Programmeren Gevorderd

Les 5 // Unit testing

Unit Testing

3 soorten (geautomatiseerde) testen

Unit testing -> kleine blokjes testen, veel cases & snel

Integration testing -> integratie testen tussen klassen, trager & minder cases

End to end testing -> volledige doorloop testen, heel traag, moeilijk onderhoudbaar, niet alle limieten

User testing -> niet geautomatiseerd, kijken hoe gebruikers applicatie begrijpen & ermee om gaan

Unit Testing 101

Unit testing geautomatiseerde, geprogrammeerde testen

Resultaat = geslaagd of gefaald

Gebeurt in de IDE (Visual Studio), enkel als development tool, zit niet in eindproduct

Testen op edge cases heel belangrijk!

Doen tijdens de ontwikkeling, niet achteraf!

Waarom?

Unit tests vergen extra werk, wat krijg je ervoor terug?

1. Geeft meer zekerheid bij aanpassingen of nieuwe features dat applicatie blijft werken
2. Afzonderlijk evalueren van verschillende delen van applicatie
3. Documentatie: door te kijken in de (goed geschreven) tests kan een ontwikkelaar vlug afleiden wat het gedrag en verwacht resultaat van elke unit is

Principes Unit Testing (FIRST)

1. **Fast**. Snelheid = belangrijk. Elke functionaliteit (hoe klein ook) vereist unit testing, kan leiden tot véél unit tests
2. **Isolated.** Tests mogen geen invloed hebben op andere, juist opkuisen, geen statics, ...
3. **Repeatable**. Test moet op elk moment precies dezelfde output geven
4. **Self-Validating**. Een goede test vertelt wat er fout liep zonder manueel te moeten nakijken met duidelijke messages
5. **Thorough**. Testing moet grondig gebeuren. Voorzie je op edge cases, nulls, exceptions die kunnen optreden

Test libraries: 2 delen

Test Harnas: User interface die toelaat om tests uit te voeren en resultaten te bekijken (visual studio > test explorer)

Assertion Framework: Bepalen of resultaten van testen correct zijn, doen we aan de hand van assertions

Test Framework: xUnit

xUnit

Open source

Niet van microsoft

Wel standaard mee geïnstalleerd in Visual Studio 2022

Veel uitgebreider dan standaard unit testing frameworks

Fact vs Theory

[Fact]

Test zonder variabele data

Geen externe input, testen onveranderlijke data

[Theory]

Krijgt externe input parameter

Wordt meestal ingevuld met annotation

[InlineData]

Geeft één of meerdere parameters mee aan test

Kan meerdere keren gebruikt worden om zelfde test meerdere keren uit te voeren met verschillende data

[MemberData]

Data meegeven via property in collection

Verschillende rijen = test meerdere keren uitvoeren

Gebruik nameof om naam van test property mee te geven

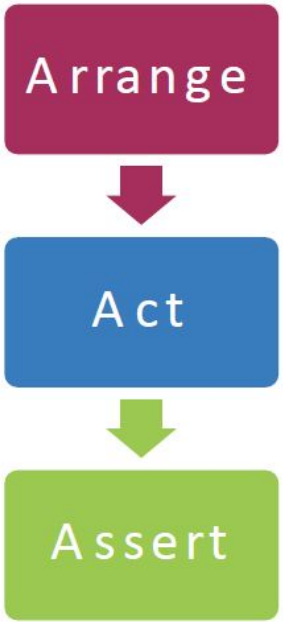
Assertion

Assertion framework

Resultaat van test controleren en failed/success state instellen

Eigenlijk resultaat van de test vergelijken met gewenst resultaat

Hoe test opbouwen? (AAA Pattern)



Set things up
Create object instances
Create test data / inputs

Execute production code
Call methods
Set properties

Check results
Test passes / fails

Assertion calls

- Assert.Equal -> Kijken of twee objecten gelijk zijn aan elkaar met equals methode
- Assert.Same -> Kijken of reference naar hetzelfde object verwijst, (strikter dan Equal)
- Assert.True -> Kijken of een value true returned
- Andere handige methodes: Assert.Contains, Assert.InRange, Assert.Empty
- De meeste assert methodes hebben een methode die het tegengestelde controleert (Assert.False bv)

