.NET Core Interview Questions and Answers 2022

hackertrail.com/talent/backend/net-core-interview-questions-answers

HackerTrail September 8, 2022

According to Stackoverflow Survey 2020, developers are using .NET and .NET Core in second and third place respectively, as framework or libraries. Mostly it is the first programming language the programmers learn after the OOPs concepts. There are multiple reasons for the success of .NET, one of the most prominent ones is its stack-layered architecture. This architecture provides flexibility to add/ update layers without disturbing the whole application.

.NET can be used to develop different types of high-performance applications such as gaming applications. It provides better monitoring, scalability, performance, and consistency for a whole range of applications such as console apps, GUI apps, web apps, web API apps, Windows services etc.

.NET MVC(Model-View-Controller), provides a large set of additional functionalities to create modular applications. It eases the whole application creation process by providing component-based development and testing.

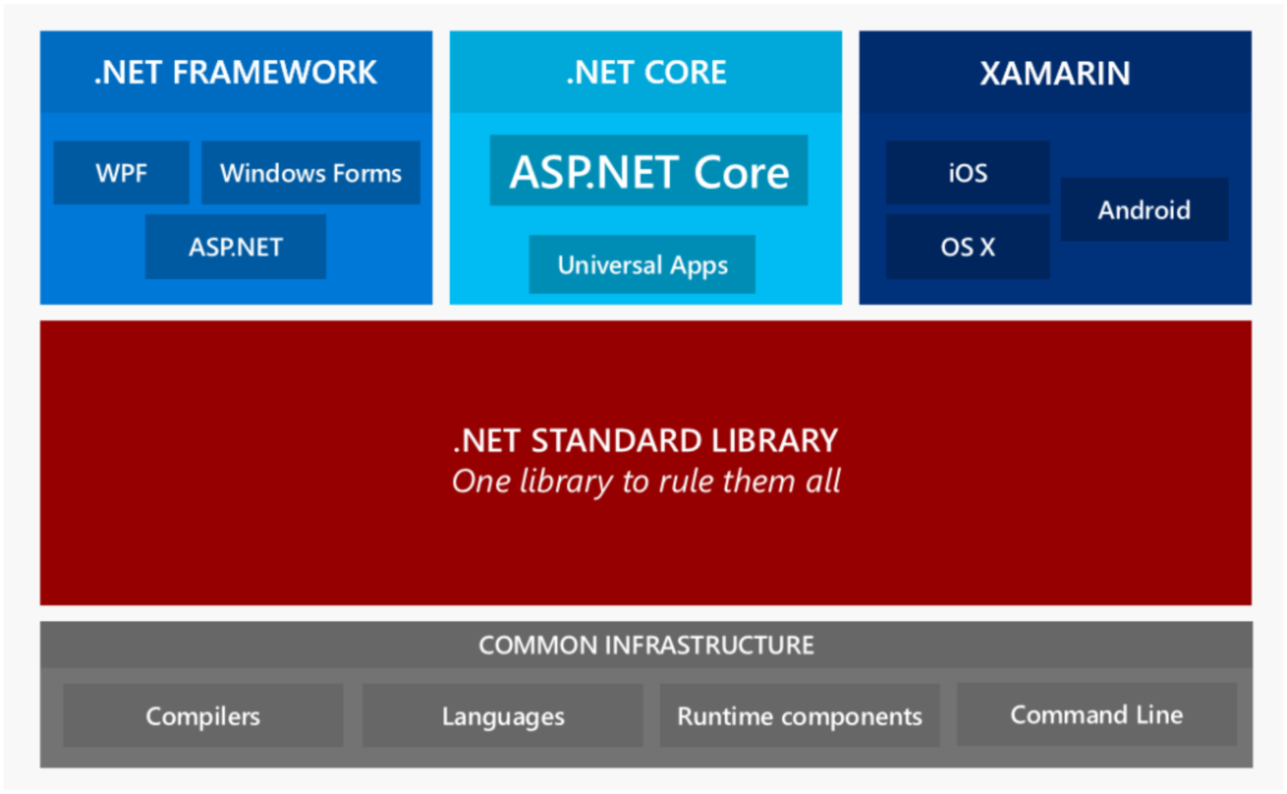


Image credit: https://codeburst.io/

In this article we will dig deeper into .NET core interview questions into the following sections:

Basic .NET Core Interview Questions

1. What is .NET Core Framework, and how does it work?

.NET Core framework provides an open-source, accessible, and general-purpose platform to create and run applications onto different operating systems. The framework follows the object-oriented programming principles that we can use C#, .NET, VB, Perl, Cobol, etc., programming languages. The framework provides various built-in tools such as packages, classes, libraries, APIs, and other functionalities. We can create a diverse range of applications.

It works as follows:

- Once you have finished developing codes for required applications, you need to compile those application codes to Common Intermediate Language.
- The framework uses an assembly file to store the compiled code with an extension (.dll or .exe)
- Now, the Common Language Runtime (CLR) of the framework convert the compiled code to machine code (executable code) using the Just In Time (JIT) compiler.
- At last, we can execute this executable code on any specific architecture used by developers.

2. What is the latest version of .NET Core? Share one specific attribute.

The latest version of .NET Core is .NET Core 6.0, and its release date is July 12 2022, according to Microsoft Documentation. The newest release includes the .NET Runtime and ASP.NET Core Runtime. It has introduced Android, iOS, and macOS SDKs for developing native applications. You can check this documentation to know the setup instructions and develop .NET MAUI applications.

3. Share specific features of .NET Core?

.NET Core has these 4 specific features:

- **Cross-platform:** It supports various platforms and is executable on windows, macOS, and Linux. You can easily port the codes from one platform to another platform.
- **Flexibility:** You can easily include codes in the desired app or install them per requirements. It means you can use one single consistent API model for all .NET applications with the help of the same library on various platforms.
- **Open Source:** You can use it by downloading it from the Github library. You don't need to pay to purchase a license. The framework has been licensed under MIT and Apache.
- **Command-line tools:** You can efficiently execute applications at the command line.

4. What is .NET Core used for?

You can use .NET Core in many ways:

- For developing and building web applications and services that run on diverse operating systems
- For creating Internet of Things applications and mobile backends
- For using any development tools on any operating system
- For creating and deploying applications to the cloud or other on-premises services.
- Flexibility, high performance, and lightweight features allow for the development of applications quickly in containers deployable on all operating systems.

5. Discuss critical components in .NET Core?

Since .NET Core is a modular platform thus, its components could be stacked into these three layers:

- A .Net runtime: It consists of different runtime libraries that allow you to perform functions such as type safety, load assemblies, garbage collections etc.
- A collection of Framework libraries: It also consists of libraries that offer utilities, primitive data types, etc.
- A collection of SDK tools and compilers: It permits you to work with .NET Core SDK quickly.

