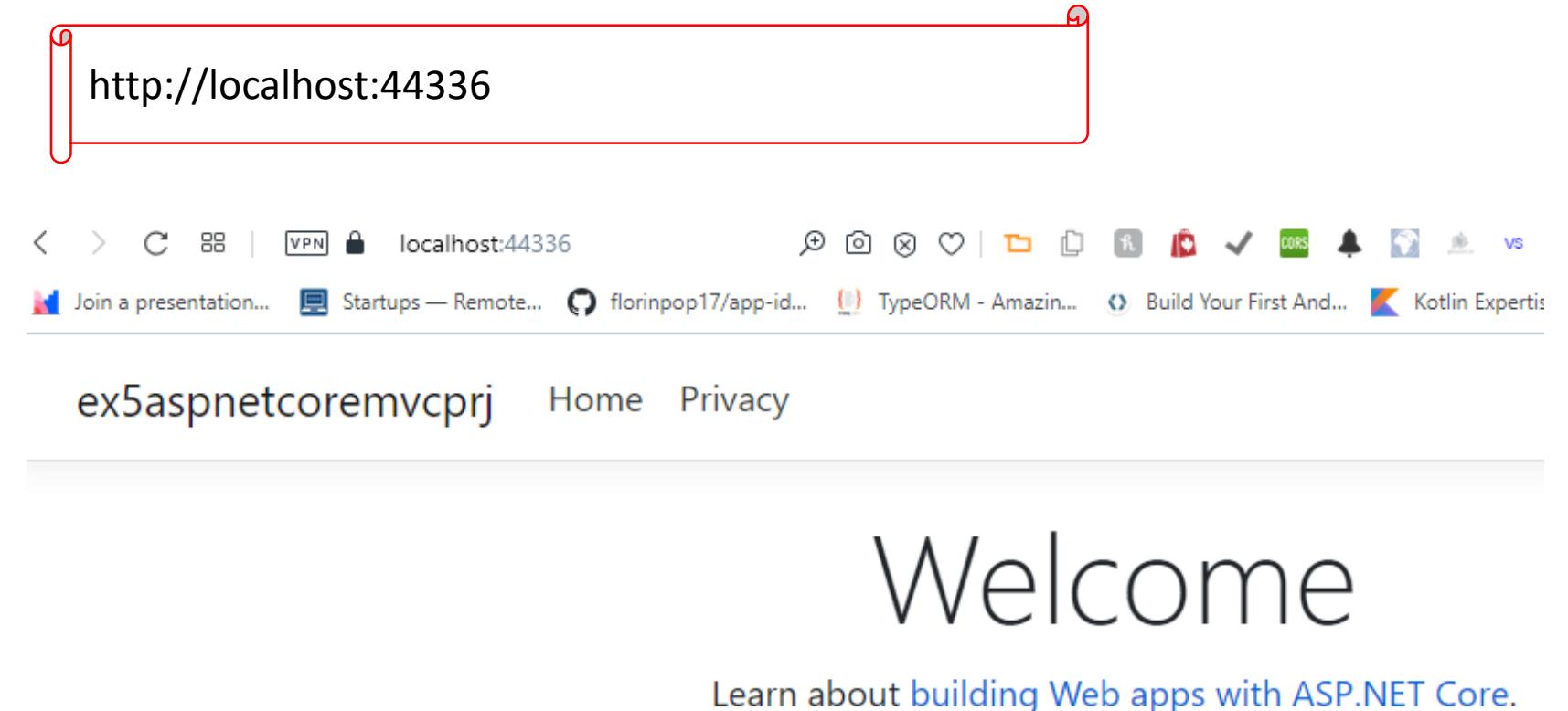
Example: 5

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar displays "File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) ex5aspnetcoremvcprj A - □ × Live Share". The main window shows the code for "Startup.cs" in the "ex5aspnetcoremvcprj" project. The code defines a "Startup" class that implements the "IConfiguration" and "IApplicationBuilder" interfaces. It uses dependency injection to set up controllers and configure the HTTP request pipeline. The Solution Explorer on the right lists the project files, including "Program.cs" and "Startup.cs". The status bar at the bottom indicates "Ready".

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5 using Microsoft.AspNetCore.Builder;
6 using Microsoft.AspNetCore.Hosting;
7 using Microsoft.AspNetCore.HttpsPolicy;
8 using Microsoft.Extensions.Configuration;
9 using Microsoft.Extensions.DependencyInjection;
10 using Microsoft.Extensions.Hosting;
11
12 namespace ex5aspnetcoremvcprj
13 {
14     public class Startup
15     {
16         public Startup(IConfiguration configuration)
17         {
18             Configuration = configuration;
19         }
20
21         public IConfiguration Configuration { get; }
22
23         public void ConfigureServices(IServiceCollection services)
24         {
25             services.AddControllersWithViews();
26         }
27
28         public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
29         {
30             // This method gets called by the runtime. Use this method to add services to the container.
31             // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
32         }
33     }
34 }
```

Example: 5





SYED AWASE KHIRNI

Example 6: ASP.NET Core Angular Stack Application

Learning Outcomes:

1. Understand the steps to create ASP.NET Core 3.x angular stack application
2. Understand the web application structure
3. Understand the key dependencies for ASP.NET Core 3.x angular stack application
4. Executing ASP.NET Core 3.x angular stack application and rendering it on <http://localhost:44360>

Create a new project

Recent project templates

- ASP.NET Core Web Application C#
- Console App (.NET Core) C#

Search for templates (Alt+S)

All languages ▾ All platforms ▾ All project types ▾

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Console

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
Visual Basic Linux macOS Windows Console

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

Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web

ASP.NET Web Application (.NET Framework)

Next

SELECT TO CREATE

Configure your new project

ASP.NET Core Web Application

C# Linux macOS Windows Cloud Service Web

Project name

ex6aspnetcoreangularstack

Location

D:\awase-harddisk\CT\Csharp2017\aspnetcore3x\codeplay\ex6aspnetcoreangularstack\



Solution

Create new solution

Solution name 

ex6aspnetcoreangularstack

 Place solution and project in the same directory[Back](#)[Create](#)

Create a new ASP.NET Core web application

The screenshot shows the 'Create a new ASP.NET Core web application' dialog. At the top, there are two dropdown menus: '.NET Core' and 'ASP.NET Core 3.1'. Below these are six project template options:

- Empty**: An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
- API**: A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
- Web Application**: A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
- Web Application (Model-View-Controller)**: A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
- Angular**: A project template for creating an ASP.NET Core application with Angular.
- React.js**: A project template for creating an ASP.NET Core application with React.js.

On the right side of the dialog, there are three sections: 'Authentication' (No Authentication, Change), 'Advanced' (Configure for HTTPS checked, Enable Docker Support unchecked), and 'Linux' (selected). At the bottom, it says 'Author: Microsoft' and 'Source: Templates 3.1.8'. There are 'Back' and 'Create' buttons at the bottom right.

Example: 6

Please read terms of use for authorized access

Original Series

The screenshot shows the Microsoft Visual Studio IDE interface. The left side features the Server Explorer and Toolbox. The main area displays the `Startup.cs` file for the project `ex6aspnetcoreangularstack`. The code in `Startup.cs` is as follows:

```
1  using Microsoft.AspNetCore.Builder;
2  using Microsoft.AspNetCore.Hosting;
3  using Microsoft.AspNetCore.HttpsPolicy;
4  using Microsoft.AspNetCore.SpaServices.AngularCli;
5  using Microsoft.Extensions.Configuration;
6  using Microsoft.Extensions.DependencyInjection;
7  using Microsoft.Extensions.Hosting;
8
9  namespace ex6aspnetcoreangularstack
10 {
11     public class Startup
12     {
13         public Startup(IConfiguration configuration)
14         {
15             Configuration = configuration;
16         }
17
18         public IConfiguration Configuration { get; }
19
20         // This method gets called by the runtime. Use this method to add services to
21         // the container.
22         public void ConfigureServices(IServiceCollection services)
23         {
24             services.AddControllersWithViews();
25             // In production, the Angular files will be served from this directory
26             services.AddSpaStaticFiles(configuration =>
27             {
28                 configuration.RootPath = "ClientApp/dist";
29             });
29
30             // This method gets called by the runtime. Use this method to configure the HT
31         }
31     }
31 }
```

The Solution Explorer on the right shows the project structure for `ex6aspnetcoreangularstack`, which includes `Connected Services`, `Dependencies`, `Properties`, `wwwroot`, `ClientApp`, `Pages`, and files like `launchSettings.json`, `package.json`, and `WeatherForecastController.cs`.



The screenshot shows the Visual Studio Solution Explorer with the following project structure:

- Solution 'ex6aspnetcoreangularstack' (1 of 1 project)
- ex6aspnetcoreangularstack
 - Connected Services
 - Dependencies
 - Analyzers
 - Frameworks
 - Packages
 - Properties
 - launchSettings.json
 - wwwroot
 - favicon.ico
 - ClientApp
 - e2e
 - src
 - .editorconfig
 - .gitignore
 - angular.json
 - browserslist
 - package.json
 - README.md
 - tsconfig.json
 - tslint.json
 - Controllers
 - WeatherForecastController.cs
 - Pages
 - _ViewImports.cshtml
 - Error.cshtml
 - .gitignore
 - appsettings.json
 - appsettings.Development.json
 - Program.cs
 - Program
 - Startup.cs
 - Startup
 - WeatherForecast.cs
 - WeatherForecast

http://localhost:44385

The screenshot shows a browser window with the URL `localhost:44385` in the address bar. The page content is as follows:

Hello, world!

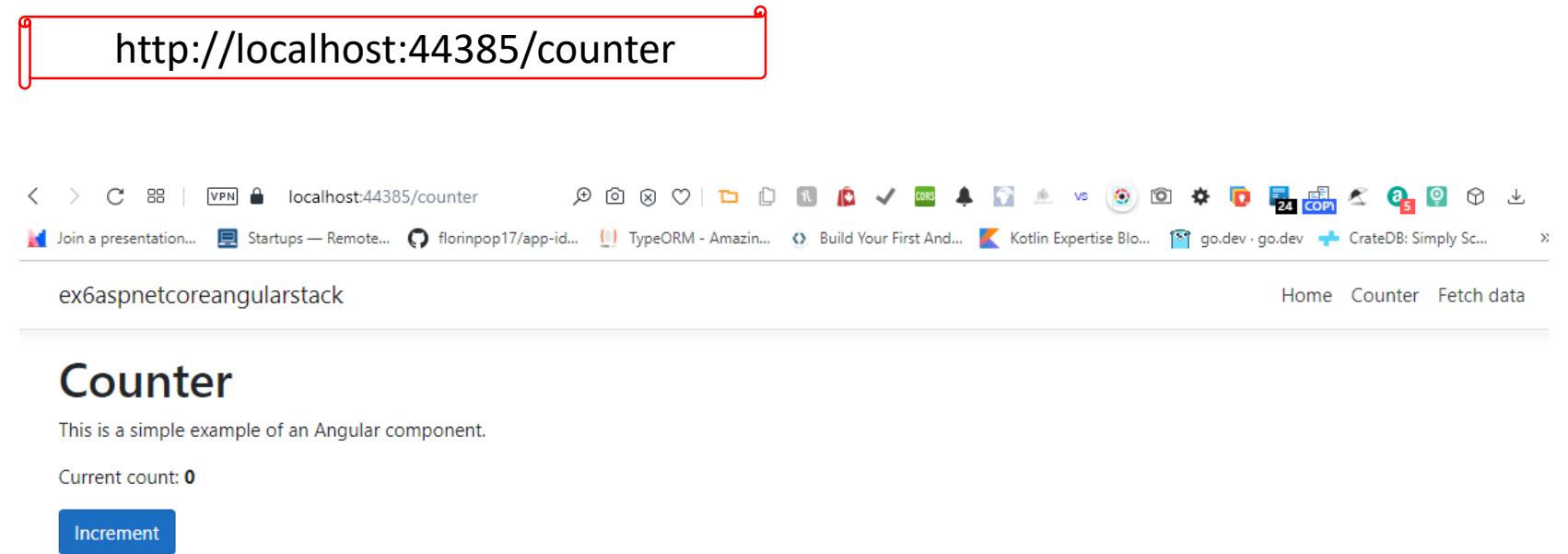
Welcome to your new single-page application, built with:

- ASP.NET Core and C# for cross-platform server-side code
- Angular and TypeScript for client-side code
- Bootstrap for layout and styling

To help you get started, we've also set up:

- Client-side navigation.** For example, click `Counter` then `Back` to return here.
- Angular CLI integration.** In development mode, there's no need to run `ng serve`. It runs in the background automatically, so your client-side resources are dynamically built on demand and the page refreshes when you modify any file.
- Efficient production builds.** In production mode, development-time features are disabled, and your `dotnet publish` configuration automatically invokes `ng build` to produce minified, ahead-of-time compiled JavaScript files.

The `ClientApp` subdirectory is a standard Angular CLI application. If you open a command prompt in that directory, you can run any `ng` command (e.g., `ng test`), or use `npm` to install extra packages into it.



The screenshot shows a browser window with the URL <http://localhost:44385/fetch-data> highlighted by a red box. The page title is "Weather forecast". The content area displays a table with five rows of weather data:

Date	Temp. (C)	Temp. (F)	Summary
2020-08-26T12:47:29.285781-04:00	39	102	Scorching
2020-08-27T12:47:29.2870991-04:00	54	129	Freezing
2020-08-28T12:47:29.2871119-04:00	18	64	Chilly
2020-08-29T12:47:29.2871136-04:00	19	66	Mild
2020-08-30T12:47:29.2871144-04:00	30	85	Sweltering

SYED AWASE KHIRNI

Example 7: ASP.NET Core React Stack Application

Learning Outcomes:

1. Understand the steps to create ASP.NET Core 3.x react stack application
2. Understand the web application structure
3. Understand the key dependencies for ASP.NET Core 3.x react stack application
4. Executing ASP.NET Core 3.x react stack application and rendering it on `http://localhost:44360`

Create a new project

Recent project templates

 ASP.NET Core Web Application

C#

 Console App (.NET Core)

C#

SELECT TO CREATE

Search for templates (Alt+S)



All languages

All platforms

All project types

 C# Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

C# Linux macOS Windows Console

 VB Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

Visual Basic Linux macOS Windows Console

 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

 Blazor App

Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).

C# Linux macOS Windows Cloud Web

 ASP.NET Web Application (.NET Framework)[Next](#)

Configure your new project

ASP.NET Core Web Application C# Linux macOS Windows Cloud Service Web

Project name

Location

Solution

Solution name i

Place solution and project in the same directory

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

Create a new ASP.NET Core web application

The screenshot shows the 'Create a new ASP.NET Core web application' dialog. At the top, there are two dropdown menus: 'NET Core' and 'ASP.NET Core 3.1'. Below these are several project template options:

- API**: A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
- Web Application**: A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
- Web Application (Model-View-Controller)**: A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
- Angular**: A project template for creating an ASP.NET Core application with Angular.
- React.js**: A project template for creating an ASP.NET Core application with React.js. This template is highlighted with a blue background.
- React.js and Redux**: A project template for creating an ASP.NET Core application with React.js and Redux.

On the right side of the dialog, there are sections for 'Authentication' (No Authentication, Change), 'Advanced' (Configure for HTTPS checked, Enable Docker Support unchecked), and build information (Author: Microsoft, Source: Templates 3.1.8). At the bottom are 'Back' and 'Create' buttons.

Example: 7

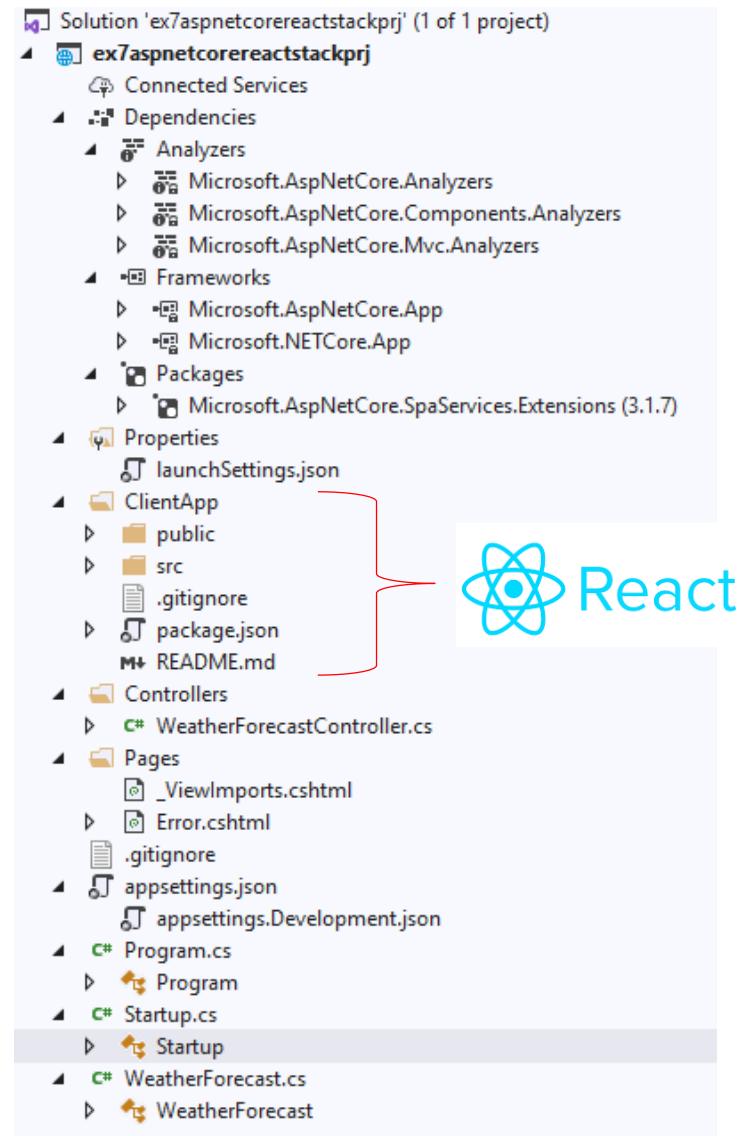
The screenshot shows the Microsoft Visual Studio IDE interface. The title bar indicates the project is named 'ex7aspnetcorereactstackprj'. The left pane displays the code for 'Startup.cs':

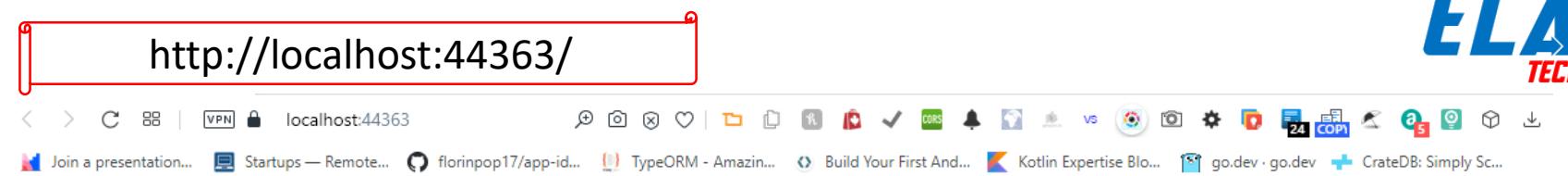
```
1  using Microsoft.AspNetCore.Builder;
2  using Microsoft.AspNetCore.Hosting;
3  using Microsoft.AspNetCore.HttpsPolicy;
4  using Microsoft.AspNetCore.SpaServices.ReactDevelopmentServer;
5  using Microsoft.Extensions.Configuration;
6  using Microsoft.Extensions.DependencyInjection;
7  using Microsoft.Extensions.Hosting;
8
9  namespace ex7aspnetcorereactstackprj
10 {
11     public class Startup
12     {
13         public Startup(IConfiguration configuration)
14         {
15             Configuration = configuration;
16         }
17
18         public IConfiguration Configuration { get; }
19
20         // This method gets called by the runtime. Use this method to add services to
