

Low Level

1. Erklären Sie die Begriffe BCL, CLI, CLR, CTS, FCL und JIT und wie diese zueinander stehen.
Tipp: Verwenden Sie Boxen um die Beziehung zwischen den Elementen zu erklären.
2. Was ist die sogenannte Common Intermediate Language (CIL)? Wieso kann man .NET mit verschiedenen Hochsprachen programmieren?
Bereiten Sie ein Beispiel vor, d. h., zeigen Sie, wie aus einer Hochsprache wie C++.NET oder C# die CIL wird.
3. Was ist die Common Language Runtime (CLR)? Wo sind die Unterschiede zu COM, DCOM, COM+ und Java?
Gehen Sie dazu auf technologische Besonderheiten wie die Garbage Collection oder Plattformunabhängigkeit ein!
4. Was ist eine Assembly? Was ist dagegen eine Satellitenassembly?
Enthält eine mit C++.NET (unverwaltet/unmanaged) entwickelte Assembly Maschinencode?
Was ist der Global Assembly Cache (GAC)? Was sind die Vor- und Nachteile vom GAC bzw. sogenannten private / shared Assemblys?
5. Was ist unverwaltet / verwalteter Code? Wer oder was führt ihn aus?
Tipp: Erklären Sie, wie aus einer Hochsprache die Intermediate Language wird (wer kompiliert, was wann) und wer sie ausführt. Gehen Sie auf Merkmale wie Typsicherheit ein. Versuchen Sie zu klären, warum beim ersten Ausführen einer Anwendung diese in der Regel „etwas Zeit benötigt“. Klären Sie dann, wo der Cache für liegt, der zukünftige Starts beschleunigt.

Mid-Level

6. Welche unterschiedlichen Frameworks gibt es bei .NET?
Gehen Sie z. B. auf Silverlight, .NET Micro Framework, .NET Compact Framework, Rotor und Mono ein.
7. Ordnen Sie die zum .NET Framework 2.0 hinzugekommen Frameworks Windows Communication Foundation, Windows Presentation Foundation und Windows Workflow Foundation sowie CardSpace technologisch ein.
Tipp: Geben Sie einen Überblick über die Funktionalitäten. Klären Sie, warum diese Frameworks nicht einfach in das .NET Framework „hinein gepackt worden“.
8. Was ist beim .NET Framework die Attributisierung wie z. B. [WebMethod]. Klären Sie dahinter das technologische Konzept, in dem Sie auf den Sinn, die Verarbeitung und drei Beispiele eingehen. Tipp: Klären Sie, wer die Attribute auswertet (Präprozessor, Laufzeitumgebung, Visual Studio etc.). Für ein Beispiel könnten Sie [Browseable] verwenden. Für ein weiteres die sogenannte bedingte Kompilierung mit einem bestimmten Attribut aus System.Diagnostics.
9. Was ist der Unterschied zwischen einem Verweis- und einem Werttyp?
Tipp: Gehen Sie auf Stack, Heap, Stack beim .NET Framework bzw. der Common Language Runtime ein. Zeigen Sie, wo die jeweiligen Werte „liegen“, insbesondere auch, wo sie

physikalisch liegen (Register, Hauptspeicher, etc.?)! Worin unterscheiden sich in diesem Zusammenhang Strukturen (struct) von Klassen (class)? Bringen Sie Beispiele warum es diesen Unterschied zwischen einem Verweis und einem Werttyp gibt. Denken Sie an Boxing, Casting, Call By Reference, Call By Value, out, ref etc.

10. Was ist System.Diagnostics und warum sollte man es bei einer professionellen Entwicklung verwenden?

Tipp: Klären Sie, ob man bei einem Web Service eine MessageBox für den Entwickler mit einer Ausnahme/Exception anzeigen kann? Wie würden Sie es gegebenenfalls ohne MessageBox machen? Wie könnte man Programmausgaben über eine Konfigurationsdatei einstellen (app.config)? Zeigen Sie Beispiele, wie sich Prozesse starten lassen und bedingte Kompilierung läuft.

11. Wo liegen die Unterschiede zwischen COM/DCOM/COM+ zu .NET? Welche Vorteile bringt es mit sich .NET anstelle der anderen Technologien einzusetzen?

Tipp: Klären Sie, welche essentiellen Konzepte das .NET Framework für verteilte Anwendungssysteme bereit hält.

High Level

12. Klären Sie was Delegates sind, wozu sie technologisch benötigt werden. Bereiten Sie zwei sinnvolle Beispiele zu deren Verwendung vor.

Klären Sie dann, was Multi-Delegates sind und zeigen ein Beispiel.

13. Gibt es beim .NET Framework Schutz des geistigen Eigentums?

Tipp: Gehen Sie auf starke / schwache Namen ein, auf Disassemblierung, Reengineering. Sie könnten zeigen, ob sich Assemblies mit starkem Namen patchen lassen.

14. Was sind die Sicherheitsaspekte beim .NET Framework?

Tipp: Gehen Sie auf die Code Access Security, AppDomains, etc. ein.

15. Wie Interoperable ist .NET-Code bzw. das .NET Framework?

Tipp: Gehen Sie auf P/Invoke, COM/Interop, .NET Remoting, RPC, .NET Enterprise Services/COM+ ein. Klären Sie auch die Frage, wie z. B. COM+-Komponenten per Windows Communication Foundation (WCF) in die .NET Framework „geholt“ werden können – ohne sie „anzufassen“.

16. Was ist Reflection? Wozu benötigt man dieses Konzept? Ist es schnell, langsam? Bringen Sie fünf sinnvolle Beispiele! Gibt es bereits Werkzeuge für Entwickler, die Reflection nutzen?

Tipp: Zeigen Sie, wie z. B. der Anwendungspfad („Ausführen in“) oder die Build-Nummer einer Assembly ermittelt werden kann. Sie könnten ein Beispiel bringen, wie ermittelt werden kann, was der aktuelle Methodenname ist und wie der Stack aussieht.

17. Ist das .NET Framework eigentlich nur ein „Wrapper“ für das jeweilige Betriebssystem? Mit anderen Worten, ist der Common Intermediate Language Code der Maschinencode, oder ruft dieser Code nur Betriebssystem-Funktionen auf? Erklären Sie, was z. B. bei MessageBox.Show(„Foo“) passiert!

Tipp: Versuchen Sie zu klären, ob beim .NET Framework und bei Mono Betriebssystem-

Funktionen aufgerufen werden, gegebenenfalls nachimplementiert werden oder Maschinencode generiert wird.

18. Was sind und wie funktionieren Generics? Bringen Sie drei sinnvolle Beispiele, eines davon, bei einer Methode einer Klasse (Rückgabe und / oder übergebene Parameter sind generisch). Tipp: Wo sind die Unterschiede zu Templates bei C++? Werden bei Generics die „Slots“ durch Visual Studio, dem Compiler oder dem JIT-Compiler mit dem jeweiligen konkreten Typ ausgeprägt („der Typ eingesetzt“).