

## Einfach

1. Was ist *Continuous Integration* (CI)? Welchen Bezug haben CI und *Source Control Management* (SCM) zueinander? Beispiel: Automatisches Vergeben von *Source Control-Tags* für erfolgreiche Release Builds.
2. Was ist ein *Build Script*? Durch wen, wo und wann wird es ausgeführt?
3. Welche Vorteile hat die Erstellung von Visual Studio-Projekten unabhängigen *Build Scripts*?
4. Was sollte man bei der Erstellung des *Source Trees* für ein Projekt beachten, um einen *Source Tree* zu erhalten der auf jeder Arbeitsstation ohne die Installation zusätzlicher Tools kompiliert werden kann? Welche Vorteile für die Entwicklung im Team ergeben sich daraus? Hinweis: <http://xrl.in/1sqe>
5. Gibt es sinnvolle bzw. bewährte Vorlagen / Strukturen für *Source Trees*? Recherchieren Sie Tools zur Erstellung von *Source Trees*. Hinweis: Prüfen Sie einige Open Source-.NET Projekte, z. B. auf *GitHub* oder *Google Code*.
6. Was ist *MSBuild* und wie findet es in Visual Studio Verwendung?
7. Was ist bei *MSBuild* unter *ItemGroups*, *PropertyGroups*, *Tasks* und *Targets* zu verstehen? Erstellen Sie eine Übersicht in Form einer Grafik wie diese Elemente zueinander stehen.
8. Was ist *NAnt* und *Rake*? Worin liegen die Unterschiede zwischen *NAnt* bzw. *MSBuild* und *Rake*? Hinweis: Mit welchem Ziel wurde *Rake* entwickelt, worin liegt bspw. der Nachteil großer XML-Dateien?
9. Was versteht man unter einem *Build Server* und welche Aufgaben hat dieser im Rahmen von *Continuous Integration*?

## Mittel

10. Was sind wesentliche Unterschiede sowie je Vor- und Nachteile von *MSBuild* und *NAnt*? Wie werden die in Frage 7 erwähnten Elemente von *MSBuild* in *NAnt* abgebildet? Was sind *NAnt*-Funktionen?
11. Aus welchen Schritten besteht ein CI-Vorgang üblicherweise? Wodurch wird er gestartet?
12. Welche Tools und Frameworks können im automatisierten CI-Prozess Einsatz finden? Mögliche Beispiele: Unit Test Runner, Dokumentation, statische Codeanalyse, Source Code-Analyse (normierte Programmierung).
13. Welcher wirtschaftliche Nutzen lässt sich durch den Einsatz von CI generieren?
14. Welche Möglichkeiten gibt es, die APIs einer Anwendung zu dokumentieren? Inwieweit dient dieses den Entwicklern?
15. Recherchieren Sie Produkte die als *Build Server* fungieren können und gehen Sie auf drei signifikante Unterschiede wie z. B. den Preis ein. Hinweis: <http://xrl.in/1sqp>

## Herausforderung

16. In welche Typen kann man CI-Vorgänge unterteilen? Kategorisieren Sie dabei bspw. über

- die Zeitdauer bis zur Rückmeldung über Erfolg oder Fehler,
- die Tiefe der Integration mit Drittsystemen,
- die Art des Zielsystems bzw. -plattform und
- die zeitliche Planung des Vorgangs.

17. Der *CruiseControl.NET Server* bietet die verschiedensten Möglichkeiten der Konfiguration. Beispielsweise können durch Projektkonfigurationen unterschiedliche Builds erzeugt werden. Die gesamte Konfiguration findet in *ccnet.config*, einer XML-Datei, statt. Welche Konfigurationsmöglichkeiten sind dort möglich? Gibt es Tools die die Handhabung der Konfiguration vereinfachen?

18. Wie kann man mit *TeamCity* CI-Vorgänge auf mehrere Rechner verteilen? Kann man über den gleichen Mechanismus Builds ein und desselben Produkts für verschiedene Versionen des .NET Frameworks realisieren?

19. Welche statistischen Auswertungen sind mithilfe von *TeamCity* möglich? Wie können benutzerdefinierte Werte verwendet werden und wozu?

20. Im Anhang befindet sich eine Taschenrechner-Anwendung. Erstellen Sie ein Build Script welches Sie über die Eingabeaufforderung starten können. Aufgabe des Scripts soll es sein, die unter Aufgabe 7 erläuterten Elemente am praktisch vorzuführen.