This stack could be divided into these four components:

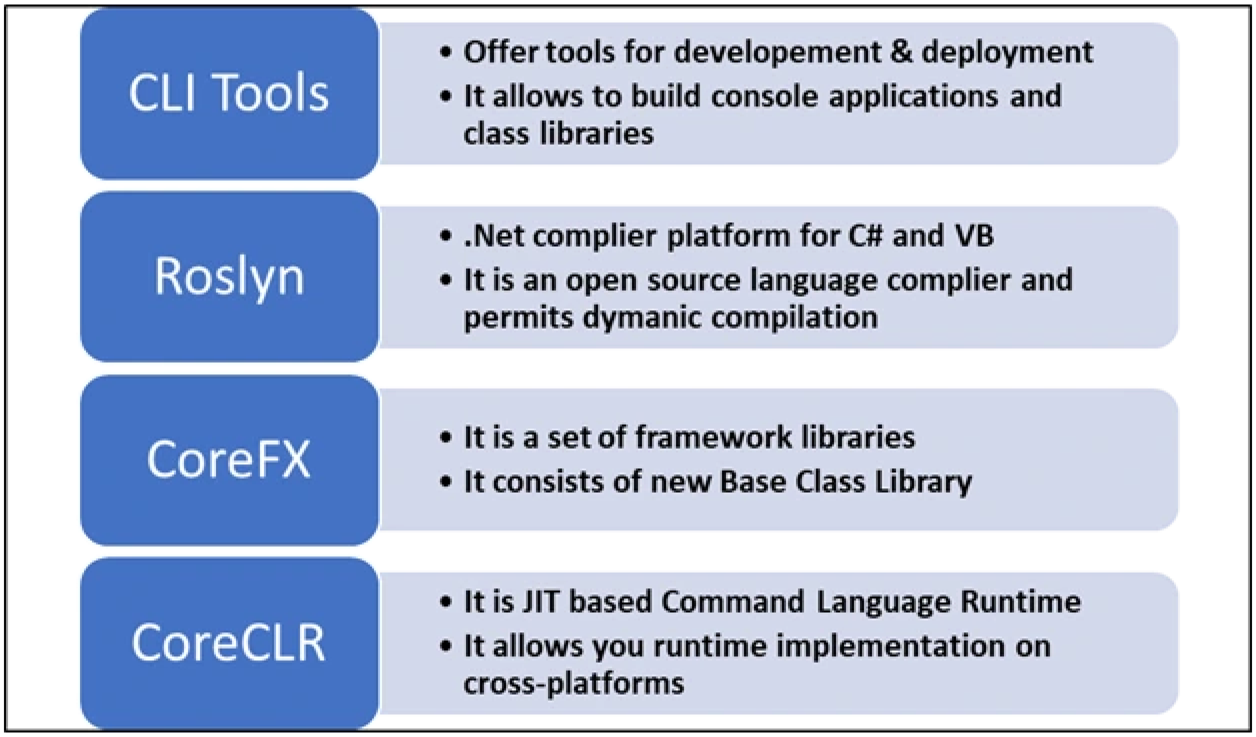

CLI Tools
• Offer tools for developement & deployment
• It allows to build console applications and class libraries
Roslyn
• .Net complier platform for C# and VB
• It is an open source language complier and permits dymanic compilation
CoreFX
• It is a set of framework libraries
• It consists of new Base Class Library
CoreCLR
• It is JIT based Command Language Runtime
• It allows you runtime implementation on cross-platforms

Image Credit: .NET Core Components

6. What is the difference between .Net Core and Mono?

Point	.Net Core	Mono
What is exactly?	A part of.NET framework which is specially optimised for designing modern apps and supporting developer workflows	It is also part of .NET family frameworks, but this framework is optimised for iOS, macOS, Android, and Windows devices by the Xamarin platform
Best application Area	• To design command line applications • Web application development • Designing background service apps • Desktop application	• Mobile app development • Designing games • Code compilation within the browser • Designing multi-platform desktop applications
Specific features	• Natural acquisition • Modular framework • Smaller deployment footprint • Fast release cycles	• Native User Interface • Native API Access • Native Performance • Productivity
App Models	UWP (Universal Windows Platform), ASP.NET Core	Xamarin iOS, Xamarin Android, Xamarin Forms, Xamarin Mac

Base library	CoreFX Class Library	Mono Class Library

7. What is .NET Core CoreFX?

CoreFX is the introductive class library for .NET Core. It consists of collection types, file systems, console, JSON, and XML for class library implementation. You can use this code as a single portable assembly. Since it provides platform-neutral code, thus you can share it across different platforms.

8. What is CoreCLR?

CoreCLR is the .NET execution engine in .NET Core. It consists of a garbage collector, JIT compiler, low-level classes, and primitive data types. Garbage collection and machine code compilation are its primary functions.

The following image shows .NET Core Compilation. You can clearly write codes in different languages that compliers like Roslyn would comply with. The compiler will generate the respective CIL code used by the JIT compiler for further compilation. Since CoreCLR is embedded in the JIT compiler, it would eventually generate machine code. Check its source code available on GitHub

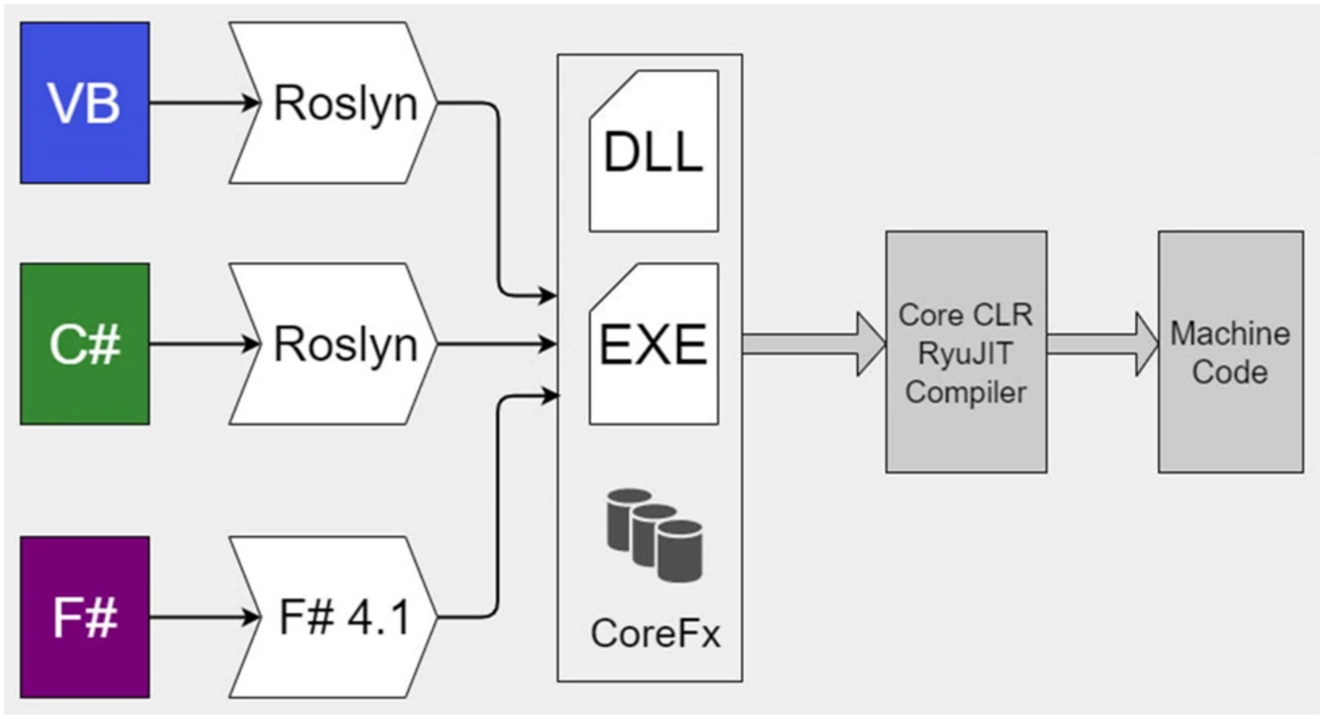


Image Credit: dotnet-talk

9. How is .NET Core SDK different from .NET Core Runtime?

.NET Core SDK builds applications, whereas .NET Core Runtime runs the application. Consider SDK is a collection of all tools and libraries you need to develop .NET Core applications quickly like a compiler, CLI. Consider Runtime as a virtual machine consisting of runtimes libraries and helps you run those applications.

