Introduction to asp.net core

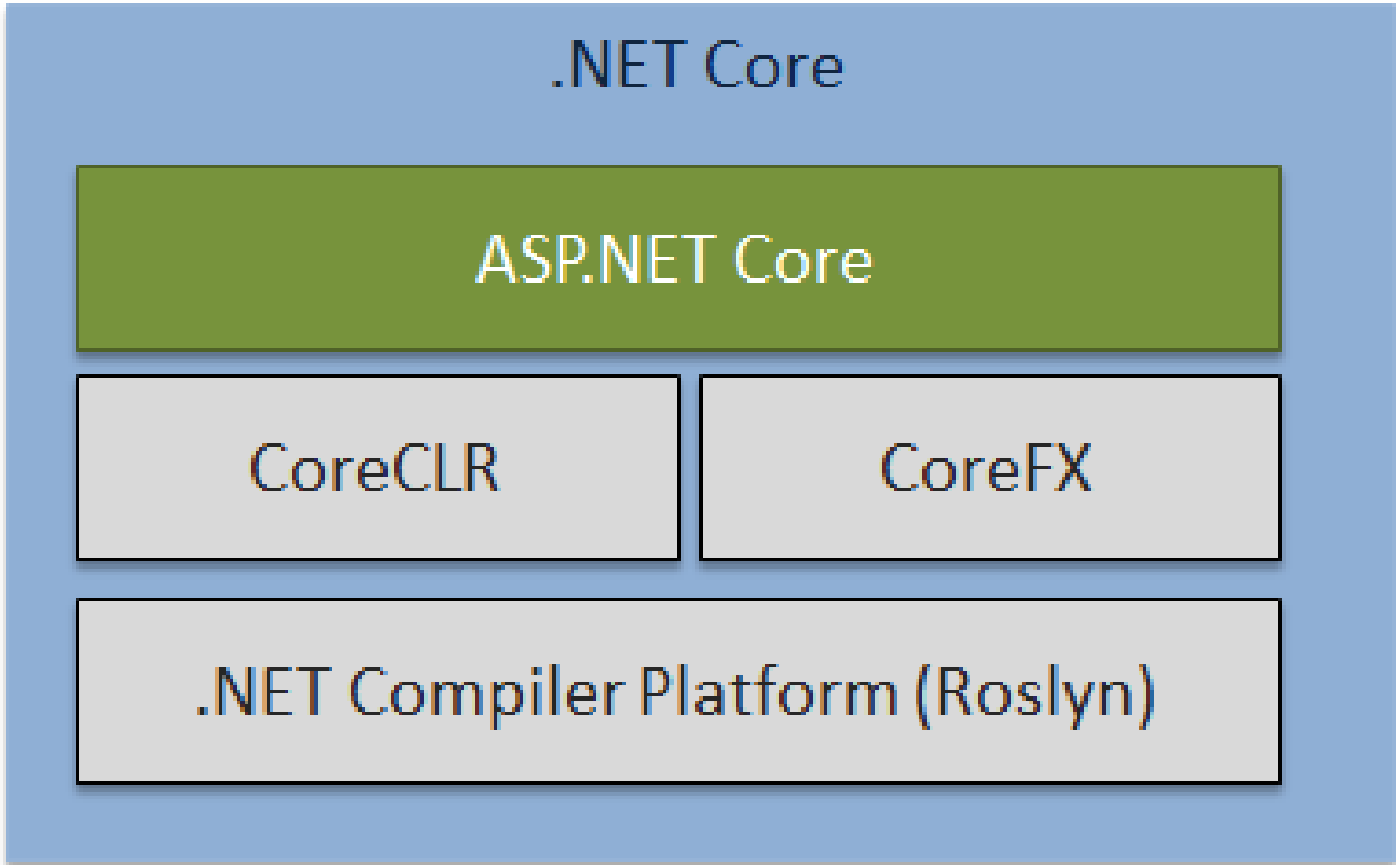
What is ASP.NET core ?

- ASP.NET Core is a cross-platform, high-performance, open-source framework for building modern, cloud-based, Internet-connected applications. With ASP.NET Core, you can:
  - Build web apps and services, IoT apps, and mobile backends.
  - Use your favourite development tools on Windows, macOS, and Linux.
  - Deploy to the cloud or on-premises.
  - Run on .NET Core or .NET Framework.

