# Thank you For Joining

</>

Interview Point

Learn Everyday

www.youtube.com/InterviewPoint

# Top 15 ASP.NET Core Interview Questions

Interview Point
Learn Everyday

## 1. What is ASP.NET Core?

- ASP.NET Core is a new version of ASP.NET and it is open source.
- It is not an upgraded version of ASP.NET it’s completely rewritten.
- It is cross platform, supporting Windows, MacOS and Linux, and can be used in device, cloud, and embedded/IoT scenarios.
- It can work with both .NET Core and .net framework via the .NET standard framework.
- It is best suitable for developing cloud-based such as web application, mobile application, IoT application.
- Command-line supports to create, build and run the application.

## 2. What are the features/characteristics provided by ASP.NET Core?

- Single programming module for MVC and Web API.
- Support Web Socket and  SignalR.
- Command Line Support to Create, build and run application.
- Built-in support for logging framework and it can be extensible.
- It has good support for asynchronous programming
- 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.
- Multiple hosting ways are supported.
- Provide protection against CSRF (Cross-Site Request Forgery)
- Built-in Support for Dependencies Injection.

## 2. What are the features/characteristics provided by ASP.NET Core?

- Single programming module for MVC and Web API.
- Support Web Socket and  SignalR.
- Command Line Support to Create, build and run application.
- Built-in support for logging framework and it can be extensible.
- It has good support for asynchronous programming
- 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.
- Multiple hosting ways are supported.
- Provide protection against CSRF (Cross-Site Request Forgery)
- Built-in Support for Dependencies Injection.

## 3. What is Dependency Injection?

- Dependency Injection (DI) is a software design pattern.
- It is a technique for achieving Inversion of control(IoC) between classes and their dependencies.
- It allows us to develop loosely-coupled code.
- It makes code maintainable.
- It also helps to reduce the tight coupling between software components.
- DI enables us to better manage future changes and other complexity in our software.

## 4. What is the role of Startup class?

- Startup class is the entry point of the ASP.NET Core application.
- It is not necessary that class name must "Startup", it can be anything, we can configure startup class in Program class.
- **It is responsible for configuration related things as below.**
- It configure the services which are required by app.
- It defines the app's request handling pipeline as a series of middleware components.
- Startup class is specified inside 'CreateHostBuilder' method when host is created.
- Multiple Startup classes can also be defined for different environments, At run time appropriate startup class is used.

## 5. What is the role of ConfigureServices and Configure method?

- **ConfigureServices** method is optional and defined inside startup class.
- It takes care of registering services that are consumed across the application using Dependency Injection (DI) or Application Services.
- It's get called by host before 'Configure' method to configure the app's services.
- **Configure** method is used to add middleware components to IApplicationBuilder instance that's available in Configure method.
- 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.
- Configure method also specify how the app respond to HTTP request and response.

## 6. What is wwwroot folder in ASP.NET Core?

- By default the wwwroot is the root folder that contains the static files such as HTML, CSS and Javascript.
- The files can be stored here and accessed with a relative path to the root.
- Only these files inside the wwwroot can be served over HTTP Requests. All other files are filtered out and cannot be served by default.

## 7. What is middleware?

- A middleware is nothing but a component (class) which is executed on every request in ASP.NET Core application.
- It is software which is injected into the application pipeline to handle request and responses.
- **Middleware component** is program that's build into an app's pipeline to handle the request and response.
- Each middleware component can decide whether to pass the request to next component and to perform any operation before or after next component in pipeline.
- It is used to implement several tasks when handling requests.

## 8. What’s the difference between .NET Core Framework and .NET Standard?

- **.Net Standard** is a specification for implementing the Base Class Library (BCL). BCL contains classes such as exception handling, XML, collections, I/O and networking. WPF, WCF and ASP.NET do not form part of BCL and so are not included in .NET Standard library.
- .NET Core is a managed framework that builds desktop and web applications in cross-platform.
- Both “.NET Core” and “.NET Framework” include .NET Standard for BCL in their managed framework.
- .NET Framework is a framework that builds desktop and web applications in Windows only. It is highly dependent on the architecture.

## 9. What is Kestrel?

- Kestrel is a cross-platform web server for ASP.NET Core based on libuv, a cross-platform asynchronous I/O library.
- Kestrel is the web server that is included by default in ASP.NET Core new project templates.
- It is fast and secure and can even be used without a reverse proxy server. However, it is still recommended to use with IIS, Nginx or Apache.
- A reverse proxy server receives HTTP requests from the Internet and forwards them to Kestrel after some preliminary handling.
- Kestrel is relatively new and does not yet have a full complement of defenses against attacks.

## 10. What is the 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 or next middleware.
- After IApplicationBuilder.Run method, system stop adding middleware in request pipeline.

## 11. What is the difference between services.AddTransient & service.AddScoped & service.AddSingleton methods are Asp.Net Core?