10. Where should you not use .NET Core?

Consider these application areas where you should prevent using .NET Core

- Avoid using current .NET framework applications in productions or migration because there is a possibility when you are unable to execute third libraries from apps running on the .NET core. Although, these libraries are executable from the .NET framework.
- Avoid using .NET Core in designing loosely coupled and new large monolithic applications. It is because of computability issues while consuming libraries with the .NET framework. You can create such applications by running on the top of the .NET framework and with the help of CLR libraries.
- Any applications that need sub frameworks like WPF, WebForms, Winforms as .NET Core don't support these.
- Prevent trying .NET Core in applications requiring higher level frameworks such as WCF, Entity Framework, and Windows Workflow Foundation.

11. What are the advantages of .NET Core?

- **Cross-platform development and deployment**: It can support application development on different platforms such as Windows, Linux, Mac, etc. Also, the deployment is supported on multiple platforms through containerization(Docker, Kubernetes, Service Fabric). This makes .NET completely portable and runnable on different platforms.
- **Open-source**: All .NET source code and documentation is freely available for download and contribution. This results in faster software releases, enormous support, and usage of the latest tools in development.
- **Supports a plethora of applications**: It has the capabilities to support a wide range of application types such as desktop, web, AI, cloud, mobile, IoT, gaming, etc.
- **Secure**: Provides easy-to-incorporate security measures like authentication, authorization, and data protection. It has mechanisms to protect the sensitive-data like keys, passwords, connection strings, etc. For e.g. in terms of authentication, ASP.NET Core Identity allows you to integrate your app with all major external providers.
- **High performance**: With every new release of the .NET core, the performance is improved for the benefit of users. For example, in .NET 5, the garbage collection is improved for faster speed, scalability, and reduction in memory resets' cost. Detailed account of performance improvement in .NET 5.
- **Flexible**: Provides the flexibility to use any database and infrastructure as per choice. It provides the ability to change, evolve and grow easily according to external factors.

12. What is Kestrel?

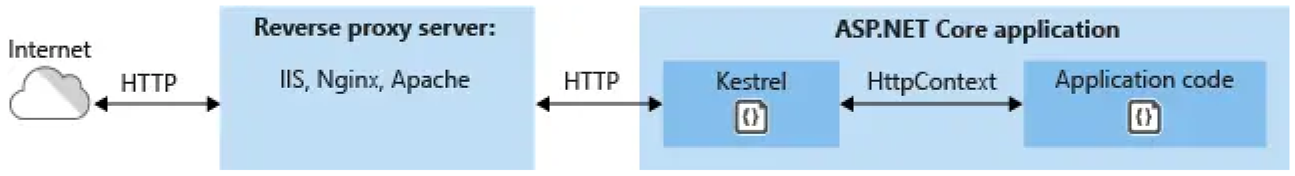


Image credit: https://docs.microsoft.com/

Kestrel is an event-driven, I/O-based, open-source, cross-platform, and asynchronous server which hosts .NET applications. It is provided as a default server for .NET Core therefore, it is compatible with all the platforms and their versions which .NET Core supports.

Usually, it is used as an edge-server, which means it is the server which faces the internet and handles HTTP web requests from clients directly. It is a listening server with a command-line interface.

Advantages of Kestrel are:

- Lightweight and fast.

- Cross-platform and supports all versions of .NET Core.
- Supports HTTPS.
- Easy configuration

13. What do you know about .NET Core middleware?

Middleware is a layer, software, or simple class through which all the requests and responses have to go through. The middleware is assembled of many delegates in an application pipeline. Each component(delegate) in the pipeline of the middleware decides :

- To pass the request to the next component.
- Perform some processing on the request before or after passing it.

The below diagram shows a middleware request pipeline consisting of many delegates called one after another. Where black arrows mark the direction of execution. Each delegate in the diagram performs some operations before or after the next delegate.

More details from Microsoft's documentation.

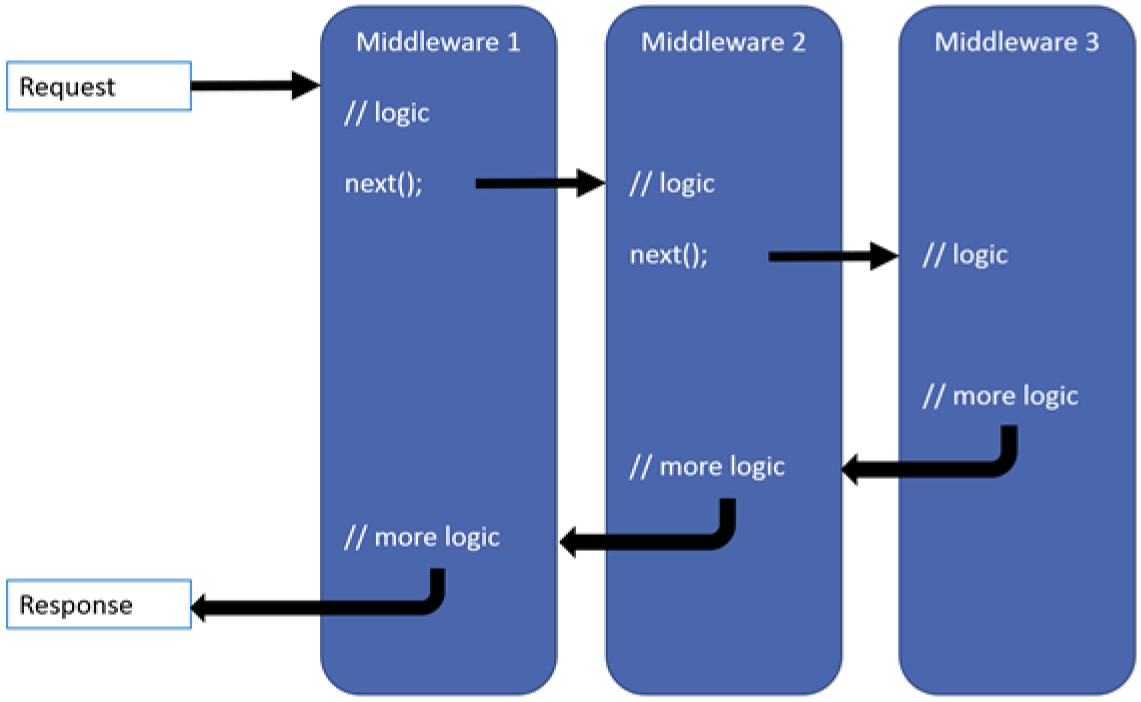


Image credit: https://docs.microsoft.com/

14. What are Razor Pages in .NET Core?

Razor Pages is a new server-side framework which works on a page-based approach to render applications in .NET Core. They are stored as a physical .cshtmlfile.