ASP.NET core History?

Version	Release date	Released with	Support Ends[18]
.NET Core 1.0	2016-06-27[19]	Visual Studio 2015 Update 3	June 27, 2019
.NET Core 1.1	2016-11-16[20]	Visual Studio 2017 Version 15.0	June 27, 2019
.NET Core 2.0	2017-08-14[21]	Visual Studio 2017 Version 15.3	October 1, 2018
.NET Core 2.1	2018-05-30[22]	Visual Studio 2017 Version 15.7	August 21, 2021
.NET Core 2.2	2018-12-04[23]	Visual Studio 2019 Version 16.0	December 23, 2019
.NET Core 3.0	2019-09-23[25]	Visual Studio 2019 Version 16.3	March 3, 2020
.NET Core 3.1	2019-12-03[26]	Visual Studio 2019 Version 16.4	**December 3, 2022**
.NET 5[29]	2020-11 (projected)		

.NET Core ARCHITECTURE



មុខវិជ្ជាដែលត្រូវបំពេញមុន

- HTML
- CSS
- JavaScript
- និង basic C#

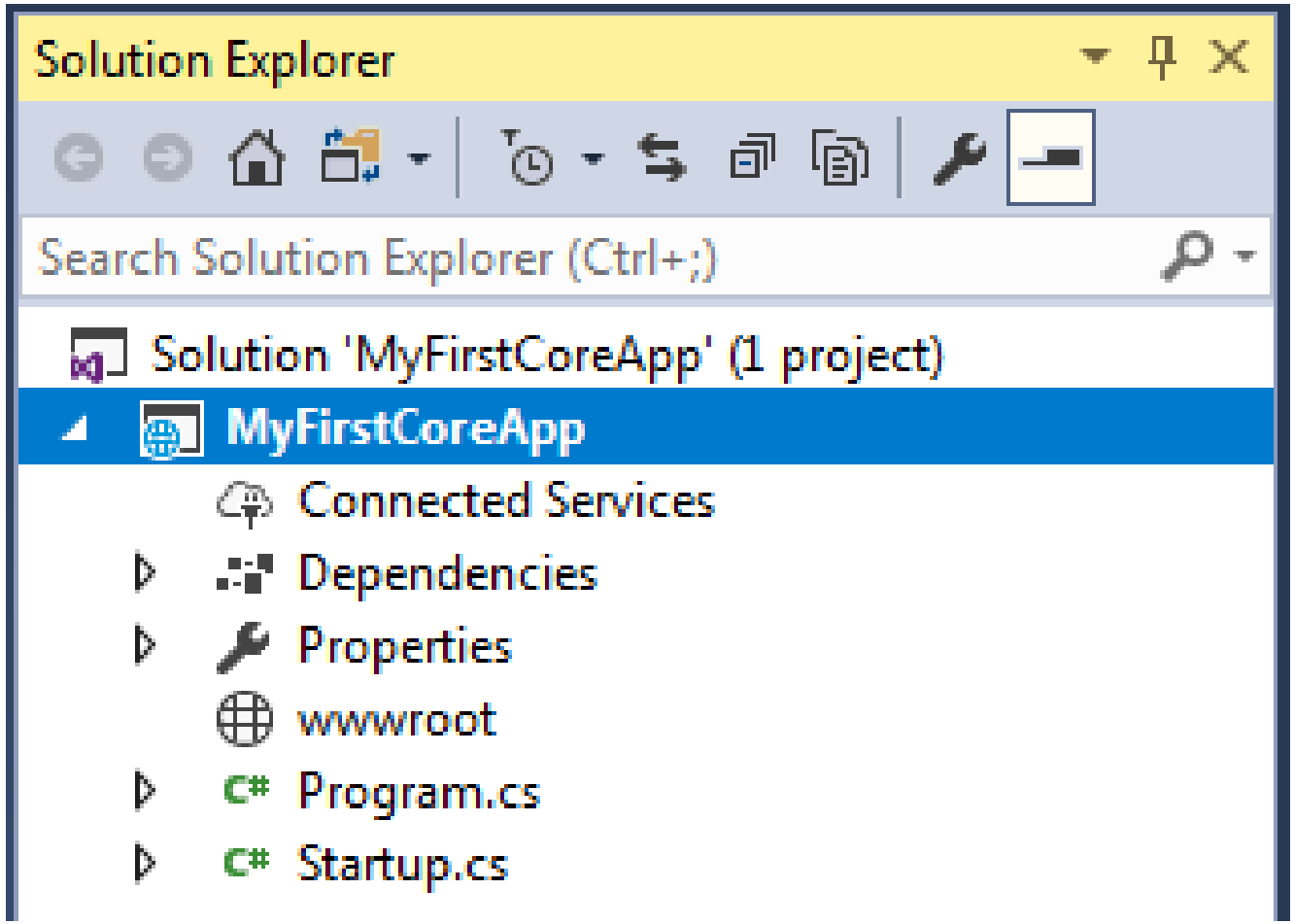
To develop asp.net core application?

