**1.What is ASP.NET MVC Core?**

ASP.NET MVC Core is an open source, cross platform framework to develop web applications. When you develop web applications using ASP.NET Core, it can run on Windows, Linux and Mac.

**2.Where Do We Store Configuration in ASP.NET Core?**

Configuration data is stored in a json file called as *Appsettings.json*.

**3.How Can We Read Configuration File in ASP.NET Core?**

To read configuration from *appsettings.json*, you need to use “IConfiguration” interface.

**4.What Is Dependency Injection?**

Dependency injection is the practice of providing dependent objects for a class from outside rather than the class creating it. So, in simple words, the object creation process is isolated from the caller. This helps to create a proper decoupled system.

**5.Why Do We Need Dependency Injection?**

By using DI classes are decoupled from each other so you make changes at one place it reflected all over the places.

**6.How to Implement Dependency Injection?**

To implement DI, we need to add the service in “ConfigureServices” method which is in *Startup.cs* file.

**7.Explain the Concept of Middleware?**

Middleware helps to add pre-processing logic before the request is sent to the controller.

**8.How to Implement Middleware in MVC Core?**

Middlewares are added in “Configure” method of *Startup.cs* file.

**9.Explain the Importance of Startup.cs File in ASP.NET Core?**

*Startup.cs* file helps to configure Dependency injection and Middleware.

**10.Explain the Various Ways of Doing DI in MVC?**

*This is an easy topic to understand but very difficult to explain to the interviewer in simple words. So, I would suggest to go through the below explanation, understand it and create your simple one liner.*

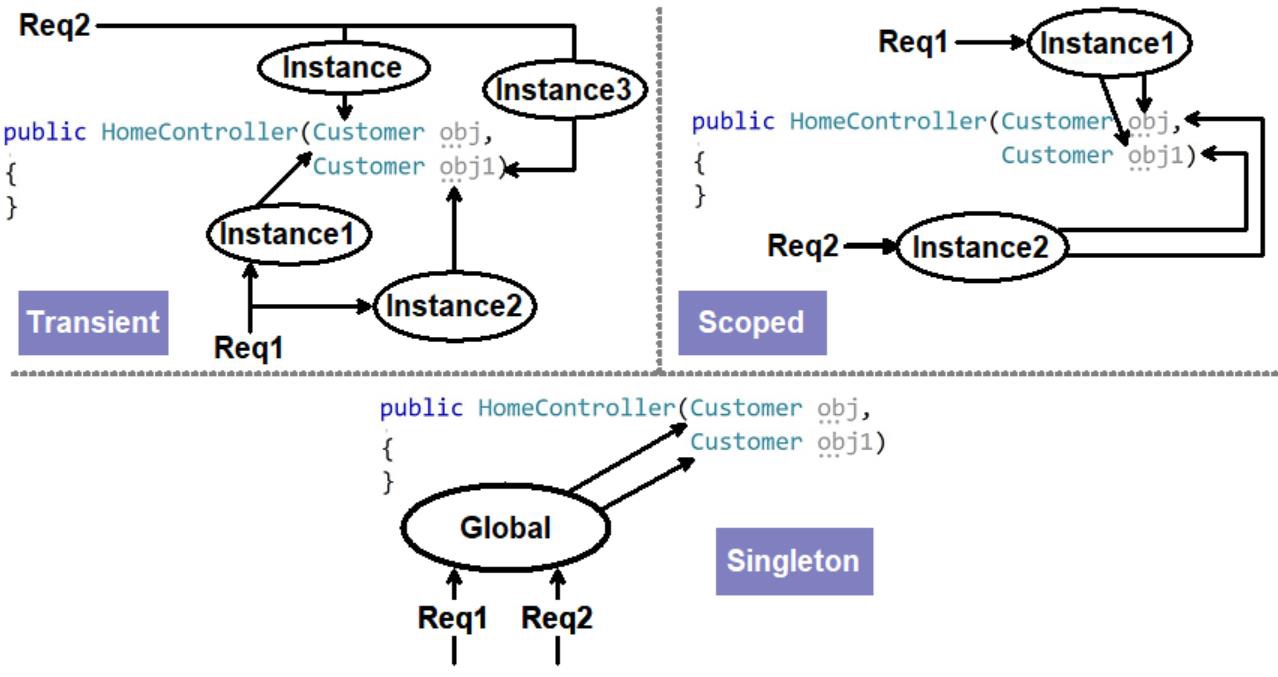
There are three ways of doing DI: Scoped, Transient and Singleton.

