

HoGent

BEDRIJF
EN
ORGANISATIE

Hoofdstuk 6 : ASP.NET MVC 4

HoGent

Pag.1

Hoofdstuk 6 : ASP.NET MVC 4

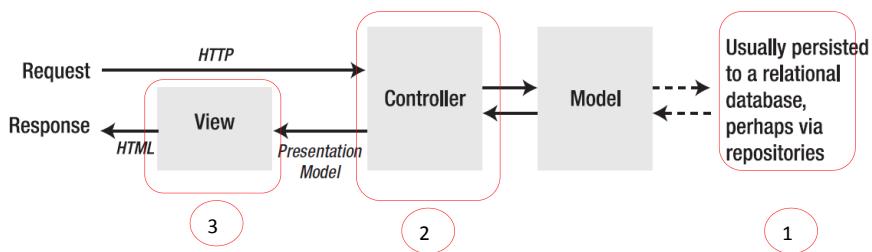
1. Inleiding
2. Domain Model
3. DAL : Repository pattern
4. Dependency Injection
5. IoC
6. Mocking
7. De MVC Applicatie Bierhalle
 - A. De Controller
 - B. Index : TDD van de Controller
 - C. Index : View
 - D. Index : Extra vraag van de gebruiker
 - E. Create GET/POST : TDD van de Controller
 - F. Create GET/POST : View
 - G. Edit GET/POST : TDD van de Controller
 - H. Edit GET/POST : View
 - I. Delete GET/POST : TDD van de Controller
 - J. Delete GET/POST : View
8. Layouts
9. Oefeningen

HoGent

Pag.2

1. Inleiding

- ASP.NET MVC framework



1. Inleiding

- Een eerste **data-driven** MVC applicatie
- Bedoeling
 - Volledige CRUD voor brouwers
 - TDD van het Model (zoals in hoofdstuk 2)
 - TDD van de Controller
 - De unit testen : mocking
 - Interacties met de databank : Entity Framework Code First
 - Aanmaken van de View : basis
 - Nog geen authenticatie en autorisatie (zie later)
 - Nog geen validatie (zie later)
 - Nog geen ajax (zie later)
 - ...

1. Inleiding : Use case

► Use Case

Use Case : Beheer brouwers

Actor : administrator

Precondities : actor is ingelogd als administrator. Brouwer gegevens zijn beschikbaar.

Postcondities : administrator heeft gegevens brouwers bekeken, eventueel de gegevens van een brouwer aangepast, een brouwer toegevoegd of een brouwer verwijderd.

Normale verloop:

1. Systeem geeft een overzicht van de brouwers (naam, gemeente, omzet) en de mogelijkheid om een brouwer te editeren, te verwijderen of een nieuwe brouwer toe te voegen
2. Herhaal
 1. Administrator kiest om brouwer te editeren
 2. Systeem geeft de gegevens van de brouwer weer : brouwernr, naam, straat, postcode, gemeente, omzet
 3. Administrator wijzigt de gegevens van de brouwer
 4. Systeem valideert de gegevens en slaat deze op in de database
 5. Systeem geeft melding dat de gegevens gewijzigd zijn

HoGent

Pag. 5

1. Inleiding : Use case

Alternatieve scenario's:

2.1.1a Administrator wenst een brouwer te creëren

1. Systeem biedt de mogelijkheid om te gegevens in te geven : naam, straat, postcode, omzet
2. Administrator geeft de gegevens in
3. Naar 2.1.4

2.3.a Administrator wenst de gegevens niet te wijzigen

1. Terug naar 2

2.3.b De administrator wenst een brouwer te verwijderen

1. Systeem vraagt om bevestiging
2. Actor bevestigt
3. Systeem verwijdert de brouwer uit de database
4. Systeem geeft melding. Terug naar 1

2.3.b.1.a. Systeem detecteert dat brouwer bieren bevat.

1. Systeem geeft melding dat de brouwer niet kan verwijderd worden zolang hij bieren heeft.
Terug naar 2

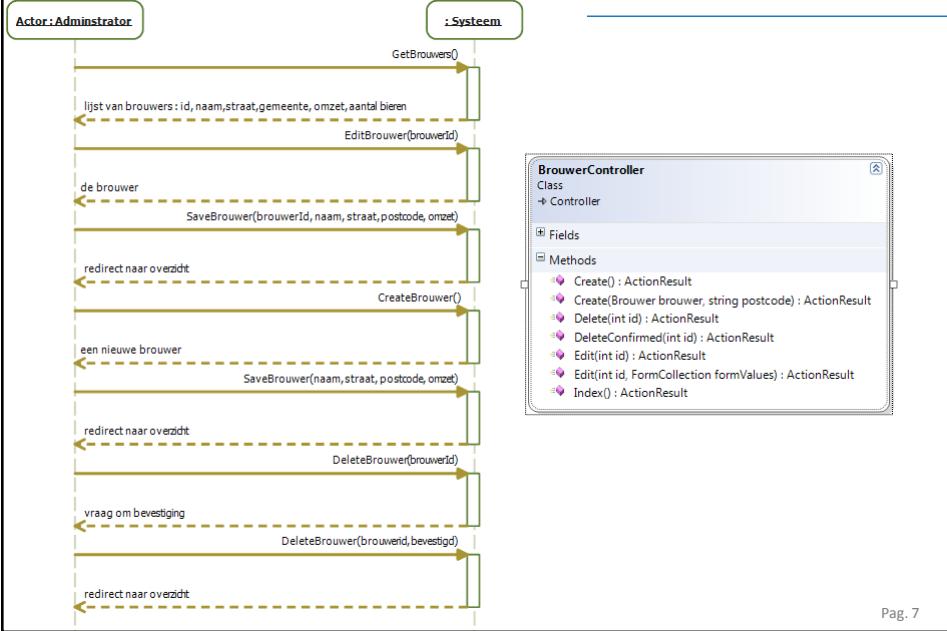
2.4.a Systeem detecteert dat niet alle gegevens correct ingevuld zijn : naam verplicht, omzet > 0 als ingevuld.)

