Unit Testing & Dependency Injection & Reflection

ISB204A-02

Sander Forrer
Christophe Dumont

Karel de Grote Hogeschool

Wat is Unit Testing?

- Test de **logica** van een applicatie
  → Business Layer

- Integratietest
  → database connecties

- End-to-end (E2E) test
  → user interface

Wat is Unit Testing?

- Opdelen van applicatie code in blokjes
- Methode tot klasse
- Per blokje een test
- Test vergelijkt verwachte resultaat
- Deze tests staan los van de applicatie

Wat maakt een Unit Test nuttig?

Een goede unit test is “A TRIP”.

- **A**utomatic
  → aanroep & resultaatverwerking automatisch
- **T**horough
  → alle mogelijke scenarios
- **R**epeatable
  → herhaalbaar & zonder oncontroleerbare parameters
- **I**ndependent
  → 1 test voor 1 specifiek blokje code. Geen dependencies
- **P**rofessional
  → qualitatieve test code met duidelijke namen

Wat zijn de voordelen?

- Betere resultaten dan manueel debuggen
- Sneller
- Automatisch
- Herhaalbaar
- Inzicht in werking applicatie
- Voorkomt foutieve code in productie

Test Driven Development (TDD)

- **Eerst** Unit Test → **Daarna** applicatie code
- Alle code is essentieel
- Alle code wordt getest
- Nadenken over aanroep
  → verbeterd ontwerp

Benodigdheden

- Framework
  → Library
  → Test Runner

- NUnit
- XUnit
- MSTest (Visual Studio built in)
- ReSharper (test runner)

Benodigdheden

- Mocking framework
  → dependencies opvangen

- NSubstitute
- Moq
- Rhino Mocks
- FakeItEasy
- NMock3

Benodigdheden

- In deze tutorial maken wij gebruik van unit testing framework **MSTest** en mocking framework **Moq**

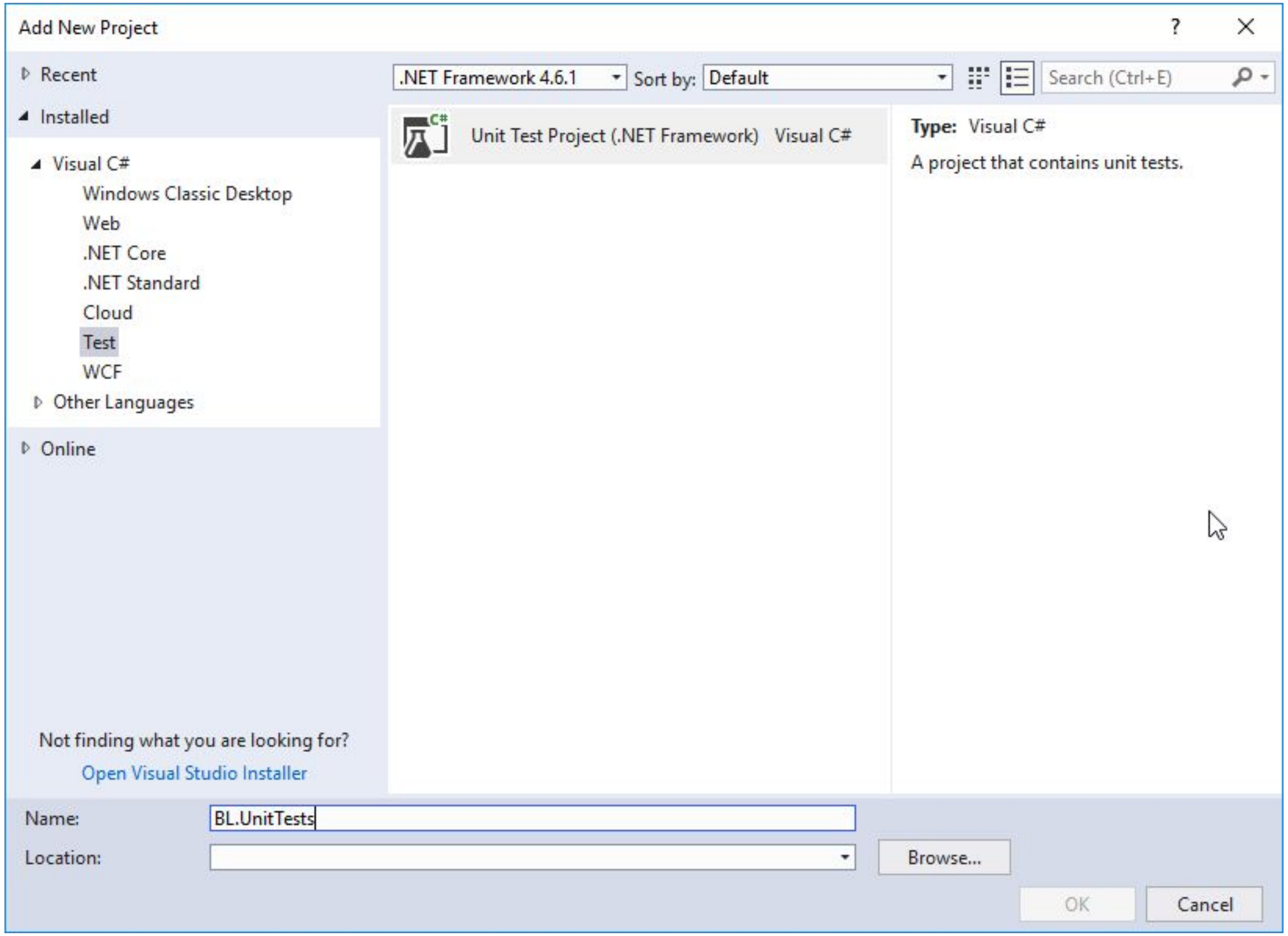
- Solution `Eg_SupportCenter`

Karel de Grote Hogeschool

Schrijven van tests

Creëren van Unit Test Project

- Solution explorer -> Rechtermuisklik op de solution -> Add -> New Project…
- Visual C# -> Test -> Unit Test Project (.NET Framework)
- Benaming gebeurd door [Projectnaam].UnitTests te gebruiken
- In dit voorbeeld gebruiken we project BL.UnitTests om methods in BL te testen



Add New Project

.NET Framework 4.6.1 · Sort by: Default · Search (Ctrl+E)

- Recent
- Installed
  - Visual C#
    - Windows Classic Desktop
    - Web
    - .NET Core
    - .NET Standard
    - Cloud
    - Test
    - WCF
  - Other Languages
- Online

Unit Test Project (.NET Framework) — Visual C#

Type: Visual C#

A project that contains unit tests.

Not finding what you are looking for?

