Enhetstester och Test Doubles

Av: Niklas Hjelm

Enhetstestning

- Enhetstestning innebär att testa minsta möjliga delar av koden, vanligtvis metoder eller funktioner, isolerat från resten av systemet.
- Målet är att verifiera att varje del fungerar som förväntat.

Arrange, Act, Assert (AAA) Pattern

Detta är en standardstruktur för att skriva enhetstester.

- **Arrange:** Sätt upp testförhållandena. Skapa objekt, mocka beroenden, och förbered data som behövs för testet.
- **Act:** Utför handlingen som ska testas, oftast genom att anropa en metod.
- **Assert:** Kontrollera att handlingen gav det förväntade resultatet.

```
//Arrange

Här skrivs all kod som behövs
för att testa en viss funktion.

//Act

Här skrivs koden som testar funktionen.

//Assert

Här skrivs koden som kontrollerar
att funktionen fungerar som den ska.
```

Test Isolation

- Enhetstester bör vara isolerade, vilket innebär att de inte ska bero på externa resurser som databaser, filsystem eller andra tester.
- Isolation säkerställer att testerna är stabila och körs konsekvent.
- Detta är viktigt för att säkerställa att vi testar rätt saker.

Test Doubles

- Test doubles används för att uppnå ”Test Isolation”
- En Test Double är ett ”spök-objekt” som vi kan konfigurera att fungera på ett förväntat sätt
- Det finns framförallt 5 olika typer av Test Doubles
  - Mock
  - Fake
  - Spy
  - Stub
  - Dummy

Mock

- **Användning**: Används för att simulera beteenden av verkliga objekt. De är användbara när du vill verifiera att vissa metoder har anropats eller att vissa interaktioner har ägt rum.
- **Egenskaper**: Mock-objekt kan ha förväntningar, som att en viss metod ska anropas ett visst antal gånger. De kan också kasta undantag för att simulera fel.
- **Exempel**: Verifiera att en metod som skickar e-post har anropats i en tjänst.

Fake

- **Användning**: Enklare och ofta handskrivna versioner av komponenter som ersätter mer komplexa, verkliga implementationer.
- **Egenskaper**: Fakes har arbetsfunktionalitet, men den är vanligtvis förenklad.
  - Till exempel kan en fake databas använda en enkel lista i minnet istället för att ansluta till en riktig databas.
- **Exempel**: En in-memory databas som används för att testa dataåtkomstlogik.

Spy

- **Användning**: Används för att inspektera hur vissa delar av koden används under testning.
  - Till skillnad från mocks, som bara verifierar att något hände, kan spies också rapportera tillbaka om hur det hände.
- **Egenskaper**: Spies kan registrera information om hur de har blivit anropade, som vilka argument som användes eller hur många gånger de anropades.
- **Exempel**: Registrera hur många gånger en metod anropas för att säkerställa att en cache-mekanism fungerar som förväntat.

Stub

- **Användning**: Används för att ge förutbestämda svar på anrop.
  - De är användbara när du behöver styra ett tests beteende genom att returnera specifika värden från anrop.
- **Egenskaper**: Stubs är passiva och används bara för att "stubba ut" vissa delar av koden som inte är relevanta för det aktuella testet.
- **Exempel**: Returnera en specifik användarinformation när en metod som hämtar data från en användardatabas anropas

Dummy

- **Användning**: Används som platsinnehållare för att uppfylla en metodsignaturs parameterkrav när den faktiska parametern inte är relevant för testet.
- **Egenskaper**: Dummies har ingen funktionalitet och används inte alls i själva testet; de är bara där för att testet ska kunna köras.
- **Exempel**: Skicka in ett objekt som inte används bara för att uppfylla en metodsignatur.

Olika ramverk

Test Doubles

- Moq
- NSubstitute
- FakeItEasy

Unit Tests

- xUnit
- NUnit
- MSTest

xUnit och FakeItEasy

xUnit

- xUnit och NUnit är de två absolut största tredjepartsramverken för enhetstestning i .NET.
- Både xUnit och NUnit har inbyggda mallar i VS 2022 för att sätta upp test-projekt för .net-projekt
- xUnit och NUnit använder något olika syntax men är i stora drag väldigt lika.

Fact

Användning:

- En Fact är en metod som representerar ett enskilt testfall i xUnit.
- Det är ett test som alltid har samma indata och förväntas alltid ge samma resultat.

Syntax:

- Annoteras med [Fact] attributet.

Exempel:

- En metod som testar om en adderingsfunktion returnerar korrekt summa för två givna tal.

Theory

Användning:

- En Theory är i xUnit en metod som representerar ett testfall som ska köras med flera olika indata.

Syntax:

- Annoteras med [Theory] attributet och kombineras ofta med [InlineData], [ClassData], eller [MemberData] för att tillhandahålla testdata.

Exempel:

- En metod som testar en adderingsfunktion med olika par av tal för att säkerställa att den fungerar korrekt i flera scenarier.

InlineData

Användning:

- Används tillsammans med Theory för att tillhandahålla konkreta värden för testparametrar.

