ASP.NET Core

ASP.NET Core is not an upgraded version of ASP.NET. ASP.NET Core is completely rewriting that work with .net Core framework. It is

* much faster,
* configurable,
* modular,
* scalable,
* extensible and
* cross-platform support.

It can work with both .NET Core and .net framework via .NET standard framework. It is best suitable for developing cloud-based such as web application, mobile application, IoT application.

Features provided by ASP.NET Core

Following are the core features that are provided by the ASP.NET Core

* Built-in supports for Dependency Injection –
* using Microsoft.Extensions.DependencyInjection;
* Built-in supports for the logging framework and it can be extensible
* Introduced new, fast and cross-platform web server - Kestrel. So, a web application can run without IIS, Apache and Nginx.- HTTP 2.0 version
* Multiple hosting ways are supported
* It supports modularity, so developer need to include the module that required by the application. However, .NET Core framework is also providing the meta package that includes the libraries
* Command-line supports to create, build and run the application
* There is no web.config file. We can store the custom configuration into an appsettings.json file
* There is no Global.asax file. We can now register and use the services into startup class
* It has good support for asynchronous programming
* Support WebSocket and SignalR
* Provide protection against CSRF (Cross-Site Request Forgery)

## Advantages of ASP.NET Core over ASP.NET?

There are following advantages of ASP.NET Core over ASP.NET :

* It is cross-platform, so it can be run on **Windows, Linux and Mac**.
* There is no dependency on framework installation because all the required dependencies are ship with our application
* ASP.NET Core can handle more request than the ASP.NET
* Multiple deployment options available with ASP.NET Core

## Metapackages

The framework .NET Core 2.0 introduced Metapackage that include all the supported package by ASP.NET code with their dependencies into one package. It helps us to do fast development as we don't require to include the individual ASP.NET Core packages. The assembly Microsoft.AspNetCore.All is a meta package provide by ASP.NET core.

## Can ASP.NET Core application work with full .NET 4.x Framework?

Yes. ASP.NET core application works with full .NET framework via the .NET standard library.

## Startup class in ASP.NET core?

Startup class is the entry point of the ASP.NET Core application. Every .NET Core application must have this class. This class contains the application configuration rated items. It is not necessary that class name must "Startup", it can be anything, we can configure startup class in Program class.

2. public class Program
3. {
4. public static void Main(string[] args)
5. {
6. CreateWebHostBuilder(args).Build().Run();
7. }
9. public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
10. WebHost.CreateDefaultBuilder(args)
11. .UseStartup<TestClass>();
12. }

## ConfigureServices method of startup class?

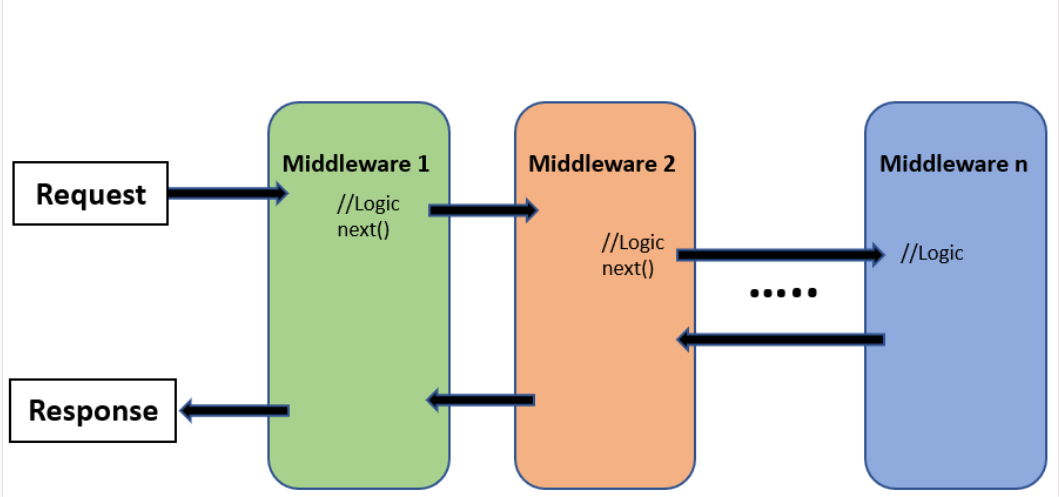
This is an optional method of startup class. It can be used to configure the services that are used by the application. This method calls first when the application is requested for the first time. Using this method, we can add the services to the DI container, so services are available as a dependency in controller constructor.

## Use of Configure method of startup class?

It defines how the application will respond to each HTTP request. We can configure the request pipeline by configuring the middleware. It accepts IApplicationBuilder as a parameter and also it has two optional parameters: IHostingEnvironment and ILoggerFactory. Using this method, we can configure built-in middleware such as routing, authentication, session, etc. as well as third-party middleware.

## Middleware?

It is software which is injected into the application pipeline to handle request and responses. They are just like chained to each other and form as a pipeline. The incoming requests are passes through this pipeline where all middleware is configured, and middleware can perform some action on the request before passes it to the next middleware. Same as for the responses, they are also passing through the middleware but in reverse order.



## Difference between IApplicationBuilder.Use() and IApplicationBuilder.Run()?

We can use both the methods in Configure methods of startup class. Both are used to add middleware delegate to the application request pipeline. The middleware adds using IApplicationBuilder.Use may call the next middleware in the pipeline whereas the middleware adds using IApplicationBuilder.Run method never calls the subsequent ore next middleware. After IApplicationBuilder.Run method, system stop adding middleware in request pipeline.

## Use of "Map" extension while adding middleware to ASP.NET Core pipeline?

It is used for branching the pipeline. It branches the ASP.NET Core pipeline based on request path matching. If request path starts with the given path, middleware on to that branch will execute.

2. public void Configure(IApplicationBuilder app)
3. {
4. app.Map("/path1", Middleware1);
5. app.Map("/path2", Middleware2);
6. }

## Routing in ASP.NET Core?

Routing is functionality that map incoming request to the route handler. The route can have values (extract them from URL) that used to process the request. Using the route, routing can find route handler based on URL. All the routes are registered when the application is started. There are two type of routing supported by ASP.NET Core

* The conventional routing
* Attribute routing

The Routing uses routes for map incoming request with route handler and Generate URL that used in response. Mostly, the application having a single collection of routes and this collection are used for the process the request. The RouteAsync method is used to map incoming request (that match the URL) with available in route collection.

## Enable Session in ASP.NET Core?

The middleware for the session is provided by the package Microsoft.AspNetCore.Session. To use the session in ASP.NET Core application, we need to add this package to csproj file and add the Session middleware to ASP.NET Core request pipeline.

2. public class Startup
3. {
4. public void ConfigureServices(IServiceCollection services)
5. {
6. ….
7. ….
8. services.AddSession();
9. services.AddMvc();
10. }
11. public void Configure(IApplicationBuilder app, IHostingEnvironment env)
12. {
13. ….
14. ….
15. app.UseSession();
16. ….
17. ….
18. }
19. }

## Various JSON files available in ASP.NET Core?

There are following JSON files in ASP.NET Core :

* global.json
* launchsettings.json
* appsettings.json
* bundleconfig.json
* bower.json
* package.json