Open Visual Studio Installer

Name: BL.UnitTests

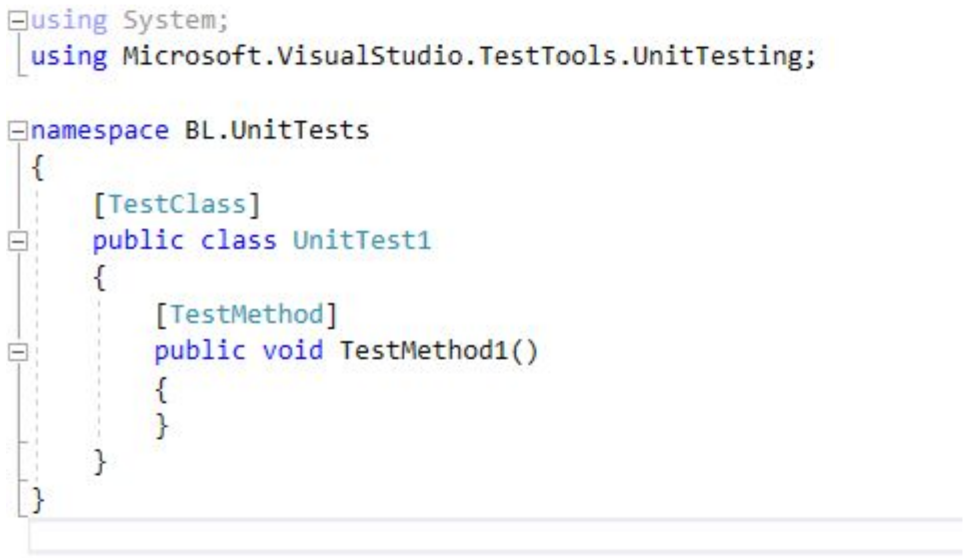
Location:

Browse... · OK · Cancel

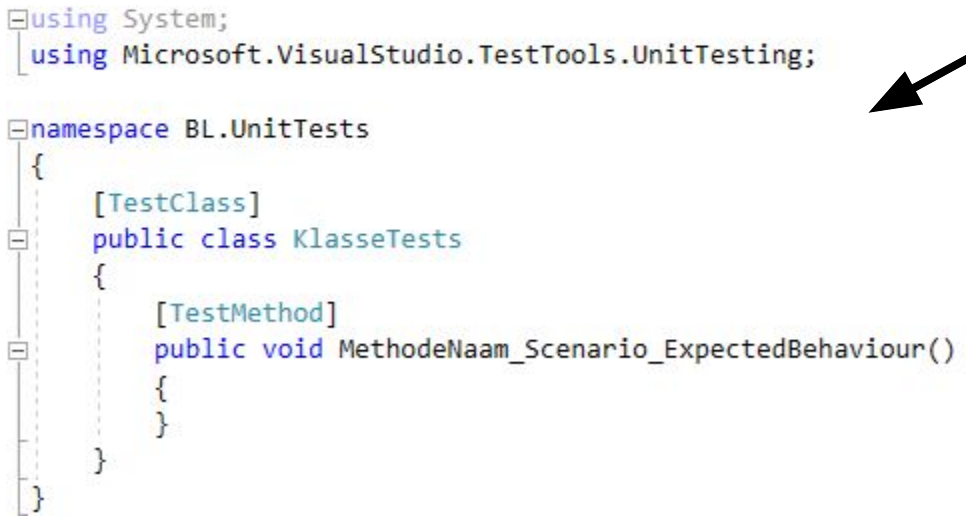
Naamgeving conventies

- Bij unit tests worden er conventies gebruikt in de benaming van de test-classes en -methods.

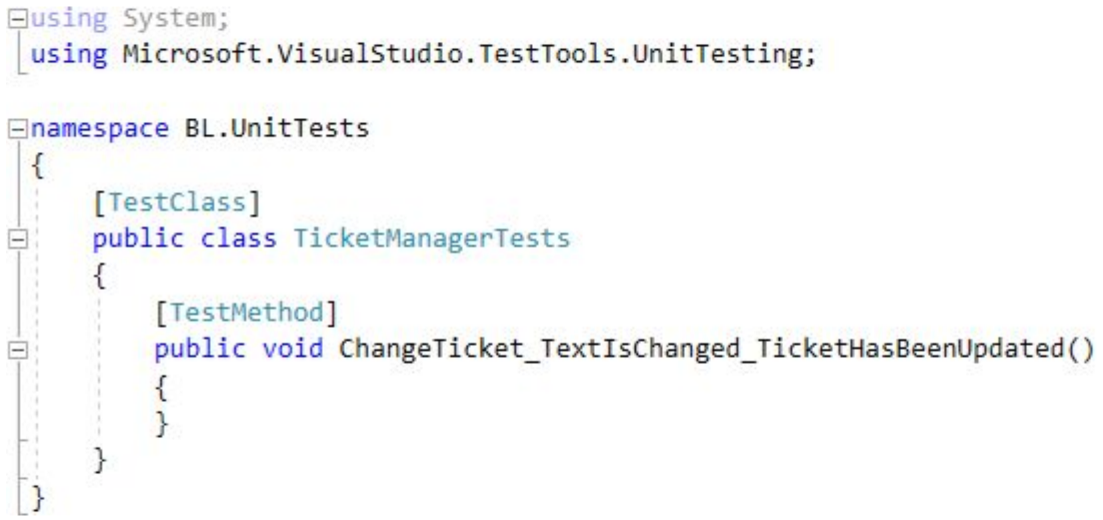
- Voor klasses: *[Class]Tests*
- voor methodes: *[Methode naam]_[Scenario]_[ExpectedBehaviour]*



```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace BL.UnitTests
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethod1()
        {
        }
    }
}
```



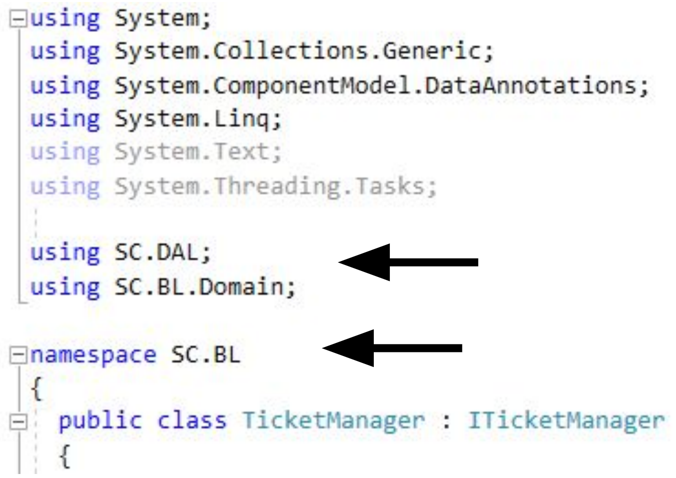
```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace BL.UnitTests
{
    [TestClass]
    public class KlasseTests
    {
        [TestMethod]
        public void MethodeNaam_Scenario_ExpectedBehaviour()
        {
        }
    }
}
```



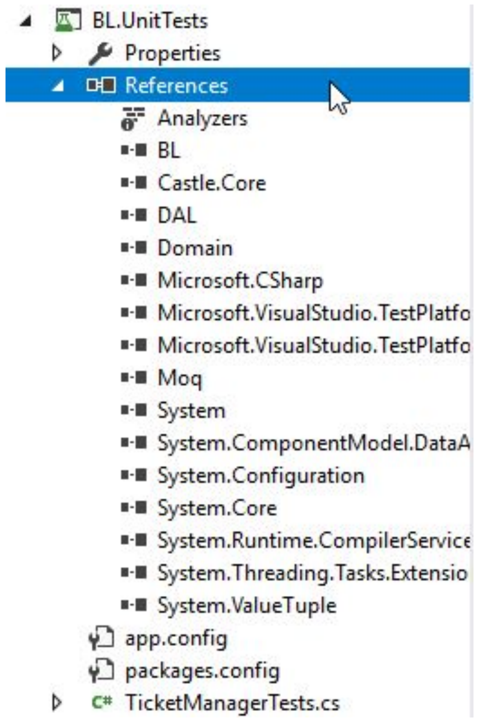
```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace BL.UnitTests
{
    [TestClass]
    public class TicketManagerTests
    {
        [TestMethod]
        public void ChangeTicket_TextIsChanged_TicketHasBeenUpdated()
        {
        }
    }
}
```

