108 .NET Core interview questions to hire top developers

Questions

1. What is .NET Core, and why should we use it instead of older .NET Framework?

2. Can you explain the difference between .NET Core and .NET Standard?

3. What are the key benefits of using .NET Core for building applications?

4. What's the role of the .NET CLI (Command Line Interface), and what are some common commands you'd use?

5. Explain what a 'cross-platform' application is, and how .NET Core enables it.

6. What is a NuGet package, and why are they important in .NET Core development?

7. How do you create a new .NET Core project using the command line?

8. What is the purpose of the 'Program.cs' file in a .NET Core console application?

9. What's the difference between 'build' and 'run' commands in .NET Core?

10. How can you add a NuGet package to your .NET Core project?

11. What is the purpose of the 'csproj' file in a .NET Core project?

12. Explain what environment variables are and how they are used in .NET Core applications.

13. How do you handle configuration settings in a .NET Core application (e.g., connection strings)?

14. What is dependency injection (DI), and why is it used in .NET Core?

15. How do you register services for dependency injection in .NET Core?

16. Can you explain the concept of middleware in .NET Core, especially within ASP.NET Core?

17. What are some common middleware components used in ASP.NET Core applications?

18. How do you create a simple API endpoint in ASP.NET Core?

19. What are HTTP request methods (like GET, POST, PUT, DELETE), and how are they used in APIs?

20. How do you handle different HTTP status codes in your API responses?

21. What is JSON, and why is it commonly used in APIs?

22. How do you serialize and deserialize JSON data in .NET Core?

23. What is exception handling, and how do you implement it in .NET Core?

24. How do you write unit tests for your .NET Core code?

25. What are some common unit testing frameworks used with .NET Core (e.g., xUnit, NUnit)?

26. What is source control, and why is it important when working on .NET Core projects?

27. Explain the basic Git commands like 'clone', 'commit', 'push', and 'pull'.

28. How would you debug a .NET Core application running in a Docker container?

29. What are some of the key differences between .NET Core and .NET 5/6/7/8?

30. What is .NET Core, and why is it special compared to the older .NET Framework?

31. Can you explain what a 'namespace' is in C# and why we use them?

32. What's the difference between 'int' and 'string' in C#, and when would you use each?

33. Imagine you're explaining 'variables' to a friend. How would you describe what they are and what they do?

34. What is an 'if' statement, and how does it help your program make decisions?

35. What is a 'loop', and why is it useful when you want to do something many times?

36. Can you explain what an 'array' is and how it helps you store multiple things?

37. What is a 'method' in C#, and why do we use them to organize our code?

38. What does 'debugging' mean, and what's one simple way you can try to find mistakes in your code?

39. What is 'source control' (like Git), and why is it important for working on projects with others?

40. What is the difference between `Console.WriteLine()` and `Console.ReadLine()`?

41. Explain the difference between a class and an object in C#?

42. What is the purpose of using comments in code, and how can they help you and others?

43. What are some basic data types you know in C# other than `int` and `string`?

44. How do you create a simple 'Hello, World!' program in .NET Core?

45. Describe the difference between `public` and `private` access modifiers.

46. Can you describe what you understand about API's and how they can be used in .NET Core?

47. What is NuGet, and why is it important in .NET Core development?

48. How would you handle a simple error in a .NET Core application?

49. Explain what you know about testing in .NET Core (e.g., unit testing).

50. What are some benefits of using .NET Core for web development?

51. Describe what you understand about dependency injection in .NET Core.

52. What are environment variables and how might you use them in a .NET Core application?

53. If you had a list of numbers, how could you find the largest number using C#?

54. Explain what you know about LINQ and how it is useful.

55. What is JSON, and why is it used in .NET Core applications?

56. How does .NET Core handle different operating systems (like Windows, macOS, and Linux)?

57. What are some tools or IDEs (like Visual Studio Code) that you can use for .NET Core development?

58. Imagine your program isn't working as expected. Walk me through your process of fixing it.

59. How does .NET Core's dependency injection work, and why is it useful?

60. Explain the difference between `Task.Run` and `Task.Factory.StartNew` in .NET Core.

61. What are middleware components in ASP.NET Core, and how are they used in the request pipeline?

62. How does .NET Core handle configuration, and what are some common configuration sources?

63. Describe the purpose of Kestrel in ASP.NET Core.

64. How can you implement logging in a .NET Core application, and what are some best practices?

65. What is the role of the `dotnet` CLI, and what are some common commands?

66. Explain the difference between `ProjectReference`, `PackageReference`, and `ProjectReference` in a .NET Core project file.

