1. What is .NET Core?

.NET Core is an open-source, cross-platform framework used to build modern, cloud-based applications. It is designed to run on Windows, macOS, and Linux.

2. What are the advantages of using .NET Core?

Some advantages of using .NET Core are: cross-platform compatibility, improved performance, open-source codebase, easy deployment and maintenance, and a large community of developers.

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

.NET Framework is a Windows-only framework, while .NET Core is cross-platform. .NET Framework requires the full .NET runtime, while .NET Core can run on a lightweight runtime.

4. What is the difference between ASP.NET and ASP.NET Core?

ASP.NET is the original framework for building web applications using .NET, while ASP.NET Core is the newer, cross-platform framework. ASP.NET Core is more modular, lightweight, and faster than ASP.NET.

5. How does dependency injection work in .NET Core?

Dependency injection is used to provide objects with their dependencies. In .NET Core, services are registered with the dependency injection container, and the container provides them to the classes that need them.

6. What is the role of the Startup.cs file in .NET Core?

The Startup.cs file is used to configure the services that the application needs and to set up the middleware pipeline. It is executed when the application starts up.

7. What is the difference between middleware and filters in .NET Core?

Middleware is software that sits between the web server and the application, while filters are used to modify the request or response as it passes through the middleware pipeline. Filters can be applied globally or to specific actions.

8. What is the purpose of appsettings.json file in .NET Core?

The appsettings.json file is used to store configuration settings for the application, such as database connection strings and API keys.

What Is Kestrel?

Kestrel is an event-driven, open-source, asynchronous, cross-platform server that hosts .Net applications. It is provided as a default server for .Net Core. As a result, it is compatible with all the platforms and their respective versions, which .Net does not support.

Some of the primary advantages of Kestrel also include but are not limited to; it is light and fast, can support HTTPS, it's easy to use, and has an exemplary configuration.

9. What is the purpose of the wwwroot folder in .NET Core?

The wwwroot folder is where static files, such as HTML, CSS, and JavaScript files, are stored. These files can be accessed by the client directly, without going through the middleware pipeline.

10. What is the difference between Razor Pages and MVC in .NET Core?

Razor Pages is a newer, simpler framework for building web applications, while MVC is a more traditional framework. Razor Pages is recommended for simple applications, while MVC is recommended for more complex applications.

11. What is Entity Framework Core?

Entity Framework Core is an object-relational mapping (ORM) framework used to interact with databases in .NET Core applications. It provides a simple, intuitive way to perform CRUD operations and query the database.

12. What is the difference between Code First and Database First in Entity Framework Core?

Code First is a method of creating the database schema based on the classes in the application, while Database First is a method of generating the classes in the application based on an existing database schema.

What are the benefits of using Entity Framework Core?

The benefits of using Entity Framework Core include reduced boilerplate code, improved developer productivity, improved performance, and better security.

What is a migration in Entity Framework Core?

Migration is a way to update the schema of a database to match the changes made to the data model. It's implemented using the EF Core command-line interface (CLI) and generates SQL scripts to apply the changes.

13. What is the difference between eager loading and lazy loading in Entity Framework Core?

Eager loading is the process of loading all related entities at once, while lazy loading is the process of loading related entities only when they are accessed.

14. What is the difference between a DbSet and a DbContext in Entity Framework Core?

A DbSet represents a table in the database, while a DbContext represents the database context, which is used to query and save data.

15. What is LINQ?

LINQ is a language-integrated query that allows developers to write queries using familiar syntax. It is used to query databases, collections, and other data sources in .NET applications.

16. What is a Lambda expression in .NET Core?

A Lambda expression is a short, anonymous function that can be used to define a delegate or expression tree. It is used to write concise and readable code in .NET Core applications.

17. What is the purpose of the using statement in C#?

The using statement is used to ensure that a disposable object is properly disposed of after it is no longer needed. It is commonly used with database connections and file streams.

18. What is the difference between an abstract class and an interface in C#?

An abstract class can contain implementation details, while an interface cannot. A class can implement multiple interfaces, but can only inherit from one abstract class.

19. What is a delegate in C#?

A delegate is a type that represents a reference to a method with a specific signature. It is used to pass methods as parameters or to define event handlers.

20. What is the purpose of the async/await keywords in C#?

The async/await keywords are used to write asynchronous code in a synchronous style. They allow code to continue executing while waiting for a long-running operation to complete.

21. What is a generic type in C#?

A generic type is a type that is defined with one or more type parameters. It allows developers to create reusable code that can work with different types.

22. What is a nullable type in C#?

A nullable type is a value type that can also be null. It is represented by adding a question mark (?) to the end of the type name.

23. What is the difference between a struct and a class in C#?

A struct is a value type, while a class is a reference type. Structs are typically used for small, simple data structures, while classes are used for larger, more complex objects.

24. What is the purpose of the const keyword in C#?

The const keyword is used to declare a constant value that cannot be changed at runtime. It is typically used for values that are known at compile time.

25. What is the difference between a static class and a non-static class in C#?

A static class cannot be instantiated, while a non-static class can. Static classes are used for utility classes that do not need to maintain state.

26. What is the purpose of the lock keyword in C#?

The lock keyword is used to synchronize access to shared resources in a multithreaded environment. It ensures that only one thread can access the resource at a time.

27. What is the difference between the StringBuilder and String classes in C#?

The StringBuilder class is used to build strings dynamically, while the String class is used to represent immutable strings. StringBuilder is more efficient when manipulating strings frequently.

