.NET

Interview Questions

Q: What is .NET?

A: .NET is a free, `cross-platform`, `open-source` developer platform from Microsoft used to build different types of applications, such as web, mobile, desktop, and cloud.

Q: What is the difference between .NET Framework, .NET Core, and .NET `5/6` /7+?

A: .NET Framework is `Windows-only`, .NET Core is `cross-platform` and modular, and .NET 5+ is a unified platform combining the best of both.

Q: What is the CLR?

A: The Common Language Runtime (CLR) is the virtual machine `component` of .NET that manages memory, executes code, handles exceptions, and provides `type` safety.

Q: What is the CTS?

A: The Common `Type` System (CTS) defines how types are declared, used, and managed in the runtime to ensure `type` safety between different .NET languages.

Q: What is the CLS?

A: The Common Language Specification (CLS) is a set of base rules to ensure interoperability among .NET languages.

Q: What are value types and reference types in .NET?

A: Value types store data directly and are stored in the stack, while reference types store references to data and are stored on the heap.

Q: What is a `namespace` in `C#`?

A: A `namespace` is a container that holds classes, structs, enums, delegates, and interfaces to organize code and avoid naming conflicts.

Q: What is the difference between ‘==’ and ‘Equals()’ in `C#`?

A: ‘==’ compares `object` references by default, unless overloaded. ‘Equals()’ is a method that compares `object` values, and can be overridden.

Q: What is boxing and unboxing in .NET?

A: Boxing is converting a value `type` to an `object`; unboxing is converting an `object` back to a value `type`.

Q: What is the difference between ‘`const`’ and ‘readonly’?

A: ‘`const`’ values are assigned at `compile-time` and are static by default, while ‘readonly’ can be assigned at runtime, usually in the `constructor`.

Q: What is the purpose of ‘using’ statement in `C#`?

A: The ‘using’ statement ensures proper disposal of IDisposable objects to free up unmanaged resources.

Q: What is garbage collection in .NET?

A: Garbage collection automatically manages memory by cleaning up unused objects in the heap.

Q: What are assemblies in .NET?

A: Assemblies are compiled code libraries (.dll or .exe) that contain metadata about types and resources.

Q: What is the Global Assembly Cache (GAC)?

A: The GAC is a `machine-wide` cache used to store assemblies that are intended to be shared across several applications on the system.

Q: What is an `interface` in `C#`?

A: An `interface` defines a contract with method and property signatures, but no implementation.

Q: What is the difference between an abstract `class` and an `interface`?

A: An abstract `class` can contain both implementation and abstraction, while an `interface` can only contain method signatures (until `C#` 8 where default implementations are allowed).

Q: What are access modifiers in `C#`?

A: Access modifiers control the visibility of `class` members. Common ones are public, private, protected, internal, and protected internal.

Q: What is a delegate in `C#`?

A: A delegate is a `type-safe` `function` pointer that refers to methods with a specific signature.

Q: What is an event in `C#`?

A: An event is a message sent by an `object` to signal the occurrence of an action, often used with delegates.

Q: What is LINQ?

A: Language Integrated `Query` (LINQ) is a feature that allows querying collections in a `SQL-like` syntax using `C#`.

Q: What is `async/await` in `C#`?

A: The `async` and `await` keywords simplify asynchronous programming by allowing code to be `non-blocking` while maintaining a sequential style.

Q: What is the difference between Task and Thread?

A: A Task is a `higher-level` abstraction for asynchronous work, while Thread represents a `lower-level` system thread.

Q: What is dependency injection in .NET?

A: Dependency Injection is a design pattern used to achieve Inversion of Control (IoC), where dependencies are injected into a `class` rather than created within it.

Q: What is `middleware` in `ASP.NET` Core?

A: `Middleware` is software that is assembled into an application pipeline to handle requests and responses in `ASP.NET` Core.

25%

You're 25% through! Keep going!

Success is built one step at a time.

Q: What is the difference between .NET Core and .NET Standard?

A: .NET Core is a runtime and framework, while .NET Standard is a specification that defines a set of APIs available across all .NET implementations.

Q: What are generics in `C#`?

A: Generics allow you to define `type-safe` data structures without committing to actual data types until the code is instantiated in the runtime.

Q: What is the difference between IEnumerable and IQueryable?

A: IEnumerable executes queries `in-memory` and is suitable for `in-memory` collections. IQueryable executes queries against a data source and supports deferred execution.