1. Systeem geeft melding. Terug naar 2.3

HoGent

Pag. 6

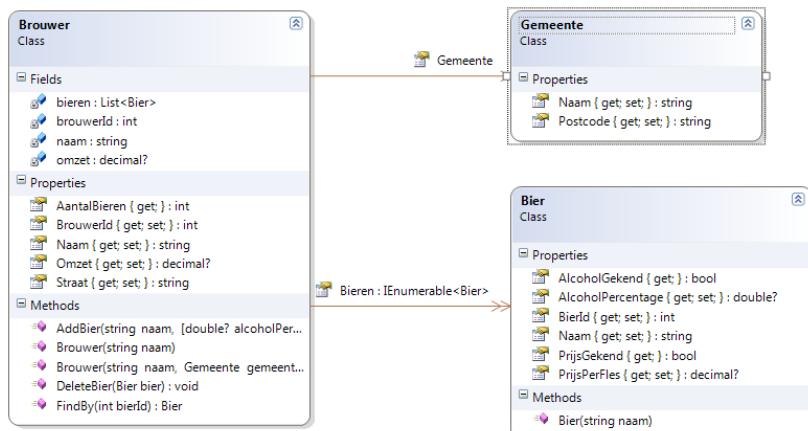
1. Inleiding : SSD - BrouwerController



Pag. 7

1. Inleiding : DCD

- Domeinklassen en testen reeds aangemaakt. Nu in Folder Models > Domain



HoGent

Pag. 8

1. Inleiding

Bierhalle

Hello, student! [Log off](#)

[Home](#) [Brouwers](#) [Over ons](#) [Contact](#)

De MVC applicatie Bierhalle

Brouwers

[Creëer brouwer](#)

Naam brouwer	Straat	Gemeente	Omzet	Aantal bieren	
Bavik	Rijksweg 33	8531 Bavikhove	20000000,00 euro	6	Detail Editeer Verwijder
De Graal			omzet niet gekend	0	Detail Editeer Verwijder
De Leeuw			omzet niet gekend	0	Detail Editeer Verwijder
Duvel Moortgat	Breendonk dorp 28	2870 Puurs	omzet niet gekend	6	Detail Editeer Verwijder
InBev	Brouwerijplein 1	3000 Leuven	omzet niet gekend	5	Detail Editeer Verwijder
Palm Breweries			500000,00 euro	4	Detail Editeer Verwijder
Roman	Hauwaart 105	9700 Oudenaarde	omzet niet gekend	4	Detail Editeer Verwijder

De totale omzet : 20500000,00 €

© 2013 - Hogeschool Gent

HoGent

Pag. 9

2. Domain Model

- ▶ Open de Starter applicatie
 - Is een MVC internet applicatie met een unit test project
 - Naam project : Bierhalle

- ▶ De domeinklassen Brouwer, Bier, Gemeente zijn reeds aangemaakt in folder Models > Domain
- ▶ De bijhorende testen (BrouwerTest.cs en BierTest.cs) eveneens
 - Run de testen

HoGent

Pag. 10

3. Data Access Layer

- ▶ De persistentielaaag is ook reeds aangemaakt
 - BierhalleContext
 - BierhalleInitializer
 - Het instellen van de DbContext/initstrategie : **Global.asax**
 - Global.asax : bevat events gettriggered op applicatie niveau.
 - Pas aan (using System.Data.Entity;)

```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();

    RegisterGlobalFilters(GlobalFilters.Filters);
    RegisterRoutes(RouteTable.Routes);
    Database.SetInitializer<BierhalleContext>(new BierhalleInitializer());
}
```

Merk op : de database wordt hierdoor nog niet gecreëerd. Dit gebeurt pas bij de eerste query!

HoGent

Pag. 11

3. Data Access Layer

- ▶ De persistentielaaag is ook reeds aangemaakt
 - Controller zal met de persistentielaaag communiceren voor de CRUD
 - Controller kan hiervoor rechtstreeks communiceren met een instantie van BierhalleContext
 - Bvb new BierhalleContext().Brouwers;
 - Nadeel : Kan niet getest worden. Unit testen werken niet rechtstreeks met databases (te traag, repeatable?)
 - Oplossing : Repositories

HoGent

Pag. 12



Data Acces Layer (persistentielaag) Repository pattern

3. Data Access Layer

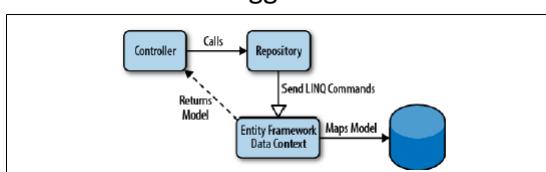
► Repository pattern

Martin Fowler writes:

"A Repository mediates between the domain and data mapping layers, acting like an *in-memory domain object collection*. Client objects construct query specifications declaratively and submit them to Repository for satisfaction. Objects can be added to and removed from the Repository, as they can from a simple collection of objects, and the mapping code encapsulated by the Repository will carry out the appropriate operations behind the scenes. Conceptually, a Repository encapsulates the set of objects persisted in a data store and the operations performed over them, providing a more object-oriented view of the persistence layer. Repository also supports the objective of achieving a clean separation and one-way dependency between the domain and data mapping layers."

- Logische groepering van objecten = aggregate
- Handelt als in memory collection van objecten, en maakt abstractie van de onderliggende data infrastructure

HoGent



3. Data Access Layer

► Repository pattern

- Maak één repository per aggregate
 - In DDD betekent aggregate

A cluster of associated objects that are treated as a unit for the purpose of data changes. External references are restricted to one member of the Aggregate, designated as the root. A set of consistency rules applies within the Aggregate's boundaries.".

- Voorbeeld : De aggregates in het domein zijn : Brouwer en Gemeente
- Tip : aggregate komt overeen met de objecten die de Controller zal opvragen uit de database (Controller zal een Brouwer opvragen, en via de Brouwer de Bieren van die brouwer opvragen, maar zal nooit rechtstreeks de Bieren uit de db opvragen.=> Bier is geen aggregate root)

3. Data Access Layer

► Repository pattern

- Een typische **Repository interface** bevat enkel wat nodig is voor de CRUD.
- In de meeste gevallen heb je 5 methodes :
 - Geef alle, eventueel met bijkomende zoekmogelijkheden (**FindAll**)
 - Geef één aggregate by id (primary key) (**FindBy**)
 - Voeg een nieuwe aggregate toe aan de repository (**Add**)
 - Verwijder een aggregate van de repository (**Delete**)
 - Opslaan van de wijzigingen (**Save**)
- MERK OP : Je hebt geen “**wijzig**” methode
 - **Reden** : de repository moet bij de **Save** opdracht zelf in staat zijn om te detecteren welke objecten gewijzigd zijn en de nodige aanpassingen in de databank door te voeren

3. Data Access Layer

► Repository pattern

- Een typische repository interface
 - conventie : interfaces starten met I
- Behoort tot de **Bierhalle.Models.Domain** namespace

```
using System.Linq;

namespace BierHalle.Models.Domain
{
    public interface IBrouwerRepository
    {
        Brouwer FindBy(int brouwerId);
        IQueryable<Brouwer> FindAll();
        void Add(Brouwer brouwer);
        void Delete(Brouwer brouwer);
        void SaveChanges();
    }
}
```

HoGent

Pag. 17

3. Data Access Layer

► IQueryable

- IEnumerable<T> : aanvullingen (include, where, orderby,...) worden uitgevoerd in het geheugen
- IQueryable<T> : aanvullingen (include, where, orderby,...) worden uitgevoerd in de database

```
public IQueryable<T> FindAll()
{
    return _objectSet.Where(e => e.Id > 0);
}

public IEnumerable<T> FindAllAsEnumerable()
{
    return _objectSet.Where(e => e.Id > 0);
}
```