- **Transient** objects are created for every request (when requested).
  - The service can be added as Transient using AddTransient method of IServiceCollection.
  - This lifetime can be used in stateless service and it is a way to add lightweight service.
- **Scoped** objects are the same within a request, but different across different requests.
  - ASP.NET Core will create and share an instance of the service per request to the application.
  - It will create a new instance in the new request.
  - The service can be added as scoped using an AddScoped method of IServiceCollection.
- **Singleton** objects created the first time they're requested (or when ConfigureServices is run and an instance is specified with the service registration).
  - The service can be added as a singleton using AddSingleton method of IServiceCollection.
  - ASP.NET Core creates service instance at the time of registration and subsequence request use this service instance.

## 12. What is WebListener?

- WebListener is a web server in ASP.NET Core that runs only on Windows host machines.
- It is an alternative to Kestrel and is built on HttpSys kernel-mode driver.
- Also, is used for direct connection to the internet without the need of an IIS as a reverse proxy server.
- It is not compatible with IIS.

## 13. What are the various JSON files available in ASP.NET Core?

**Following JSON files are available in the ASP.NET Core**

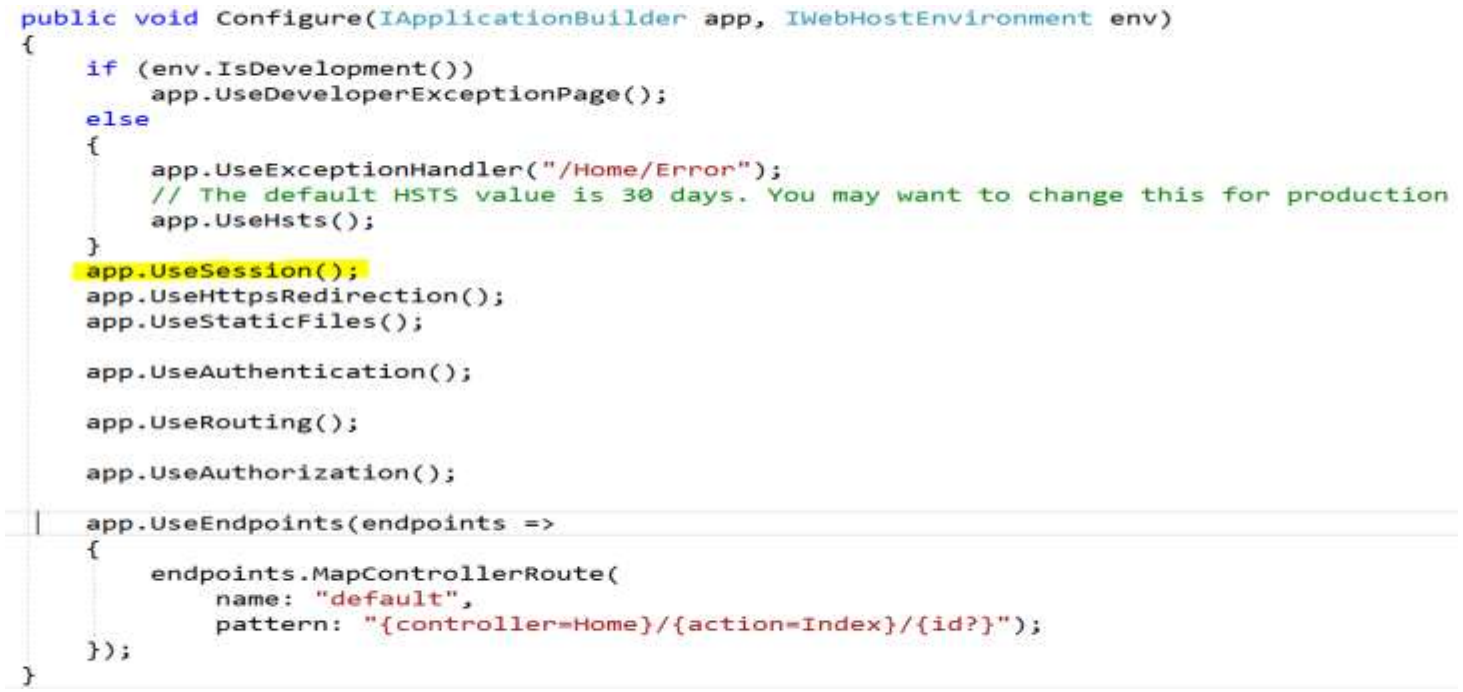
- appsettings.json
- bundleconfig.json
- launchsettings.json
- bower.json
- package.json
- global.json

## 14. Explain routing in ASP.NET Core

- Routing is functionality that map incoming request to the route handler.
- The Routing uses routes for map incoming request with route handler and Generate URL that used in response.
- 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 types of routing supported by ASP.NET Core.
  - The conventional routing
  - Attribute routing

## 15. How to 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.



```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
        app.UseDeveloperExceptionPage();
    else
    {
        app.UseExceptionHandler("/Home/Error");
        // The default HSTS value is 30 days. You may want to change this for production
        app.UseHsts();
    }
    app.UseSession();
    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseAuthentication();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```