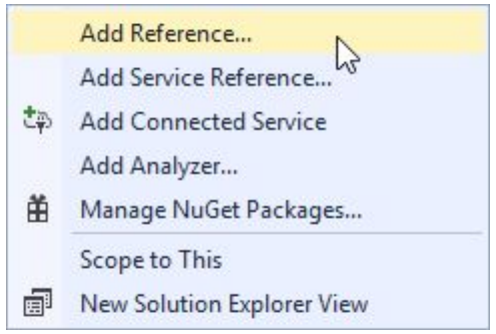
Referenties

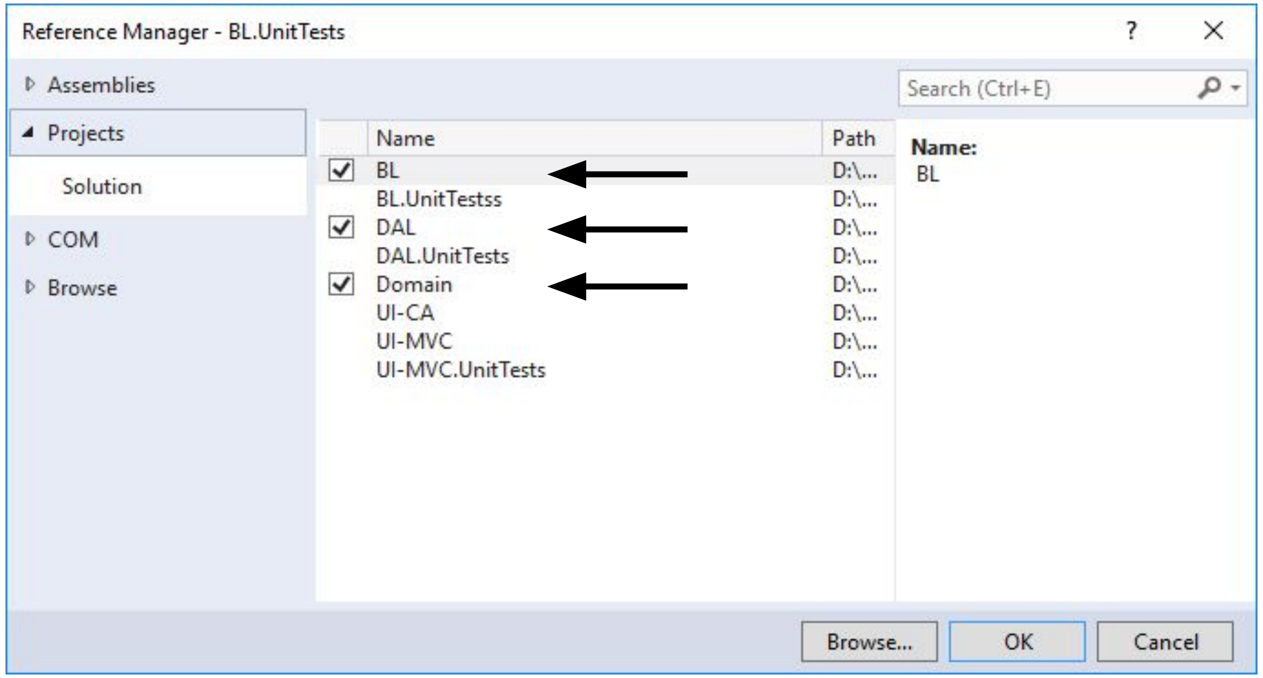
- References koppelen naar te testen class
- Rechtermuisknop References -> Add Reference



```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using SC.DAL;
using SC.BL.Domain;

namespace SC.BL
{
    public class TicketManager : ITicketManager
    {
```







Reference Manager - BL.UnitTests

	Name	Path
✓	BL	D:\...
	BL.UnitTestss	D:\...
✓	DAL	D:\...
	DAL.UnitTests	D:\...
✓	Domain	D:\...
	UI-CA	D:\...
	UI-MVC	D:\...
	UI-MVC.UnitTests	D:\...