Assert Exceptions

Unit tests kunnen exceptions verwachten

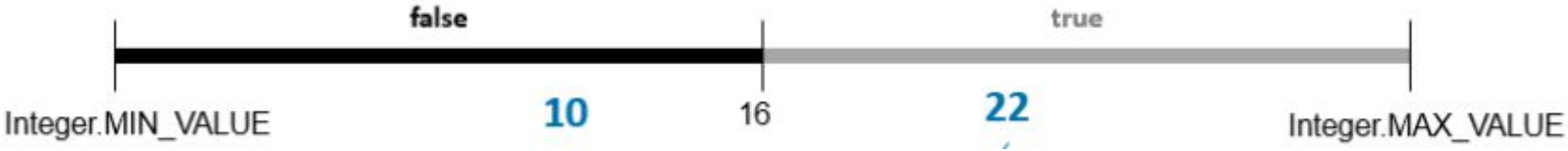
Met Assert.Throws<T> waar T type exception is

Unit test ontwerp

Equivalentie partitionering

Partitioneren van waarden die gelijke resultaten terug geven

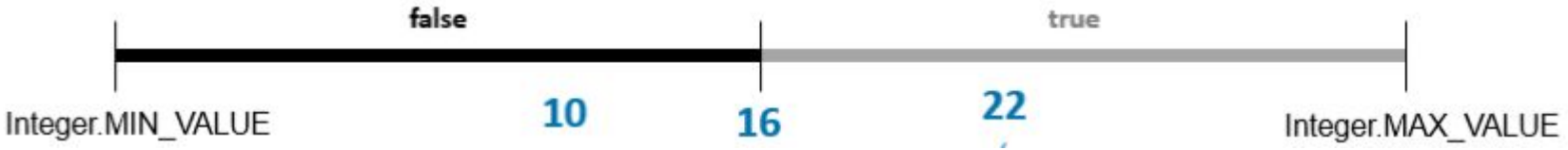
Voorbeeld: alcohol pas drinken vanaf 16



Grenswaarden analyse

Waarden die op de grens liggen worden best ook getest

Meestal heeft een grens twee zijden, dus je kan twee grenswaarden testen



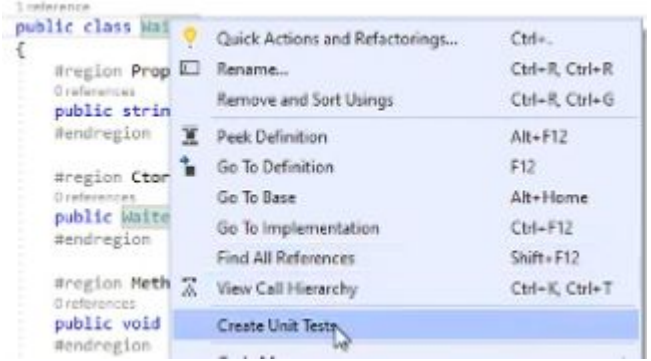
Tests schrijven

Project genereren

Vereist installatie van xUnit Test Generator

Via Extensions > Manage Extensions > Online > Search > Ctrl + E > “xunit” > xUnit.net.TestGenerator2022 > Install

Niet nodig, je kan het ook manueel doen



Test naam conventie

Afspraak over naamgeving: probeer functionaliteit zo duidelijk mogelijk te beschrijven in test methode

Bouw naam op in deze volgorde:

- de naam van de geteste method (bijvoorbeeld Reverse)
- korte beschrijving van test scenario (bijvoorbeeld ShouldThrowArgumentException)
- verwacht resultaat van test (bijvoorbeeld IfWordIsNull)

Elk onderdeel gescheiden wordt door een underscore

Business code & test assembly

Unit test klasse steeds public!

xUnit

Apart test project

Default behaviour -> throw exception ipv assert failure

Test -> Test Explorer (CTRL + E, CTRL + T)

Live unit testing -> voert tests uit terwijl je ontwikkelt

Voorbeeld: Restaurant

Voorbeeld: IntegrationTestMocking

Test Driven Development (TDD)

Wat is het?