They have the HTML and code in a single file, without the need to maintain separate controllers, view models, action methods, etc. We can also have the code separate from the HTML in a different file which is attached to the Razor Page. Both types are shown below in the diagram:

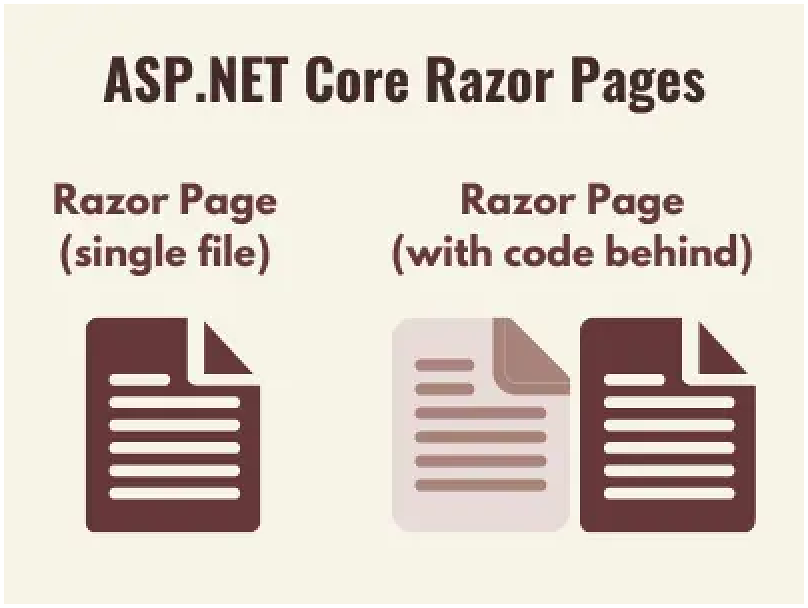


Image credit: https://www.ezzylearning.net/

Razor Pages framework is flexible, lightweight, cohesive, page-based, easy to learn and maintain compared to MVC. It can be used in conjunction with traditional MVC (Model-View-Controller) architecture or Web-API controllers.

15. What are service lifetimes in .NET Core?

.NET Core supports a design pattern called ‘Dependency Injection’ which helps in the implementation of IoC(Inversion of Control). During registration, dependencies require their lifetime to be defined. The lifetime of service decides under what condition the instance of the service will be created and till what time it will be live.

There are three types of service lifetimes supported by .NET Core:

- **Transient Service:** Instance is created each time it is requested.
- **Scoped Service:** User-specific instance is created once per user and shared across all the requests.
- **Singleton Service:** Single Instance is created once a lifetime of the application.

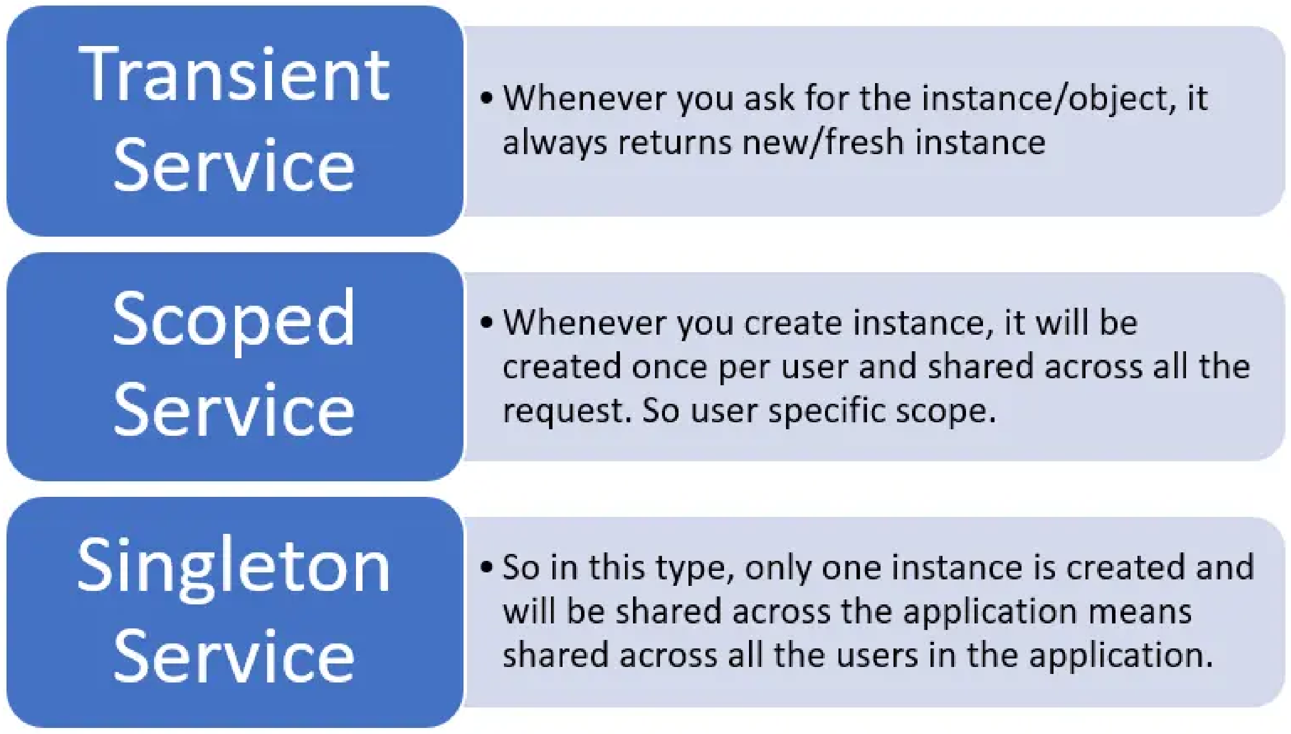

Transient Service
• Whenever you ask for the instance/object, it always returns new/fresh instance
Scoped Service
• Whenever you create instance, it will be created once per user and shared across all the request. So user specific scope.
Singleton Service
• So in this type, only one instance is created and will be shared across the application means shared across all the users in the application.

Image credit: https://www.c-sharpcorner.com/

16. What are the differences between .NET Core and .NET Framework?

.NET Core	.NET Framework
Completely open-source.	Few components are open-source.
Compatible with Linux, Windows, and Mac operating systems.	Compatible with only Windows.
Does not support desktop application development.	Supports web and desktop application development.
Supports microservices development.	Does not support microservices development.
Lightweight for Command Line Interface(CLI).	Heavy for Command Line Interface.

Advanced .NET Core Interview Questions

17. Explain Docker in .NET Core.

Docker is an open platform for developing, shipping, and running applications. It allows you to quickly isolate your applications from the infrastructure to transmit software. You should leverage this feature for managing infrastructure and deploying codes fast. It would help reduce the time needed between writing and running codes in infrastructure.