Q: What is the difference between `var`, dynamic, and `object` in `C#`?

A: `var` is statically typed and resolved at `compile-time`, dynamic is resolved at runtime, and `object` is the base `class` of all types requiring casting for specific operations.

Q: What is the purpose of `async` streams in `C#`?

A: `Async` streams (using `IAsyncEnumerable<T>`) allow asynchronous iteration over a stream of data using `await foreach`.

Q: What is the difference between `string` and StringBuilder in `C#`?

A: `string` is immutable, so each modification creates a `new` instance. StringBuilder is mutable and more efficient for repeated modifications.

Q: What is reflection in .NET?

A: Reflection is the ability of a program to inspect and modify its own structure and behavior at runtime.

Q: What is the difference between early binding and late binding?

A: Early binding happens at `compile-time`, while late binding occurs at runtime using reflection or the dynamic keyword.

Q: What is the IDisposable `interface` used for?

A: IDisposable is used to release unmanaged resources explicitly via the Dispose method.

Q: How does the 'yield' keyword work in C#?

A: The 'yield' keyword returns each element one at a time in a custom iterator method without creating an entire collection.

Q: What is the difference between == and ReferenceEquals in `C#`?

A: ‘==’ can be overloaded to compare values, while ReferenceEquals checks if two references point to the exact same `object` in memory.

Q: What are attributes in .NET?

A: Attributes are metadata annotations that provide additional information about code elements like classes or methods, used during reflection or by frameworks.

Q: What is dependency injection `middleware` in `ASP.NET` Core?

A: It's part of the `built-in` DI system in `ASP.NET` Core that injects dependencies into classes through `constructor` or method injection at runtime.

Q: What is the difference between transient, scoped, and singleton services in `ASP.NET` Core?

A: Transient: `new` instance every time. Scoped: one per `request`. Singleton: one for the lifetime of the app.

Q: What are filters in `ASP.NET` Core?

A: Filters are `components` that run before or after specific stages in the `request` pipeline, such as authorization, action execution, or result generation.

Q: What is `Model` Binding in `ASP.NET` Core?

A: `Model` Binding maps data from HTTP requests (route, `query string`, form, etc.) to action method parameters or `model` properties.

Q: What is the difference between TempData, ViewData, and ViewBag in `ASP.NET` `MVC`?

A: TempData persists for a single `request` or redirect. ViewData is a dictionary accessible during the current `request`. ViewBag is a dynamic wrapper over ViewData.

Q: What is Kestrel?

A: Kestrel is the `cross-platform` web server used by `ASP.NET` Core to handle HTTP requests directly without relying on IIS or other external servers.

Q: What is SignalR?

A: SignalR is a library for `ASP.NET` that enables `real-time` web functionality using WebSockets or fallback protocols.

Q: What is Entity Framework Core?

A: EF Core is an `open-source` `ORM` that allows developers to work with databases using .NET objects, eliminating the need for most SQL code.

Q: What is the difference between eager loading and lazy loading in EF Core?

A: Eager loading fetches related data immediately with the main `query`. Lazy loading fetches it only when accessed, usually via proxies.

Q: How do you create a `migration` in EF Core?

A: You use the `dotnet ef migrations add <MigrationName>` command to scaffold a `new migration` based on `model` changes.

Q: What is the difference between AddSingleton, AddScoped, and AddTransient in dependency injection?

A: AddSingleton: one instance for app lifetime. AddScoped: one per `request`. AddTransient: `new` instance every time it's requested.

Q: What is the purpose of `appsettings.json` in `ASP.NET` Core?

A: `appsettings.json` stores configuration data such as connection strings, settings, and environment variables in a structured JSON format.

Q: What is the purpose of `Middleware` Order in `ASP.NET` Core?

A: The order determines how requests and responses flow through the pipeline. Earlier `middleware` can `short-circuit` requests or modify responses.

50%

Halfway there! Every expert was once a beginner.

Q: What are tag helpers in `ASP.NET` Core?

A: Tag Helpers `enable` `server-side` code to participate in creating and rendering HTML elements in Razor views.

Q: What is the difference between ConfigureServices and Configure in `Startup.cs`?

A: ConfigureServices is used to register services with the dependency injection container. Configure defines the `middleware` pipeline that handles HTTP requests.

Q: What is the Host in `ASP.NET` Core?

A: The Host is an `object` that encapsulates app resources like DI container, configuration, logging, and `middleware` pipeline. It controls the app lifecycle.