HoGent

Pag. 18

3. Data Access Layer

► IQueryable

- Repository

```
public IQueryable<T> FindAll()
{
    return _objectSet.Where(e => e.Id > 0);
}

public IEnumerable<T> FindAllAsEnumerable()
{
    return _objectSet.Where(e => e.Id > 0);
}
```

- Controller

```
public ViewResult Index()
{
    var model = _repository.FindAll()
        .OrderBy(e => e.HireDate);
    return View(model);
}

public ViewResult Index()
{
    var model = _repository.FindAllAsEnumerable()
        .OrderBy(e => e.HireDate)
        .Take(10);
    return View(model);
}
```

```
SELECT
[Extent1].[Id] AS [Id],
[Extent1].[Name] AS [Name],
[Extent1].[HireDate] AS [HireDate]
FROM [dbo].[Employees] AS [Extent1]
WHERE [Extent1].[Id] > 0
ORDER BY [Extent1].[HireDate] ASC
```

```
SELECT
[Extent1].[Id] AS [Id],
[Extent1].[Name] AS [Name],
[Extent1].[HireDate] AS [HireDate]
FROM [dbo].[Employees] AS [Extent1]
WHERE [Extent1].[Id] > 0
```

HoGent

Pag. 19

3. Data Access Layer

- ### ► Maak de interface IBrouwerRepository aan. De interface IGemeenteRepository is reeds voorzien
- folder Models > Domain > rechtsklik > Add > New Item > Interface

```
using System.Linq;

namespace BierHalle.Models.Domain
{
    public interface IBrouwerRepository
    {
        Brouwer FindBy(int brouwerId);
        IQueryable<Brouwer> FindAll();
        void Add(Brouwer brouwer);
        void Delete(Brouwer brouwer);
        void SaveChanges();
    }
}
```

```
using System.Linq;

namespace BierHalle.Models.Domain
{
    public interface IGemeenteRepository
    {
        Gemeente FindBy(string postcode);
        IQueryable<Gemeente> FindAll();
    }
}
```

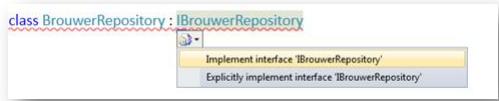
HoGent

Opm : Delete kan ook met parameter brouwerId

Pag. 20

3. Data Access Layer

- ▶ Implementeren van de repository klasse
 - De implementatie van de interface is onderdeel van de DAL!
want in deze applicatie zal gebruik gemaakt worden van EF
 - Maak een nieuwe klasse BrouwerRepository aan in Models > DAL folder
 - Erf van IBrouwerRepository
 - Kies Implement interface (klik op rechthoekje onder I)



3. Data Access Layer

- ▶ Implementeren van de Repository klasse
 - Constructor met BierhalleContext als parameters.
Verschillende repositories werken met dezelfde DbContext.
 - Anders 2 caches, identity maps, ...

```
public class BrouwerRepository : IBrouwerRepository
{
    private BierhalleContext context;
    private DbSet<Brouwer> brouwers;

    public BrouwerRepository(BierhalleContext context)
    {
        this.context = context;
        brouwers = context.Brouwers;
    }
}
```

3. Data Access Layer

- ▶ Implementeer Repository klasse

```
public void Add(Brouwer brouwer)
{
    brouwers.Add(brouwer);
}
public void Delete(Brouwer brouwer)
{
    brouwers.Remove(brouwer);
}
public Brouwer FindBy(int brouwerId)
{
    return brouwers.Find(brouwerId);
}
public IQueryable<Brouwer> FindAll()
{
    return brouwers.Include(b=>b.Gemeente).OrderBy(b => b.Naam);
}
public void SaveChanges()
{
    context.SaveChanges();
}
```

HoGent

Pag. 23

3. Data Access

- ▶ De Gemeente Repository klasse

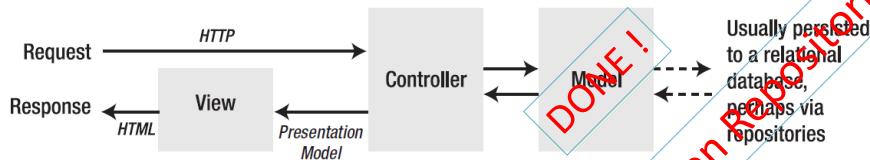
```
using System.Data.Entity;
using System.Linq;
using BierHalle.Models.Domain;
namespace Bierhalle.Models.DAL
{
    public class GemeenteRepository : IGemeenteRepository
    {
        private BierhalleContext context;
        private DbSet<Gemeente> gemeenten;

        public GemeenteRepository(BierhalleContext context)
        {
            this.context = context;
            gemeenten = context.Gemeenten;
        }
        public Gemeente FindBy(string postcode)
        {
            return gemeenten.Find(postcode);
        }
        public IQueryable<Gemeente> FindAll()
        {
            return gemeenten;
        }
    }
}
```

HoGent

Pag. 24

3. Data Access Layer



HoGent

Pag. 25

Dependency Injection

4. Dependency injection

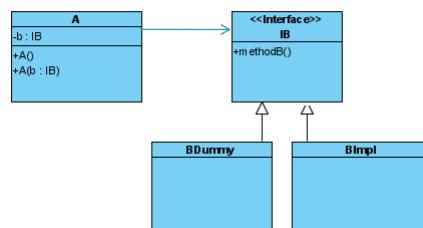
► Dependency injection

- Als klasse A een methode uit klasse B oproept heeft A een **dependency** op B
- Problemen bij unit test
 - Als je een unit test doet van klasse A voer je ook code van klasse B uit. Je test niet 1 maar meerdere klassen!
 - Als klasse B nog niet geschreven is kan je klasse A niet testen
 - Het is moeilijk het gedrag A te testen als B exceptions throwt
 - Testen met database zijn vaak te traag

4. Dependency injection

► Dependency injection

- Oplossing
 - Bij het unit testen van A zou A een **dummy implementatie** van B moeten gebruiken, die dezelfde methodes als B bevat maar in deze methodes zo weinig mogelijk doet
 - De tester van klasse A gaat dummyB schrijven
 - De dummy implementatie wordt door **dependency injection** in de code geplaatst. (low coupling)

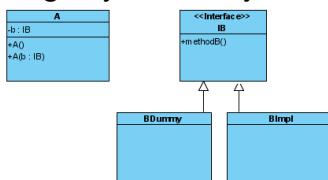


4. Dependency injection

► Dependency injection

- BImpl en BDummy implementeren eenzelfde interface IB
- Je drukt de dependency in A uit
 - Met een private variabele van het type IB
 - Een constructor met een parameter van het type IB en/of
 - Als bij instantiatie al gekend welke implementatie gebruikt wordt
 - Een set method met een parameter van het type IB
 - Als de gebruikte implementatie kan veranderen tijdens bestaan object
- Bij het echt uitvoeren geef je A een object BImpl
- Bij het testen geef je A een object BDummy

