GUIDANCE 24

Your Success, Our Guidance

100 RAPID-FIRE QUESTIONS WITH ANSWERS IN C# & ASP.NET

WEB FORMS | .NET FRAMEWORK | INTERVIEW GUIDE

ASP.NET

- 100 Most Important Interview Questions
- Concise & Easy to Understand Answers
- Covers Fundamentals to Advanced Topics
- Perfect for Freshers & Experienced Professionals

PRESENTED BY :

SANDEEP T.

Part 1: C# Fundamentals

1. What is C#?

Answer: C# is a modern, object-oriented programming language developed by Microsoft. It is mainly used with the .NET platform to build web, desktop, API and enterprise applications.

2. What are the main features of C#?

Answer: Object-oriented programming, type safety, exception handling, generics, delegates, LINQ, asynchronous programming and automatic garbage collection.

3. What is .NET Framework?

Answer: .NET Framework is a Microsoft software framework used to develop and run Windows and web applications. It provides CLR, class libraries and runtime services.

4. What is CLR?

Answer: CLR stands for Common Language Runtime. It executes managed .NET code and provides services like garbage collection, exception handling, security and memory management.

5. What is CTS?

Answer: CTS stands for Common Type System. It defines how data types are declared and used in .NET so that different .NET languages can work together.

6. What is CLS?

Answer: CLS stands for Common Language Specification. It defines a set of common rules that .NET languages should follow for language interoperability.

7. What is Managed Code?

Answer: Code executed under the control of CLR is called managed code. CLR handles memory management, security and exception handling for it.

8. What is Unmanaged Code?

Answer: Code that runs outside CLR is called unmanaged code. C/C++ native code is a common example.

9. What is JIT Compiler?

Answer: JIT stands for Just-In-Time compiler. It converts Intermediate Language (IL) code into machine code at runtime.

10. Difference between C# and .NET?

Answer: C# is a programming language, whereas .NET is the platform/framework used to develop and execute applications written in C# and other .NET languages.

11. What are Value Types and Reference Types?

Answer: Value types store the actual value, while reference types store a reference to an object in memory. Examples of value types are `int`, `bool`, and `struct`; examples of reference types are `class`, `string`, and arrays.

12. What is Boxing?

Answer: Boxing converts a value type into an object/reference type.

```
int x = 10;
object obj = x;
```

13. What is Unboxing?

Answer: Unboxing converts a boxed object back to its original value type.

```
object obj = 10;
int x = (int)obj;
```

14. Difference between `var`, `dynamic` and `object`?

Answer: `var` is compile-time typed, `dynamic` is resolved at runtime, and `object` is the base type of all .NET types and usually requires casting to access specific members.

15. What is a Constant?

Answer: A constant is a value that cannot be changed after compilation.

```
const double PI = 3.14;
```

16. Difference between `const` and `readonly`?

Answer: `const` is assigned at compile time and cannot be changed. `readonly` can be assigned at declaration or inside a constructor and then cannot be changed.

17. What are access modifiers?

Answer: Access modifiers control the accessibility of classes and members. Common modifiers are `public`, `private`, `protected`, `internal` and `protected internal`.

18. Difference between public, private, protected and internal?

Answer:

- **public:** accessible from anywhere
- **private:** accessible only within the class
- **protected:** accessible within the class and derived classes
- **internal:** accessible within the same assembly

19. What is a Namespace?

Answer: A namespace logically groups related classes and prevents naming conflicts.

```
namespace MyApplication
{
    class Employee { }
}
```

20. What is an Assembly?

Answer: An assembly is a compiled .NET unit containing code, metadata and resources. It is generally an `.exe` or `.dll` file.

□ Part 2: OOPs

21. What is OOP?

Answer: OOP stands for Object-Oriented Programming. It organizes software around objects and classes.

22. What are the four pillars of OOP?

Answer:

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

23. What is Encapsulation?

Answer: Encapsulation means wrapping data and methods together inside a class and controlling access to the data.

24. What is Abstraction?

Answer: Abstraction means hiding implementation details and exposing only the required functionality.

25. What is Inheritance?

Answer: Inheritance allows one class to acquire properties and methods of another class.

```
class Employee : Person
{
}
```

26. What is Polymorphism?

Answer: Polymorphism means "one interface, multiple implementations." In C#, it can be achieved through method overloading and overriding.

27. Abstraction vs Encapsulation?

Answer: Abstraction focuses on **what an object does**, while encapsulation focuses on **how data and implementation are protected**.

28. What is a Class?

Answer: A class is a blueprint or template used to create objects.

29. What is an Object?

Answer: An object is an instance of a class.

```
Employee emp = new Employee();
```

30. What is a Constructor?

Answer: A constructor is a special method automatically called when an object is created. It has the same name as the class and has no return type.

31. Can a class have multiple constructors?

Answer: Yes. This is called constructor overloading.

32. What is a Static Constructor?

Answer: A static constructor initializes static members and is executed automatically only once for a type.