Three main functions:

- Quick and constant delivery of applications
- Responsive deployment and scaling
- Efficiently run more workloads on the same hardware

Take care following points while using Docker in .NET Core

- You can use the Docker client's CLI for managing images and containers
- You must adequately integrate Docker images, containers, and registries while designing and containerising applications or microservices
- Use Dockerfile for rebuilding images and distribute them with others

18. What is .NET Core CLI?

.NET Core CLI is part of .NET SDK that provides a cross-platform toolset to develop, create, and run .NET Core applications. You can install multiple versions of the toolset on your machine. You can use the following standard syntax to use CLI:

dotnet [verb] [arguments]

It provides four types of commands

- **Basic commands**: All commands required to develop applications like new, restore, build, run, etc.
- **Project Modification commands**: It allows you to use existing packages or add packages for developing applications.
- **Advanced commands**: It gives various commands to perform additional functions such as deleting nuget.
- **Tool management commands**: You can use these commands to manage tools.

19. What is Hosting Environment Management?

It is a new feature of .NET Core that permits you to work with multiple environments with no friction. You can use this feature through the available interface, Hosting Environment. The interface has been open since the first run of the application. Its execution depends on the environment variable and switch between the configuration files during runtime.

The interface reads a specific environment variable named "ASPNETCORE_ENVIRONMENT" and checks its value. Check its following values:

- **If value**: Development – You are running the application in Dev mode
- **If value**: Staging – You are running the application in staging mode

This feature permits you to manage the different environments as well.

20. Briefly explain Garbage Collection, its benefits, and its condition.

Garbage collection is another powerful feature of .NET Core. The primary function of this feature is to manage memory allocation and release. The .NET Core has "Zero Garbage Collector" to execute this function. You can call it Automatic Memory Manager.

Benefits:

- You don't need to put effort into releasing memory manually
- Efficient object allocation on the heap
- Ensure memory security by ensuring object's usage
- You can reclaim objects that are no longer needed, free the memory, and use it for other purposes.

Three conditions that allow garbage collection

- The system has low physical memory
- In case of an acceptable threshold
- When the GC method has been called

21. Discuss CTS types in .NET Core.

Common Type System or CTS standard defines and explains how to use data types in the .NET framework. The "System.Object" is the base type that derives other types in the singly rooted object hierarchy. It is a collection of data types, and Runtime uses it to implement cross-language integration.

You can categorise this into two types:

> **Value types:** This data type uses an object's actual value to represent any object. If you assign instance of value type to a variable, that variable is given a fresh copy of the value.

Examples: Built-in value types, User-defined value types, Enumeration, Structure

> **Reference types:** This data type uses a reference to the object's value to represent the objects. You can say it follows the concept of pointers. It doesn't create any copy if you assign a reference type to a variable that further points to original values.

Examples: Self-defining types like array, Pointer type, Interface Type

22. Explain CoreRT.

In .NET Core, CoreRT has been used as a native toolchain that performs compilation to translation. In other words, it compiles CIL byte code to machine code. The CoreRT uses ahead-of-complier, RyuJIT for compilation. You can also use it with other compilers to perform native compilation for UWP apps.

As a developer, you can utilise its following benefits:

- It is easy to work with one single file generated during compilation along with app, CoreRT, and managed dependencies.
- It works fast because of the prior execution of compiled code. You don't need to generate machine code or load the JIT compiler at runtime.
- Since it uses an optimised compiler, thus it generates faster output from higher quality code.

23. Why is Startup Class important?

The Startup is a critical class in the application. The following points make it imperative:

- It describes the pipeline of the web applications.
- You can use individual startups for each environment.
- It helps to perform the registration of all required middleware components.
- Reading and checking thousands of lines in different environments is tough, but you can use various startup classes to resolve it.

24. What do you mean by state management?

Regarding .NET Core frameworks, state management is a kind of state control object to control the states of the object during different processes. Since stateless protocol, HTTP has been used, which is unable to retain user values; thus, different methods have been used to store and preserve the user data between requests.

Approach Name	Storage Mechanism
Cookies	HTTP Cookies, Server-side app code
Session state	HTTP Cookies, Server-side app code
Temp Data	HTTP Cookies, Session State
Query Strings	HTTP Query Strings
Hidden Fields	HTTP Form Fields
HTTPContext.Items	Server-side app code

Cache	Server-side app code

25. What is the best way to manage errors in .NET Core?

There are mainly four ways to manage errors in .NET Core for web APIs.

- Developer Exception Page
- Exception Handler Page
- Exception Handle Lambda
- UseStatusCodePages

But, in all these four, the best way is "Developer Exception Page" as it provides detailed information (stacks, query string parameters, headers, cookies) about unhandled request exceptions. You can easily enable this page by running your applications in the development environment. This page runs early in the middleware pipeline, so you can easily catch the exception in middleware.

26. IS MEF still available in .NET Core?

Yes, MEF or Managed Extensibility Framework is still available. This library plays a major role in developing lightweight and extensible applications. You can easily use extensions without configuration. You can restore the extensions within and outside the application. You can smoothly perform code encapsulation and prevent fragile complex dependencies.

It has been considered outdated but is still available. If you want to use it, you must use it using some plugins systems and namespaces like "System.Composition", "System.ComponnetModel.Composition", and "Microsoft.Composition".

27. What is response caching in .NET Core?

During response caching, cache-related headers are mentioned in the HTTP responses of .NET Core MVC actions. Using these headers, we can specify how the client/proxy machine will cache responses to requests. This, in turn, reduces the number of client/proxy requests to the web server because the responses are sent from the cache itself.

As we can see in the below diagram, the first request has a complete cycle from client browser to proxy server and then subsequently to web server. Now, the proxy server has stored the response in the cache. For all the subsequent requests, the proxy server sends the response from the cache itself. Hence, the number of proxy/client requests to the web server is reduced.