HoGent



Pag. 29

4. Dependency injection

► BrouwerController zal gebruik maken van de repositories

- Maak de BrouwerController aan
 - Rechtsklik op de map “Controllers”
 - Kies Add > Controller
 - Geef de naam “BrouwerController”
 - Kies Empty MVC Controller
 - Klik op Add
- Voorzie dependency injection van de repositories

HoGent

Pag. 30

4. Dependency injection

► Pas BrouwerController aan

```
namespace BierHalle.Controllers
{
    public class BrouwerController : Controller
    {
        private IBrouwerRepository brouwerRepository;
        private IGemeenteRepository gemeenteRepository;

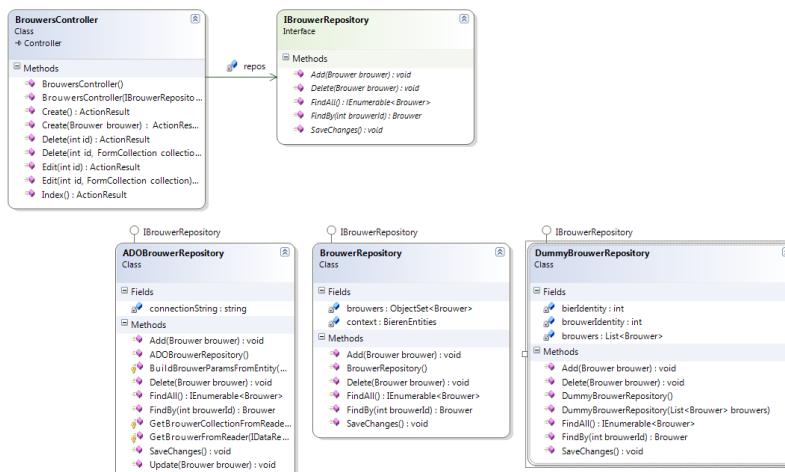
        public BrouwerController()
        {
            BierhalleContext context = new BierhalleContext();
            this.brouwerRepository = new BrouwerRepository(context);
            this.gemeenteRepository = new GemeenteRepository(context);
        }

        public BrouwerController(IBrouwerRepository brouwerRepository, IGemeenteRepository gemeenteRepository)
        {
            this.brouwerRepository = brouwerRepository;
            this.gemeenteRepository = gemeenteRepository;
        }
    }
}
```

Pag. 31

4. Dependency injection

- Dit betekent dat de Controller en Unit Test kan kiezen welke Repository gebruikt zal worden



Pag. 32

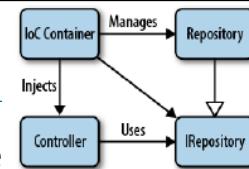
5. IoC Containers

- ▶ Ipv een tweede constructor
 - => gebruik **dependency injection container (IoC container)**
 - = software library die dienst doet als een factory voor componenten. Verantwoordelijk voor de creatie en life-cycle management van de afhankelijkheden die objecten in de applicatie nodig hebben.

Taak van IoC

```
public BrouwerController()  
{  
    BierhalContext context = new BierhalContext();  
    this.brouwerRepository = new BrouwerRepository(context);  
    this.gemeenteRepository = new GemeenteRepository(context);  
}  
  
public BrouwerController(IBrouwerRepository brouwerRepository, IGemeenteRepository gemeenteRepository)  
{  
    this.brouwerRepository = brouwerRepository;  
    this.gemeenteRepository = gemeenteRepository;  
}
```

Pag. 33



5. IoC Containers

- ▶ MVC voorziet IDependencyResolver, die geïmplementeerd moet worden voor IoC containers
- ▶ IoC (Inversion Of Control) Containers voor .Net :
 1. Ninject: <http://www.ninject.org>
 2. Castle Windsor: <http://www.castleproject.org/container/index.html>
 3. Autofac: <http://code.google.com/p/autofac/>
 4. StructureMap: <http://structuremap.net/structuremap/index.html>
 5. Unity: <http://unity.codeplex.com>
 6. MEF: <http://msdn.microsoft.com/en-us/library/dd460648.aspx>

5. IoC Containers

► Ninject:



Ninject is a lightning-fast, ultra-lightweight dependency injector for .NET applications. It helps you split your application into a collection of loosely-coupled highly-cohesive pieces, and then glue them back together in a flexible manner. By using Ninject to support your software's architecture, your code will become easier to write, reuse, test, and modify.

► Toevoegen aan project via Nuget

- Nuget = package installer
- Solution Explorer > Rechtsklik op Bierhalle project > Manage Nuget Packages > zoek naar Ninject en installeer Ninject en Ninject.MVC3



HoGent

35

5. IoC Containers

► Ninject

- Dit voegt toe
 - De references

■■ Ninject
■■ Ninject.Web.Common
■■ Ninject.Web.Mvc

- App_Start folder : NinjectWebCommon.cs
 - Bekijk de code
 - Ninject bootstrapper : beheert de modules, basic bootstrapper used to set up web applications

HoGent

36

5. IoC Containers

► IKernel : Ninject interface klasse

- = is the core van de Ninject framework
- Definieert de methodes
 - Get<T> : vragen van een instantie van een bepaald type (interface of concreet) aan Ninject
 - Bind<IT>.To<T> : Mappen van een interface op de te instantiëren klasse
 - InRequestScope() : object wordt 1 keer per HttpRequest geïnstantieerd
 - InSingletonScope() : object wordt 1 * in de applicatie geïnstantieerd
 - InThreadScope() : object wordt 1 * per thread geïnstantieerd
 - WithConstructorArgument : Binden aan klasse met constructor parameter
 - ninjectKernel.Bind<IT>
 - .To<T>.WithConstructorArgument("nameParameter", value)
 - WithPropertyValue : binden aan klasse en opgeven van property waarde
 - .WithPropertyValue("propertynname",value)
 - ToSelf() : self binding
 - ...

HoGent

37

5. IoC Containers

► Ninject

◦ StandardKernel :

- Implements IKernel. This class is the container. Bindings are added to the container. An instance of the StandardKernel can be used to request concrete instances of our service types. The StandardKernel instance can resolve the requested type to a type configured in the bindings. It then provides an instance of the resolved type. The StandardKernel manages the life time of the resolved type instances.