Syntax:

- Annoteras med [InlineData] följt av värdena som ska användas.

Exempel:

- Används för att ge olika par av tal till en testmetod som testar en adderingsfunktion.

Assert

Användning:

- Assert-klassen används för att verifiera att ett visst villkor uppfylls. Om villkoret inte uppfylls, misslyckas testet.

Syntax:

- Assert följt av en metod som motsvarar det specifika villkoret som ska testas (t.ex. Assert.Equal, Assert.True).

Exempel:

- Använd Assert.Equal för att kontrollera att resultatet av en adderingsfunktion är lika med det förväntade värdet.

Test Fixture

Användning:

- En test fixture är en uppsättning resurser som behövs för att köra testerna. Det kan inkludera objektinstanser, databasanslutningar, filresurser etc.

Syntax:

- Ofta implementeras genom att skapa en konstruktor och en Dispose-metod (om IDisposable implementeras) i testklassen.

Exempel:

- Skapa en instans av en klass som ska testas eller en mockad databasanslutning som används i flera testfall.

Test Runner

Användning:

- Verktyget som faktiskt kör dina testfall. Det kan vara en kommandoradsapplikation, en del av din IDE, eller en del av en CI/CD-pipeline.

Exempel:

- Visual Studio Test Runner, dotnet test-kommandot.

Test Collection

Användning:

- Testfall i xUnit delas som standard in i testkollektioner baserat på testklassen. Testfall i samma testkollektion körs sekventiellt för att undvika delning av testkontext och därmed undvika sidoeffekter.

Exempel:

- Alla testfall i en klass körs som en del av samma testkollektion, om inte annat anges

FakeItEasy

- FakeItEasy är ett populärt bibliotek för att skapa test doubles i .NET för att möjliggöra isolation.
- Likt Moq och NSubstitute hjälper det oss att sätta upp och konfigurera Test Doubles.

Mock

- **Terminologi**: I FakeItEasy-termer är en "mock" vanligtvis en "fake" som har konfigurerats för att förvänta sig vissa anrop.
- **Skapa**: Skapa en fake med A.Fake<T>(), där T är typen som ska mockas.
- **Konfigurera**: Använd A.CallTo(() => fake.Method()).MustHaveHappened() för att sätta upp förväntningar och verifiera att vissa anrop gjorts.
- **Exempel**: Verifiera att en metod på ett fake-objekt har anropats en gång med A.CallTo(() => fake.Method()).MustHaveHappened(Once.Exactly).

Fake

- **Terminologi**: En "fake" är en enklare ersättning för en verklig implementation.
- **Skapa**: Använd A.Fake<T>() för att skapa en fake av T.
- **Användning**: Fakes används för att ersätta komplexa beroenden. De kan konfigureras för att returnera specifika värden eller bete sig på ett visst sätt.
- **Exempel**: Skapa en fake av en tjänst och konfigurera den att returnera en specifik värde vid anrop av en metod.

Spy

- **Terminologi**: En "spy" i FakeItEasy är en fake som också spårar hur den har använts.
- **Skapa och Användning**: Skapa en fake och använd A.CallTo() för att spåra anrop. Till skillnad från en strikt mock, kan en spy tillåta anrop utan fördefinierade förväntningar.
- **Exempel**: Skapa en fake och använd A.CallTo(() => fake.Method()).MustHaveHappened() för att se om och hur en metod har anropats.

Stub

- **Terminologi**: En "stub" är en fake som har konfigurerats för att returnera specifika värden.
- **Skapa**: Använd A.Fake<T>() för att skapa en fake.
- **Konfigurera**: Använd A.CallTo(() => fake.Method()).Returns(value) för att konfigurera stubben att returnera ett specifikt värde.
- **Exempel**: Skapa en fake av en databasaccess-tjänst och konfigurera den att alltid returnera en specifik användarlista.

Dummy

- **Terminologi**: En "dummy" är det enklaste formen av en fake, ofta använd för att fylla ut argumentlistor där det verkliga värdet inte är viktigt för testet.
- **Skapa**: Använd A.Dummy<T>() för att skapa en dummy av T.
- **Användning**: Används när ett objekt krävs men dess beteende eller data är irrelevant för testet.
- **Exempel**: Skapa en dummy-användarobjekt för att fylla en parameter i en metod som testas, där användarens detaljer inte påverkar testresultatet.

Länkar och resurser

- UnitTests med xUnit och FakeItEasy https://youtube.com/playlist?list=PL82C6-O4XrHeyeJcI5xrywgpfbrqdkQd4&si=b719G66CUJb2-0By
- NUnit or xUnit - Nick Chapsas https://youtu.be/JD2ZMxCPnqc?si=z3W4i1WqLjpAXjYg
- Moq vs NSubstitute vs FakeItEasy https://blog.elmah.io/moq-vs-nsubstitute-vs-fakeiteasy-which-one-to-choose/
- Comparing Mocking libraries - Dan Clarke https://www.danclarke.com/comparing-dotnet-mocking-libraries