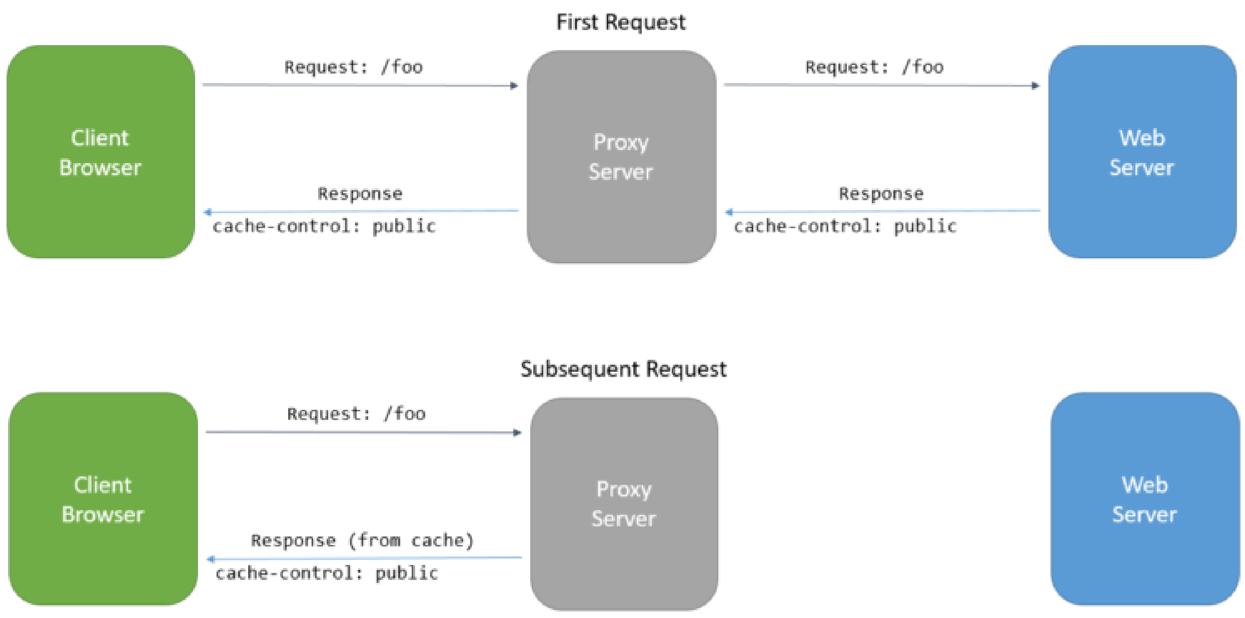


Image credit: https://jakeydocs.readthedocs.io/

28. What is a generic host in .NET Core?

The generic host was previously present as 'Web Host', in .NET Core for web applications. Later, the 'Web Host' was deprecated and a generic host was introduced to cater to the web, Windows, Linux, and console applications.

Whenever a new application is started we are required to take care of the below points:

- Dependency Injection
- Configuration
- Logging
- Service lifetime management

.NET generic host called 'HostBuilder' helps us to manage all the above tasks since it is built on the original abstraction of these tools.

29. What is routing in .NET Core?

It is a process through which the incoming requests are mapped to the corresponding controllers and actions. The .NET Core MVC has a routing middleware to perform this task. This middleware matches the incoming HTTP requests to the executable request-handling code. We can define the routing in the middleware pipeline in the 'Startup.Configure' file.

As we can see in the below code snippet, there are two methods or pair of middleware to define routing:

- **UseRouting:** Adds route which matches the middleware pipeline.
- **UseEndpoints:** Adds end execution point to the middleware pipeline and runs the delegate of the endpoint.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
if (env.IsDevelopment())
{
app.UseDeveloperExceptionPage();
}
app.UseRouting();
app.UseEndpoints(endpoints =>
{
endpoints.MapGet("/", async context =>
{
await context.Response.WriteAsync("Hello World!");
});
});
}
```

30. What is Dependency Injection in .NET Core? Explain its advantages.

.NET Core has been designed to support Dependency Injection(DI), which means the application is loosely coupled. It is a technique to introduce Inversion Control(IoC) between the classes and their dependencies. In other words, the object maintains only that dependency which is required during that particular task. A dependency is an object on which another object depends, by dependency injection, the application becomes better testable, maintainable, and reusable.

Dependency Injection has three steps:

- An interface or base class is present to provide an abstraction for dependency implementation.
- Dependency is registered in a service container, a built-in container IServiceProvider is present in .NET Core.
- Service is injected into the constructor of the class where dependency is used.

Advantages of Dependency Injection:

- Code is flexible, implementation can be changed without much overhead.
- Code becomes easy to test because of the use of interfaces.
- Code is loosely coupled, clean, and easy to maintain.

Practice Skills: *Developers love practice; Do your practice here and crack your coding interview*

ASP.NET MVC Interview Questions

31. What role does IIS manager play for ASP.NET MVC?

The application deployment process requires a windows server with an installed IIS manager. You need to use the IIS manager to perform deployment after the development of the applications. Without deployment, you can't bring any application to the market; thus, the IIS manager plays a primary role in completing this process. Click this link to know all steps of deployment using IIS manager.

Another deployment option is to use the Docker environment, which first deploys the docker package on any server machine and then implements the next deployment stage.

32. Discuss role-based authentication in ASP.NET MVC?

Roles define the permission to access something. A user can access any resource if they have permission. Role-based authentication is essential to ensure the security of applications and their data. It defines the role of providers and membership. The main task of providers is to give permission and assign roles to users to ensure authentication and authorisation.

Check this image to know how it works and establish security in applications.

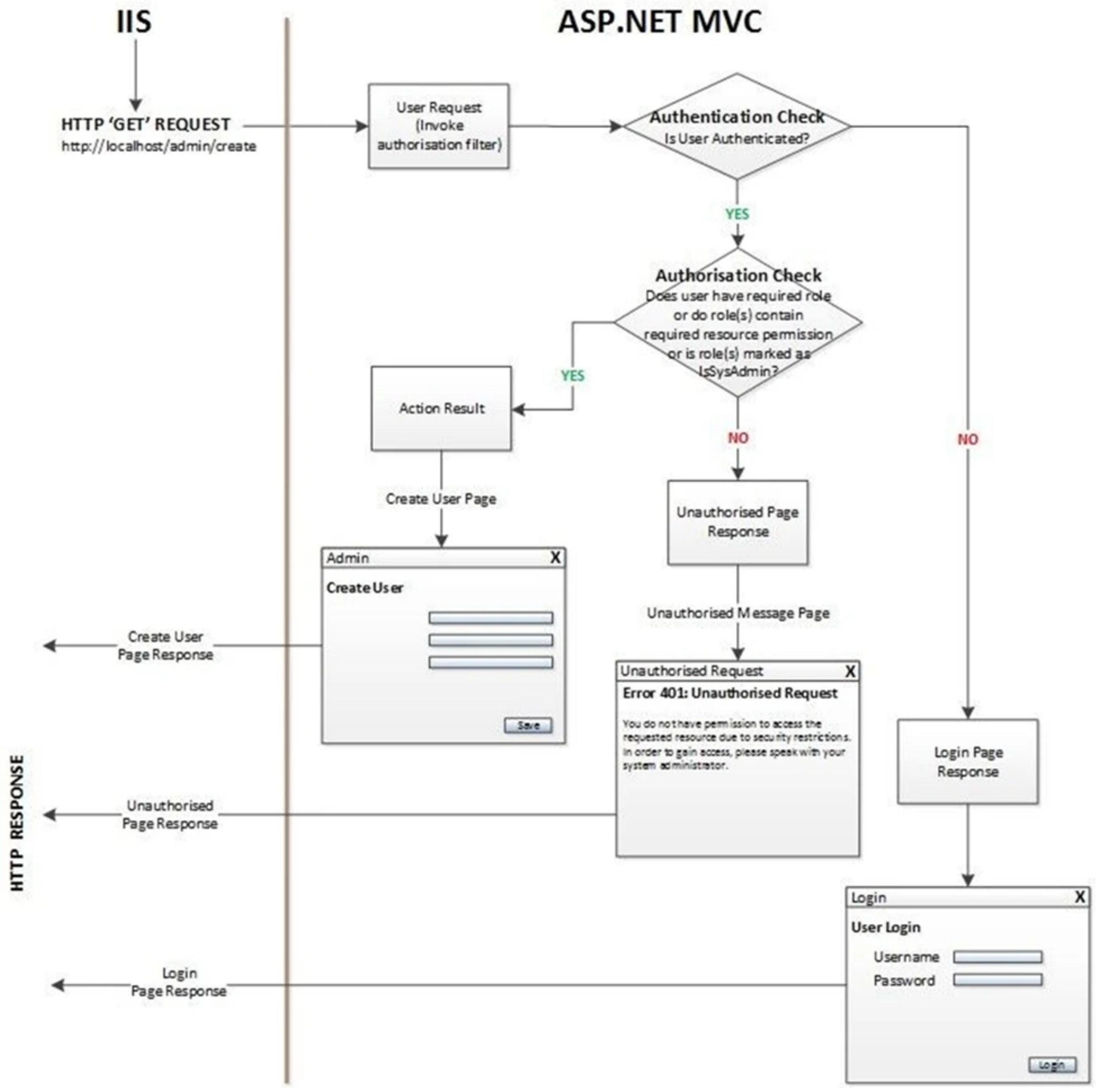

IIS
ASP.NET MVC
HTTP 'GET' REQUEST
http://localhost/admin/create
User Request (Invoke authorisation filter)
Authentication Check
Is User Authenticated?
YES
Authorisation Check
Does user have required role or do role(s) contain required resource permission or is role(s) marked as IsSysAdmin?
YES
Action Result
NO
NO
Create User Page
Unauthorised Page Response
Unauthorised Message Page
Admin
Create User
Save
Create User Page Response
Unauthorised Request
Error 401: Unauthorised Request
You do not have permission to access the requested resource due to security restrictions. In order to gain access, please speak with your system administrator.
Login Page Response
HTTP RESPONSE
Unauthorised Page Response
Login
User Login
Username
Password
Login
Login Page Response

Image Credit: C-metric

33. How would you differentiate ASP.NET from ASP.NET MVC?

Check the following points to understand how ASP.NET is different from ASP.NET MVC:

- ASP.NET is a web platform, whereas ASP.NET MVC is an application framework for building web applications.
- ASP.NET offers a layer that resides on the web server's top layer for creating web applications and services. Conversely, ASP.NET MVC framework stays on top of ASP.NET to design web applications with the help of ASP.NET's APIs.

- ASP.NET is based on a simple event-driven programming model, whereas ASP.NET MVC is based on the "Model-View-Controller" architectural model.

34. Which feature of ASP.NET Core MVC has been used as a new way of exposing server-side code that renders HTML elements?

The new "Tag helper" feature of ASP.NET Core MVC helps expose server-side code that renders HTML elements. It brings the same features of "HTML Razor helpers, " which looks like standard HTML elements. There is no need to switch context between HTML and Razor Syntax. Tag helpers are objects, and you can bound them to the models and dynamically render HTML elements according to their properties. Some of the common Tag-helper objects are as follows:

- **asp-action** – To use action methods
- **asp-for** – To use model binding
- **asp-route-id** – To use route expression
- **asp-validation-summary** – For validations

If you are a front-end designer working on CSS, JS frameworks or libraries, this feature can help you to quickly change or update the "View" without knowing the programming language. Additionally, they are reliable and reusable, which could be used in multiple views.

35. What is the view component feature?

View Component is another new feature that has been considered a powerful version of partial views. It is used for solving many problems. The primary function of this feature is to split the complex views into reusable parts. With the help of partial views, you can also access the parent page's view model.

But, one drawback of this feature is that it can't access the page model and can operate on the passed arguments. Thus, the best application of this feature is to use it to render reusable pieces of pages that might consist of logic. Use this feature through dependency injection, which makes it robust and reusable.

36. What do you mean by MVC application life cycle?

MVC application life cycle has two stages of executing applications.

First stage: Creating the request object

- Fill the route table using route collection
- Fetch the route to create "RouteData" object
- Now use this object to create "RequestContext" object
- At last, create a controller instance to control the class instance

Second stage: Creating the response object

- Execute the action
- Send the response to the browser as a result

37. What are the different return types used by the "controller action" method in MVC