HoGent

38

5. IoC Containers

► Ninject

- App_Start folder : NinjectWebCommon.cs
 - Pas de methode RegisterServices aan

```
private static void RegisterServices(IKernel kernel)
{
    kernel.Bind<IBrouwerRepository>().To<IBrouwerRepository>().InRequestScope();
    kernel.Bind<IGemeenteRepository>().To<IGemeenteRepository>().InRequestScope();
    kernel.Bind<BierhalleContext>().ToSelf().InRequestScope();
}
```

- Meer op :
<https://github.com/ninject/ninject.web.mvc/wiki/MVC3>

5. IoC Containers

► Ninject

- Binden van types aan interfaces

```
kernel.Bind<IBrouwerRepository>().To<IBrouwerRepository>().InRequestScope();
```

- Ninject zal als het een request ontvangt voor een implementatie van IBrouwerRepository, een instantie van BrouwerRepository retourneren.
- Bind en To zijn beide methodes gedefinieerd in IKernel interface
- Bij aanroepen van een constructor die als parameter een interface bevat zal Ninject de juiste instantie aanmaken. BrouwerController bevat bijgevolg maar 1 constructor meer!

```
public BrouwerController(IBrouwerRepository brouwerRepository, IGemeenteRepository gemeenteRepository)
{
    this.brouwerRepository = brouwerRepository;
    this.gemeenteRepository = gemeenteRepository;
}
```

5. IoC Containers

► Ninject

◦ Self binding

- Vragen van een concrete klasse aan Ninject. Ninject zal default een instantie van die klasse zelf aanmaken.
 - Je hoeft dus geen bindings aan te maken voor instantiatie van concrete klassen (zonder interface).
 - Je kan de default werking wel overschrijven
 - `ninjectKernel.Bind<IBierhalleContext>.ToSelf().InRequestScope()`
 - Maakt maar 1 BierhalleContext aan per request!!!!
 - Of je zou bepaalde argumenten kunnen meegeven

HoGent

41



BDummy?
Mocking

Mock Objects:

**Mock objects are simulated objects
that mimic the behavior of real
objects in controlled ways**

-Wikipedia



6. Mocking

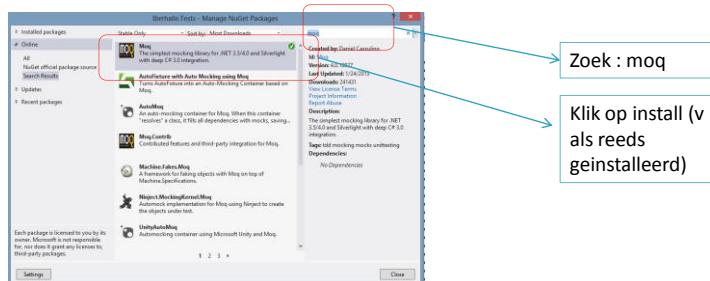
- ▶ Hoe BDummy aanmaken?
 - Mock
 - Eenvoudige dummy implementatie van een interface
 - Laat verificatie toe: controleren of de dependencies
 - opgeroepen werden met de juiste parameters
 - Een juist aantal keer opgeroepen werden. Een class die zijn dependencies niet oproept, is niet OK.
 - Je hoeft een Mock class niet zelf te schrijven. Er bestaan Mock frameworks die Mock klassen voor je maken
 - Typemock, Rhino Mocks, Moq, ...
 - In de oefeningen maken we gebruik van moq.
 - <http://code.google.com/p/moq>



6. Mocking

► Installeren van Moq

- Nuget = package installer. Voegt een framework toe aan je project
- Solution Explorer > Bierhalle.Tests > Manage nuget packages



- Moq.dll wordt toegevoegd aan de References

HoGent

Pag. 45

6. Mocking

- Stap 1 : Zorg voor een Interface klasse
 - Voorbeeld IBrouwerRepository, IGemeenteRepository
- Stap 2 : Voeg dependency injection toe via constructor of via een setter
 - In de BrouwerController : via de constructor
- Stap 3 : Maak testklasse aan
 - Installeer moq package via Nuget
 - Maak een testklasse aan voor BrouwerController
 - Rechtsklik op het unit test project, Folder Controllers > Add > New item > Visual C# > Test > Kies Basic Unit Test > geef naam BrouwerControllerTest.cs
 - Voeg using statement toe : using Moq;

HoGent

46

6. Mocking

- ▶ Stap 4 : Maak de testgevallen aan
 - Wat hebben we nodig? Zie testscenario's
 - Een lijst van brouwers en gemeenten.
 - We maken een DummyDataContext aan met de testgevallen uit scenario's
 - Bestaat reeds , maar nog toevoegen aan project. Klap Controllers folder in test project open. Klik in toolbar van Solution Explorer op « Show All» incoontje. Toont DummyDataContext.cs. RechtsKlik > Include in Project
 - Instantieer. Hier in TestInitialize

HoGent

```
[TestClass]
public class BrouwerControllerTest
{
    [TestInitialize]
    public void SetUpContext()
    {
        DummyDataContext context = new DummyDataContext();
```

47

6. Mocking

- ▶ Stap 5 : Maak Mock aan in de testklasse
 - Declareer en instantieer mock en injecteer mock in het te testen object. Hier in de TestInitialize methode

```
[TestClass]
public class BrouwerControllerTest
{
    private BrouwerController controller;
    private Mock<IBrouwerRepository> mockBrouwerRepository;
    private Mock<IGemeenteRepository> mockGemeenteRepository;
```

Declaratie

```
[TestInitialize]
public void SetUpContext()
{
    DummyDataContext context = new DummyDataContext(); ;
    mockBrouwerRepository = new Mock<IBrouwerRepository>();
    mockGemeenteRepository = new Mock<IGemeenteRepository>();
    controller = new BrouwerController(mockBrouwerRepository.Object, mockGemeenteRepository.Object);
```

Instantiatie

```
}
```

Injectie

HoGent

48

6. Mocking

► Stap 5 : Maak Mock aan in de testklasse

```
mockBrouwerRepository = new Mock<IBrouwerRepository>();
mockGemeenteRepository = new Mock<IGemeenteRepository>();
controller = new BrouwerController(mockBrouwerRepository.Object, mockGemeenteRepository.Object);
```

- Mock is een proxy class (die zogezegd IBrouwerRepository implementeert)
- Object is een instantie van de implementatie

6. Mocking

► Stap 6 : Train de mock

- Voeg onderstaande methode toe aan de BrouwerController (bij wijze van voorbeeld)
 - Deze methode zal alle brouwers ophalen uit de BrouwerRepository, gesorteerd op naam

```
public IEnumerable<Brouwer> GetBrouwers()
{
    throw new NotImplementedException();
}
```

- Alvorens we de methode implementeren schrijven we eerst de testen
- Deze methode zal de methode GetAll uit de BrouwerRepository aanroepen. En deze retourneert alle brouwers. => We zullen onze mock moeten trainen zodat bij aanroep van de methode GetAll() alle brouwers gereturneerd worden

6. Mocking

► Stap 6 : Train de mock

- Train de mock, zodat bij aanroep van de methode FindAll() alle brouwers geretourneerd worden.
- Train de mock
 - Setup(x).Returns(y)
 - Setup(x).Throws(z)
 - x : de methode uit de mock die je aanroeft met zijn parameterwaarden
 - y : wat de methode zal retourneren
 - z : de exception die de methode dan zal retourneren
- Het trainen kan je in de Initialize doen of in de testmethodes zelf.