Aanpak van ontwikkelen waarbij testing belangrijk is

Test worden geschreven **voor** de implementatie die ze zullen testen

TDD Workflow

1. Denk na over de nieuwe functionaliteit die je wil schrijven
2. Unit test schrijven voor de nieuwe functionaliteit
3. Test uitvoeren -> moet mislukken -> programma heeft nog niet nodige functionaliteit
4. Schrijf een eenvoudig mogelijke implementatie om test te doen slagen
5. Voer de unit test opnieuw uit
6. Refactor eenvoudige implementatie naar goede code
7. Slaagt test? Ga verder naar volgende test
   Slaagt test niet? Terug naar stap 4

Wat is refactoring?

Kritisch kijken naar code

Zoeken naar verbeteringen in:

- Leesbaarheid
- Consistentie
- Structuur
- Herhaling

Implementeren van deze verbeteringen

Waarom TDD gebruiken?

Voordelen van TDD

Moedigt eenvoudige & modulaire code aan

Vroeger fouten vinden & oplossen

Unit tests zijn een soort van documentatie

Snellere development op lange termijn

Eenvoudigste oplossingen vinden wordt gestimuleerd

Nadelen

Traag bij opzet

Goede unit tests schrijven is niet eenvoudig

Aanpassingen in functionaliteiten vergen ook aanpassingen in tests

Extras

Code Quality

Code Coverage via Test Explorer (80 %)

Analyze -> Calculate code metrics

Maintainability index

Kijk naar metrics van methodes, project wide zegt minder

Code coverage

Code coverage (Test > Analyze code coverage for all tests)

Streefdoel = +80%

Code Snippets

XML formaat (gelijkaardig aan HTML)

Bevat <Header> & <Snippet> delen

Header kan oa. <Title>, <Author>, <Description> & <Shortcut> bevatten

Snippet bevat code en taal

Met shortcut kan je gemakkelijk snippet invoegen

Surround with snippets bestaan ook

Voorbeeld

```
<?xml version="1.0" encoding="utf-8"?>
<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
    <CodeSnippet Format="1.0.0">
        <Header>
            <Title>Example</Title>
        </Header>
        <Snippet>
            <Code Language="CSharp">
                <![CDATA[
        throw new NotImplementedException("Put something here!");
        ]]>
            </Code>
        </Snippet>
    </CodeSnippet>
</CodeSnippets>
```