In ASP.NET MVC, controllers, controller actions, and action results are linked. You can consider that the action is a method on the controller which is called whenever someone requests URL in the browser address bar. The controller responds to the requests and also exposes controller actions. In simple, this action returns action results in different return types. Check this following table to know these return types, which inherit from the base Action Result class.

Return Type	Meaning
ViewResult	It represents HTML and markup
EmptyResult	It represents no result
RedirectResult	It represents a redirection to a new URL
JsonResult	It represents a JavaScript Object Notation result which could be used in an AJAX application
JavaScriptResult	It represents a JavaScript script
ContentResult	It represents a text result
FileContentResult	It represents a downloadable file (with the binary content)
FilePathResult	It represents a downloadable file (with a path)
FileStreamResult	It represents a downloadable file (with a file stream)

38. What is Scaffolding in ASP.NET MVC?

One of the essential concepts of ASP.NET MVC that help developers like me generate code to perform basic operations – Create, Read, Update, Delete. You can make changes in the codes as per needs. That's why, we call it a "code-generation framework" for developing MVC applications. It helps enhance the code which interacts with the data model of applications. It also supports reducing the development time to execute data operations.

Additionally, the framework includes multiple templates, including page templates, field templates, , entity page templates, and filter templates. You can call them Scaffold templates. These templates permit you to design and build a fully functional website.

39. What is the role of Action Filters?

The central role of Action Filters in ASP.NET MVC is the execution of filtering logic after an action method is called. You can call these filters to "custom attributes" which helps to clarify declarations of pre-action or post-action behaviour to the controller's action methods. These attributes are derived from the "System.Attribute" which could be attached to classes, methods, fields, or properties. You can utilise any of these filters to implement filtering.

Filter type	Function
OutputCache	It caches the output of a controller action for a specific period
HandleError	It handles errors raised when a controller action executes
Authorise	It enables you to restrict access to a particular user or role

40. How to intercept exceptions using ASP.NET MVC?

An intercepting exception is an essential part of application development and execution. The exception handling's job is to respond to exceptional conditions. ASP.NET MVC has various ways to intercept exceptions, including

- **HandleError attribute on controllers and action method** – A simple method to handle errors and exception
- **Try-catch-finally** – A simple three blocks to catch the exception
- **Overriding OnException Method** – A void method that takes an argument as an object of ExceptionContext to manage exception
- **Setting a goal exception handling filter** – You have to take care of HandleErrorAttribute and need to add it RegisterGlobalFilters
- **Extending HandleErrorAttribute** – It permits you to create your Exception Handler to manage the errors

41. What is ASP.NET MVC? Explain its components.

It is a lightweight and open-source web development framework, which is used to decouple data(Model), interface (View), and logic(Controller). It provides a pattern-based way to create dynamic websites and supports TDD-based development.

An MVC(Model-View-Controller) architectural pattern separates the application into three components and provides separation of concerns.

- **Model**: Represents the state of application/logic, where the business logic and implementation logic is encapsulated.
- **View**: It is responsible for providing the view through the user interface.
- **Controller**: Handles user interaction, works in tandem with model and view components.

It provides the latest web standards and many features like routing, model binding, model validation, dependency injection, web APIs, razor view engine, filters, etc.

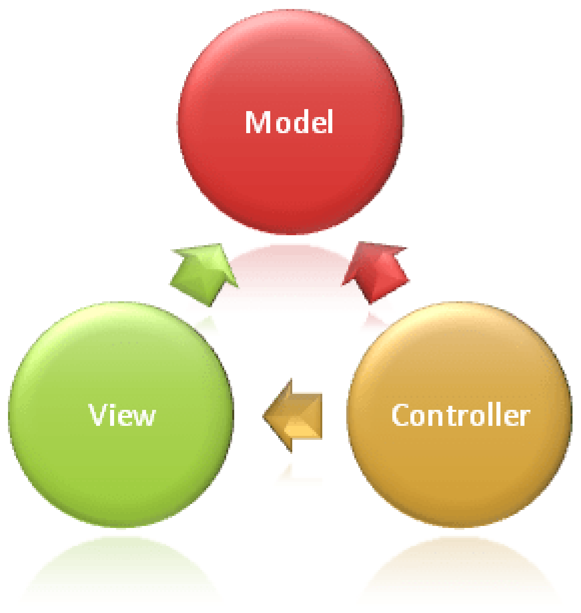


Image credit: https://docs.microsoft.com/

42. What are the advantages of ASP.NET MVC?

- Provides full control over HTML rendering.
- Provides separation of concerns(SoC).
- Reduction of complexity by dividing the application into three components.
- Supports test-driven development(TDD).
- Easy integration with JavaScript, JSON, jQuery, etc.
- Uses the latest technology and supports the latest trends.

43. Why use an area in ASP.NET MVC?

Any large ASP.NET MVC project has many controllers, views, and model classes. With time, it will become very difficult to manage it using the default MVC project structure.

The area is used to physically partition the large application into small functional units. Each unit has its own MVC folder structure and model, view, and controller folders.

The below example shows how each area - admin, finance, HR has its own set of model, view, and controller folders.

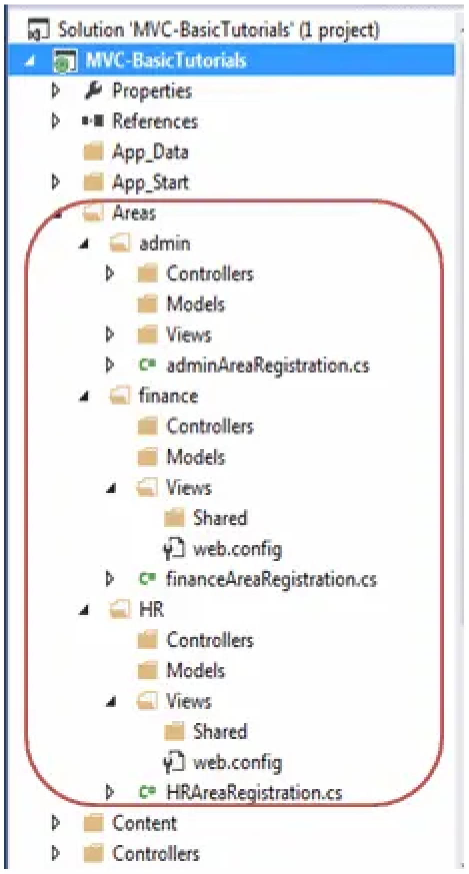


Image credit: https://www.tutorialsteacher.com/

44. What is the difference between ViewData and ViewBag in ASP.NET MVC?

ViewData and ViewBag in ASP.NET MVC are used for transferring data from controller to view. Below are the differences between them:

ViewData	ViewBag
It is a dictionary object of the 'ViewDataDictionary' class having key-value.	It is a wrapper around ViewData and is a dynamic property.
Faster than ViewBag.	Slower than ViewData.
Type conversion code is required.	Dynamic hence type conversion code is not required.

45. Describe the request flow in the ASP.NET MVC framework.

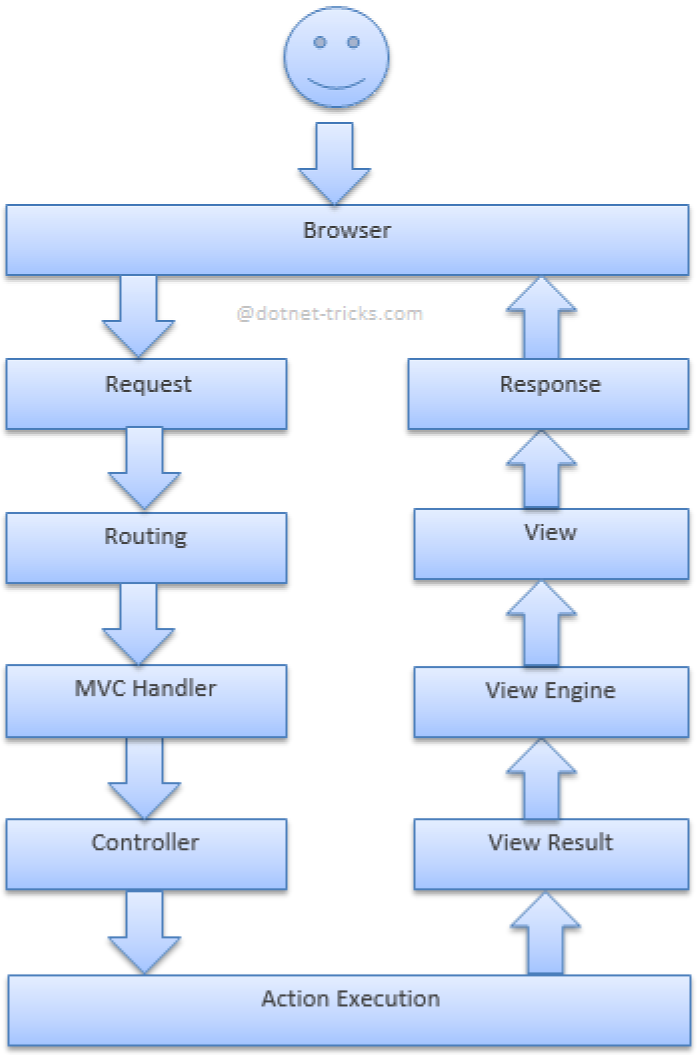
The request flow has below stages in the MVC framework:

- **Routing**: It is the first step which matches the pattern of the request's URL against the URL present in the route table.

- **MvcHandler**: It starts the processing of the request using the ProcessRequest method.
- **Controller**: Uses 'IControllerFactory' instance and calls the 'Execute' method, where 'IControllerFactory' is a default controller factory or a custom factory can be defined.
- **Action execution**: After controller instantiation, 'ActionInvoker' defines which action to be performed on the controller.
- **View result**: The 'action' method prepares the response and then returns a result.
- **View engine**: 'IViewInterface' of the view engine selects a view engine to render the result.
- **View**: 'ViewResult' returns and renders an HTML page on the browser.

Practice Skill: *Expertise comes from practice. Develop your coding skills to become an expert.*

Refresh your backend development skills here.



ASP.NET MVC Request Life Cycle

Image credit:
https://dotnettrickscloud.blob.core.windows.NET