

A

- abstract test driver class pattern 144–145
- abstract test infrastructure class pattern 137–140
- acceptance testing
 - Cucumber 251
 - FitNesse 250
 - overview 250
 - SpecFlow 251
 - TickSpec 251
 - using before refactoring legacy code 216
- action-driven testing 76
- actions, separating from asserts 183–184
- Add() method 44
- agent of change
 - choosing smaller teams 191
 - creating subteams 192
 - identifying blockers 191
 - identifying champions 190–191
 - identifying possible entry points 191
 - pilot project feasibility 192
 - preparing for tough questions 190
 - using code reviews as teaching tool 192
- AlwaysValidFakeExtensionManager class 56
- AnalyzedOutput class 178
- AnalyzeFile method 181
- antipatterns, in isolation frameworks
 - complex syntax 120–121
 - concept confusion 118–119
 - record and replay style 119–120
 - sticky behavior 120
- API for tests
 - AutoFixture helper API 242–243
 - documenting 149–150
 - Fluent Assertions helper API 243
 - MSTest API
 - extensibility 241–242
 - lack of Assert.Throws 242
 - overview 241
 - MSTest for Metro Apps 242
 - NUnit API 242
 - overview 241
 - SharpTestsEx helper API 243
 - Shouldly helper API 243
 - test class inheritance patterns
 - abstract test driver class pattern 144–145
 - abstract test infrastructure class pattern 137–140
 - overview 136–137
 - refactoring for test class hierarchy 146–147
 - template test class pattern 140–144
 - using generics 147–148
 - utility classes and methods 148
 - when to change tests 154–155
 - xUnit.NET 242–243
- Arg class 97
- ArgumentException 37
- arguments, ignoring by default 115–116
- arrange-act-assert 93, 95–96
- Assert class 28–29
- asserts
 - avoiding custom assert messages 182–183
 - avoiding multiple on different concerns
 - overview 174–175
 - using parameterized tests 175–176
 - wrapping with try-catch 176
 - separating from actions 183–184
- Assert.Throws function 38, 242
- attributes, NUnit 27
- Autofac 60, 245
- AutoFixture helper API 242–243