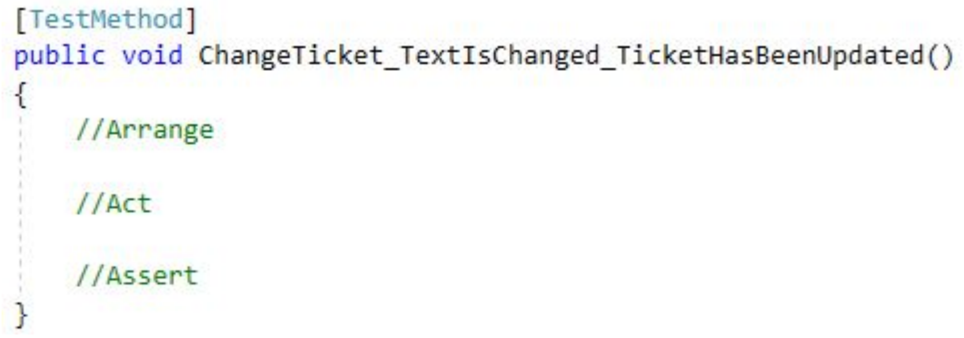
Name:
BL

Karel de Grote Hogeschool

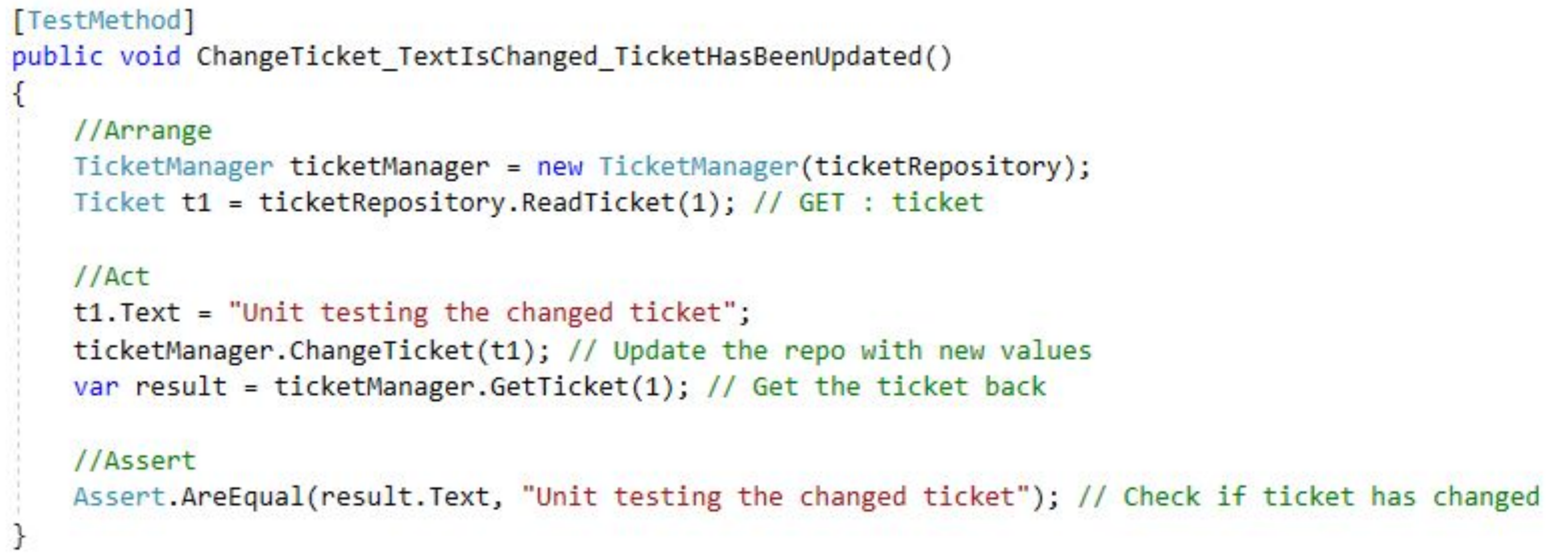
Triple A conventie

- **Triple A** conventie:

- **A**rrange: initialiseren objecten
- **A**ct: objecten testen en resultaat bijhouden
- **A**ssert: verifiëren resultaat



```
[TestMethod]
public void ChangeTicket_TextIsChanged_TicketHasBeenUpdated()
{
    //Arrange

    //Act

    //Assert
}
```

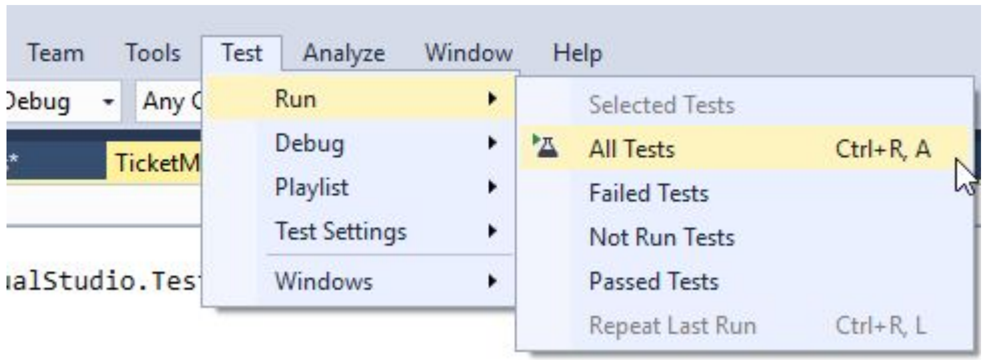


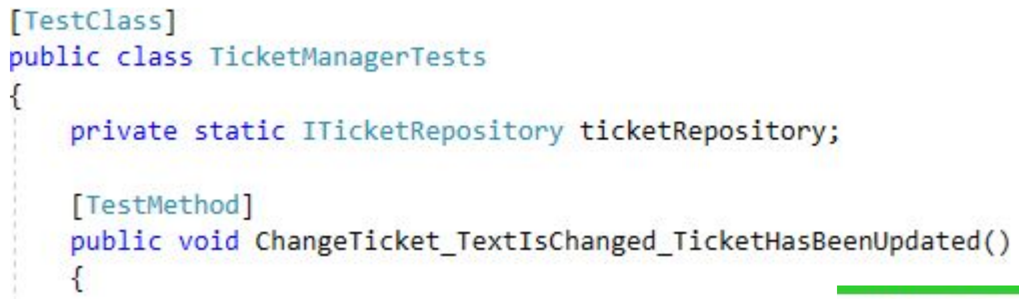
```
[TestMethod]
public void ChangeTicket_TextIsChanged_TicketHasBeenUpdated()
{
    //Arrange
    TicketManager ticketManager = new TicketManager(ticketRepository);
    Ticket t1 = ticketRepository.ReadTicket(1); // GET : ticket

    //Act
    t1.Text = "Unit testing the changed ticket";
    ticketManager.ChangeTicket(t1); // Update the repo with new values
    var result = ticketManager.GetTicket(1); // Get the ticket back

    //Assert
    Assert.AreEqual(result.Text, "Unit testing the changed ticket"); // Check if ticket has changed
}
```

De unit test uitvoeren

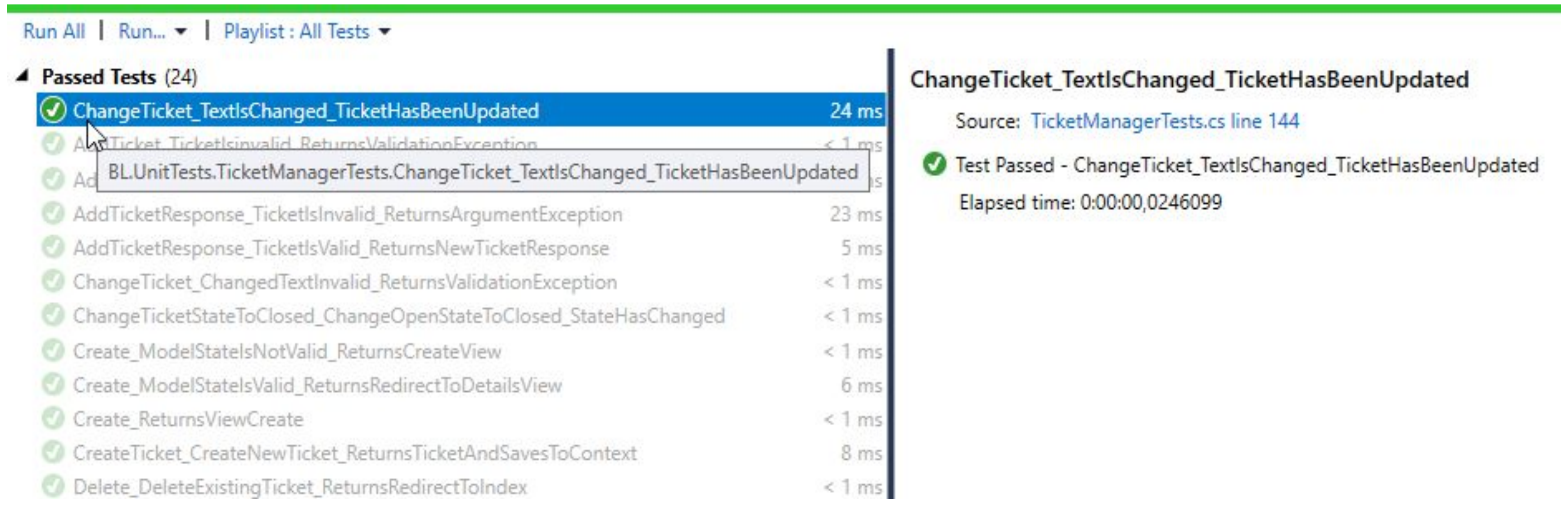
- Visual Studio -> Test -> Run -> All Tests
  OR shortcut Ctrl + R, A

- Noot: [TestClass] en [TestMethod] brackets zijn nodig!





```
[TestClass]
public class TicketManagerTests
{
    private static ITicketRepository ticketRepository;

    [TestMethod]
    public void ChangeTicket_TextIsChanged_TicketHasBeenUpdated()
    {
```