33. What is a Destructor?

Answer: A destructor is used for cleanup before an object is removed by the garbage collector. In C#, it is written using `~ClassName()`.

34. What is Method Overloading?

Answer: Defining multiple methods with the same name but different parameters is called method overloading.

35. What is Method Overriding?

Answer: Overriding means providing a new implementation of a base class virtual or abstract method in a derived class.

36. Overloading vs Overriding?

Answer: Overloading is compile-time polymorphism; overriding is runtime polymorphism.

37. What is `virtual`?

Answer: `virtual` allows a derived class to override a method of the base class.

38. What is `override`?

Answer: `override` provides a new implementation of a virtual or abstract base-class method.

39. What is Method Hiding?

Answer: Method hiding uses the `new` keyword to hide a base class member.

40. What is an Abstract Class?

Answer: An abstract class cannot be instantiated directly. It can contain both abstract and non-abstract methods.

41. What is an Interface?

Answer: An interface defines a contract that implementing classes must follow.

```
interface IEmployee
{
    void Save();
}
```

42. Abstract Class vs Interface?

Answer: An abstract class can contain implementation, fields and constructors, whereas an interface primarily defines a contract. A class can implement multiple interfaces.

43. Can an interface contain fields?

Answer: Traditionally, interfaces don't contain instance fields. Modern C# interfaces can contain certain members with implementations, but they still don't work like normal class storage fields.

44. Does C# support multiple inheritance?

Answer: C# does not support multiple inheritance between classes.

45. How can multiple inheritance be achieved?

Answer: By implementing multiple interfaces.

```
class Employee : IEmployee, IManager
{
}
```

□ Part 3: Advanced C#

46. What is a Delegate?

Answer: A delegate is a type-safe reference to a method. It is commonly used for callbacks and events.

47. What is a Multicast Delegate?

Answer: A multicast delegate can reference multiple methods and invoke them sequentially.

48. What is an Event?

Answer: An event provides a mechanism for a class to notify other classes when something happens.

49. Delegate vs Event?

Answer: A delegate can generally be invoked by its owner/reference holder, whereas an event restricts invocation to the class that declares it.

50. What is a Lambda Expression?

Answer: A lambda is a concise way of writing an anonymous function.

```
x => x * 2
```

51. What is LINQ?

Answer: LINQ stands for Language Integrated Query. It allows querying collections, objects, databases and other data sources using C# syntax.

52. What is LINQ to Objects?

Answer: LINQ used to query in-memory collections such as `List<T>`, arrays and dictionaries is called LINQ to Objects.

53. What is LINQ to SQL?

Answer: LINQ to SQL is a technology that allows querying SQL Server data using LINQ syntax and mapped objects.

54. `IEnumerable` VS `IQueryable`?

Answer: `IEnumerable` is generally used for in-memory iteration, while `IQueryable` can build an expression that a remote provider, such as a database provider, can translate and execute.

55. What are Generics?

Answer: Generics allow classes, methods and collections to work with different data types while maintaining type safety.

```
List<int> numbers = new List<int>();
```

56. Why use Generics?

Answer: They provide type safety, code reusability and reduce unnecessary boxing/unboxing.

57. What is a Generic Class?

Answer: A class that accepts a type parameter is called a generic class.

```
class Repository<T>
{
}
```

58. What is a Generic Method?

Answer: A method that accepts a type parameter is called a generic method.

59. What is an Extension Method?

Answer: An extension method allows us to add methods to an existing type without modifying the original type.

```
public static bool IsValid(this string value)
{
    return !string.IsNullOrEmpty(value);
}
```

60. What is a Nullable Type?

Answer: Nullable types allow value types to contain `null`.

```
int? age = null;
```

61. Difference between == and `.Equals()`?

Answer: == is an operator whose behavior depends on the type/operator overload, while `.Equals()` is a method used for equality comparison.

62. String vs StringBuilder?

Answer: `string` is immutable, so repeated modifications can create new objects. `StringBuilder` is mutable and is better for many string modifications.

63. Why is String immutable?

Answer: String immutability provides safety and predictable behavior and makes strings suitable for interning and hashing. Any modification creates a new string.

64. What is Exception Handling?

Answer: Exception handling manages runtime errors using `try`, `catch`, `finally` and `throw`.

65. Difference between `throw` and `throw ex`?

Answer: Inside a catch block, `throw` preserves the original stack trace, while `throw ex` resets the stack-trace information from that point.

66. What is `try-catch-finally`?

Answer: `try` contains risky code, `catch` handles exceptions and `finally` executes cleanup code whether an exception occurs or not.

67. Can we have multiple catch blocks?

Answer: Yes. Multiple catch blocks can handle different exception types.

68. What is a Custom Exception?

Answer: A custom exception is an application-specific exception class derived from `Exception`.

69. What is Garbage Collection?

Answer: Garbage Collection automatically identifies and releases memory occupied by objects that are no longer reachable.