21         // the container.
22         public void ConfigureServices(IServiceCollection services)
23         {
24
25             services.AddControllersWithViews();
26
27             // In production, the React files will be served from this directory
28             services.AddSpaStaticFiles(configuration =>
29             {
30                 configuration.RootPath = "ClientApp/build";
31             });
32         }
33     }
34 }
```

The right pane shows the 'Solution Explorer' with the project structure:

- Solution 'ex7aspnetcorereactstackprj' (1 of 1 project)
 - Connected Services
 - Dependencies
 - Analyzers
 - Microsoft.AspNetCore.Analyzers
 - Microsoft.AspNetCore.Components.Analyzers
 - Microsoft.AspNetCore.Mvc.Analyzers
 - Frameworks
 - Microsoft.AspNetCore.App
 - Microsoft.NETCore.App
 - Packages
 - Microsoft.AspNetCore.SpaServices.Extensions (3.1.7)
 - Properties
 - launchSettings.json
 - ClientApp
 - public
 - src
 - .gitignore
 - package.json
 - README.md
 - Controllers
 - WeatherForecastController.cs
 - Pages
 - _ViewImports.cshtml
 - Error.cshtml
 - .gitignore
 - appsettings.json
 - appsettings.Development.json
 - Program.cs
 - Program
 - Startup.cs
 - WeatherForecast.cs
 - WeatherForecast

Example: 7





ex7aspnetcorereactstackprj

[Home](#) [Counter](#) [Fetch data](#)

Hello, world!

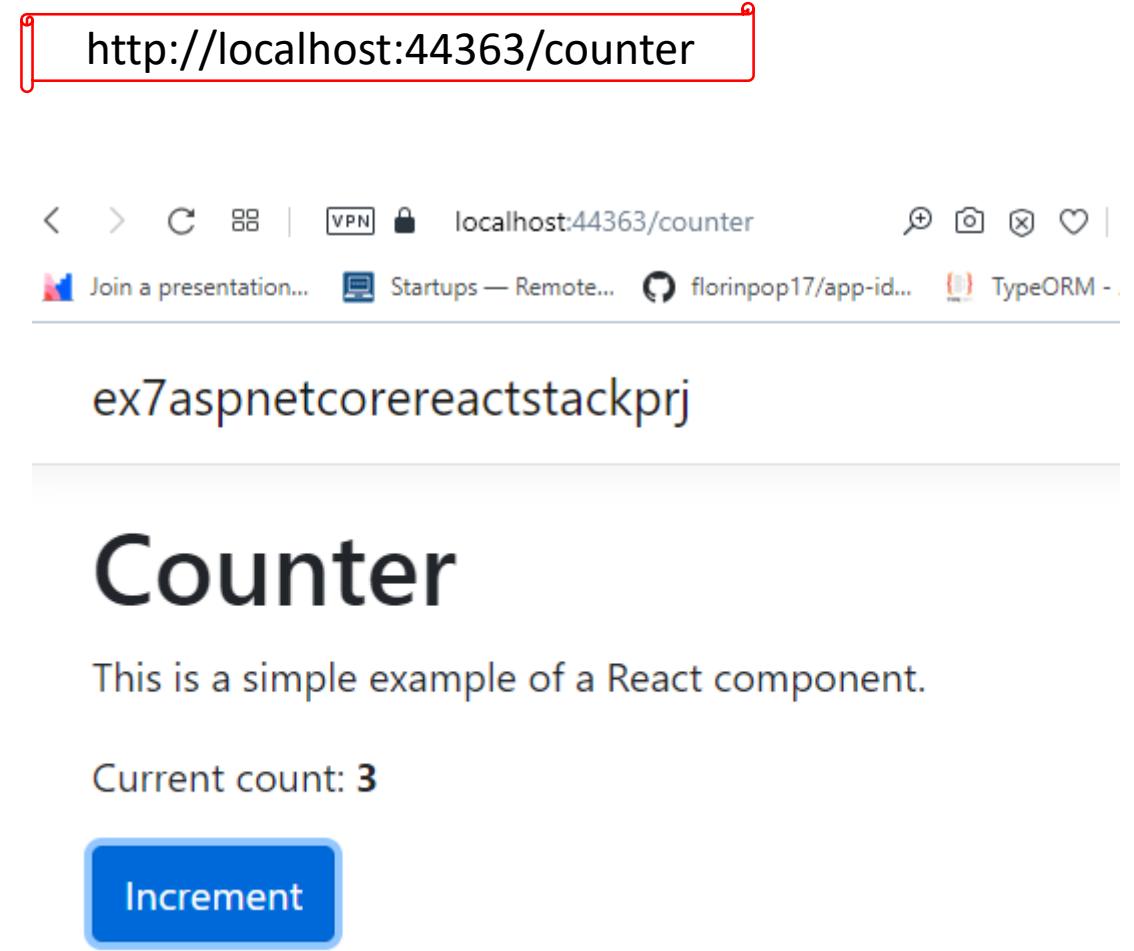
Welcome to your new single-page application, built with:

- [ASP.NET Core](#) and [C#](#) for cross-platform server-side code
- [React](#) for client-side code
- [Bootstrap](#) for layout and styling

To help you get started, we have also set up:

- **Client-side navigation.** For example, click [Counter](#) then [Back](#) to return here.
- **Development server integration.** In development mode, the development server from [create-react-app](#) runs in the background automatically, so your client-side resources are dynamically built on demand and the page refreshes when you modify any file.
- **Efficient production builds.** In production mode, development-time features are disabled, and your [dotnet publish](#) configuration produces minified, efficiently bundled JavaScript files.

The `ClientApp` subdirectory is a standard React application based on the [create-react-app](#) template. If you open a command prompt in that directory, you can run `npm` commands such as `npm test` or `npm install`.

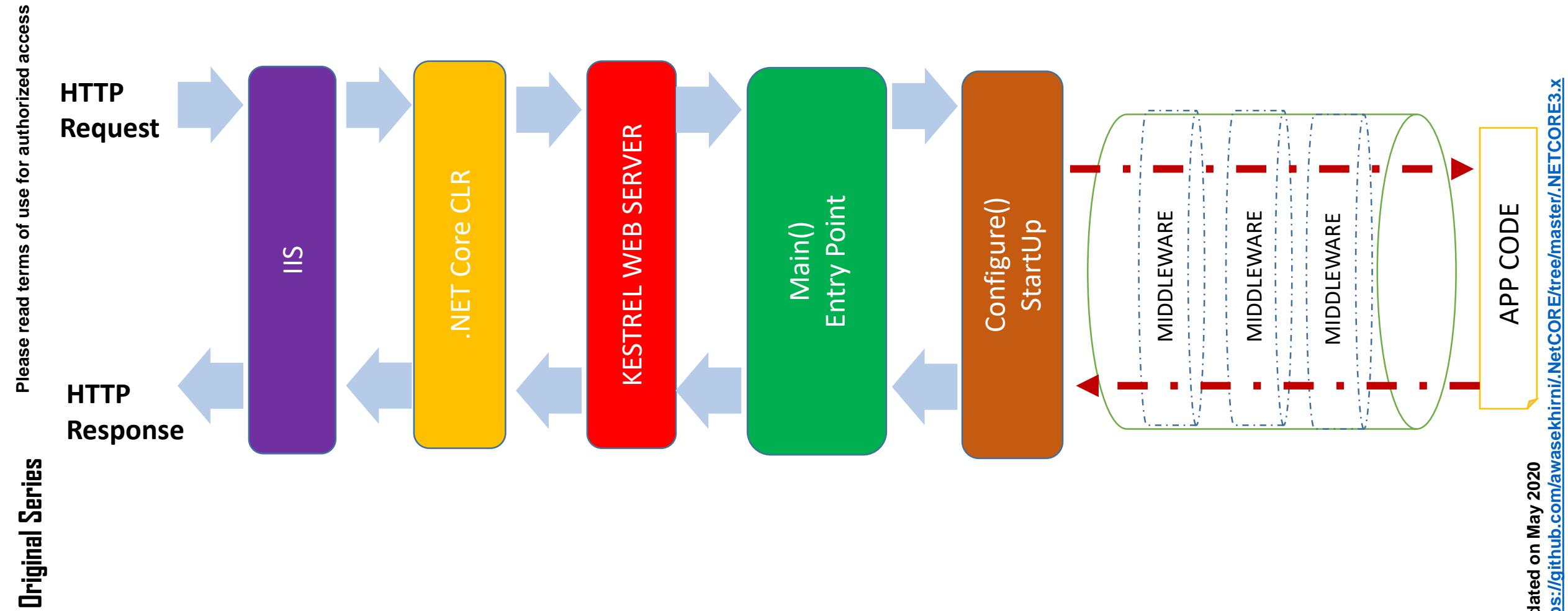


SYED AWASE KHIRNI

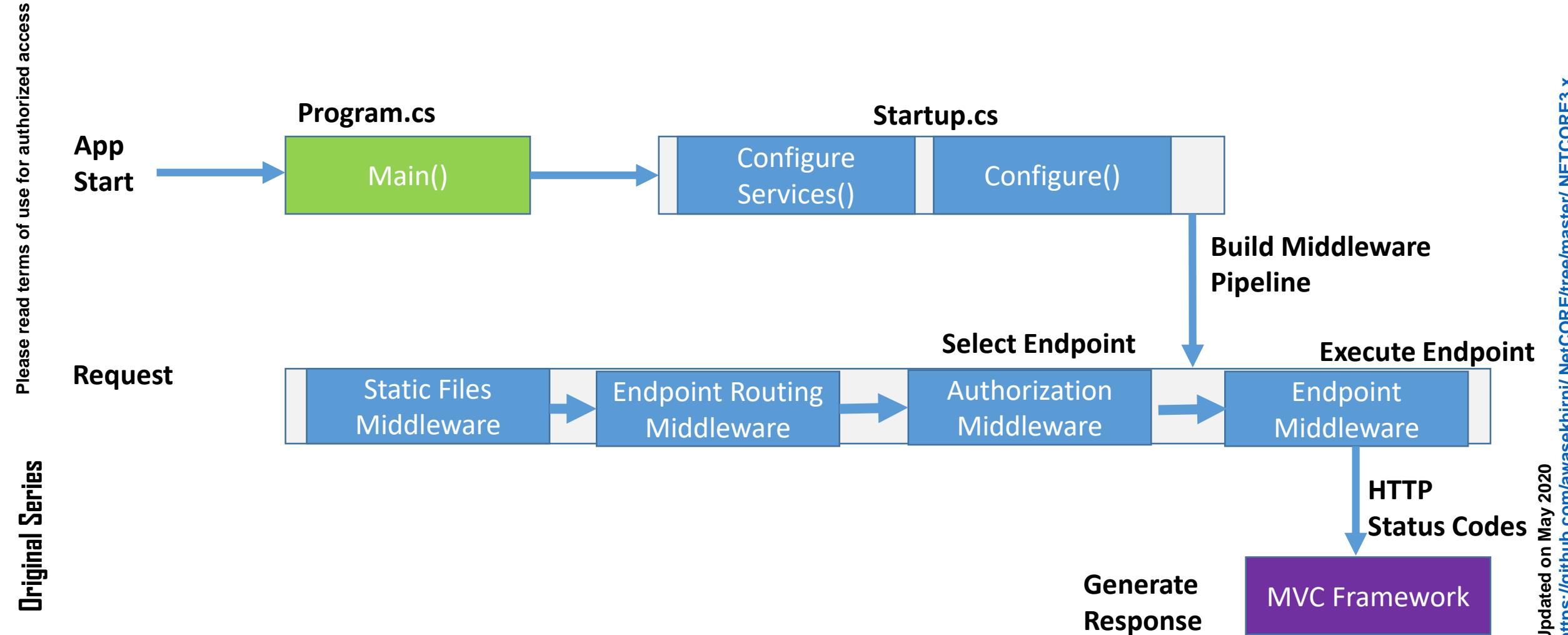
ASP.NET Core 3.0 MVC Request Life Cycle

Learning Outcomes:

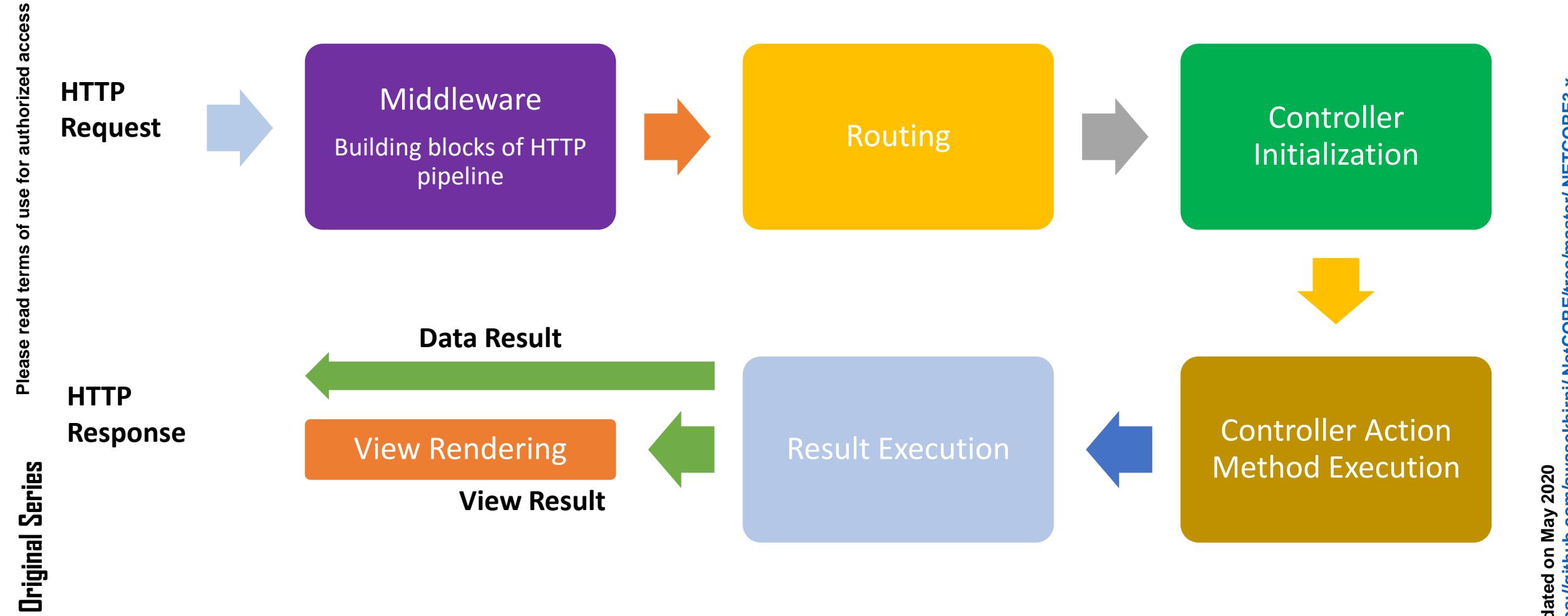
1. Refer to ASP.NET Core 2.x Slides about the evolution of web architectures and MVC foundations
<https://github.com/awasekhirni/.NetCORE>
2. Learning Outcome 2



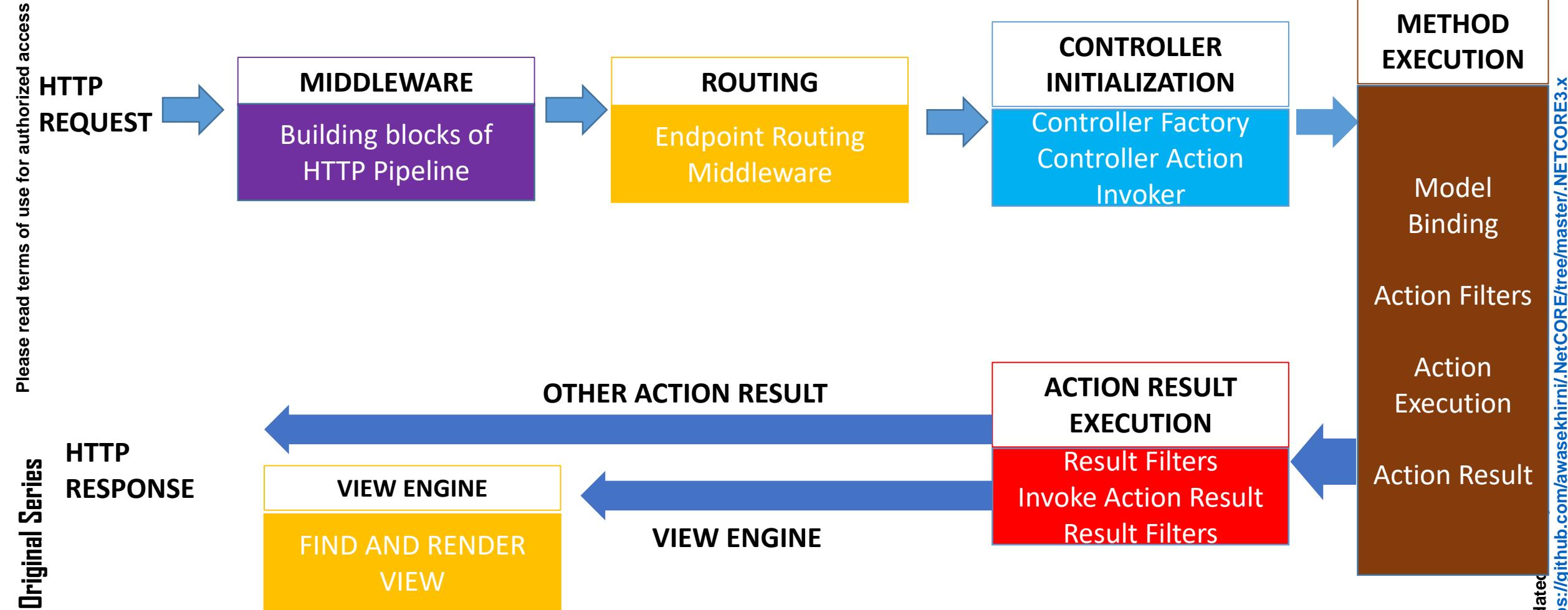
MVC Middleware Pipeline



MVC Request Life Cycle



MVC Request Life Cycle



The Middleware

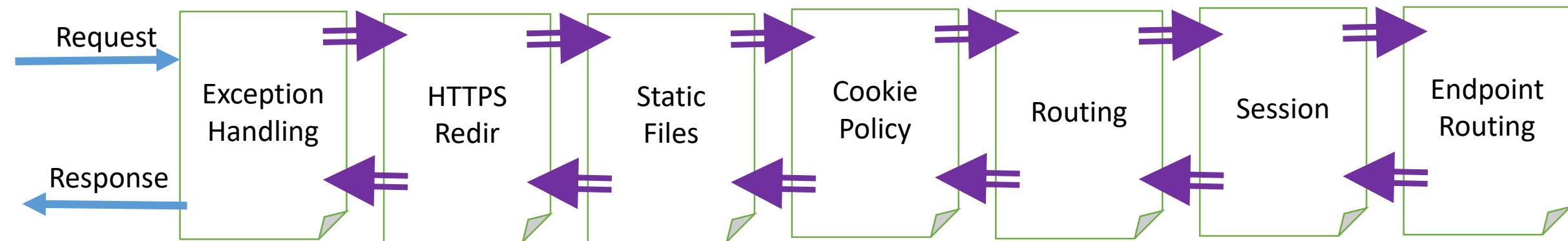


- It is software service provided to software applications which serves as a glue between different components with i/o communication channels that serves a specific purpose.
- The middleware provides the essential building blocks how an application handles HTTP requests.
- A series of components that form the application request pipeline.
- The middleware provides all kinds of foundational application-level infrastructural components for .NET Core application development.
- Examples of middleware are: routing, session, CORS, authentication, caching

Middleware Pipeline

A typical order of middleware layers in an ASP.NET Core web application

Please read terms of use for authorized access

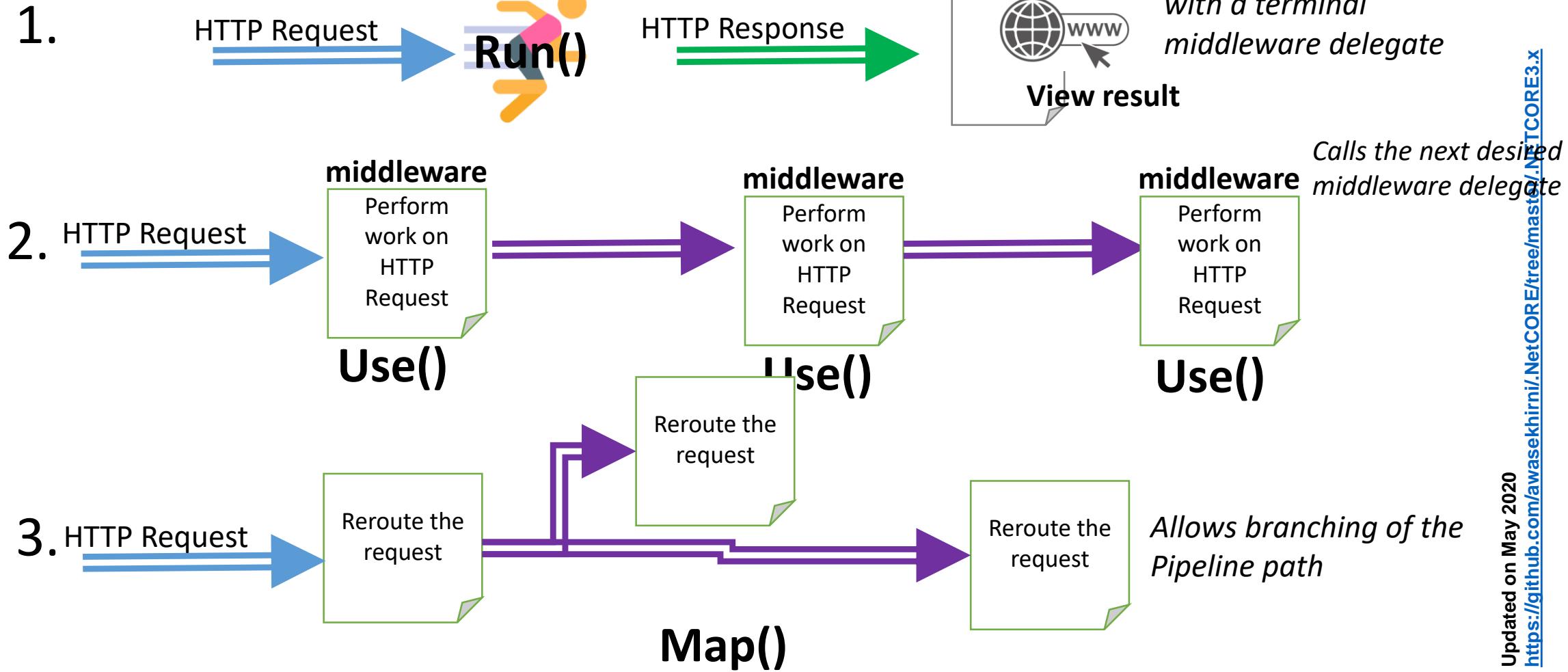


Original Series

Registering Middleware Components using Configuration Options

Please read terms of use for authorized access

Original Series



Middleware Configuration Options

Please read terms of use for authorized access

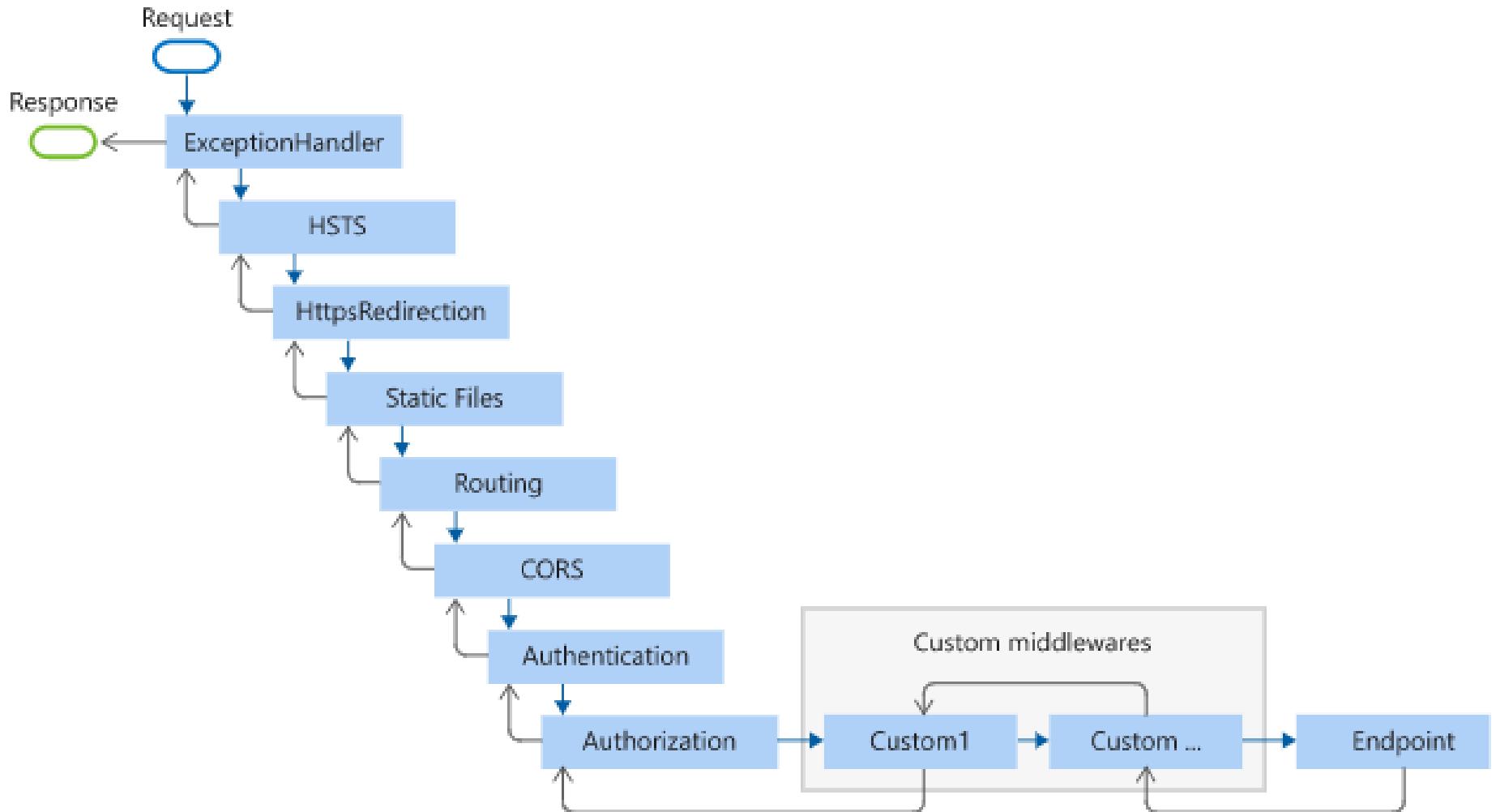
Original Series

Option	Description	Usage
Run	It terminates the middleware pipeline to render view result.	<pre>app.Run(async context =>{ await context.Response.WriteAsync("Render to Webpage") })</pre>
Use	It forwards the request onto the next delegate	<pre>app.Use(async (context,next) =>{ await next.Invoke(); })</pre>
Map	It redirect incoming requests which end in “search” to execute the GlobalSearch method.	<pre>app.Map("/search", GlobalSearch); Private static void GlobalSearch (IApplicationBuilder app){ app.Run(async context=>{ Await context.Response.WriteAsync("search results here") }); }</pre>

Updated on May 2020

<https://github.com/awasekhirni/NetCORE/tree/master/.NETCORE3.x>

Middleware Order



Built-in Middleware

Please read terms of use for authorized access

Original Series

Middleware	Description
Authentication	Adds authentication support
CORS	Configures Cross-Origin Resource Sharing
Routing	Adds routing capabilities for MVC or web forms
Session	Adds support for user session
StaticFiles	Adds support for serving static files and directory browsing
Diagnostics	Adds support for reporting and handling exceptions and errors

Middleware Methods

Please read terms of use for authorized access		
Original Series		
Exception Handling	UseDeveloperExceptionPage()	Used in development to catch run-time exceptions
	UseDatabaseErrorResponse()	
	UseExceptionHandler()	Used in production for run-time exceptions
HSTS & HTTPS Redirection	UseHsts()	Used in production to enable HSTS(HTTP Strict Transport Security protocol) and enforces HTTPS
	UseHttpsRedirection()	Forces HTTP calls to automatically redirect to equivalent HTTPS addresses.
Static Files	UseStaticFiles()	Used to enable static files such as HTML, javascript, CSS and graphics files called early to avoid the need for authentication, session or MVC middleware
Cookie Policy	UseCookiePolicy()	Used to enforce cookie policy and display GDPR-friendly messaging
Authentication, Authorization and Sessions	UseAuthentication()	Used to enable authentication and then subsequently allow authorization
	UseSession()	Manually added to the Startup file to enable the Session middleware.

Middleware Methods

Please read terms of use for authorized access

Original Series

Endpoint Routing	UseEndpoints()	Usage varies on project templates used for MVC, Razor Pages and Blazor
	MapControllerRoute()	Set the default route and any custom routes when using MVC
	BlazorHub()	Sets up Blazor Hub

Diagnostics Middleware

Please read terms of use for authorized access

Original Series

Middleware	Extension Method	Description
DeveloperExceptionPageMiddleware	UseDeveloperExceptionPage()	Captures synchronous and asynchronous exceptions from the pipeline and generates HTML error responses
ExceptionHandlerMiddleware	UseExceptionHandler()	Catch exceptions, log them and re-execute in an alternate pipeline
StatusCodePagesMiddleware	UseStatusCodePages()	Check for response with status codes between 400 and 599
WelcomePageMiddleware	UseWelcomePage()	Display Welcome page for the root path.

Custom Middleware

- Create a custom middleware class
- A constructor accepting the “next” middleware delegate.
- Invoke method to execute logic
- Perform work on the request before forwarding or generating a response.

