Top .NET Core Interview Questions and Answers (2024)

Gaurav Gandhi

Introduction

NET Core is a free, open-source, cross-platform framework for building modern, cloud-based, and internet-connected applications. It was developed by Microsoft and released in 2016.

.Net Core is a modular framework that allows developers to choose the libraries and frameworks they need for their applications. It supports multiple programming languages, including C#, F#, and Visual Basic.

Overall, .NET Core is a powerful framework for building modern applications running on any platform and in any environment.

Its flexibility, cross-platform support, and ease of use make it a popular choice for developers building various applications, from the web to mobile and gaming.

Today, we will discuss the most asked Net Core Interview Questions and answers. Let us start with our journey towards cracking the .Net core Interview.

.NET Core Interview Questions for Freshers

In this section, we are going to look at Beginner level Net Core Interview Questions.

1. What is .NET Core?

.NET is a software framework created by Microsoft that helps developers build applications for computers, mobile devices, and the web. It provides a set of tools and libraries that make it easier for developers to create software. .NET Core is a free, open-source, cross-platform framework for building modern applications.

2. What is ASP.NET Core?

ASP.NET Core is a framework for building web applications using .NET Core. It includes a rich set of tools and libraries for building scalable and secure web applications.

Some of the key features of ASP.NET Core include middleware components for handling HTTP requests and responses, support for web sockets and SignalR for real-time communication, and support for multiple authentication and authorization schemes.

Overall, ASP.NET Core is a powerful and versatile web framework that enables developers to build fast, scalable, and cross-platform web applications and services.

3. What are the advantages of ASP.NET Core over ASP.NET?

ASP.NET Core has several advantages over ASP.NET. Some of these include:

- **Cross Platform Compatibiility**
- **Lightweight and less resource consuming**
- **Better efficiency and performance**
- **Open-source**
- **Built-in API Support**

4. What are the features provided by ASP.NET Core?

Some of the key features of ASP.NET Core include middleware components for handling HTTP requests and responses, support for web sockets and SignalR for real-time communication, and support for multiple authentication and authorization schemes.

5. How do you handle errors in .NET Core?

.NET Core provides a built-in middleware for handling errors called the Exception Handling Middleware. Developers can also write custom middleware to handle specific types of errors.

6. What are the key features of .Net Framework?

Some key features of the .NET Framework include:

- **Common Language Runtime (CLR)**, which manages the execution of code and provides features such as memory management and security.
- **Class Library**, a collection of pre-built code that developers can use to build applications.
- **Language-Integrated Query (LINQ)**, a feature that allows developers to query data from various data sources using a common syntax.
- **Windows Presentation Foundation (WPF)**, a framework for building graphical user interfaces for desktop applications.
- ASP.NET, a framework for building web applications and services.

7. What is the difference between .NET Framework and .NET Core?

Below is the major difference between .Net Framework and .Net Core:

- .NET Framework is a Windows-only framework, while .NET Core is cross-platform and can run on Windows, Linux, and macOS.
- .NET Framework applications require the framework to be installed on the target machine, whereas .NET Core applications can be deployed as self-contained executables that include all of the necessary dependencies.

8. What is a namespace in .NET?

In .NET, a namespace is a way to organize and group related classes, structures, interfaces, enumerations, and other types within a common naming context. It provides a way to avoid naming conflicts between types and helps in better organization of the code.

9. What is an assembly in .NET?

In .NET, an assembly is a unit of deployment that contains one or more files, such as executable files, DLLs, or resource files, that form a logical unit of functionality. It is the smallest unit of deployment and versioning in .NET, and can be shared among multiple applications.

10. What is the difference between value types and reference types in .NET?

Below is the difference between value types and reference types in .Net:

Value Types	Reference Types
Value types are types that store their data directly in memory rather than storing a reference to an object.	Reference types, on the other hand, are types that store a reference to an object in memory rather than storing the object's data directly.
Value types are managed by the compiler.	Reference types are managed by the garbage collector, which periodically frees up memory that is no longer being used
They are cleaned up automatically when they go out of scope.	They are cleaned up by the Garbage collector.

Value Types	Reference Types
Store data directly in memory.	Store a reference to an object in memory
Small in size	Large in size

.NET Core Interview Questions for Intermediate

In this section, we are going to look at Intermediate level Net Core Interview Questions.