Q: What is IHttpContextAccessor used for?

A: IHttpContextAccessor allows access to the current HttpContext in classes where dependency injection of HttpContext is not directly available.

Q: How do you implement `caching` in `ASP.NET` Core?

A: `Caching` can be implemented using `in-memory` `caching` (IMemoryCache), distributed `caching` (IDistributedCache), or `response` `caching` `middleware`.

Q: What is `Policy-based` Authorization?

A: `Policy-based` authorization allows complex rules to be defined using policies containing requirements, evaluated by handlers.

Q: What is a Hosted `Service` in `ASP.NET` Core?

A: A Hosted `Service` is a background task that runs in the background as long as the application is running, implementing IHostedService or BackgroundService.

Q: What is a CancellationToken and how is it used?

A: CancellationToken is used to propagate cancellation requests to `async` operations or background tasks to stop execution gracefully.

Q: What is the difference between synchronous and asynchronous code?

A: Synchronous code blocks execution until it finishes, while asynchronous code allows the application to continue processing other tasks using `await`.

Q: What is the use of ConfigureAwait(false)?

A: ConfigureAwait(false) avoids capturing the synchronization context, improving performance and preventing deadlocks in certain scenarios.

Q: What are Nullable Reference Types in `C#`?

A: Nullable Reference Types allow you to explicitly indicate whether a reference variable may be `null`, improving `null` safety during `compile-time` checks.

Q: What is the difference between record and `class` in `C#`?

A: Records are reference types like classes but are immutable and provide `built-in` `value-based` equality by default.

Q: What is Minimal `API` in .NET 6+?

A: Minimal APIs provide a lightweight syntax to build HTTP APIs with less boilerplate code compared to traditional controllers in `ASP.NET` Core.

Q: What is source generation in .NET?

A: Source generators are `compile-time` code generation tools that generate `C#` source files to improve performance or `reduce` repetitive code.

Q: How does .NET implement memory management?

A: .NET uses automatic garbage collection to manage memory. It divides heap memory into generations (0, 1, 2) to optimize collection frequency.

Q: What is Span<T> and why is it useful?

A: Span<T> provides a `memory-safe`, fast way to handle slices of arrays or memory regions without allocations, improving performance in `high-throughput` scenarios.

Q: What are value tuples and how are they different from regular tuples?

A: Value tuples are lightweight, mutable structs used to `return` multiple values from methods. Unlike `System.Tuple`, they support deconstruction and named fields.

Q: What are .NET analyzers?

A: .NET analyzers are `Roslyn-based` tools that examine your code during compilation to enforce coding standards or provide `suggestions/errors`.

Q: What is reflection emit?

A: Reflection emit allows dynamic creation of types and methods at runtime, useful for advanced metaprogramming scenarios like dynamic proxies.

Q: What is the difference between Code First and Database First in EF Core?

A: Code First generates the database from code classes. Database First generates code (models and context) from an existing database schema.

Q: What is the Repository pattern and how is it used in .NET?

A: The Repository pattern abstracts data access logic, centralizing queries and interactions with a data store behind a consistent `interface`.

Q: What is the Unit of Work pattern?

A: The Unit of Work pattern groups multiple operations under a single transaction, ensuring either all changes succeed or none are applied.

Q: How do you handle versioning in Web APIs?

A: Web `API` versioning can be handled via URL segments (`e.g`., /v1/), `query` strings, or headers using the `Microsoft.AspNetCore`.`Mvc.Versioning` library.

Q: What is the use of IHttpClientFactory?

A: IHttpClientFactory is used to manage the lifecycle of HttpClient instances, improving performance and reliability of outbound HTTP requests.

Q: What is Blazor?

A: Blazor is a framework for building interactive web UIs using `C#` instead of `JavaScript`, supporting both `client-side` (WebAssembly) and `server-side` hosting models.

75%

You're at 75%! Almost done, push through and finish strong!

Q: What are Razor Pages and how are they different from `MVC`?

A: Razor Pages are a `page-based` programming `model` in `ASP.NET` Core, simplifying the structure by combining `view` and `code-behind` into one file.

Q: How does .NET Core handle `platform-specific` dependencies?

A: .NET Core uses Runtime Identifiers (RIDs) and `runtime-specific` NuGet packages to load `platform-specific` dependencies at runtime.

Q: What is IL (Intermediate Language) in .NET?

A: IL is a `CPU-independent` set of instructions compiled from .NET languages and executed by the CLR using `Just-In-Time` (JIT) compilation.