```
public class CustomMiddleware{  
  
    private RequestDelegate _next;  
  
    public CustomMiddleware(RequestDelegate next) {  
        _next = next;  
    }  
  
    public Task Invoke(HttpContext context) {  
        await _next.Invoke(context)  
    }  
}
```

When/Where Middleware Components Execute



PROGRAM CLASS

- Program class is used as the entry point to start the entire application

STARTUP CLASS

- It is used for configuring the middleware components.

SYED AWASE KHIRNI

Example 8: ASP.NET Core Custom Middleware Demo

Learning Outcomes:

1. Learning Outcome 1
2. Learning Outcome 2

Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

Search for templates (Alt+S) 

All languages All platforms All project types



Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

C# Linux macOS Windows Console



Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

Visual Basic Linux macOS Windows Console



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



Blazor App

Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).

C# Linux macOS Windows Cloud Web



ASP.NET Web Application (.NET Framework)

Back

Next

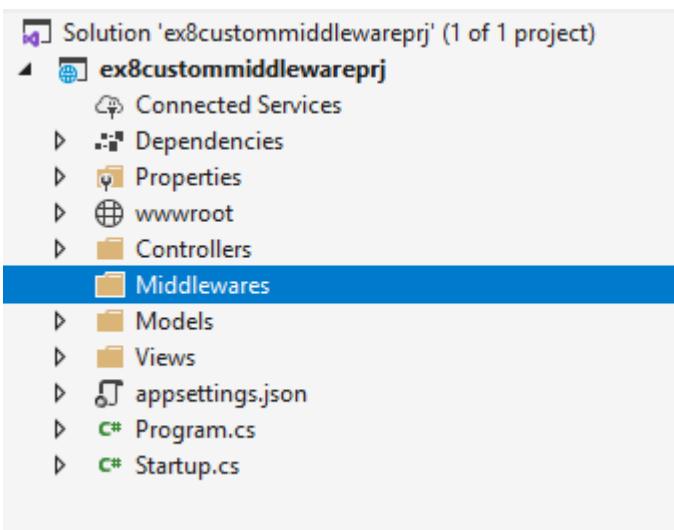
Configure your new project

ASP.NET Core Web Application C# Linux macOS Windows Cloud Service Web

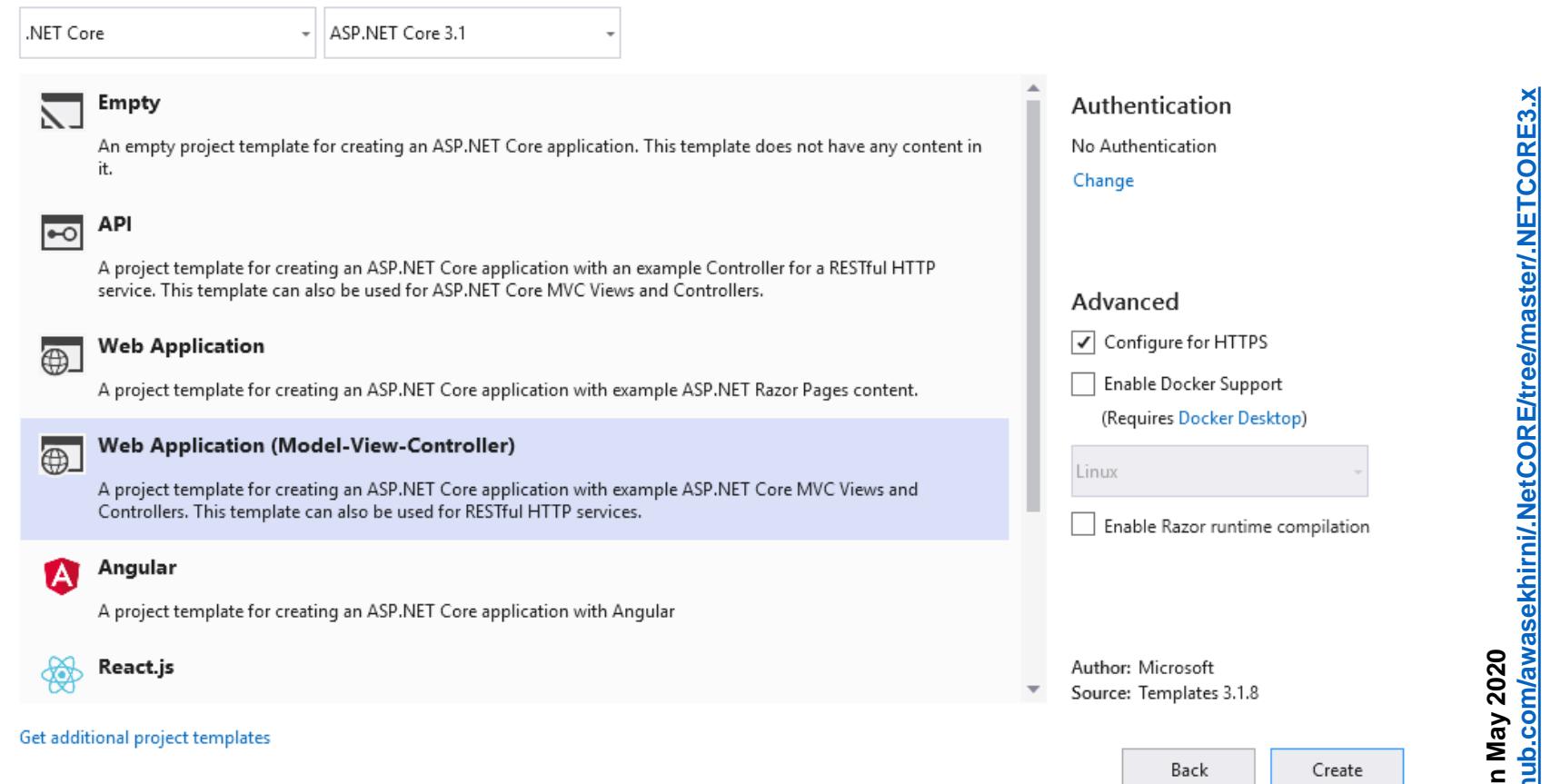
Project name

Location

Solution name  Place solution and project in the same directory



Create a new ASP.NET Core web application



.NET Core ASP.NET Core 3.1

Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Web Application
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

Web Application (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Angular
A project template for creating an ASP.NET Core application with Angular

React.js

Get additional project templates

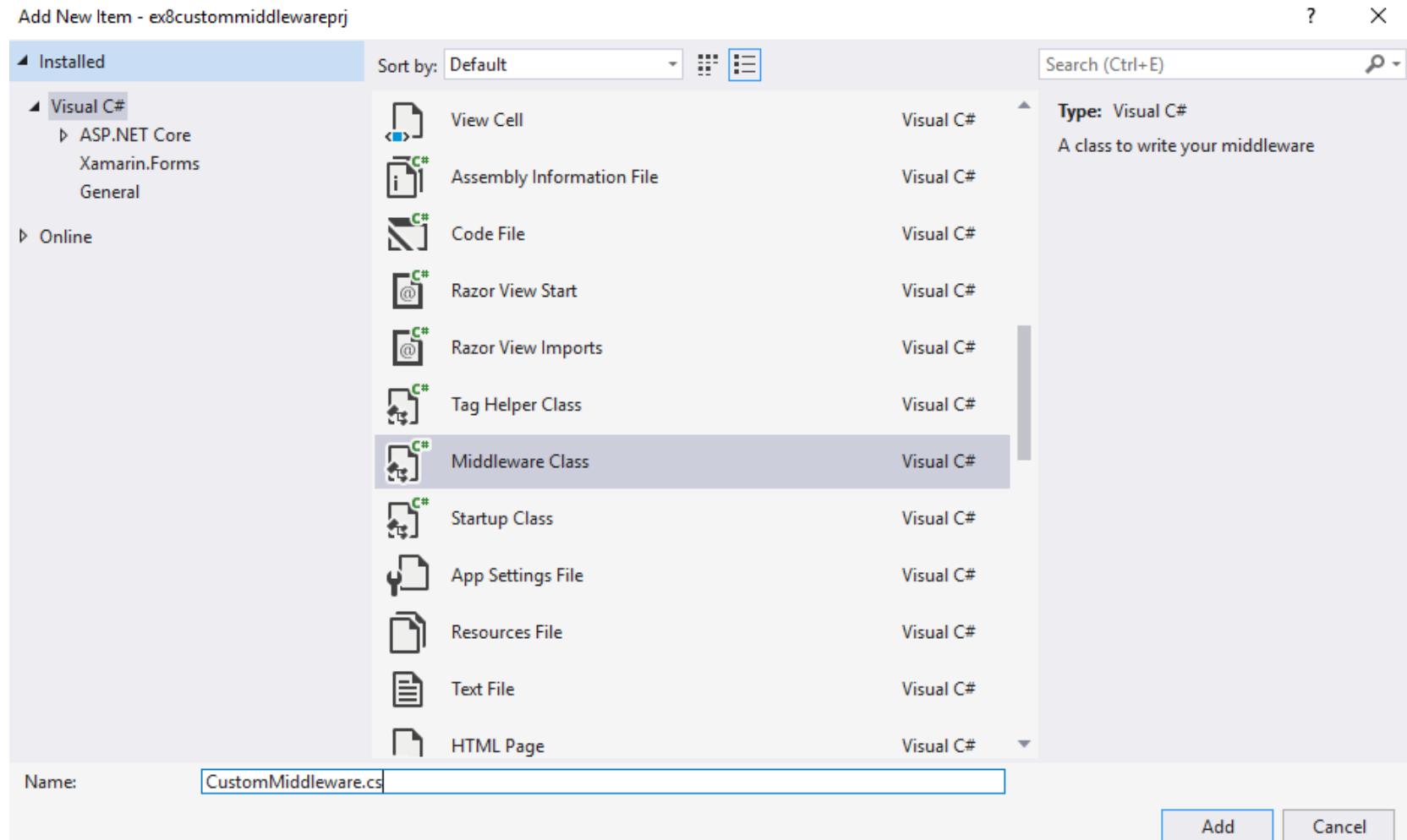
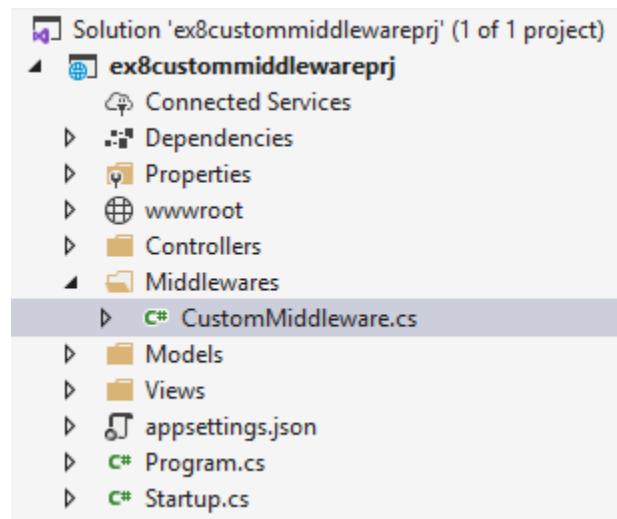
Authentication
No Authentication
Change

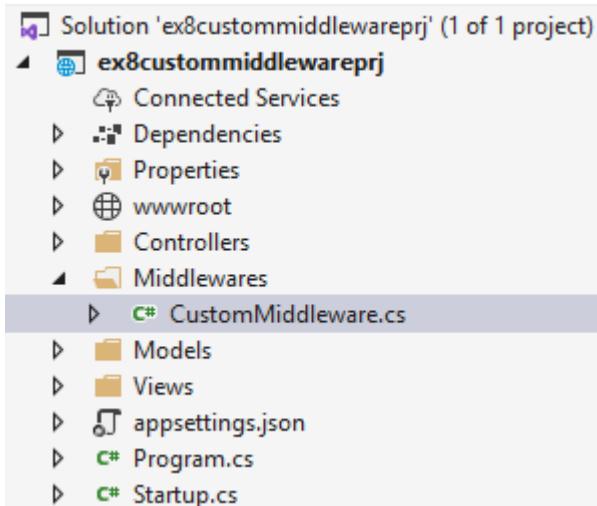
Advanced
 Configure for HTTPS
 Enable Docker Support
(Requires Docker Desktop)
Linux
 Enable Razor runtime compilation

Author: Microsoft
Source: Templates 3.1.8

Back Create

Example: 8



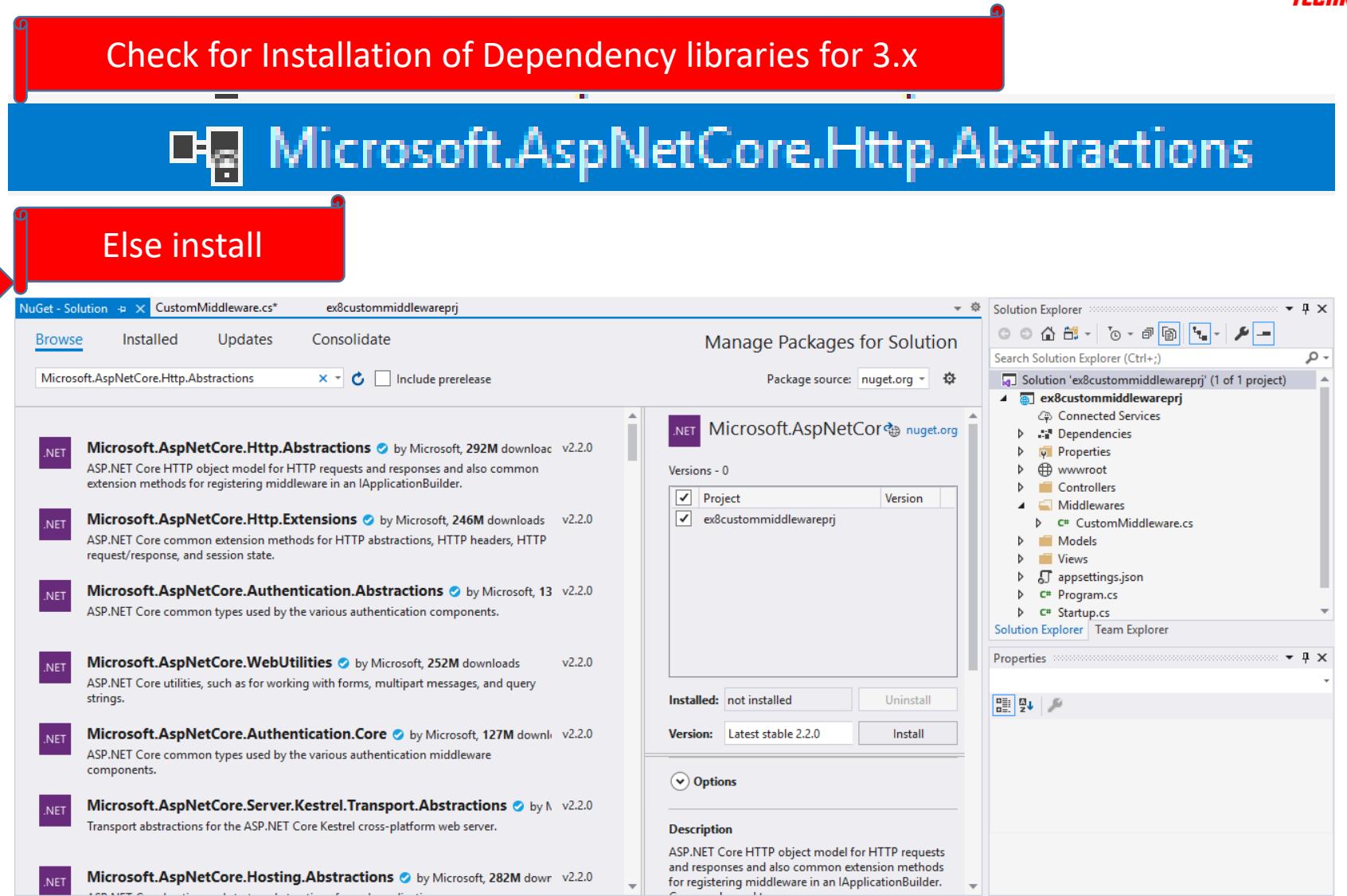