* **Transient**: Creates new instances, every single time service is injected
* **Scoped**: Creates new instances, once per request made to server
* **Singleton**: Instantiates one global object for all requests coming to server from any user

The below image shows the same visually. In transient for every dependency injection, new instance will be injected. For example, in the below image for “obj” and “obj1”, fresh new instances will be injected for every request.

In case of scoped for every request, the same instance will be injected for every dependency injection. So “obj” and “obj1” will have the same instance injected.

In case of singleton, one big global instance is created. For all request, for all Dependency injection, same instance will be injected.

****

**11.What are the Various Ways of Doing Session Management in ASP.NET Core?**

* Session variables
* Tempdata
* ViewData/ViewBag

**12.ConfigureServices vs Configure Method?**

In ConfigureServices, we add the objects to be dependency injected while in Configure method, we specify middlewares.

**13.Explain Razor Pages?**

Razor is a view engine in which you can write both C# and HTML code. C# code is rendered at the server side.

**14.How Can We Pass Model Data to the View?**

You can pass by using viewdata or using strongly typed views.

**15.What Is a Strongly Typed View?**

Strongly typed views are Razor views where we get intellisense for the model used in the view.

**16.Explain the Concept of ViewModel?**

View Model represents the data to be displayed on the view.

**17.Explain Kestrel Web Server?**

Kestrel is an open source default web server used by ASP.NET core application. Kestrel ships with ASP.NET Core setup and works as in-process web server to handle web request. Kestrel is cross platform it works on Linux, Windows and Mac.

Every request of ASP.NET Core goes through kestrel first.

**18.Why Kestrel Web Server When We Have IIS?**

ASP.NET core is meant to run cross platform. So, IIS as default server will not work , because IIS works only on Windows. Web server like NGINX, Apache and so on have their own way to running the application startup and ASP.NET core cannot satisfy each one of them. So kestrel acts like a in-process web server, takes a request, sends to MVC core application, gets response and sends it back to the main web server.

**19.Explain the Concept of Reverse Proxy?**

Reverse proxy is a concept where the server acts like a mediator. Kestrel works on reverse proxy concept it takes the request from the main web server like IIS / Apache and forwards its to MVC application and vice-versa.

**20.What Are Cookies?**

Cookies are small text file where browser can store user related information.

**21.What Is the Need for Session Management?**

HTTP protocol is stateless. To maintain states, we need to use session management.

**22.What are the Various Ways of Doing Session Management in ASP.NET?**

There three ways of doing session management:

* ViewData/ViewBag
* Session variables
* Tempdata

**23.What Exactly Is a Session?**

Session is user interaction which happens with a website over a period of time. Its interaction which happens right when browser opens and browser closes.

**24.Explain "HTTP is a Stateless Protocol"?**

HTTP does not remember states between request and response. User sends a request, server processes it and sends response. Now if the user comes again in the same session, the server treats it as a new request.

**25.What are the Various Ways of Doing Session Management?**

There are three primary ways of doing session management, Session variables , viewdata/viewbag and Tempdata.

**26.Are Sessions Enabled by Default?**

To keep ASP.NET lightweight, session variables are not enabled by default.

**27.How to Enable Sessions in MVC Core?**

To enable session variables in *Startup.cs*, we need to "AddSession" and "UseSession".

**28.Are Sessions Variables Shared(global) Between Users?**

Session variables are created for a particular user. Data of session variable is not shared between users.

**29.Do Session Variables Use Cookies?**

Yes, [session variables use cookies](https://www.youtube.com/embed/AVeq77e4BSw).

**30.What Is a Cookie?**

Cookies are small text files information which is stored in the end users' computer.

**31.Explain Idle Time Out in Sessions?**

Idle timeout is the time when user does not do any activity. So if user is inactive, you would like to ensure session variables expire.

**32.What Does a Context Mean in HTTP?**

HttpContext holds information like session, request data and response information. It has all the data needed for a particular request and response.