Q: What is JIT compilation in .NET?

A: JIT (`Just-In-Time`) compilation converts IL code into native machine code at runtime for execution on the host machine.

Q: What is AOT compilation in .NET?

A: AOT (`Ahead-of-Time`) compilation converts IL into native code at build time, improving startup performance and reducing memory usage.

Q: What is the difference between Full AOT and ReadyToRun?

A: Full AOT compiles all code to native at publish time, while ReadyToRun (R2R) precompiles some code, falling back to JIT when needed.

Q: How does gRPC compare to `REST` in .NET?

A: gRPC is faster and uses binary protocol buffers, ideal for internal `microservices`. `REST` is `text-based` and better for public APIs due to wide support.

Q: How does .NET handle multithreading and concurrency?

A: .NET provides the Thread, Task Parallel Library (TPL), `async/await`, and Parallel LINQ (PLINQ) for managing concurrency and multithreading.

Q: What is a thread pool in .NET?

A: The thread pool is a pool of worker threads maintained by .NET to efficiently manage background tasks and avoid the overhead of creating `new` threads.

Q: What is lock and how does it work?

A: lock is a `C#` keyword that ensures that a block of code runs by only one thread at a time, using a monitor under the hood to prevent race conditions.

Q: What are deadlocks and how can they be avoided in .NET?

A: Deadlocks occur when two or more threads wait indefinitely for resources locked by each other. Avoid nested locks and use timeout or ordered locking.

Q: What is memory leak in managed code and how to detect it?

A: Memory leaks occur when references to unused objects persist, preventing GC. Tools like dotMemory and Visual Studio diagnostics help detect leaks.

Q: What is a weak reference in .NET?

A: A weak reference allows the garbage collector to collect an `object` even if a reference to it exists, avoiding memory leaks.

Q: What is Roslyn?

A: Roslyn is the .NET compiler platform that exposes APIs for code analysis, compilation, and refactoring at compile time and runtime.

Q: What is the use of analyzers and code fixes in Roslyn?

A: Analyzers inspect code during compilation and enforce rules. Code fixes suggest automated fixes or improvements to the developer.

Q: What is reflection metadata trimming in .NET?

A: Metadata trimming removes unused metadata at publish time to `reduce` application size, especially important in AOT and Blazor WebAssembly.

Q: How do you secure an `ASP.NET` Core application?

A: Use HTTPS, validate input, use `authentication/authorization` `middleware`, implement `CSRF/XSS` protection, and secure `configuration/secrets` with tools like `Azure` Key Vault.

Q: How do you integrate OAuth2 or OpenID Connect in .NET?

A: Use `middleware` like `AddAuthentication().AddJwtBearer()` or external libraries like IdentityServer, with configuration for authority and client credentials.

Q: What is IdentityServer and how is it used?

A: IdentityServer is an OpenID Connect and `OAuth` `2.0` framework for `ASP.NET` Core used to issue security tokens for `authentication` and authorization.

Q: How do you handle secrets in `production` apps?

A: Use Secret Manager during `development` and environment variables, `Azure` Key Vault, or `AWS` Secrets Manager in `production` to securely store secrets.

Q: What is distributed tracing in `microservices` with .NET?

A: Distributed tracing tracks requests across `microservices` using tools like OpenTelemetry, Application Insights, or Jaeger to monitor system performance and issues.

Q: What is rate limiting and how is it implemented in `ASP.NET` Core?

A: Rate limiting restricts the `number` of requests a client can make. It can be implemented using custom `middleware` or libraries like `AspNetCoreRateLimit`.

Q: What is CQRS and how is it implemented in .NET?

A: CQRS (Command `Query` Responsibility Segregation) separates read and write logic. In .NET, it’s often implemented with MediatR and separate data models.

Q: What is DDD and how does .NET support it?

A: DDD (`Domain-Driven` Design) focuses on complex business logic with domain models. .NET supports DDD with aggregates, repositories, value objects, and bounded contexts.

Q: What are bounded contexts in DDD?

A: A bounded context defines a boundary around a specific domain `model` where terms and rules are consistent, allowing separation between subdomains.

Q: How do you scale `ASP.NET` Core applications?

A: Scale via `load balancing`, horizontal `scaling` (multiple instances), `caching`, database optimization, stateless services, and `containerization` with `Kubernetes` or `Docker`.

Thank You!

You've completed all the questions.

GM SUNSHINE