- Integrated Development Environment
  - Visual studio 2019
- Dot Net core SDK
  - .NET core 3.1
- Database
  - SQL Server
  - MySQL
  - SQLite

Create ASP.NET MVC Project

- Open visual studio 2019
- File → new → project
- Choose → asp.net core web application
- Next → name your project→create
  - .NET Core
  - ASP.NET Core 3.1
- →Create

Project structure



Project structure

- Dependencies
- Properties
- wwwroot Folder
- Program.cs
- Startup Class

wwwroot

- It is web root folder
- It store all static files
  - Js
  - Css
  - Html
  - Image
  - And others files

Program.cs

- វាជាចំនុចចាប់ផ្តើមនៃ ASP.NET Core applicationsដែលមាន main methodៗ
- គោលបំណងសំខាន់នៃ Program class គឺដើម្បី configure the applications infrastructure.

Startup.cs

- វាជាកន្លែងមួយដែលគេប្រើសម្រាប់ configure the request pipeline & middleware។
- យើងក៏អាច configure the services និងបន្ថែមពួកវាto the dependency injection container។
- វាមាន methodsចំនួនពីរគឺConfigure ដែលសំខាន់និងមានក៏បានមិនមានក៏បាននូវ Configure Services method។

ConfigureService method

- The ConfigureServices method ជាកន្លែងមួយដែលគេអាចregister នូវ dependent classes របស់យើងចូលក្នុង built-in IoC container.
- ConfigureServices method វាមាន IServiceCollection parameter ដើម្បី register services ចូលទៅ the IoC container.

Configure method

- The Configure methodជាកន្លែងដែលគេប្រើសម្រាប់ configure application request pipeline សម្រាប់ applicationរបស់យើងដោយប្រើន្ូវIApplicationBuilder instance ដែលវាបានផ្តល់ដោយbuilt-in IoC container។

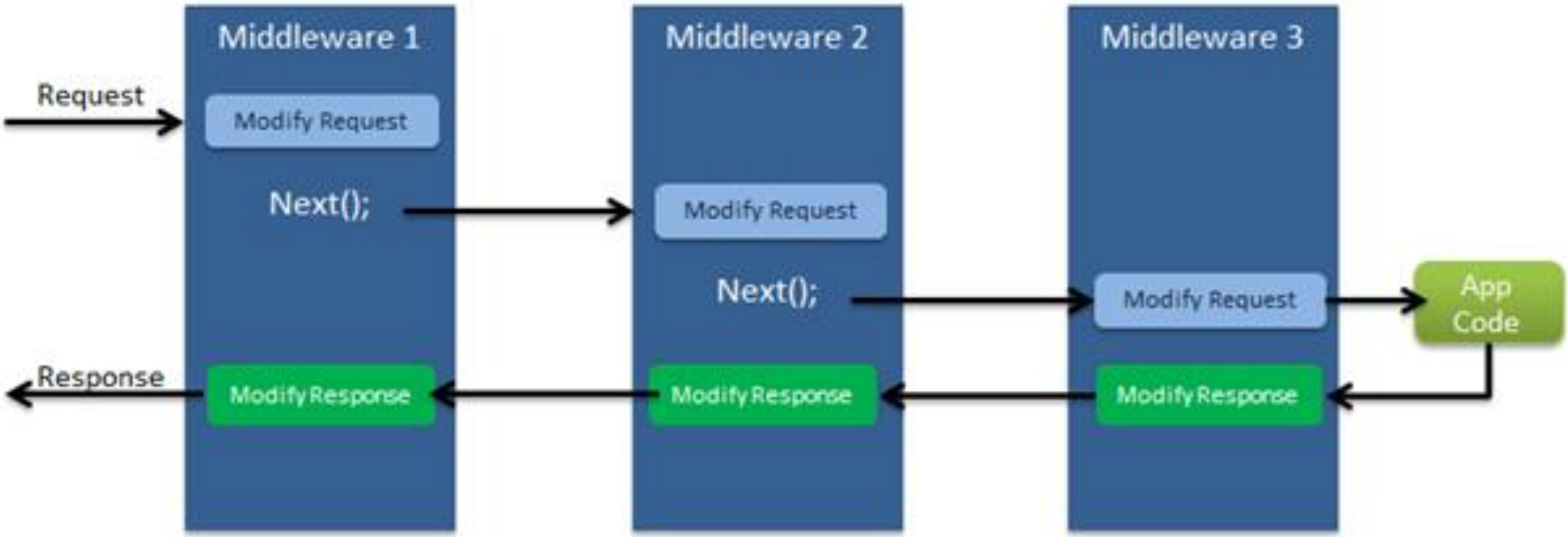
Middleware ?