**33.When Should We Use ViewData ?**

To pass data from controller to view.

**34.How to Pass Data from Controller to View?**

By using Viewdata.

**35.In Same Request, Can Viewdata Persist Across Actions?**

No, it will not persist. Viewdata scope only from action to view.

**36.ViewBag vs ViewData?**

ViewBag is syntactic sugar over viewdata.

**37.How does ViewBag Work Internally?**

ViewBag uses dynamic variables.

**38.Explain viewModels?**

They represent data for a view.

**39.ViewBag vs ViewModel - What's the Best Practice?**

For passing data from action to view, use viewmodel as ViewBag is dynamic and we can have mistakes.

**40.Explain tempdata?**

Tempdata persists for the current and we can control whether we want the state in the next request or not.

**41.Can Tempdata Persist Across Action Redirects?**

Yes.

**42.How Is Tempdata Different from Viewdata?**

Tempdata can persist from action to action (i.e., action redirects) while viewdata is only from action to the view.

**43.If Tempdata Is Read, Is It Available for Next Request?**

No.

**44.How to Persist tempdata?**

We can persist tempdata by calling the keep method.

**45.What does Keep do in tempdata?**

Keep method persists the data for the next request.

**46.Explain Peek in tempdata?**

Peek = Keep + Read. It will read as well as keep the data.

**47.How is tempdata Different from Session Variables?**

Session variables persist right from browser open to browser close. While tempdata persists across action redirect and also, we can decide using keep, can we persist state for new request.

**48.If I Restart the Server Does Tempdata, Session Stay?**

No, tempdata and session do not persist in server restarts.

**49.Is Tempdata Private to a User?**

Yes, Tempdata is private to a user.

**50.ViewData vs ViewBag vs Tempdata vs Session Variables**

* Session variables scope is from browser open to close.
* Viewdata scope is only from action to view.
* Viewdata does not persist in action or controller redirects.
* Viewbag is a syntactic sugar over ViewData.
* Tempdata maintains data in single request irrespective we have redirects or not.
* Once tempdata is read, state is not maintained across request.
* If we call Keep method, tempdata state is maintained across requests even if tempdata is read.

**51.What is WebAPI ?**

Web API controller delivers data and services easily by using HTTP REST services.

**52.What is the Advantage of WebAPI ?**

HTTP is the most used protocol so by exposing your service over HTTP, any nature of clients can consume it, making it a true client server model. So, your client can now be a mobile app or app made in JS frameworks like react/Angular or must be a simple browser, everyone can integrate with Web API seamlessly.

**53.Explain REST and Architectural Constraints of REST?**

REST stands for representational state transfer. REST is an architectural style/principle where client and server talks in representations and states.

Some important principle of REST is Client Server, Statelessness, Unique URI, Manipulation happens through representation. WebAPI helps to expose services / data using HTTP protocol.

**54.Can We Use UDP/TCPIP Protocol with Web API?**

No.

**55.How WebAPI is Different from MVC Controller?**

In MVC controller, we can return RAZOR views but in WebAPI, we cannot return Views. WebAPI follows REST principles and does things like content negotiations.

**56.What are Content Negotiations in Web API?**

Content negotiations look at client accept type and depending on the client accept type, it provides data format. So, if in the accept, client sends XML it will send XML, if it sends JSON it will send JSON.

**57.WebAPI vs WCF?**

WCF was meant for SOA, XML is compulsory format and support any protocol.  
WebAPI was meant for REST, no compulsion on Format and supports only HTTP protocol.

**58.WCF REST vs WebAPI REST?**

WCF REST was created to ensure that old legacy WCF can be REST enabled. While WebAPI was built from scratch to serve REST architecture. For new project, use WebAPI. If you want to make legacy WCF compatible with REST, you will use WCF REST.

**59.How to Return HTTP Status Codes?**

JavaScript

return StatusCode(StatusCodes.Status200OK,"Test");

For Error, Which Status Code Is Returned?

500 internal server error.

**60.How Did You Secure Your Web API?**

By using JWT token.

**61.How Do Current JS Frameworks Work with Web API?**

Client side is built using JS frameworks like Angular, React and they communicate to MVC using HTTP.