```
public class CustomMiddleware
{
    private readonly RequestDelegate _next;
    private readonly ILogger _logger;

    public CustomMiddleware(RequestDelegate next, ILoggerFactory logFactory)
    {
        _next = next;
        _logger = logFactory.CreateLogger("CustomMiddlewareLog");
    }

    public async Task Invoke(HttpContext httpContext)
    {
        _logger.LogInformation("Syed_Awase_Khirni--Custom_MiddlewareLog is Executing Now---");
        await _next(httpContext); //calling the next middleware in the pipeline
    }
}
```

- Install
"Microsoft.AspNetCore.Http.Abstractions" to
the solutions



The screenshot shows the Visual Studio IDE interface. On the left, the Solution Explorer displays a project named "ex8custommiddlewareprj" with files like Connected Services, Dependencies, Properties, wwwroot, Controllers, Middlewares (containing CustomMiddleware.cs), Models, Views, appsettings.json, Program.cs, and Startup.cs. A red arrow points from the "Middlewares" folder to the "CustomMiddleware.cs" file in the Solution Explorer. On the right, the code editor shows the "Startup.cs" file with the following code:

```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.  
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)  
{  
    //adding the custom middleware here  
    app.UseCustomMiddleware();  
}
```

Example: 8

The screenshot shows a .NET Core application development environment. On the left, the Solution Explorer displays the project structure for 'ex8custommiddlewareprj'. The 'Middlewares' folder contains 'CustomMiddleware.cs' and 'CustomMiddlewareExtensions'. The 'Startup.cs' file is also visible. In the center, a browser window shows the 'Home Page - ex8custommi...' page at 'localhost:44300'. The page title is 'ex8custommiddlewareprj' and it features a large 'Welcome' heading and a link to 'Learn about building Web apps with ASP.NET Core.'. At the top, the Output window shows a message from 'CustomMiddlewareLog': 'Information: Syed Awase Khirni--Custom_MiddlewareLog is Executing Now---'. A status bar at the bottom indicates the build was successful.

- Lets build our custom middleware

SYED AWASE KHIRNI

Example 9: ASP.NET Core 3 Online Users Middleware

Learning Outcomes:

1. Learning Outcome 1
2. Learning Outcome 2

Create a new project

Recent project templates

A list of your recently accessed templates will be displayed here.

Search for templates (Alt+S) 

All languages All platforms All project types



Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

C# Linux macOS Windows Console



Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

Visual Basic Linux macOS Windows Console



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



Blazor App

Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).

C# Linux macOS Windows Cloud Web



ASP.NET Web Application (.NET Framework)

Back

Next

Configure your new project

ASP.NET Core Web Application C# Linux macOS Windows Cloud Service Web

Project name

Location

Solution

Solution name i

Place solution and project in the same directory

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

Create a new ASP.NET Core web application

.NET Core ASP.NET Core 3.1

Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Web Application
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

Web Application (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Angular
A project template for creating an ASP.NET Core application with Angular

React.js

[Get additional project templates](#)

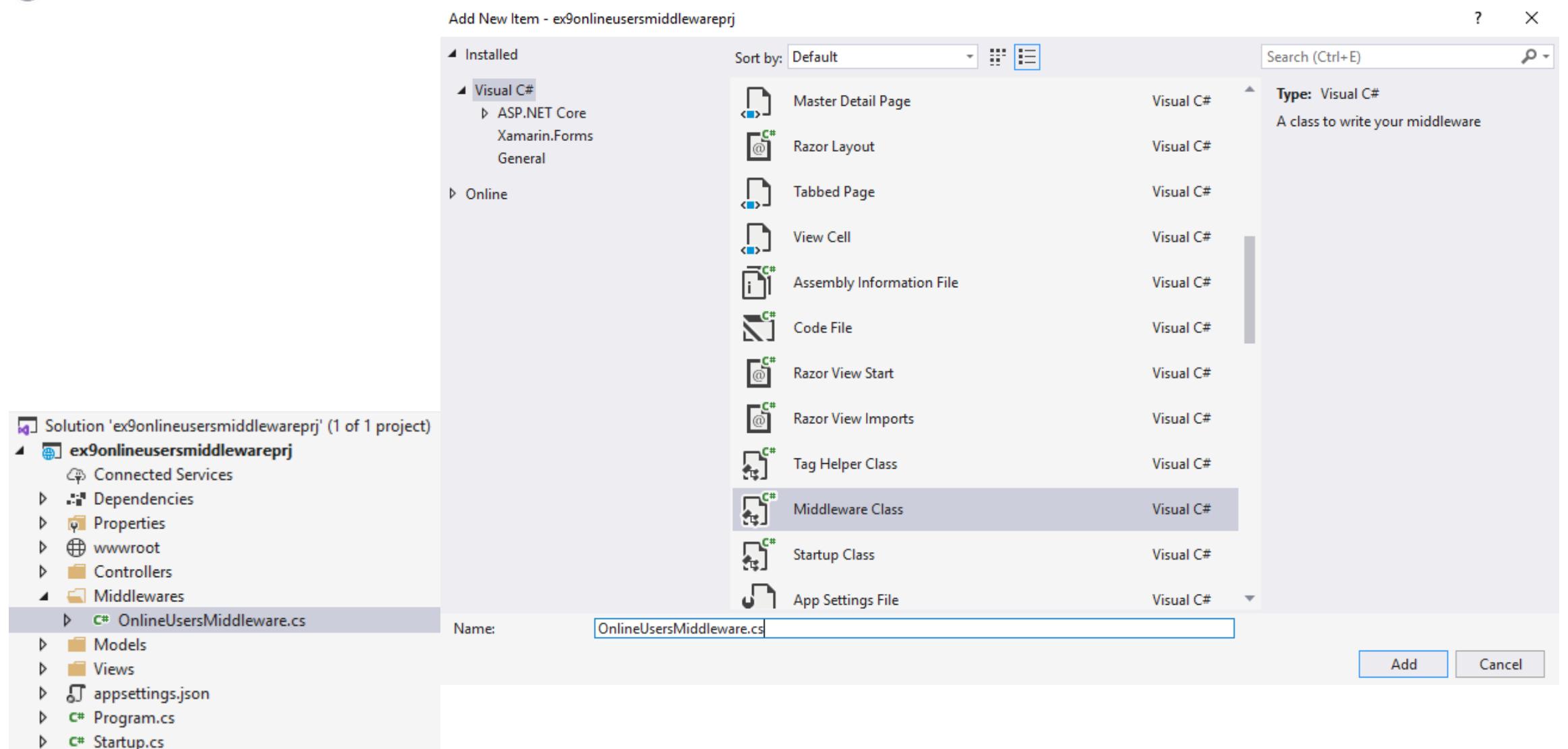
Authentication
No Authentication
[Change](#)

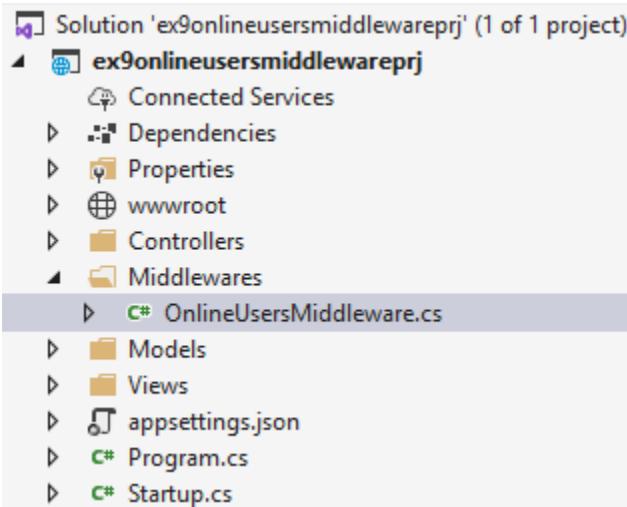
Advanced

- Configure for HTTPS
- Enable Docker Support
(Requires [Docker Desktop](#))
- Linux
- Enable Razor runtime compilation

Author: Microsoft
Source: Templates 3.1.8

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





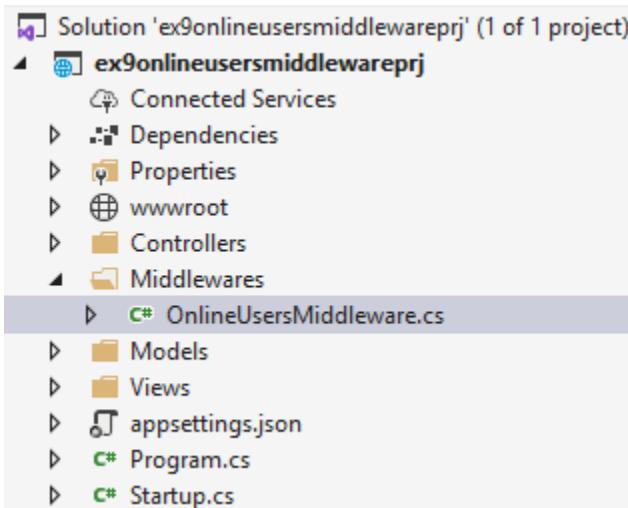
OnlineUsersMiddleware.cs

```
namespace ex9onlineusersmiddlewareprj.Middlewares
{
    // You may need to install the Microsoft.AspNetCore.Http.Abstractions package into your project
    3 references
    public class OnlineUsersMiddleware
    {
        private readonly RequestDelegate _next;

        private readonly string _cookieName;
        private readonly int _lastActivityMinutes = 20;

        private static readonly ConcurrentDictionary<string, bool> _allKeys = new ConcurrentDictionary<string, bool>();

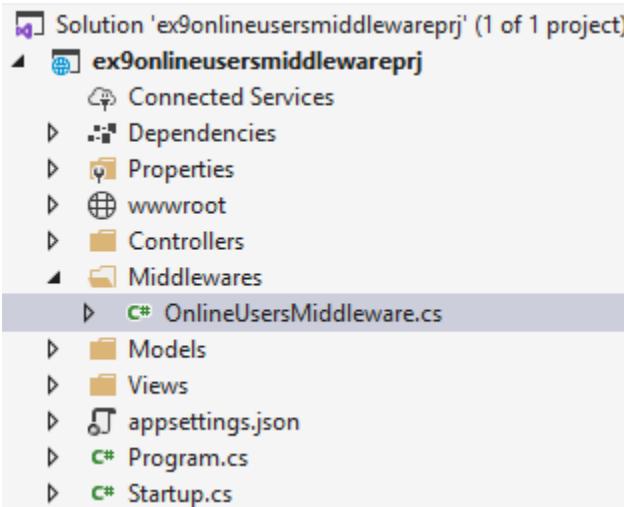
        0 references
        public OnlineUsersMiddleware(RequestDelegate next, string cookieName ="UserGuid", int lastActivityMinutes=20)
        {
            _next = next;
            _cookieName = cookieName;
            _lastActivityMinutes = lastActivityMinutes;
        }
    }
}
```