6. Mocking

► Stap 6 : Train de mock

- Enkele voorbeelden
 - Alle brouwers retourneren

```
mockBrouwerRepository.Setup(m => m.FindAll()).Returns(context.BrouwerLijst);
```

- Brouwer met id 1

```
mockBrouwerRepository.Setup(m => m.FindBy(1))  
    .Returns(context.BrouwerLijst.FirstOrDefault(b=>b.BrouwerId==1));
```

- Onbestaande brouwer. (Mock retourneert null indien niet getraind voor een bepaald geval, maar je kan hier ook een regel voor maken). Als de methode null retourneert, moet je de null casten naar het juiste type

```
mockBrouwerRepository.Setup(m => m.FindBy(10)).Returns((Brouwer)null);
```

6. Mocking

► Stap 6 : Train de mock

- Enkele voorbeelden
 - Als de methode void retourneert heb je geen Returns

```
mockBrouwerRepository.Setup(m => m.Add(new Brouwer("test")));
```

- Als de methode een exception throwt, gebruik Throws

```
mock.Setup(m => m.Add(productInCart)).Throws(new ApplicationException ("product bestaat reeds"));
```

6. Mocking

► Stap 7 : Schrijf de test

- De testmethode

```
[TestMethod]
public void GetBrouwersMustReturnBrouwers()
{
    IEnumerable<Brouwer> brouwers = controller.GetBrouwers();
    Assert.AreEqual(7, brouwers.Count());
}
```

Merk op : Count() is een extension methode. I~~E~~numerable bevat geen property Count zoals ICollection, IList.

- Je kan ook verifiëren of een methode uit mock werd aangeroepen

6. Mocking

► Stap 8 : Verifieer

- Na elke test – Verify : Ga na of de test wel effectief gebruik gemaakt heeft van de juiste methodes uit de repository

```
// Method should be called with specified parameter  
mock.Verify(foo => foo.Execute("ping"))  
  
// Method should never be called  
mock.Verify(foo => foo.Execute("ping"), Times.Never());  
  
// Called at least once  
mock.Verify(foo => foo.Execute("ping"), Times.AtLeastOnce());  
  
// Verify getter invocation of property Name, regardless of value.  
mock.VerifyGet(foo => foo.Name);  
  
// Verify setter invocation of property Name, regardless of value.  
mock.VerifySet(foo => foo.Name);  
  
// Verify setter called with specific value  
mock.VerifySet(foo => foo.Name = "foo");  
  
// Verify setter with an argument matcher  
HoGent mock.VerifySet(foo => foo.Value = It.IsInRange(1, 5, Range.Inclusive));
```

55

6. Mocking

► Stap 8 : Verifieer

- Met Verify

```
[TestMethod]  
public void GetBrouwersMustReturnBrouwers()  
{  
    IEnumerable<Brouwer> brouwers = controller.GetBrouwers();  
    Assert.AreEqual(7, brouwers.Count());  
    mockBrouwerRepository.Verify(m=>m.FindAll(), Times.Once());  
}
```

6. Mocking

- ▶ Stap 9 : Run test – faalt
- ▶ Stap 10 : Pas de methode GetBrouwers aan

```
public IEnumerable<Brouwer> GetBrouwers()
{
    return brouwerRepository.FindAll().OrderBy(b => b.Naam);
}
```

- ▶ Stap 11 : Run – Test slaagt



6. Mocking

- ▶ Generischer maken van setup
 - - It : laat toe om matching conditie op te geven
 - Mogelijkheden

```
//IsAny<T> matches if parameter is any instance of type T
mock.Setup(foo => foo.Execute(It.IsAny<string>())).Returns(true);

// Is<T> matches based on a specified predicate
mock.Setup(foo => foo.Add(It.Is<int>(i => i % 2 == 0))).Returns(true);

// IsInRange<T> matches if parameter is between the defined values
mock.Setup(foo => foo.Add(It.IsInRange<int>(0, 10, Range.Inclusive))).Returns(true);

// IsRegex : matches a string parameter if it matches the specified regular expression
mock.Setup(x => x.Execute(It.IsRegex("[a-d]+", RegexOptions.IgnoreCase))).Returns("foo");
    ◦ Meer op : http://code.google.com/p/moq/wiki/QuickStart
```

**If it's worth building, it's
worth testing.**



**If it's not worth testing,
*why are you wasting your
time working on it?***



7A. De Controller

► Controllers, Action Methods en Routing

- Controllers groeperen verschillende Actions Methods
 - Het is de bedoeling dat action methods op een logische manier in verschillende controllers samen te brengen
- De Action Methods voeren de door de client gevraagde requests uit
 - Updaten het model waar nodig
 - Communiceren met de DAL
 - en vragen aan View om model te renderen
- Wanneer wordt welke Action Method aangeroepen ?
 - Hier zorgt het Routing System voor
 - D.i. een stuk .NET MVC infrastructuur dat binnenvkomende HTTP requests analyseert en de juiste Action Method in de juiste Controller aanroeft.
 - Ook bijkomende informatie (parameters) wordt geanalyseerd door de routing en doorgegeven

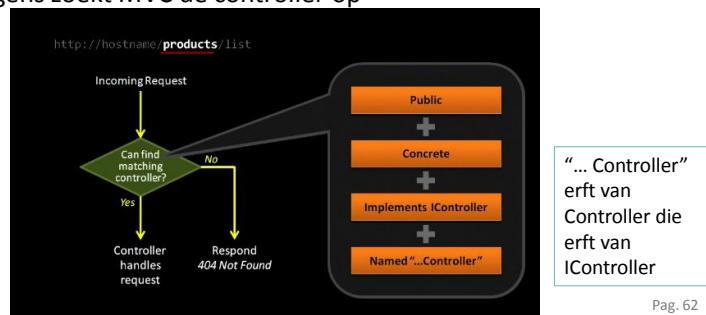
HoGent

Pag. 61

7A. De Controller

► De routing : Hoe vindt MVC de Controller?