Run All | Run... ▾ | Playlist : All Tests ▾

```
Passed Tests (24)
✓ ChangeTicket_TextIsChanged_TicketHasBeenUpdated                    24 ms
✓ AddTicket_TicketIsInvalid_ReturnsValidationException               < 1 ms
✓ Ad  BL.UnitTests.TicketManagerTests.ChangeTicket_TextIsChanged_TicketHasBeenUpdated
✓ AddTicketResponse_TicketIsInvalid_ReturnsArgumentException         23 ms
✓ AddTicketResponse_TicketIsValid_ReturnsNewTicketResponse            5 ms
✓ ChangeTicket_ChangedTextInvalid_ReturnsValidationException        < 1 ms
✓ ChangeTicketStateToClosed_ChangeOpenStateToClosed_StateHasChanged < 1 ms
✓ Create_ModelStateIsNotValid_ReturnsCreateView                     < 1 ms
✓ Create_ModelStateIsValid_ReturnsRedirectToDetailsView               6 ms
✓ Create_ReturnsViewCreate                                          < 1 ms
✓ CreateTicket_CreateNewTicket_ReturnsTicketAndSavesToContext         8 ms
✓ Delete_DeleteExistingTicket_ReturnsRedirectToIndex                < 1 ms
```

ChangeTicket_TextIsChanged_TicketHasBeenUpdated

Source: TicketManagerTests.cs line 144

✓ Test Passed - ChangeTicket_TextIsChanged_TicketHasBeenUpdated

Elapsed time: 0:00:00,0246099

Dependency Injection

Dependency Injection

- Object heeft ***dependencies*** of ***afhankelijkheden***
- Code onafhankelijk maken of ***Decoupling***

  → constructor injectie

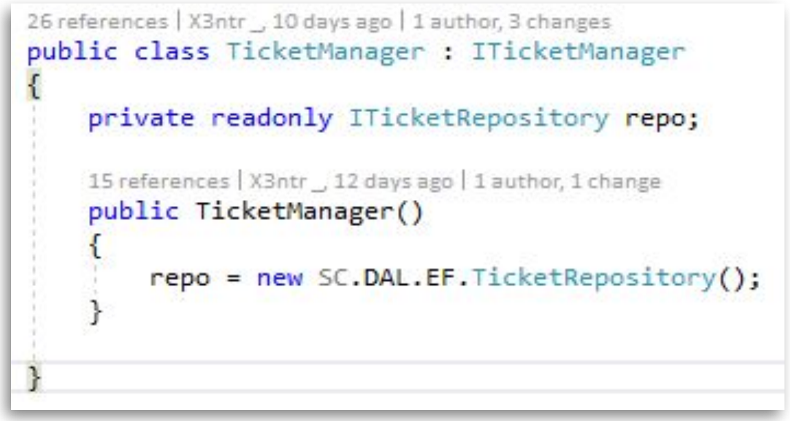
  → setter injectie

- Database queries omzeilen & andere externe bronnen
- Minder gebruik van ***mocks***

Dependency Injection

- **TicketManager** heeft dependency *ITicketRepository*

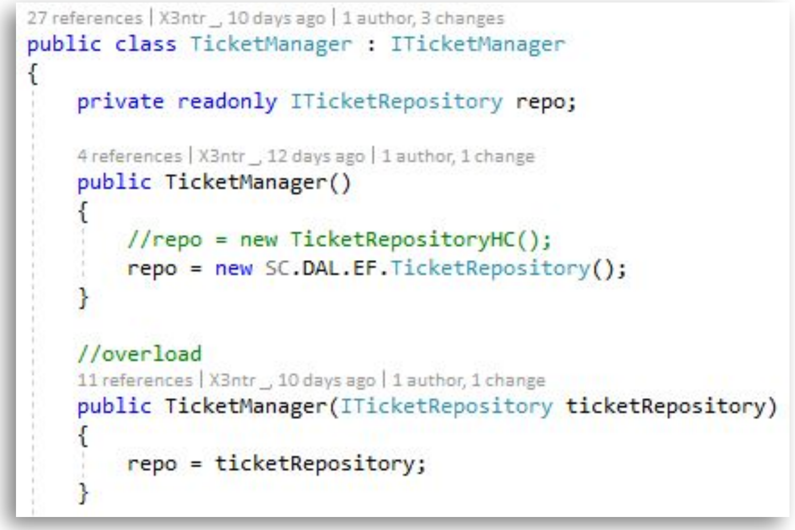
  → Dependency zelf maken



```
26 references | X3ntr_, 10 days ago | 1 author, 3 changes
public class TicketManager : ITicketManager
{
    private readonly ITicketRepository repo;

    15 references | X3ntr_, 12 days ago | 1 author, 1 change
    public TicketManager()
    {
        repo = new SC.DAL.EF.TicketRepository();
    }

}
```

  → Dependency geven (DI)

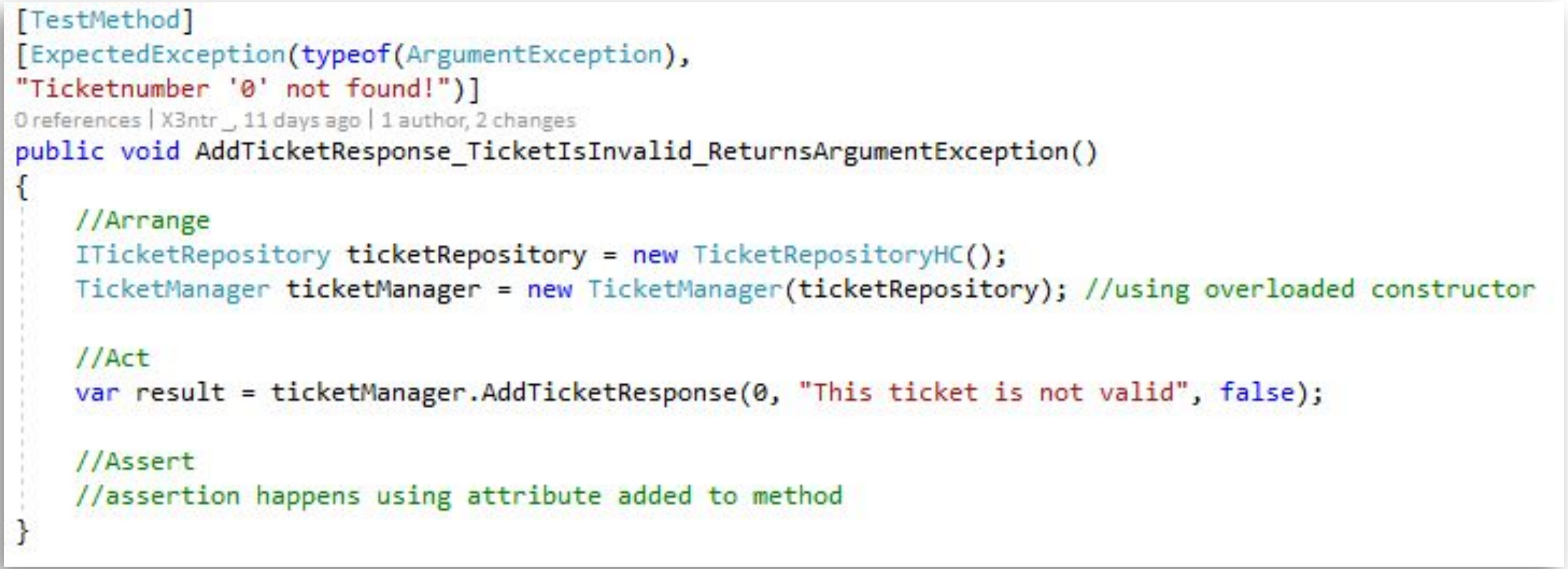
    → constructor injectie



```
27 references | X3ntr_, 10 days ago | 1 author, 3 changes
public class TicketManager : ITicketManager
{
    private readonly ITicketRepository repo;

    4 references | X3ntr_, 12 days ago | 1 author, 1 change
    public TicketManager()
    {
        //repo = new TicketRepositoryHC();
        repo = new SC.DAL.EF.TicketRepository();
    }

    //overload
    11 references | X3ntr_, 10 days ago | 1 author, 1 change
    public TicketManager(ITicketRepository ticketRepository)
    {
        repo = ticketRepository;
    }
```