11. Explain about major components of the .NET framework.

The major components of the .NET framework include the **Common Language Runtime (CLR), .NET Languages, Base Class Library (BCL),** and the **Just-In-Time Compiler.**

- **Common Language Runtime (CLR)** - It provides the environment for the user to run .NET applications.
- **Base Class Library (BCL)** - This is a collection of classes, methods, etc., that provide the functionality to the applications.
- **.NET Languages** - It is the collection of languages in the .NET Framework. It includes languages like C#, VB.NET, etc.
- **Just-In-Time Compiler** - This compiler converts the user code into the Machine code during runtime.

12. What is the role of the Startup class in .NET Core?

The Startup class is a fundamental component of the .NET Core web application architecture. It is responsible for configuring the application and its dependencies.

When the application starts, the Startup class is executed, and it performs the following tasks:

- **Sets up the environment**: The Startup class sets up the hosting environment for the application, such as development, staging, or production. It also loads the configuration settings for the application.
- **Configures services**: The Startup class is responsible for configuring the services that the application needs to function correctly.

Although it is not being embedded now, there is a file known as Program.cs file that performs the same operation as Startup.cs. However, you can create your own Startup.cs.

13. What is middleware in .NET Core?

In .NET Core, middleware is a software component that sits between a web application's client-side and server-side components. It is responsible for processing incoming HTTP requests and generating appropriate HTTP responses. Middleware is a key feature of the ASP.NET Core framework, which is used to develop web applications.

14. What is the difference between a managed and an unmanaged code?

Managed code is code that is executed within a managed execution environment, such as the .NET Common Language Runtime (CLR).Whereas Unmanaged code, on the other hand, is code that is executed directly by the operating system without the use of a managed execution environment.

15. Explain the architecture of .NET Core.

.NET Core is an open-source, cross-platform development framework that can be used to create applications for Windows, Linux, and macOS operating systems.

The architecture of .NET Core can be broken down into several key components:

- **CoreCLR:** The CoreCLR is the runtime environment for .NET Core. It is responsible for loading, compiling, and executing managed code.
- **Class Library:** The Class Library is a collection of reusable code that provides a set of APIs for common programming tasks, such as working with strings, files, and networking.
- **Host:** The Host is responsible for initializing and running the application. It includes a command-line interface (CLI) that allows developers to build, package, and deploy applications.

16. What are the different types of hosting models supported in .NET Core?

.NET Core supports several different hosting models, which allow developers to deploy and run their applications in a variety of environments.

The following are the different types of hosting models supported in .NET Core:

1. **In-process hosting**: In-process hosting allows an appli**What is NET Core features?** cation to be hosted within the same process as the hosting application. This is the default hosting model used by ASP.NET Core applications.
2. **Out-of-process hosting**: Out-of-process hosting allows an application to be hosted in a separate process from the hosting application.

17. How do you implement middleware in .NET Core?

Steps to implement middleware in .NET Core are:

1. **Create a middleware class:** Create a new class that implements the IMiddleware interface or the Microsoft.AspNetCore.Http.IMiddleware interface.
2. **Implement the middleware logic:** In the middleware class, implement the logic for handling requests and responses. This could involve adding headers, modifying the response, or performing some other task.
3. **Register the middleware:** In the Configure method of the Startup class, use the UseMiddleware extension method to register the middleware.

18. What is the role of the Program.cs file in .NET Core?

The Program.cs file is the entry point of a .NET Core application. It contains the Main method, which is the starting point of the application. The Main method is responsible for configuring the application and starting the application's host.

The host is responsible for managing the application's lifetime, including starting and stopping the application. In addition to configuring the host, the Main method can also configure services that are used by the application. These services can include middleware, logging, and other components that the application needs to run.

19. What is the Common Language Runtime (CLR) in .NET?

The Common Language Runtime (CLR) is a component of the .NET Framework that provides a runtime environment for managed code. The CLR is responsible for managing the execution of .NET programs, including memory management, security, and exception handling.

It is the component that actually executes compiled .NET code.

20. Explain the concept of Razor Pages in .NET Core.

Razor Pages is a way of building web applications in .NET Core that makes it easier for developers to create web pages.

With Razor Pages, developers can create a page that combines the HTML markup and the C# code needed to generate the page in a single file.

This approach simplifies the programming model by removing the need for a separate controller and view. Instead, each Razor Page has its own page model that handles the business logic for that page.

Razor Pages are built on top of the existing Razor view engine that is used for rendering HTML, which makes it easier for developers who are already familiar with Razor syntax to work with them.

.NET Core Interview Questions for Advanced

In this section, we are going to look at Advance level Net Core Interview Questions.