- Middleware ជាសមាសធាតុ software មួយដែលភ្ជាប់ជាប់ជាមួយនឹង request pipeline ដើម្បីត្រួតពិនិត្យនូវការស្នើរសុំនិងបង្កើតការឆ្លើយតប(Middleware is a software component that hooks into the request pipeline to handle web requests and generate responses)។
- Middlewareនិមួយៗអាតំណើរការនិងរៀបចំនូវសំណើរដែលវាទទួលបានពីmiddlewareមុនៗ(Each middleware Process and manipulates the request as it is received from the previous middleware)ដែលវាសម្រេចថាត្រូវហៅ middleware បន្ទាប់ឬបញ្ជូនចំឡើយទៅmiddlewareមុនវិញ(It may decide to call the next middleware in the pipeline or send the response back to the previous middleware (terminating the pipeline))។

Request pipeline

- Request pipeline គឺជាយន្តការដែលសំណើត្រូវបានដំណើរការដំបូងនិងបញ្ចប់ដោយមានការឆ្លើយតបត្រឡប់មកវិញ (The Request Pipeline is the mechanism by which requests are processed beginning with a Request and ending with a Response) ។

DotNet Core Application Http Pipeline


Middleware 1
Middleware 2
Middleware 3
Request
Modify Request
Next();
Modify Request
Next();
Modify Request
App Code
Response
Modify Response
Modify Response
Modify Response

Middleware

- app.UseStaticFiles();
- app.UseRouting();
- app.UseAuthentication();
- app.UseAuthorization();
- app.UseCookiePolicy();
- app.UseEndpoints();

ការអនុវត្តន៍ជាមួយ Middleware

Request pipeline configuration

- request pipeline ត្រូវបាន configureនៅក្នុង configure method ក្នុង startup class។

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{

    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("<div> Hello World from the middleware 1 </div>"
    });

}
```

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{

    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("<div> Hello World from the middleware 1 </div>");
    });

    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("<div> Hello World from the middleware 2 </div>");
    });

}
```

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{

    app.Use(async (context, next) =>
    {
        await context.Response.WriteAsync("<div> Hello World from the middleware 1 </div>");
        await next.Invoke();
    });

    app.Run(async (context) =>
    {
        await context.Response.WriteAsync("<div> Hello World from the middleware 2 </div>");
    });

}
```

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
  {

      app.Use(async (context, next) =>
      {
          await context.Response.WriteAsync("<div> Hello World from the middleware 1 </div>")
          await next.Invoke();
          await context.Response.WriteAsync("<div> Returning from the middleware 1 </div>");
      });

      app.Use(async (context, next) =>
      {
          await context.Response.WriteAsync("<div> Hello World from the middleware 2 </div>")
          await next.Invoke();
          await context.Response.WriteAsync("<div> Returning from the middleware 2 </div>");
      });

      app.Run(async (context) =>
      {
          await context.Response.WriteAsync("<div> Hello World from the middleware 3 </div>")
      });

  }
```

ការប្រើប្រាស់ static files