Dependency Injection

- Gebruik van ***constructor injectie*** in Unit Test



```
[TestMethod]
[ExpectedException(typeof(ArgumentException),
"Ticketnumber '0' not found!")]
0 references | X3ntr_, 11 days ago | 1 author, 2 changes
public void AddTicketResponse_TicketIsInvalid_ReturnsArgumentException()
{
    //Arrange
    ITicketRepository ticketRepository = new TicketRepositoryHC();
    TicketManager ticketManager = new TicketManager(ticketRepository); //using overloaded constructor

    //Act
    var result = ticketManager.AddTicketResponse(0, "This ticket is not valid", false);

    //Assert
    //assertion happens using attribute added to method
}
```

Karel de Grote Hogeschool

Reflection

Reflection

"Reflectie is de mogelijkheid van een applicatie om ***at runtime*** zijn eigen gedrag en structuur te bekijken en eventueel aan te passen"

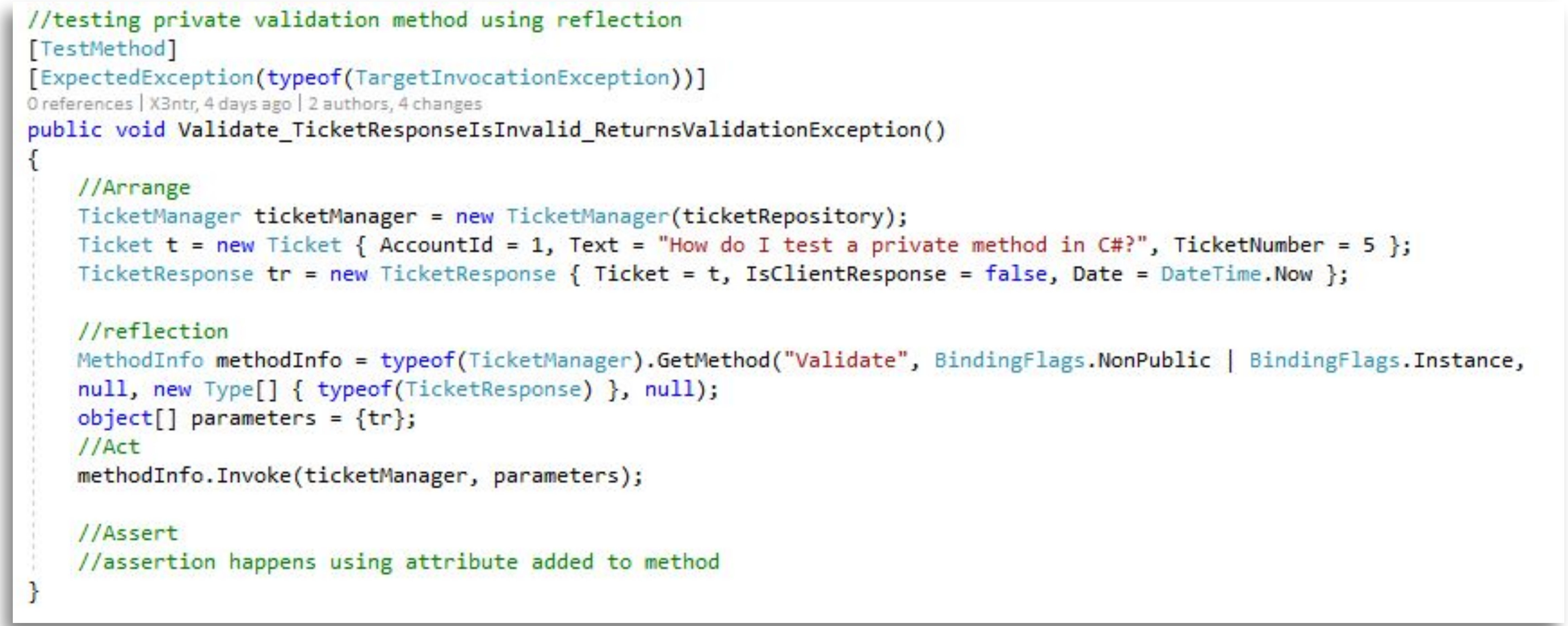
- "Type" object
- System.Reflection namespace

Reflection en Unit Testing

- Gebruik reflectie om *private* methode te testen
  → directe toegang ⇔ *public* wrapper
  → kleinste blokje

  → al getest via *public* methodes
  → test aanpassen naargelang *refactoring*

Reflectie toepassing



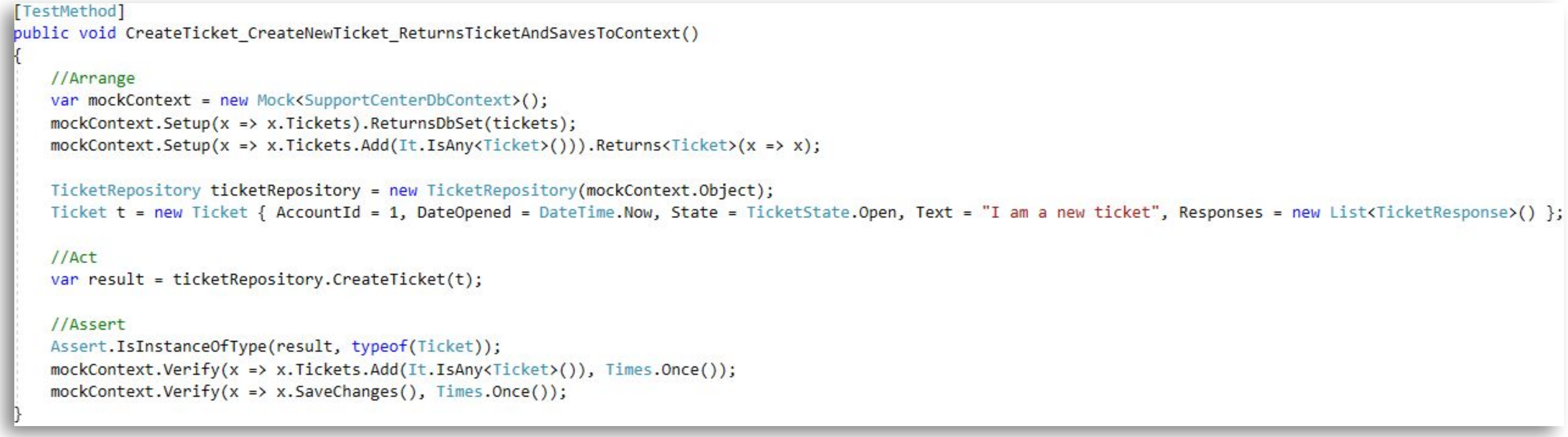
```
//testing private validation method using reflection
[TestMethod]
[ExpectedException(typeof(TargetInvocationException))]
0 references | X3ntr, 4 days ago | 2 authors, 4 changes
public void Validate_TicketResponseIsInvalid_ReturnsValidationException()
{
    //Arrange
    TicketManager ticketManager = new TicketManager(ticketRepository);
    Ticket t = new Ticket { AccountId = 1, Text = "How do I test a private method in C#?", TicketNumber = 5 };
    TicketResponse tr = new TicketResponse { Ticket = t, IsClientResponse = false, Date = DateTime.Now };

    //reflection
    MethodInfo methodInfo = typeof(TicketManager).GetMethod("Validate", BindingFlags.NonPublic | BindingFlags.Instance,
    null, new Type[] { typeof(TicketResponse) }, null);
    object[] parameters = {tr};
    //Act
    methodInfo.Invoke(ticketManager, parameters);

    //Assert
    //assertion happens using attribute added to method
}
```