67. How does .NET Core support cross-platform development, and what are some considerations?

68. What is the purpose of the `IConfiguration` interface, and how is it used?

69. Explain how you would implement exception handling in an ASP.NET Core application.

70. How can you secure a .NET Core web API?

71. What are Razor Pages in ASP.NET Core, and how do they differ from traditional MVC?

72. How does .NET Core handle memory management, and what are some common memory leaks?

73. Explain the concept of 'options pattern' in .NET Core.

74. How would you implement unit testing in a .NET Core project, and what are some common testing frameworks?

75. What is the purpose of the `IHttpClientFactory` interface in .NET Core?

76. How can you publish a .NET Core application to different platforms?

77. Describe the role of environment variables in a .NET Core application.

78. How do you implement caching in .NET Core to improve performance?

79. How does .NET Core's garbage collector handle large object heaps, and what strategies can you employ to minimize fragmentation in these heaps?

80. Describe the differences between `Task.Run` and `Task.Factory.StartNew` in .NET Core, focusing on their thread pool usage and potential performance implications.

81. Explain the concept of middleware in ASP.NET Core and how you would implement custom middleware to handle authentication or request logging.

82. How can you implement efficient caching strategies in .NET Core applications, considering both in-memory and distributed caching options?

83. Discuss the role of dependency injection in .NET Core and how it promotes loose coupling and testability in your applications.

84. Explain how to use health checks in ASP.NET Core to monitor the status of your application and its dependencies.

85. What are the benefits of using Kestrel as a web server in .NET Core, and when might you choose to use a reverse proxy like Nginx or IIS in front of it?

86. Describe how you would handle errors and exceptions in a .NET Core application, including global exception handling and logging strategies.

87. How can you secure a .NET Core API using JWT (JSON Web Tokens), and what are some best practices for storing and managing tokens?

88. Explain the concept of asynchronous programming in .NET Core using `async` and `await`, and how it improves the responsiveness of your applications.

89. Discuss the advantages of using gRPC over REST APIs in .NET Core, considering factors like performance, efficiency, and code generation.

90. How would you implement a message queue using .NET Core, and what are some popular message queue technologies you might consider (e.g., RabbitMQ, Kafka)?

91. Explain how to use the .NET Core CLI for building, testing, and deploying your applications, and what are some common CLI commands you use?

92. Describe the different hosting models available in ASP.NET Core (e.g., InProcess, OutOfProcess) and their implications on performance and deployment.

93. How do you optimize performance in .NET Core applications, specifically regarding memory allocation and CPU usage?

94. What are the various ways to configure a .NET Core application, and how do you manage environment-specific configurations?

95. How does the .NET Core runtime handle cross-platform compatibility, and what are some potential challenges when deploying to different operating systems?

96. What are `Span<T>` and `Memory<T>` in .NET Core, and what problems do they solve when dealing with memory management and data processing?

97. Describe the role of logging frameworks (e.g., Serilog, NLog) in .NET Core, and how you would configure structured logging in your applications.

98. How do you implement unit testing and integration testing in .NET Core, and what testing frameworks do you prefer (e.g., xUnit, NUnit)?

99. Explain the concept of minimal APIs in .NET 6+ and how they differ from traditional controller-based APIs.

100. Describe how you would implement background tasks in a .NET Core application, considering options like `IHostedService` and Hangfire.

101. How can you use reflection in .NET Core to dynamically inspect and manipulate types and members at runtime?

102. Discuss the various options for deploying .NET Core applications to cloud platforms like Azure and AWS.

103. How would you implement a CQRS (Command Query Responsibility Segregation) pattern in a .NET Core application?

104. Explain the purpose of the `ConfigureAwait(false)` method in asynchronous .NET Core code.

105. Describe the differences between authorization and authentication in ASP.NET Core and how to implement custom authorization policies.