- De routing configuratie wordt opgegeven in *Global.asax.cs*
 - URL's met syntax **X/Y/id** worden gerouteerd naar
 - Controller **XController** – of naar HomeController indien X ontbreekt
 - Actiemethode **Y(...)** – of naar Index(...) indien Y ontbreekt
 - Met parameter **id** – geen fout indien id ontbreekt
 - Vervolgens zoekt MVC de controller op



HoGent

Pag. 62

7A. De Controller

- ▶ De routing : Hoe vindt MVC de Controller?
 - Global.asax.cs, Application_Start event

```
RouteConfig.RegisterRoutes(RouteTable.Routes);
```

- App_Config/RouteConfig.cs

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
    );
}
```

7A. De Controller

- ▶ URL design

URL's	Controller	Action Method
/Brouwer/Index	BrouwerController	Index()
/Brouwer		
/Brouwer/Create	BrouwerController	Create()
/Brouwer/Delete/12	BrouwerController	Delete(int id)
/Brouwer/Edit/12	BrouwerController	Edit(int id)

- ▶ Welke routing nodig om dit URL design te realiseren ?

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Brouwer", action = "Index", id = UrlParameter.Optional }
```

7A. De Controller

► De routing : Hoe vindt MVC de Controller?

- App_Config/RouteConfig.cs
 - IgnoreRoutes : routing wordt niet uitgevoerd indien een file op het filesysteem wordt opgevraagd, of i.g.v. .axd files (tracing files, bestaan niet op file systeem, zijn virtuele files)
 - Kan meerdere routes bevatten.
 - Bij binnenkomende request wordt in de routing tabel gezocht naar de eerste match (volgorde is belangrijk)
- Voorbeeld

```
public static void RegisterRoutes(RouteCollection routes)
{
    routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
    routes.MapRoute(
        name: "Regions",
        url: "Regions/{name}",
        defaults: new { controller = "Brouwer", action = "Search", name = UrlParameter.Optional }
    );
    routes.MapRoute(
        name: "Default",
        url: "{controller}/{action}/{id}",
        defaults: new { controller = "Brouwer", action = "Index", id = UrlParameter.Optional }
    );
}
```

url /Regions/Oostvlaanderen =>
Controller Brouwers,
Action=Search, parameter
name=Oostvlaanderen

HoGent

Pag. 65



7A. De Controller

- ▶ Startpage(URL "/")=> BrouwerController, action Index()
 - De startpagina van de BierHalle geeft een overzicht van alle brouwers met mogelijkheid om nieuwe brouwers aan te maken of bestaande te editeren of te verwijderen.
 - Methode Index()



7A. De Controller

- ▶ De action method
 - De action methode moet de **request afhandelen**
 - In dit vb : de brouwers ophalen (via de repository)
 - En moet een **antwoord** (response) **terugsturen** naar de gebruiker.
 - De response kan een HTML pagina zijn, maar ook een XML bestand, een JSON bestand, een HTTP redirect , ...
 - De controller maakt hiervoor een instantie aan van **ActionResult**. Hoe het renderen naar de browser gebeurt is *niet* de verantwoordelijkheid van de Controller. De controller dient enkel de data die moet gerenderd worden in de browser en de naam van de view, waarin de data wordt gemerged doorgeven aan ActionResult.

7A. De Controller

► De action method

- Retourneert steeds een ActionResult
 - Een abstracte basisklasse met verschillende concrete implementaties
 - Elke implementatie stelt een andere afhandeling van request voor
 - Een ActionResult zal door het framework worden uitgevoerd (Command pattern) en zal een resultaat terugsturen naar de browser. De controller moet hem hiervoor alle nodige informatie aanleveren

ActionResult()
ExecuteResult(System.Web.Mvc.ControllerContext)

```
public abstract void ExecuteResult(System.Web.Mvc.ControllerContext context)
    Member of System.Web.Mvc.ActionResult
```

Summary:
Enables processing of the result of an action method by a custom type that inherits from the System.Web.Mvc ActionResult class.

Parameters:
context: The context in which the result is executed. The context information includes the controller, HTTP content, request context, and route data.

Dit kan je bekijken in de Object Browser. Menu > View > Object browser

HoGent

Pag. 69

7A. De Controller

► De action method

- ActionResult subklassen

ACTIONRESULT TYPE	DESCRIPTION
ContentResult	Writes the specified content directly to the response as text.
EmptyResult	Represents a null or empty response. It doesn't do anything.
FileContentResult	Derives from FileResult and writes a byte array to the response.
FilePathResult	Derives from FileResult and writes a file to the response based on a file path.
FileResult	Serves as the base class for a set of results that writes a binary response to the stream. Useful for returning files to the user.
FileStreamResult	Derives from FileResult and writes a stream to the response.
HttpNotFound	Derives from HttpStatusCodeResult. Returns an HTTP 404 response code to the client, indicating that the requested resource is not found.
HttpStatusCodeResult	Returns a user-specified HTTP code.

HoGent

Pag. 70

7A. De Controller

► De action method

- ActionResult subklassen

ACTIONRESULT TYPE	DESCRIPTION
HttpUnauthorizedResult	Derives from HttpStatusCodeResult. Returns an HTTP 401 response code to the client, indicating that the requestor does not have authorization to the resource at the requested URL.
JavaScriptResult	Used to execute JavaScript code immediately on the client sent from the server.
JsonResult	Serializes the objects it is given into JSON and writes the JSON to the response, typically in response to an Ajax request.
PartialViewResult	This is similar to ViewResult, except it renders a partial view to the response, typically in response to an Ajax request.
RedirectResult	Redirects the requestor to another URL by returning either a temporary redirect code 302 or permanent redirect code 301, depending upon a Boolean Permanent flag.
RedirectToRouteResult	Similar to RedirectResult, but redirects the user to a URL specified via Routing parameters.
ViewResult	Calls into a view engine to render a view to the response.

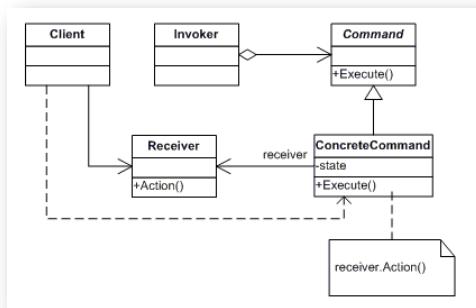
HoGent

Pag. 71

7A. De Controller

► De action method

- ActionResult en het command pattern
 - Command = ActionResult met ExecuteResult methode
 - ConcreteCommand = ViewResult, JsonResult,....



HoGent

Pag. 72

7A. De Controller

► De action method

◦ ActionResult

- De Controller beschikt over **helper methodes** die een dergelijke instantie aanmaken en uitvoeren :

METHOD	DESCRIPTION
Redirect	Returns a <code>RedirectResult</code> , which redirects the user to the appropriate URL.
RedirectPermanent	The same as <code>Redirect</code> but returns a <code>RedirectResult</code> with the <code>Permanent</code> property set to <code>true</code> , thus returning an HTTP 301 status code.
RedirectToAction	Returns a <code>RedirectToRouteResult</code> , which redirects the user to an action using the supplied route values.
RedirectToActionPermanent	The same as <code>RedirectToAction</code> but returns a <code>RedirectResult</code> with the <code>Permanent</code> property set to <code>true</code> , thus returning an HTTP 301 status code.
RedirectToRoute	Returns a <code>RedirectToRouteResult</code> , which redirects the user to the URL that matches the specified route values.
RedirectToRoutePermanent	The same as <code>RedirectToRoute</code> but returns a <code>RedirectResult</code> with the <code>Permanent</code> property set to <code>true</code> , thus returning an HTTP 301 status code.