Karel de Grote Hogeschool

Functionaliteit demo-app

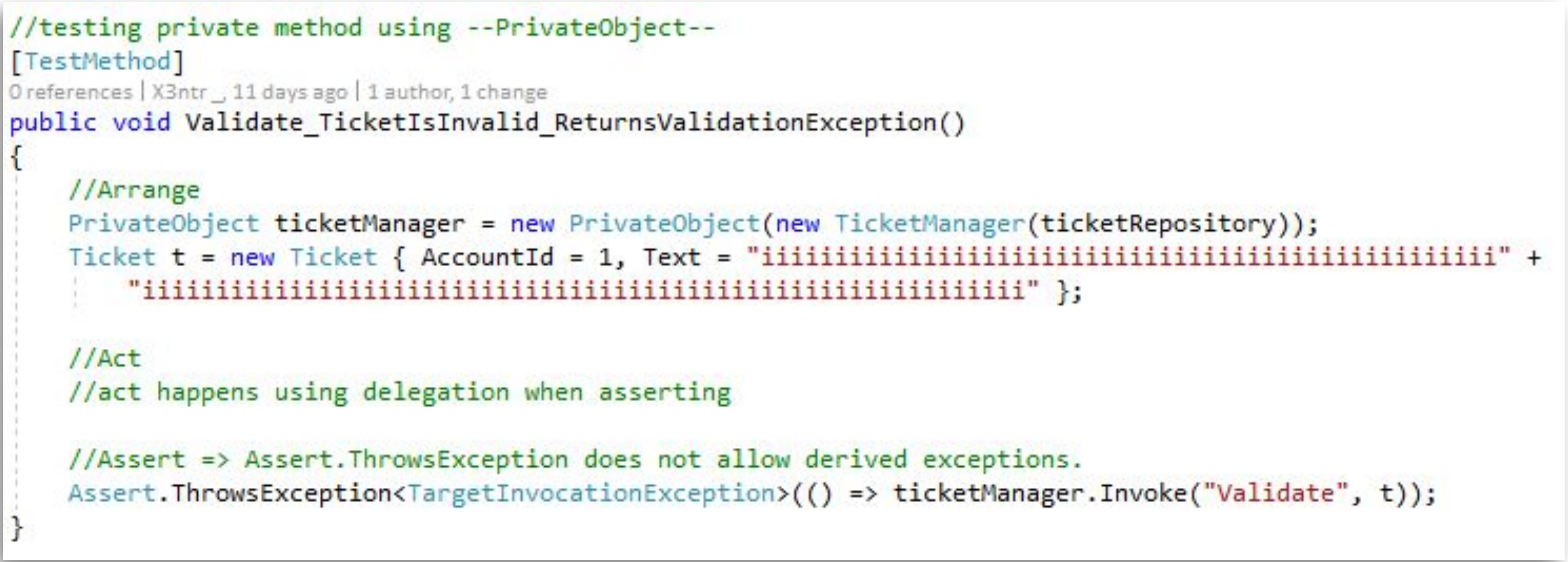
Gebruik van Moq

- Unit tests voor `Eg_SupportCenter` geschreven



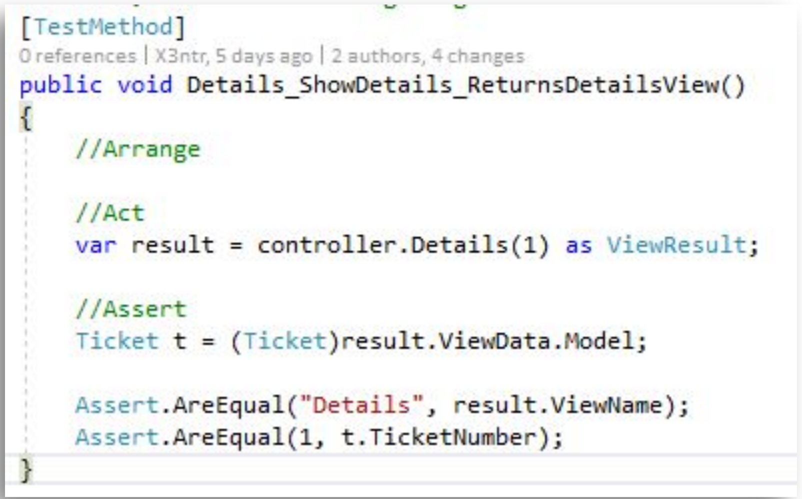
```
[TestMethod]
public void CreateTicket_CreateNewTicket_ReturnsTicketAndSavesToContext()
{
    //Arrange
    var mockContext = new Mock<SupportCenterDbContext>();
    mockContext.Setup(x => x.Tickets).ReturnsDbSet(tickets);
    mockContext.Setup(x => x.Tickets.Add(It.IsAny<Ticket>())).Returns<Ticket>(x => x);

    TicketRepository ticketRepository = new TicketRepository(mockContext.Object);
    Ticket t = new Ticket { AccountId = 1, DateOpened = DateTime.Now, State = TicketState.Open, Text = "I am a new ticket", Responses = new List<TicketResponse>() };

    //Act
    var result = ticketRepository.CreateTicket(t);

    //Assert
    Assert.IsInstanceOfType(result, typeof(Ticket));
    mockContext.Verify(x => x.Tickets.Add(It.IsAny<Ticket>()), Times.Once());
    mockContext.Verify(x => x.SaveChanges(), Times.Once());
}
```