70. What are GC Generations?

Answer: .NET garbage collection groups managed objects into generations, mainly Gen 0, Gen 1 and Gen 2, to optimize memory collection.

□ Part 4: ASP.NET Web Forms

71. What is ASP.NET?

Answer: ASP.NET is Microsoft's web application framework for building dynamic web applications and services using .NET.

72. What is ASP.NET Web Forms?

Answer: Web Forms is an ASP.NET framework based on pages, server controls, events and postbacks. It uses `.aspx` pages and code-behind files.

73. Web Forms vs MVC?

Answer: Web Forms uses page/control-based event-driven development and ViewState, while MVC separates the application into Model, View and Controller and provides more direct control over HTTP and HTML.

74. What is Page Life Cycle?

Answer: The Web Forms page lifecycle includes stages such as:

Init → Load → Postback Events → PreRender → SaveStateComplete → Render → Unload

75. What is `Page_Init`?

Answer: `Page_Init` occurs during initialization. It is commonly used for initializing controls and early setup.

76. What is `Page_Load`?

Answer: `Page_Load` occurs when the page is loaded. It is commonly used to load data and perform page-level operations.

77. What is `Page_PreRender`?

Answer: `PreRender` occurs just before the page is rendered. It is useful for making final changes to controls or data.

78. What is `IsPostBack`?

Answer: `IsPostBack` tells whether the page is being loaded for the first time or as a result of a postback.

```
if (!IsPostBack)
{
    BindData();
```

```
}
```

79. What is PostBack?

Answer: PostBack occurs when a Web Forms page sends a request back to the server for processing.

80. What is ViewState?

Answer: ViewState stores page/control state between postbacks, generally in a hidden field on the page.

```
ViewState["UserID"] = 10;
```

81. What is ControlState?

Answer: ControlState stores essential control information that must be maintained across postbacks even when ViewState is disabled.

82. ViewState vs Session?

Answer: ViewState is maintained for a page/control and travels with the page, while Session stores user-specific data on the server.

83. What is Session State?

Answer: Session stores data associated with a particular user's session.

```
Session["UserID"] = 101;
```

84. What is Application State?

Answer: Application State stores data that is shared across all users of the application.

```
Application["CompanyName"] = "ABC";
```

85. What is Cache?

Answer: ASP.NET Cache temporarily stores frequently used data or generated content to improve application performance.

86. Session vs ViewState vs Cache vs Application?

Feature	Scope	Typical Storage
ViewState	Page/control	Client
Session	Individual user	Server/state provider
Application	All users	Server
Cache	Shared/cacheable data	Server

87. What is `web.config`?

Answer: `web.config` contains configuration settings for an ASP.NET application, such as connection strings, authentication, compilation and custom settings.

88. What is `machine.config`?

Answer: `machine.config` contains machine-level .NET configuration settings and acts as a higher-level configuration file.

89. What is Global.asax?

Answer: Global.asax contains application-level and session-level events such as `Application_Start`, `Application_End`, `Session_Start` and `Session_End`.

90. What is Master Page?

Answer: A Master Page provides a common layout for multiple ASP.NET Web Forms pages.

Example:

```
Master Page
 ├── Header
 ├── Menu
 ├── ContentPlaceHolder
 └── Footer
```

91. What is User Control?

Answer: A User Control is a reusable UI component created using an `.ascx` file.

92. What is a Server Control?

Answer: A server control is an ASP.NET control that runs on the server and can be accessed programmatically from code-behind.

```
<asp:Button ID="btnSave" runat="server" Text="Save" />
```

93. HTML Server Control vs Web Server Control?

Answer: HTML server controls are standard HTML elements with `runat="server"`, while Web Server Controls are ASP.NET controls such as `GridView`, `TextBox` and `Button` with richer server-side functionality.

94. What are Validation Controls?

Answer: ASP.NET validation controls validate user input on the client and/or server.

Examples:

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator
- ValidationSummary

95. What is RequiredFieldValidator?

Answer: It ensures that a required field is not left empty.

96. What is Authentication?

Answer: Authentication verifies **who the user is**.

Example: Login using username and password.

97. What is Authorization?

Answer: Authorization determines **what an authenticated user is allowed to access**.

Example: Admin can access Admin Panel, while a normal user cannot.

98. What is Forms Authentication?

Answer: Forms Authentication is an ASP.NET authentication mechanism where a user's authenticated state is maintained using an authentication ticket, typically represented by a cookie.

99. What is Role-Based Authorization?

Answer: Role-based authorization restricts access according to a user's role.

```
Admin       → Full Access
Manager     → Reports + Management
User        → Limited Access
```

100. How do you improve ASP.NET application performance and security?

Answer: I would use caching, optimized SQL queries, proper indexing, connection pooling, pagination, asynchronous operations where appropriate, compression, validation, parameterized queries, HTTPS, secure authentication/authorization, proper session management and avoid exposing sensitive information.