21. How does .NET Core support cross-platform development?

.NET Core is designed from the ground up to support cross-platform development, which means that developers can write code that can run on multiple platforms, including Windows, macOS, and Linux.

This is achieved through several key features of the .NET Core platform:

- **Cross-platform runtime**: .NET Core includes a runtime that can be installed on multiple platforms, allowing .NET Core applications to run on any platform that supports the runtime.
- **Cross-platform tooling**: .NET Core includes a set of command-line tools that can be used on multiple platforms, including the .NET Core CLI and Visual Studio Code.

22. What are some of the new features introduced in the latest version of .NET Core?

The latest version of .NET Core, as of my knowledge cutoff date of September 2021, is .NET 6.0. Some of the new features introduced in this version include:

1. **Performance improvements:** .NET 6.0 includes several performance improvements, including faster startup times and reduced memory usage for ASP.NET Core applications.
2. **Single-file applications:** .NET 6.0 includes support for building single-file applications, which can simplify deployment and distribution of .NET Core applications.
3. **Hot reload:** .NET 6.0 includes support for hot reload, which allows developers to make changes to code while an application is running and see the changes immediately without restarting the application.
4. **HTTP3 support:** .NET 6.0 includes support for HTTP3, the latest version of the HTTP protocol, which can improve performance and security for web applications.

23. What are delegates in .NET?

Delegates in .NET are type-safe function pointers that allow you to reference methods or functions in a way that is similar to C/C++ function pointers. Delegates can be used to pass methods as parameters to other methods or to store references to methods in a data structure.

A delegate is a reference type that encapsulates a method**What is NET Core features?** with a specific signature, consisting of a return type and zero or more input parameters. When a delegate is invoked, the method it references is executed with the specified input parameters, and the result is returned.

24. What is the difference between Debug and Release mode in .NET?

Below is the difference between value types and reference types in .Net:

Debug Mode	Release Mode
Debug mode is used during development, and testing stages	Release mode is used when an application is ready for distribution.
Debug mode is used to identify and fix bugs.	Release mode is used to optimize the performance of an application and prepare it for deployment.
When an application is built in Debug mode, the compiler includes additional debugging information, such as symbols and metadata, that can be used to help identify and diagnose problems in the code.	When an application is built in Release mode, the compiler applies optimization techniques to generate code that runs faster and more efficiently.

25. How do you configure logging in .NET Core?

In .NET Core, logging can be configured using the built-in logging framework, which provides a simple and extensible way to log messages from your application. The following steps outline how to configure logging in .NET Core:

1. Add the required logging provider NuGet packages to your project.
   These packages include Microsoft.Extensions.Logging.Console, Microsoft.Extensions.Logging.Debug, Microsoft.Extensions.Logging.EventLog, Microsoft.Extensions.Logging.Abstractions, and others, depending on your needs.

2. In your application's Program.cs file, create a HostBuilder instance and configure logging by calling the ConfigureLogging() method on the HostBuilder object.
   For example:

```
using Microsoft.Extensions.Logging;
public static void Main(string[] args) {
  var hostBuilder = Host.CreateDefaultBuilder(args)
    .ConfigureLogging(logging => {
      logging.ClearProviders();
      logging.AddConsole();
      logging.AddDebug();
    })
    .ConfigureWebHostDefaults(webBuilder => {
      webBuilder.UseStartup < Startup > ();
    })
    .Build();

  hostBuilder.Run();
}
```

26. How is a Managed code executed?

In .NET Core, managed code is executed using the .NET Core Runtime, which is a cross-platform runtime that includes the Common Language Runtime (CLR) and the Base Class Library (BCL).

When you compile a .NET Core application, the source code is compiled into an intermediate language called Common Intermediate Language (CIL), which is also known as Microsoft Intermediate Language (MSIL). This is a high-level, CPU-independent representation of the code that can be executed on any platform that has the .NET Core runtime installed.

27. What are the different parts of an Assembly?

In .NET Core, an assembly is the primary unit of deployment, and it contains one or more files that make up a logical unit of functionality.

The different parts of an assembly in .NET Core include:

- **Manifest:** This is a metadata that describes the assembly, including its version number, strong name, dependencies, and security permissions. The manifest is stored in the assembly's main file, which has the extension .dll or .exe.

- **MSIL code:** This is the actual executable code of the assembly, which is written in Common Intermediate Language (CIL) and is stored in one or more files with the extension .dll or .exe. The MSIL code is compiled by the Just-In-Time (JIT) compiler into native code that can be executed by the CPU.