28. What is the purpose of the yield keyword in C#?

The yield keyword is used to create an iterator that can be used to generate a sequence of values on demand. It is typically used to generate large sequences of data.

29. What is garbage collection in .NET Core?

Garbage collection is the process of automatically freeing memory that is no longer being used by the application. It helps prevent memory leaks and improves the stability of the application.

30. What is the difference between a value type and a reference type in .NET Core?

A value type contains the actual value, while a reference type contains a reference to the value. Value types are typically smaller and faster than reference types.

31. What is the purpose of the using directive in C#?

The using directive is used to import namespaces into a C# file. It allows developers to use classes and types without having to specify the fully qualified name.

---

32. What is a try-catch block in C#?

A try-catch block is used to handle exceptions that may occur during runtime. The code in the try block is executed first, and if an exception occurs, the code in the catch block is executed.

33. What is the purpose of the finally block in a try-catch-finally statement in C#?

The finally block is used to execute code that should always be executed, regardless of whether an exception was thrown or not. It is typically used to release resources.

34. What is a lambda expression in C#?

A lambda expression is a concise way to define an anonymous method. It is often used to define event handlers and to pass methods as parameters.

35. What is an extension method in C#?

An extension method is a static method that can be called as if it were an instance method on a particular type. It allows developers to add functionality to existing types without having to modify the original code.

36. What is the difference between a synchronous and asynchronous method in C#?

A synchronous method blocks the current thread until it completes, while an asynchronous method does not. Asynchronous methods allow code to continue executing while waiting for a long-running operation to complete.

37. What is a LINQ query in C#?

A LINQ query is a query that is written using Language Integrated Query syntax. It allows developers to query and manipulate data from different sources, including arrays, collections, and databases.

38. What is a tuple in C#?

A tuple is a data structure that can hold multiple values of different types. It is often used to return multiple values from a method.

39. What is a CancellationToken in .NET Core?

A CancellationToken is an object that can be used to cancel a long-running operation. It is typically passed as a parameter to asynchronous methods.

40. What is a middleware in .NET Core?

Middleware is a component that sits between the web server and the application. It can be used to perform various tasks, such as authentication, logging, and caching.

41. What is the difference between a repository and a service in .NET Core?

A repository is a class that is responsible for data access, while a service is a class that performs business logic. Repositories typically work with a specific data source, while services can work with multiple sources.

42. What is dependency injection in .NET Core?

Dependency injection is a design pattern that allows dependencies to be injected into an object at runtime. It allows for loosely coupled code and easier testing.

43. What is the difference between a singleton and a scoped service in .NET Core?

A singleton service is created once and shared across the entire application, while a scoped service is created once per request. Singleton services are typically used for stateless services, while scoped services are used for stateful services.

44. What is the purpose of the ASP.NET Core pipeline?

The ASP.NET Core pipeline is used to handle requests and responses in a web application. It consists of middleware components that can be added or removed to modify the behavior of the application.

45. What is a JWT token in .NET Core?

A JWT (JSON Web Token) token is a secure way to transmit information between parties as a JSON object. It is commonly used for authentication and authorization in web applications.

46. What is a CORS policy in .NET Core?

A CORS (Cross-Origin Resource Sharing) policy is used to control access to resources from different domains. It allows developers to specify which domains are allowed to access resources in the application.

47. What is the purpose of the appsettings.json file in .NET Core?

The appsettings.json file is used to store configuration data for the application. It allows developers to configure various settings, such as database connection strings and logging options.

48. What is the purpose of the Startup.cs file in .NET Core?

The Startup.cs file is used to configure the services and middleware components for the application. It is typically the entry point for the application and is responsible for setting up the dependency injection container and registering middleware.

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

.NET Framework is a Windows-only framework that has been around since 2002, while .NET Core is a cross-platform framework that was first released in 2016. .NET Core is designed to be lightweight, modular, and open-source, while .NET Framework is a more monolithic framework that is tightly integrated with Windows.

50. What are some advantages of using .NET Core?

Some advantages of using .NET Core include:

- Cross-platform support
- Lightweight and modular
- Open-source
- Improved performance and scalability
- Better support for modern web development, including containerization and microservices.

51. AddSingleton() vs AddTransient() vs AddScope() in ASP.Net Core

---

All these 3 methods are very important in ASP.Net Core, they have there own significance of using in your application. Also, this is on of the most important ASP.Net Core interview question. Let's see them one by one.

AddSingleton() Method

In AddSingleton() method, only one instance of service is created, when the service is executed at very first time. So, a single instance is created for a service , no matter how many times the service is being executed.

OR

Singleton objects are the same for every object and every request.

OR

Singleton lifetime services are created the first time they are requested (or when ConfigureServices is run if you specify an instance there) and then every subsequent request will use the same instance.

AddTransient() Method

With AddTransient() method, it creates new instance every time a service request is raised, doesn't matter it is in the same HTTP request or different HTTP request.

OR

Transient objects are always different; a new instance is provided to every controller and every service.

AddScope() Method

With AddScope() method, we get new instance with different http requests. But we get same instance if it is within the same scope.

OR

Scoped objects are the same within a request, but different across different requests.

OR

Scoped lifetime services are created once per request.

52. What is the difference between app.Run and app.Use in asp.net core middleware pipeline?

---

app.Run() will end the request, and app.Use() will pass the request to next middleware.

First of all both methods are used to registered middleware, to the application request pipeline.

The only difference is middleware defined using app.Use may call next middleware component in the pipeline. On the other hand, middleware defined using app.Run will never call subsequent middleware.

It means app.Run acts as a terminal middleware, and so no other middleware method will run after this.