Testen met PrivateObject

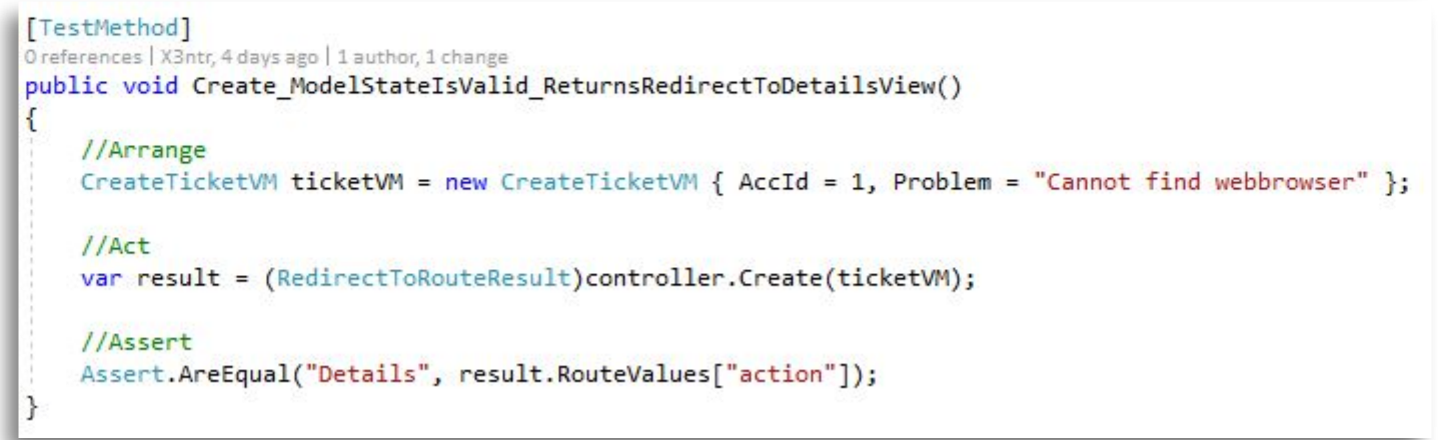


```
//testing private method using --PrivateObject--
[TestMethod]
0 references | X3ntr_, 11 days ago | 1 author, 1 change
public void Validate_TicketIsInvalid_ReturnsValidationException()
{
    //Arrange
    PrivateObject ticketManager = new PrivateObject(new TicketManager(ticketRepository));
    Ticket t = new Ticket { AccountId = 1, Text = "iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii" +
        "iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii" };

    //Act
    //act happens using delegation when asserting

    //Assert => Assert.ThrowsException does not allow derived exceptions.
    Assert.ThrowsException<TargetInvocationException>(() => ticketManager.Invoke("Validate", t));
}
```

Testen van MVC controllers



```
[TestMethod]
0 references | X3ntr, 5 days ago | 2 authors, 4 changes
public void Details_ShowDetails_ReturnsDetailsView()
{
    //Arrange

    //Act
    var result = controller.Details(1) as ViewResult;

    //Assert
    Ticket t = (Ticket)result.ViewData.Model;

    Assert.AreEqual("Details", result.ViewName);
    Assert.AreEqual(1, t.TicketNumber);
}
```



```
[TestMethod]
0 references | X3ntr, 4 days ago | 1 author, 1 change
public void Create_ModelStateIsValid_ReturnsRedirectToDetailsView()
{
    //Arrange
    CreateTicketVM ticketVM = new CreateTicketVM { AccId = 1, Problem = "Cannot find webbrowser" };

    //Act
    var result = (RedirectToRouteResult)controller.Create(ticketVM);

    //Assert
    Assert.AreEqual("Details", result.RouteValues["action"]);
}
```

Conclusie

Conclusie

- Geeft een duidelijk inzicht in de applicatie
- Soms onnodig veel werk voor simpele code
- Moeilijk om in te komen
- Bestaande code testen ⇔ Test Driven Development
- Mocking framework vereist kennis

Bronnen

Bronnen

- “Creating Unit Tests for ASP.NET MVC Applications (C#)”. *docs.microsoft.com*. 19 augustus 2008. [Online]. Beschikbaar: https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/unit-testing/creating-unit-tests-for-asp-net-mvc-applications-cs. [Geraadpleegd op 28 juli 2018].
- Shining Dragon, “Unit Testing Interfaces in .NET”. *codeproject.com*. 13 augustus 2014. [Online]. Beschikbaar: https://www.codeproject.com/Tips/609259/Unit-Testing-Interfaces-in-NET. [Geraadpleegd op 28 juli 2018].
- TimStall, “How to Test Private and Protected methods in .NET”. *codeproject.com*. 1 maart 2005. [Online]. Beschikbaar: https://www.codeproject.com/Articles/9715/How-to-Test-Private-and-Protected-methods-in-NET. [Geraadpleegd op 30 juli 2018].
- “C# How do I invoke a private overloaded method using System.Reflection when number of arguments are equal”. *stackoverflow.com*. 1 augustus 2018. [Online]. Beschikbaar: https://stackoverflow.com/questions/51631254/c-sharp-how-do-i-invoke-a-private-overloaded-method-using-system-reflection-when. [Geraadpleegd 1 augustus 2018].

Bronnen

- Molly Alger. “Intro to Mocking with Moq”. *spin.atomicobject.com*. 7 augustus 2017. [Online]. Beschikbaar: https://spin.atomicobject.com/2017/08/07/intro-mocking-moq/. [Geraadpleegd op 2 augustus 2018].
- Ben Lucas. “Effective Unit Testing - Part 2: Dependency Injection”. *info.obsglobal.com*. 3 maart 2014. [Online]. Beschikbaar: https://info.obsglobal.com/blog/2014/03/effective-unit-testing-part-2-dependency-injection. [Geraadpleegd 2 augustus 2018].
- “Entity Framework Testing with a Mocking Framework (EF6 onwards)”. *msdn.microsoft.com*. 23 oktober 2016. [Online]. Beschikbaar: https://info.obsglobal.com/blog/2014/03/effective-unit-testing-part-2-dependency-injection. [Geraadpleegd 2 augustus 2018].
- Tim Larson. “A Simple interface for fluently mocking a DbSet”. *codethug.com*. 20 maart 2015. [Online]. Beschikbaar: https://codethug.com/2015/03/20/mocking-dbset/. [Geraadpleegd 3 augustus 2018].

Bronnen

- “What makes a good Unit Test?”, *stackoverflow.com*. 14 september 2008. [Online]. Beschikbaar: https://stackoverflow.com/questions/61400/what-makes-a-good-unit-test. [Geraadpleegd op 05 augustus 2018].
- “What is dependency injection?”, *stackoverflow.com*. 25 september 2008. [Online]. Beschikbaar: https://stackoverflow.com/questions/130794/what-is-dependency-injection [Geraadpleegd op 05 augustus 2018].
- “How should one unit test a .NET MVC controller?”. *stackoverflow.com*. 11 januari 2012. [Online]. Beschikbaar: https://stackoverflow.com/questions/8818207/how-should-one-unit-test-a-net-mvc-controller.
- “How do I use Assert to verify that an exception has been thrown?”. *stackoverflow.com*. 1 juni 2009. [Online]. Beschikbaar: https://stackoverflow.com/questions/933613/how-do-i-use-assert-to-verify-that-an-exception-has-been-thrown. [Geraadpleegd op 05 augustus 2018].
- “Invoking Overloaded Methods Using Reflection”. *blackwasp.co.uk*. 2 juni 2013. [Online]. Beschikbaar: http://www.blackwasp.co.uk/ReflectionInvokeOverload.aspx. [Geraadpleegd op 06 augustus 2018].