```
0 references
public Task InvokeAsync(HttpContext httpContext, IMemoryCache memoryCache)
{
    if (httpContext.Request.Cookies.TryGetValue(
        _cookieName,
        out var userGuid) == false)
    {
        userGuid = Guid.NewGuid().ToString();
        httpContext.Response.Cookies.Append(_cookieName, userGuid,
            new CookieOptions { HttpOnly = true, MaxAge = TimeSpan.FromDays(10) });
    }
    memoryCache.GetOrCreate(userGuid, cacheEntry => {

        if (_allKeys.TryAdd(userGuid, true) == false)
        {
            //if add key failed, set the expiration to the past
            cacheEntry.AbsoluteExpiration = DateTimeOffset.MinValue;
        }
        else
        {
            cacheEntry SlidingExpiration = TimeSpan.FromSeconds(10);
            cacheEntry.RegisterPostEvictionCallback(RemoveKeyWhenExpired);
        }
        return string.Empty;
    });

    return _next(httpContext);
}
```

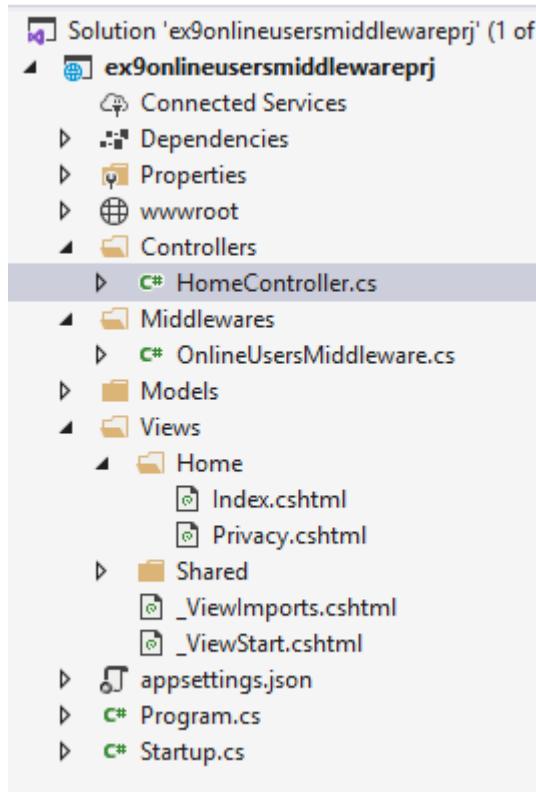


OnlineUsersMiddleware.cs

```
//step4- add the method to remove the key when expired
1 reference
private void RemoveKeyWhenExpired(object key, object value, EvictionReason reason, object state)
{
    string strKey = (string)key;
    if (!_allKeys.TryRemove(strKey, out _))
        _allKeys.TryUpdate(strKey, false, true);
}

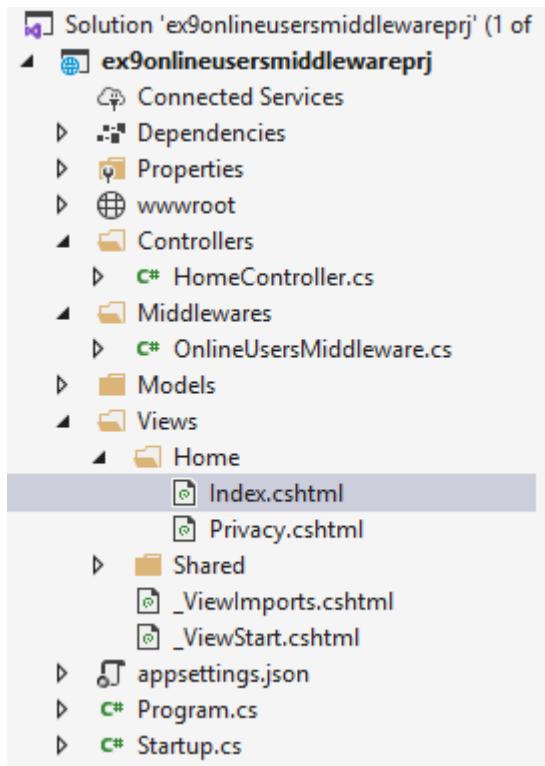
//step 5- get the no of online users count to render on the view pass this value to the controller
1 reference
public static int GetOnlineUsersCount() => _allKeys.Count(p => p.Value);

// Extension method used to add the middleware to the HTTP request pipeline.
0 references
public static class OnlineUsersMiddlewareExtensions
{
    1 reference
    public static IApplicationBuilder UseOnlineUsersMiddleware(this IApplicationBuilder builder, string
        cookieName="UserGuid", int lastActivityMinutes=20)
    {
        return builder.UseMiddleware<OnlineUsersMiddleware>(cookieName,lastActivityMinutes);
    }
}
```



HomeController.cs

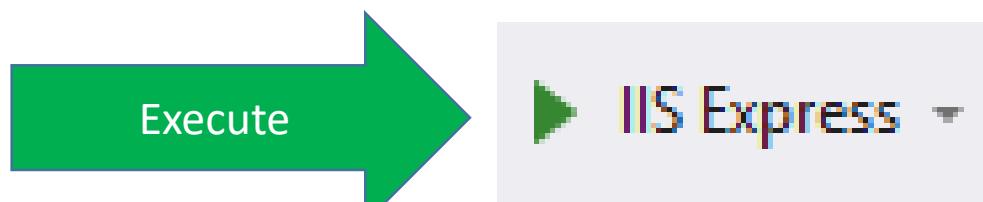
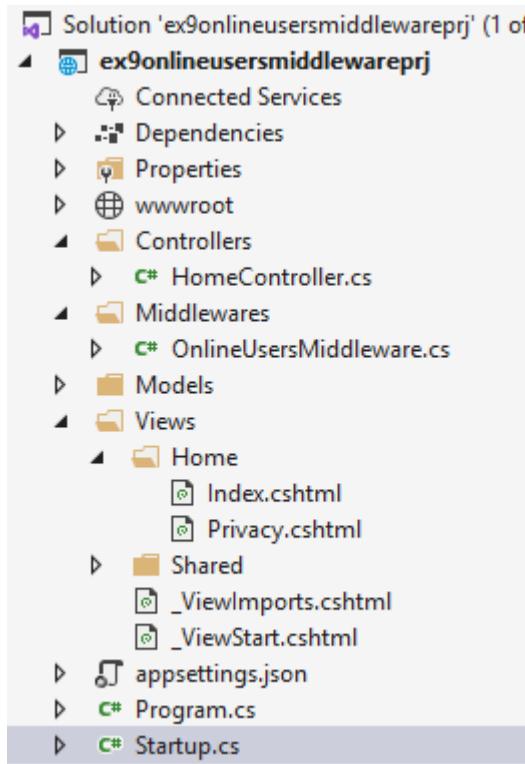
```
0 references
public IActionResult Index()
{
    //step 6- modelbinding the count of users to render to the view model
    var onlineUsersCount = OnlineUsersMiddleware.GetOnlineUsersCount();
    return View(onlineUsersCount);
}
```



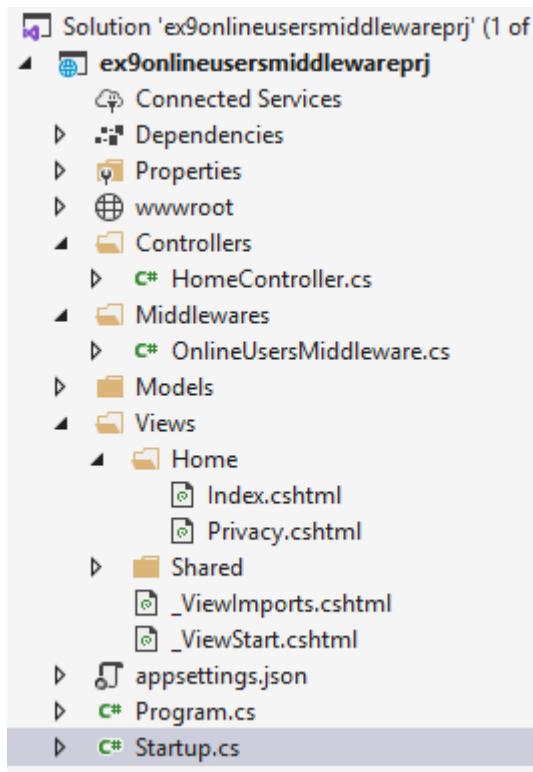
Views/Home/Index.cshtml

```
@{  
    ViewData["Title"] = "Home Page";  
}  
  
<div class="text-center">  
    <h1 class="display-4">Welcome</h1>  
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>  
</div>  
  