- **Type metadata:** This is information about the types defined in the assembly, including their names, properties, methods, and fields. Type metadata is used by the CLR to load and execute types at runtime.

- **Resources:** These are data files, such as images, icons, and configuration files, that are embedded in the assembly and can be accessed at runtime using the System.Reflection.Assembly class.

28. What is the difference between synchronous and asynchronous programming in .NET Core? When should you use each approach?

In .NET Core, synchronous programming is the traditional programming approach, where the program execution is blocked until the operation completes, and the calling thread is suspended until the result is returned.

In contrast, asynchronous programming allows the program to continue executing while the operation is in progress, freeing up the calling thread to perform other tasks.

Synchronous programming is suitable when the program does not have a large number of concurrent tasks, and the tasks are relatively short-lived, and the program does not need to wait for the results of other tasks. It is easier to understand and debug compared to asynchronous programming.

29. What are the benefits of using Entity Framework Core for data access in a .NET Core application?

There are several benefits of using Entity Framework Core for data access in a .NET Core application:

1. **Productivity:** Entity Framework Core can simplify the data access layer, making it easier and faster to develop the application. It provides a high-level abstraction of the database, allowing developers to work with objects instead of SQL statements.

2. **Performance:** Entity Framework Core can automatically generate efficient SQL queries that can improve the performance of the application.

3. **Cross-platform support:** Entity Framework Core is a cross-platform framework that can be used with .NET Core applications on Windows, Linux, and macOS.

4. **Database independence:** Entity Framework Core can work with multiple database providers, including Microsoft SQL Server, MySQL, PostgreSQL, and SQLite.

5. **Security:** Entity Framework Core can help prevent SQL injection attacks by automatically sanitizing user input and generating parameterized queries.

30. What is Dependency Injection, and how is it used in .NET Core?

Dependency Injection (DI) is a way of organizing code so that it is easier to reuse and maintain. In a program, there are often many different objects that need to work together to get things done.

For example, imagine you are building a car. You would need parts like the engine, wheels, and steering system to all work together.

In a **traditional programming approach**, each part of the program might create the other parts it needs directly. For example, the engine might create the wheels and the steering system. This approach can make the code more difficult to maintain and test, especially as the program gets

larger.

With DI, each part of the program doesn't create the other parts it needs directly. Instead, it gets the parts it needs from an external provider, which is responsible for creating and managing all the parts. This ewxternal provider is called the "dependency injector".

Using the car example, the engine, wheels, and steering system would all be separate objects, and the dependency injector would be responsible for creating them and making sure they work together properly. This approach makes the code easier to maintain and test because each part is more modular and can be tested separately.

In .NET Core, the DI container is built into the platform and can be used to manage dependencies between objects. Developers can register dependencies with the DI container, and it will automatically create and manage those objects for them. This allows developers to focus on writing their code without worrying about managing dependencies between objects.

Frequently Asked Questions

How should I prepare for a .NET interview?

Some essential points for the .NET interview are revising the fundaments of .NET, like Common Language Runtime, .NET Framework, and OOPS concepts. Solve Coding problems related to. NET. Research the last few years' interview questions and Create small projects or contribute to open-source projects using .NET technologies.

Why .NET Core is faster than .NET Framework?

Many people believe that.NET Core is faster than.NET Framework because it has essential features such as cross-platform capabilities, allowing it to run on various operating systems. Furthermore,.NET, Core features optimization techniques and uses sophisticated hardware capabilities to improve code execution.

What is the difference between .NET Core and .NET framework?

.NET Core and .NET framework are two separate frameworks with some key differences. .NET Core is a cross-platform, lightweight, open-source framework that is mainly used for cloud-based applications. On the other hand, the .NET framework is a Windows-only, fully-featured framework used for building desktop applications.

What is .NET Core features?

.NET Core provides multiple features like cross-platform support for Windows, Linux, and MacOS, lightweight architecture, multi-language support, high performance, a large community with active support, etc. This makes .NET Core a great choice for cloud application development.

Conclusion

To conclude the discussion, we have discussed Net core Interview Questions. These questions cover almost every important aspect of .Net Core.

After reading about the Net Core Interview Questions, are you not excited to read or explore more

articles on other interview-related articles? Don't worry, and Coding Ninjas has your back:

- oop interview questions
- System Design Interview Questions
- SQL Query Interview Questions
- Excel Interview Questions
- C# Interview Questions

Happy Learning Ninja!