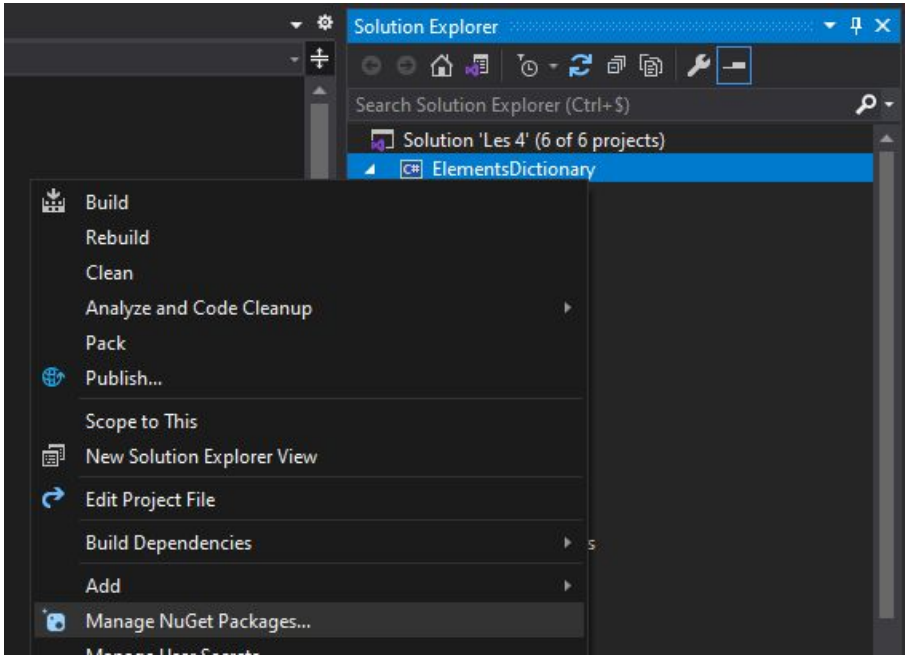
NuGet

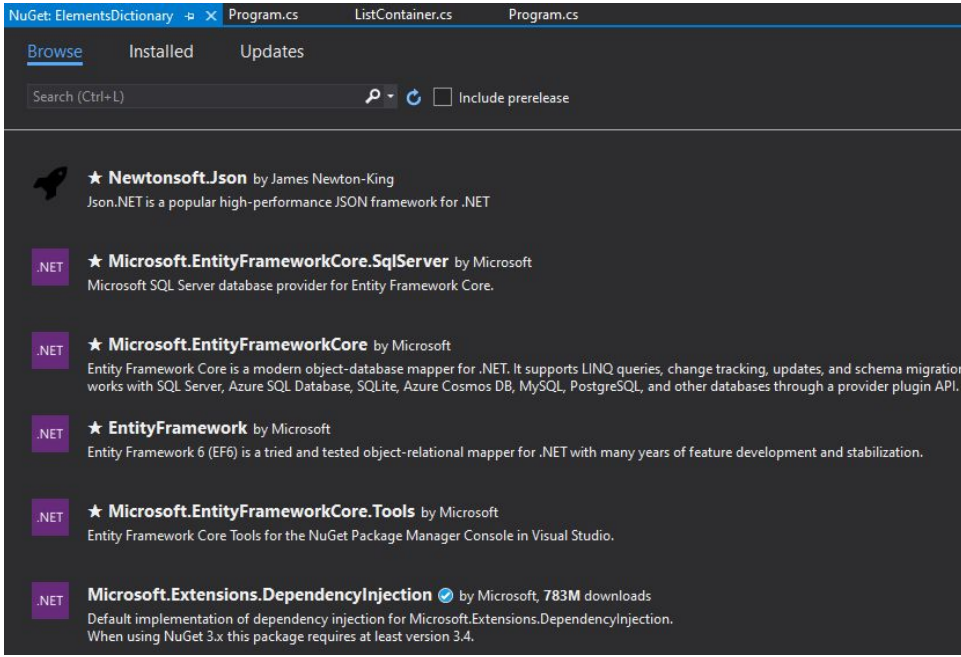
Wat is  nuget ?

.NET package manager

Importeren, verwijderen, up to date houden, compatibility checken van externe packages (dependencies)

Waar zit nuGet





Externe package/dependency

Project hangt af van externe code (is dependent on) = dependency

Package kan zelf ook dependencies hebben, worden mee geinstaleerd

Meestal in eigen namespace

Heb je een externe package nodig?

Voordelen

Tijd besparen -> code die iemand anders al geschreven heeft

Soms betere code dan je zelf zou schrijven -> grote packages, security

Onderhoud -> deel van je code wordt extern onderhouden

Gratis -> bijna allemaal kostenvrij

Nadelen

Package doet niet altijd volledig wat je wil -> bespaar of verlies je tijd als je nog aanpassingen moet doen in code die je niet kent?

Onderhoud -> package updates kunnen plots stoppen, heel moeilijk als je afhangkelijk bent

Black box -> begrijpt een bepaald deel niet volledig

Waar op te letten bij een package

Grootte -> package kan veel dependencies bevatten

Licentie -> niet alle packages mogen gratis gebruikt worden in commerciële projecten

Onderhoud -> wanneer laatst geupdate?

Details te bekijken via nuget.org

Registers

Globaal NuGet Register

Privaat register (GitHub, Azure DevOps)

Ook lokaal te installeren zonder register

Voorbeeld: arrange, act, assert snippet

https://learn.microsoft.com/en-us/visualstudio/ide/walkthrough-creating-a-code-snippet?view=vs-2022