<!--Step 7- pass the online users here modelbinding happens here --&gt;<br/><div class="text-center">  
    <h1 class="display-2">Online Users:@Model</h1>  
    <p> Displaying the number of online users using middleware</p>  
</div>
```





Example: 9



Opera
Join a presentation... Startups — Remote... florinpop17/app-id... TypeORM - Amazing...
ex9onlineusersmiddlewareprj
Welcome
Learn about building Web apps with ASP.NET Core.
Online Users:3
Displaying the number of online users using middleware

Chrome
Home Page ~ ex9onlineusersmiddlewareprj
localhost:44370 Apps Core Java Topics &... Apps Learn ASP.NET MV...
ex9onlineusersmiddlewareprj
Welcome
Learn about building Web apps with ASP.NET Core.
Online Users:3
© 2020 - ex9onlineusersmiddlewareprj - Privacy
Displaying the number of online users using middleware

FireFox
https://localhost:44370 ex9onlineusersmiddlewareprj Home Privacy
Welcome
Learn about building Web apps with ASP.NET Core.
Online Users:3
Displaying the number of online users using middleware

<https://vicluar.wordpress.com/2020/06/01/implement-health-check-in-asp-net-core-services/>

SYED AWASE KHIRNI

Example 10 : ASP.NET Core Health Check Services

Learning Outcomes:

1. Learning Outcome 1
2. Learning Outcome 2

Create a new project

Recent project templates



ASP.NET Core Web Application

C#

 [All languages](#) [All platforms](#) [All project types](#) C# Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

[C#](#) [Linux](#) [macOS](#) [Windows](#) [Console](#) VB Console App (.NET Core)

A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.

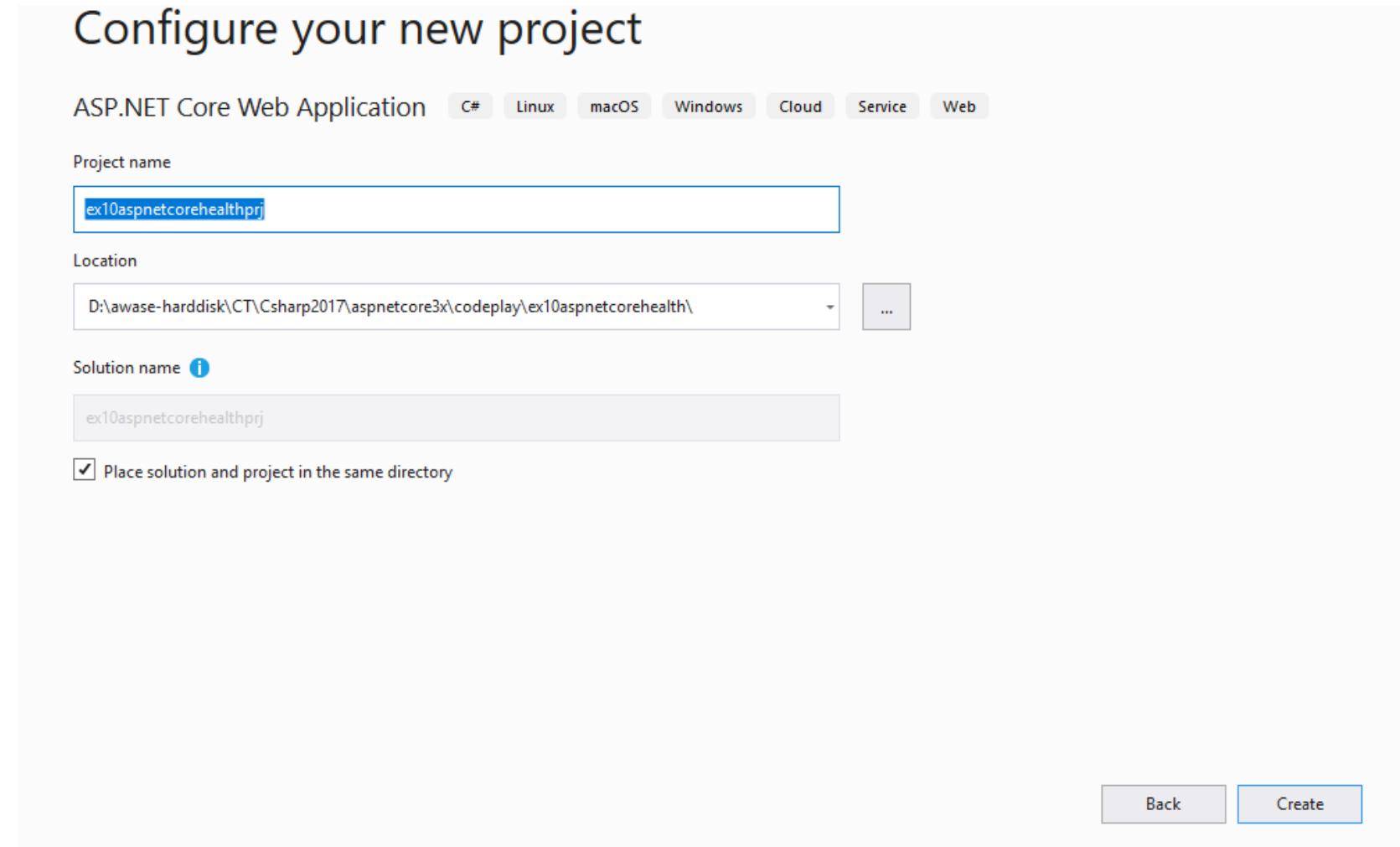
[Visual Basic](#) [Linux](#) [macOS](#) [Windows](#) [Console](#) ASP.NET Core Web Application

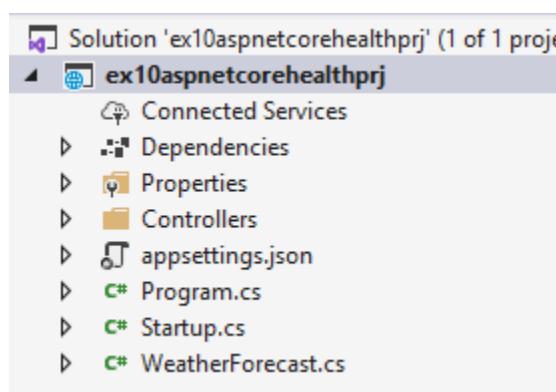
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.

[C#](#) [Linux](#) [macOS](#) [Windows](#) [Cloud](#) [Service](#) [Web](#) Blazor App

Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).

[C#](#) [Linux](#) [macOS](#) [Windows](#) [Cloud](#) [Web](#) ASP.NET Web Application (.NET Framework)[Back](#) [Next](#)





Create a new ASP.NET Core web application

.NET Core ASP.NET Core 3.1

Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Web Application
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

Web Application (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core Views and Controllers. This template can also be used for RESTful HTTP services.

Angular
A project template for creating an ASP.NET Core application with Angular

React.js

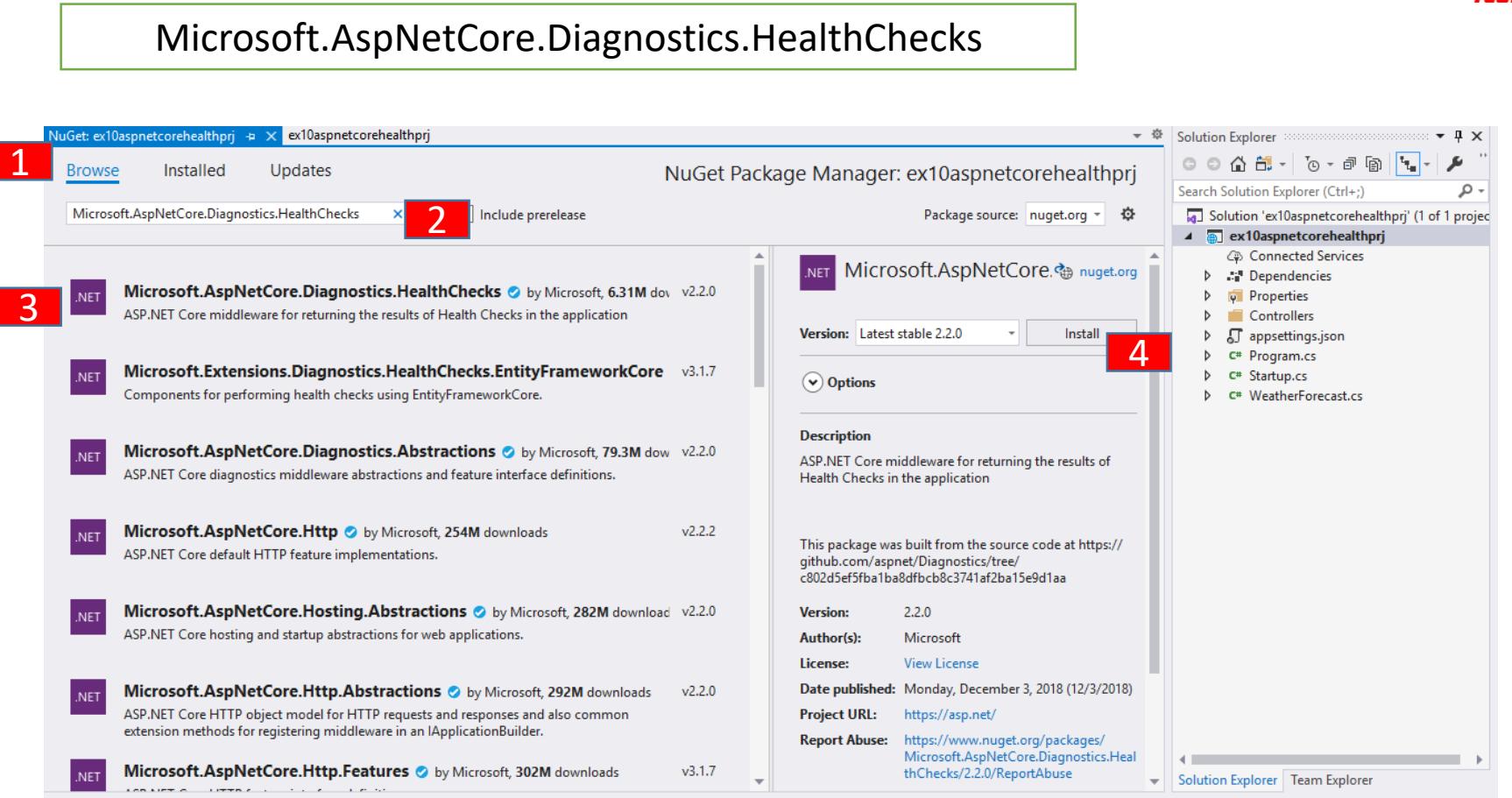
Get additional project templates

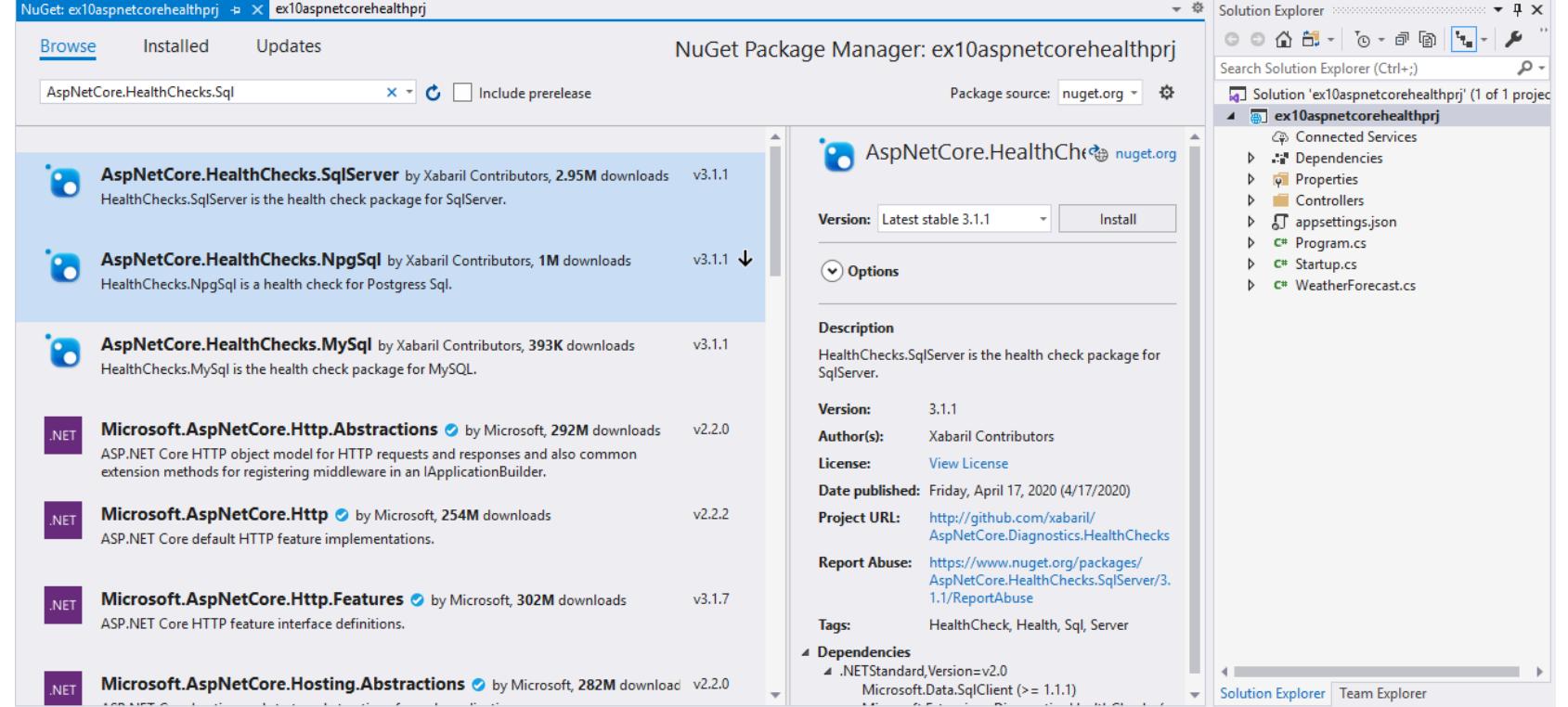
Authentication
No Authentication
[Change](#)

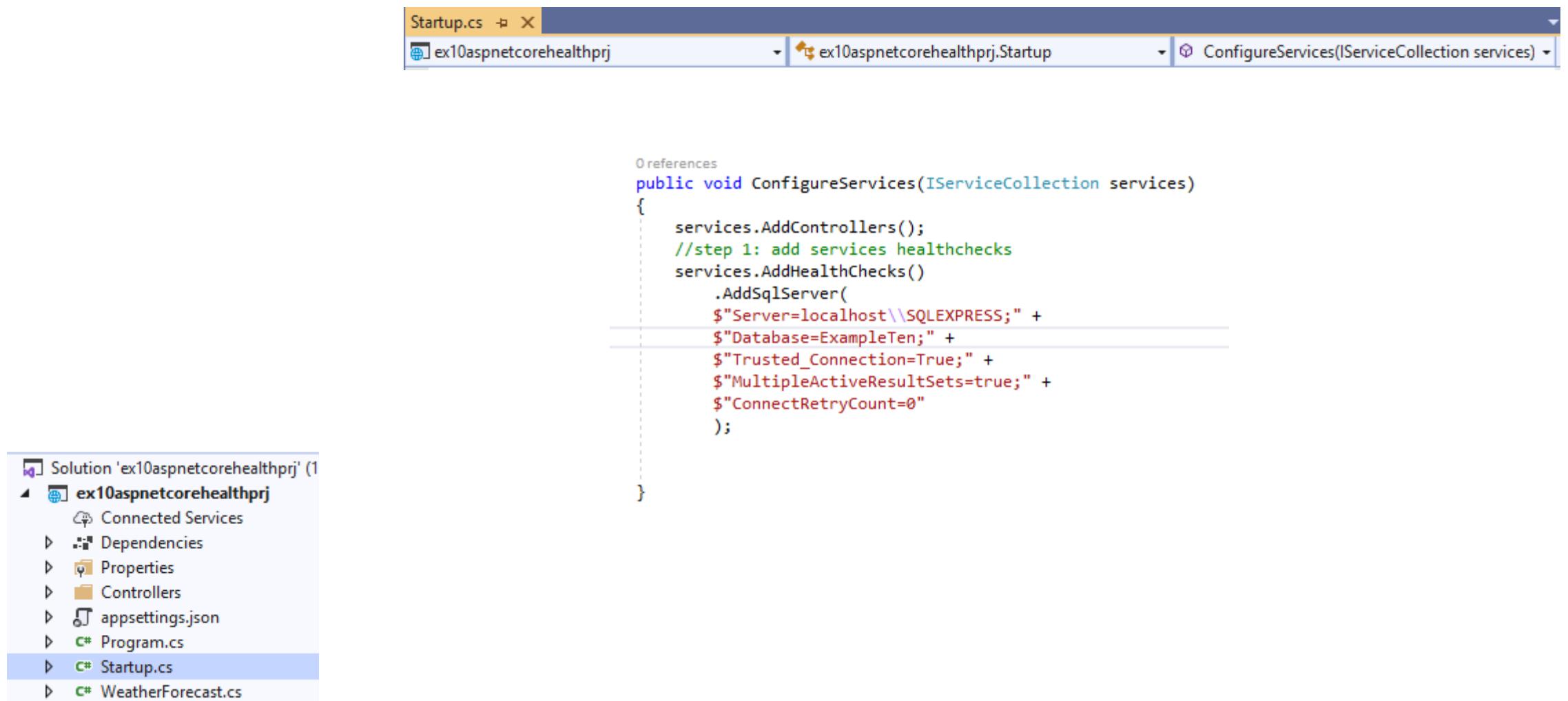
Advanced
 Configure for HTTPS
 Enable Docker Support
(Requires [Docker Desktop](#))
Linux

Author: Microsoft
Source: Templates 3.1.8

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



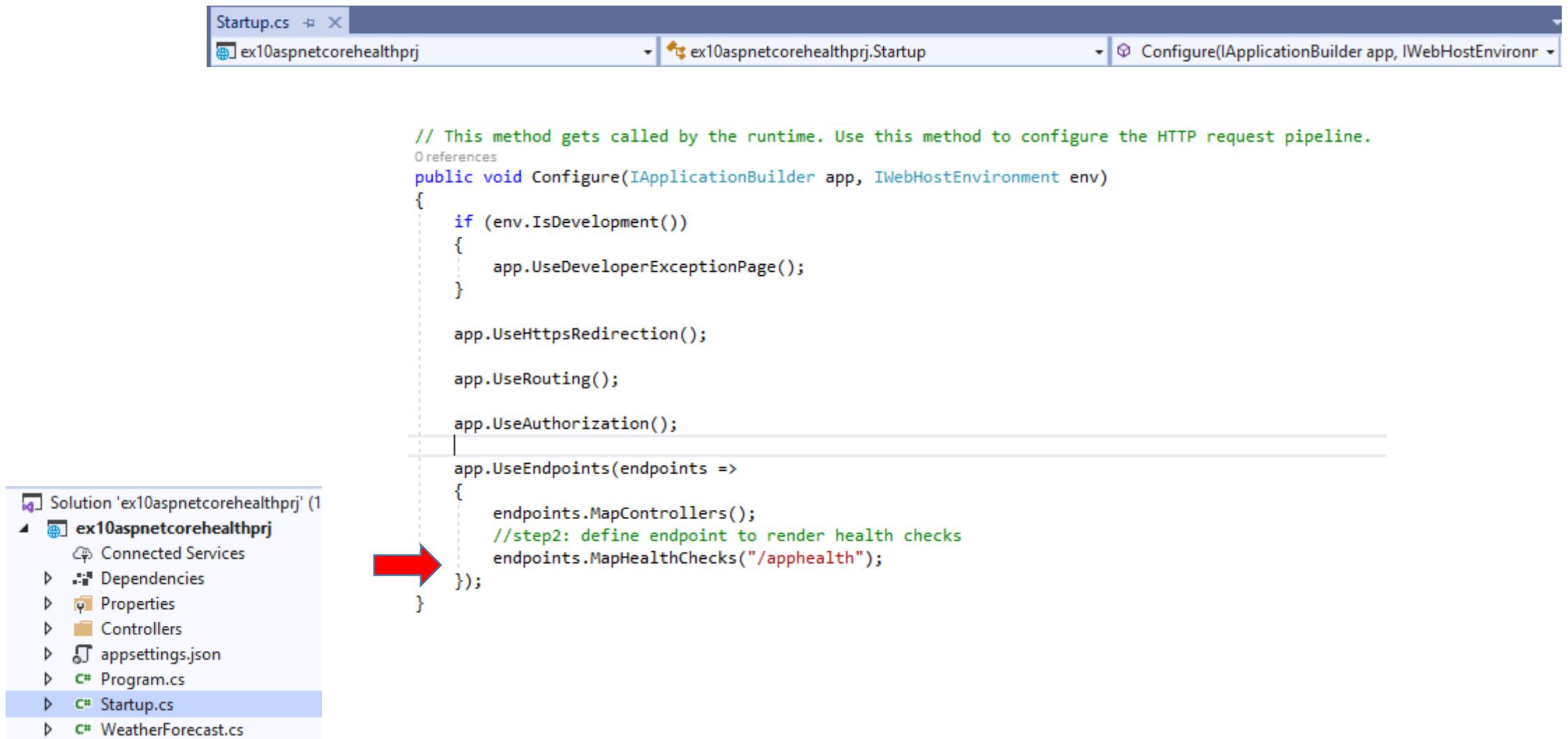




The screenshot shows a Visual Studio interface. The title bar says "Startup.cs" and "ex10aspnetcorehealthprj.Startup". The code editor displays the following C# code:

```
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllers();
    //step 1: add services healthchecks
    services.AddHealthChecks()
        .AddSqlServer(
            $"Server=localhost\\SQLEXPRESS;" +
            $"Database=ExampleTen;" +
            $"Trusted_Connection=True;" +
            $"MultipleActiveResultSets=true;" +
            $"ConnectRetryCount=0"
        );
}
```

The "ConfigureServices" method is highlighted. In the Solution Explorer on the left, the "Startup.cs" file is selected.



The screenshot shows the Visual Studio IDE. In the top navigation bar, the file 'Startup.cs' is selected. The tabs below show the project name 'ex10aspnetcorehealthprj' and the class 'ex10aspnetcorehealthprj.Startup'. The status bar indicates the method 'Configure(IApplicationBuilder app, IWebHostEnvironment env)'. The code editor displays the following C# code:

```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.  
0 references  
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)  
{  
    if (env.IsDevelopment())  
    {  
        app.UseDeveloperExceptionPage();  
    }  
  
    app.UseHttpsRedirection();  
  
    app.UseRouting();  
  
    app.UseAuthorization();  
  
    app.UseEndpoints(endpoints =>  
    {  
        endpoints.MapControllers();  
        //step2: define endpoint to render health checks  
        endpoints.MapHealthChecks("/apphealth");  
    });  
}
```

A red arrow points to the line 'endpoints.MapHealthChecks("/apphealth");' in the code.

In the bottom-left corner, the Solution Explorer shows the project structure:

- Solution 'ex10aspnetcorehealthprj' (1)
- ex10aspnetcorehealthprj
 - Connected Services
 - Dependencies
 - Properties
 - Controllers
 - appsettings.json
 - Program.cs
 - Startup.cs**
 - WeatherForecast.cs

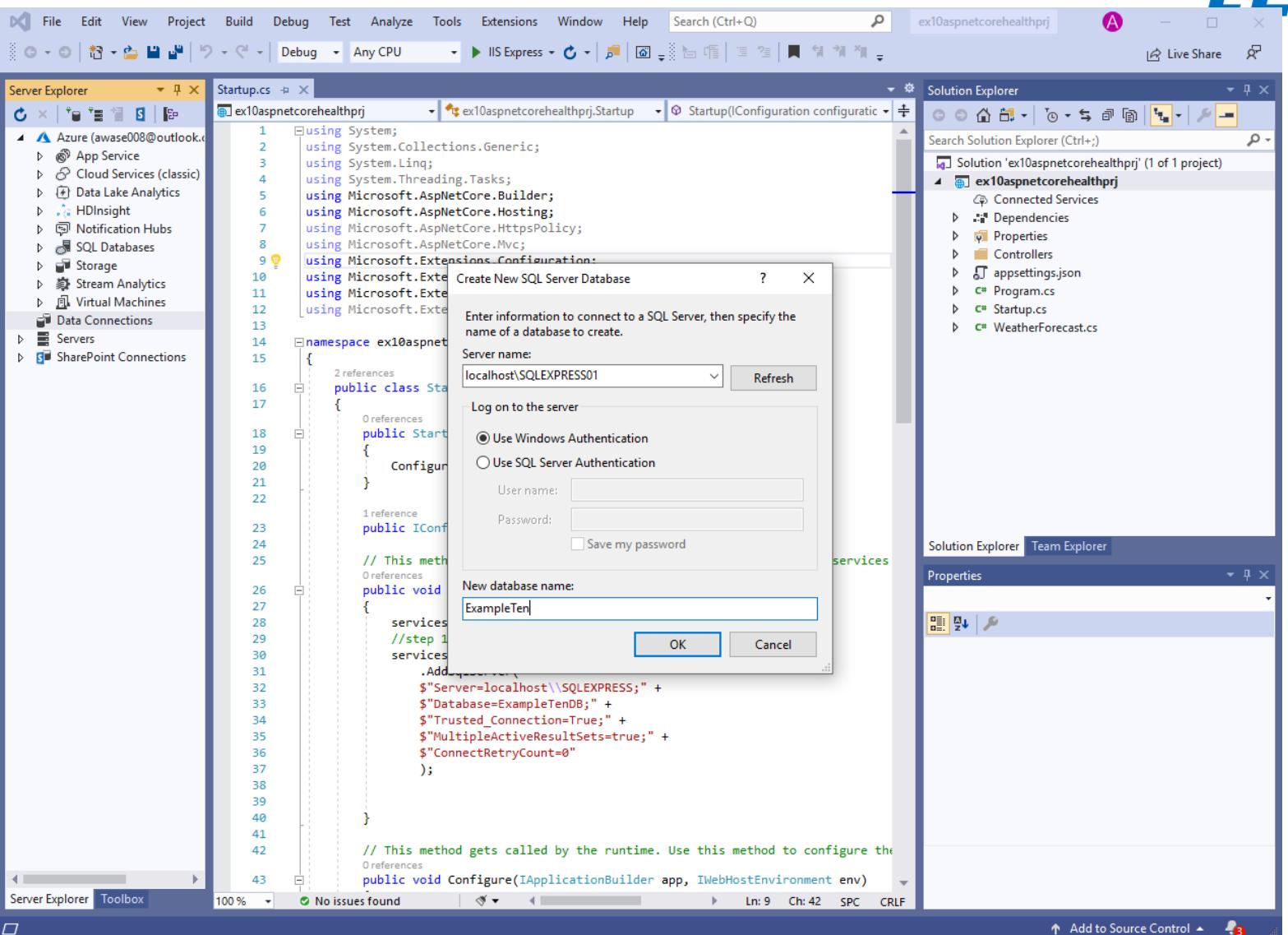
Example:

► Azure (awase008@outlook.com - 0 subscriptions)

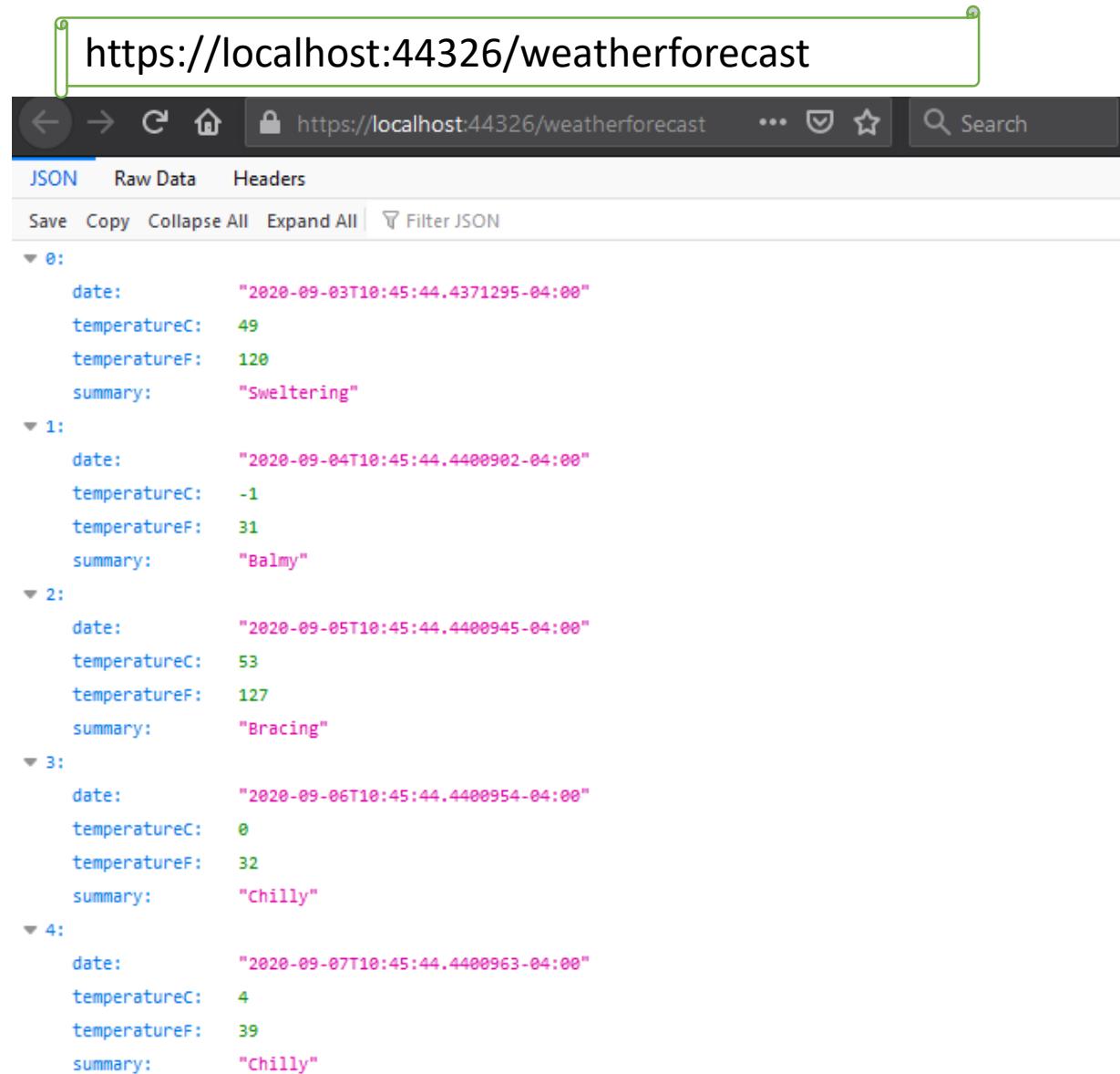
- ▷ App Service
- ▷ Cloud Services (classic)
- ▷ Data Lake Analytics
- ▷ HDInsight
- ▷ Notification Hubs
- ▷ SQL Databases
- ▷ Storage
- ▷ Stream Analytics
- ▷ Virtual Machines

► Data Connections

- ▷ desktop-5g6inot\sqlexpress01.ExampleTen.dbo
- ▷ Servers
- ▷ SharePoint Connections



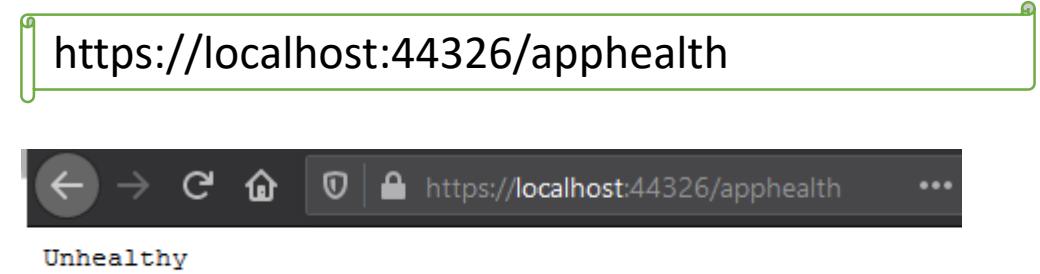
Example:



The screenshot shows a browser window displaying a JSON response from a local host. The URL in the address bar is <https://localhost:44326/weatherforecast>. The browser interface includes a back button, forward button, refresh button, home icon, and a search bar labeled "Search". Below the address bar, there are tabs for "JSON", "Raw Data", and "Headers", with "JSON" being the active tab. Under the "JSON" tab, there are buttons for "Save", "Copy", "Collapse All", "Expand All", and "Filter JSON". The main content area displays a JSON array with five elements, indexed from 0 to 4. Each element contains four properties: "date", "temperatureC", "temperatureF", and "summary". The data is as follows:

```
[{"date": "2020-09-03T10:45:44.4371295-04:00", "temperatureC": 49, "temperatureF": 120, "summary": "Sweltering"}, {"date": "2020-09-04T10:45:44.4400902-04:00", "temperatureC": -1, "temperatureF": 31, "summary": "Balmy"}, {"date": "2020-09-05T10:45:44.4400945-04:00", "temperatureC": 53, "temperatureF": 127, "summary": "Bracing"}, {"date": "2020-09-06T10:45:44.4400954-04:00", "temperatureC": 0, "temperatureF": 32, "summary": "Chilly"}, {"date": "2020-09-07T10:45:44.4400963-04:00", "temperatureC": 4, "temperatureF": 39, "summary": "Chilly"}]
```

Example:



dotnet tool install --global dotnet-counters --version 1.0.3-preview5.19251.2

```
PS C:\Users\Awase Khirni Syed> dotnet tool install --global dotnet-counters --version 1.0.3-preview5.19251.2
Welcome to .NET 5.0!
-----
SDK Version: 5.0.100-preview.7.20366.6
Telemetry
-----
The .NET tools collect usage data in order to help us improve your experience. The data is anonymous. It is collected by Microsoft and shared with the community. You can opt-out of telemetry by setting the DOTNET_CLI_TELEMETRY_OPTOUT environment variable to '1' or 'true' using your favorite shell.

Read more about .NET CLI Tools telemetry: https://aka.ms/dotnet-cli-telemetry
-----
Installed an ASP.NET Core HTTPS development certificate.
To trust the certificate run 'dotnet dev-certs https --trust' (Windows and macOS only).
Learn about HTTPS: https://aka.ms/dotnet-https
-----
Write your first app: https://aka.ms/dotnet-hello-world
Find out what's new: https://aka.ms/dotnet-whats-new
Explore documentation: https://aka.ms/dotnet-docs
Report issues and find source on GitHub: https://github.com/dotnet/core
Use 'dotnet --help' to see available commands or visit: https://aka.ms/dotnet-cli
-----
You can invoke the tool using the following command: dotnet-counters
Tool 'dotnet-counters' (version '1.0.3-preview5.19251.2') was successfully installed.
```

Dotnet-counters --help

```
PS C:\Users\Awase Khirni Syed> dotnet-counters --help
Usage:
  dotnet-counters [options] [command]

Options:
  --version    Display version information

Commands:
  monitor <counter_list>      Start monitoring a .NET application
  list          Display a list of counter names and descriptions, grouped by provider.
```

Dotnet tool install --global dotnet-trace

```
PS C:\Users\Awase Khirni Syed> dotnet tool install --global dotnet-trace
You can invoke the tool using the following command: dotnet-trace
Tool 'dotnet-trace' (version '3.1.141901') was successfully installed.
```

```
PS C:\Users\Awase Khirni Syed> dotnet-trace --help
Usage:
  dotnet-trace [options] [command]

Options:
  --version   Display version information

Commands:
  collect      Collects a diagnostic trace from a currently running process
  ps          Lists the dotnet processes that traces can be collected
  list-profiles  Lists pre-built tracing profiles with a description of what providers and filters are in each profile
  convert <input-filename>    Converts traces to alternate formats for use with alternate trace analysis tools. Can only convert from the nettrace format
```

```
PS C:\Users\Awase Khirni Syed> dotnet-trace ps
126992 iisexpress C:\Program Files\IIS Express\iisexpress.exe
```

```
PS C:\Users\Awase Khirni Syed> dotnet-trace collect -p 126992
No profile or providers specified, defaulting to trace profile 'cpu-sampling'

Provider Name           Keywords          Level        Enabled By
Microsoft-DotNETCore-SampleProfiler 0x0000F00000000000 Informational(4)  --profile
Microsoft-Windows-DotNERTRuntime    0x00000014c14FCCBD  Informational(4)  --profile

Process      : C:\Program Files\IIS Express\iisexpress.exe
Output File  : C:\Users\Awase Khirni Syed\trace.nettrace
```

```
dotnet tool install --global dotnet-dump --version 1.0.3-preview5.19251.2
```

```
PS C:\Users\Awase Khirni Syed> dotnet tool install --global dotnet-dump --version 1.0.3-preview5.19251.2
You can invoke the tool using the following command: dotnet-dump
Tool 'dotnet-dump' (version '1.0.3-preview5.19251.2') was successfully installed.
```