HoGen

Pag. 73

7A. De Controller

► De action method

◦ ActionResult – helper methodes Controller

METHOD	DESCRIPTION
View	Returns a <code>ViewResult</code> , which renders the view to the response.
PartialView	Returns a <code>PartialViewResult</code> , which renders a partial view to the response.
Content	Returns a <code>ContentResult</code> , which writes the specified content (string) to the response.
File	Returns a class that derives from <code>FileResult</code> , which writes binary content to the response.
Json	Returns a <code>JsonResult</code> containing the output from serializing an object to JSON.
JavaScript	Returns a <code>JavaScriptResult</code> containing JavaScript code that is immediately executed when returned to the client.

HoGent

Pag. 74

7A. De Controller

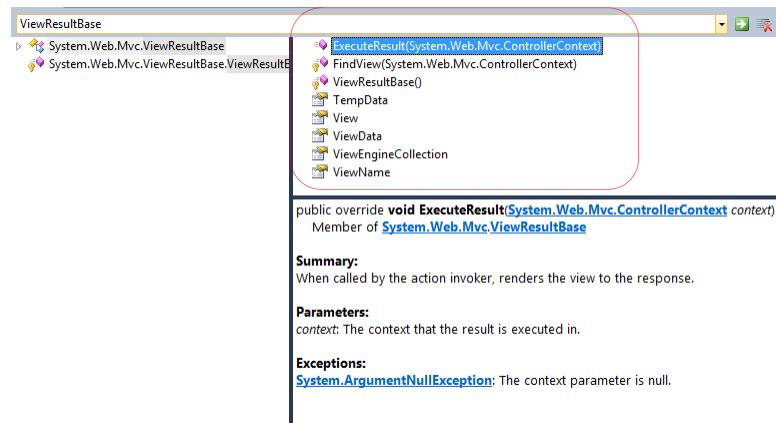
► De action method

- ViewResult
 - Wensen we html terug te sturen naar de browser => ViewResult
 - ViewResult zal een View renderen aan de hand van gegevens die het van de Controller heeft ontvangen (ViewName, ViewBag, TempData)
 - Een View (ViewPage) is een template die HTML en inline code bevat. De View wordt gerendered => genereert html pagina voor browser
 - Als de controller methode View() aanroeft zonder ViewName parameter dan wordt de controller's Default View gerendered. Het framework zoek in volgorde naar
 - ~\Views\{ControllerNaam\}\{ActionMethodNaam}.cshtml
 - ~\Views\Shared\{ActionMethodNaam}.cshtml
 - In ons concreet geval wordt dat
 - ~\Views\Brouwer\Index.cshtml
 - ~\Views\Shared\Index.cshtml
 - Indien niet gevonden => runtimefout

7A. De Controller

► De action method

- ActionResult : ViewResult erft van ViewResultBase



7A. De Controller

- ▶ Doorgeven van informatie Controller → View
 - De controller en de View delen een object **ViewData** van type **ViewDataDictionary**. De Controller maakt het object aan en geeft dit door aan de view. De view maakt van dit object gebruik om de data te renderen. Data kan doorgegeven worden via
 - ViewData dictionary
 - ViewData["bierId"] = bier.BierId
 - ViewData["biernaam"] = bier.Naam
 - Vanaf MVC3 wordt ViewBag gebruikt. Dit een wrapper rond ViewData
 - ViewBag.BierId = bier.BierId;
 - Wordt eerder gebruikt om boodschappen en 'extra informatie' door te geven
 - Strongly typed views (ViewPage<T>) => geef je **Model T** door
 - Beschikbaar via een strongly typed property op ViewData, nl. ViewData.Model
 - In Model property worden meestal de weer te geven gegevens doorgegeven. In ons voorbeeld is het type **IEnumerable<Brouwer>**

HoGent

Pag. 77

7B. TDD van de Controller

- ▶ TDD van Controller
 - Wat verwachten we van een actie?
 - Moet request afhandelen : domein aanpassen
 - Moet een ActionResult aanmaken en de nodige informatie doorgeven aan de ActionResult

Naam	Straat	Omzet	AantalBieren	
Bavik	Rijksweg 33	2000000	6	Edit Details Delete
De Graal		0	0	Edit Details Delete
De Leeuw		0	0	Edit Details Delete
Duvel Moortgat	Breendonk dorp 28		6	Edit Details Delete
InBev	Brouwerijplein 1		5	Edit Details Delete
Palm Breweries		500000	4	Edit Details Delete
Roman	Hauwaart 105		4	Edit Details Delete

HoGent

Pag. 78

7B. Index : TDD van de Controller

► TDD van Controller

- Wat verwachten we van de Index action methode?
 1. De default View wordt gerendered.
 2. De informatie die de View nodig heeft, hier een lijst van brouwers wordt doorgegeven.

7B. Index : TDD van de Controller

► Test 1 : Moet de default view renderen

```
[TestMethod]
public void IndexUsesConventionsToChooseView()
{
    ViewResult result = controller.Index() as ViewResult;
    Assert.IsNotNull(result);
    Assert.IsTrue(String.IsNullOrEmpty(result.ViewName));
}
```

7B. Index : TDD van de Controller

► Test 2 :

- moet als Model een `IEnumerable<Brouwer>` doorgeven.
- In de repos zijn er 7 brouwers

```
[TestMethod]
public void IndexMustReturnBrouwers()
{
    ViewResult result = controller.Index() as ViewResult;
    Assert.IsInstanceOf(result.Model, typeof(IEnumerable<Brouwer>));
    Assert.AreEqual(7, (result.Model as IEnumerable<Brouwer>).Count());
    mockBrouwerRepository.Verify(m => m.FindAll(), Times.Once());
}
```

7B. Index : TDD van de Controller

► Index action methode

- Pas de code aan:
 - Onze startpagina moet een overzicht van brouwers geven
 - Deze komen uit de repository

```
public ActionResult Index()
{
    IEnumerable<Brouwer> brouwers = brouwerRepository.FindAll();
    return View(brouwers);
}
```

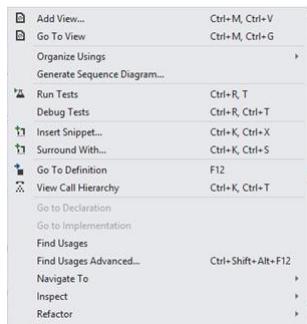
Stelt de Model property in van de View

- De testen slagen
- Je kan de lijst sorteren via `OrderBy` extension method => Vergeet using `System.Linq` niet! anders is de methode `OrderBy` niet gekend

7C. Index : View

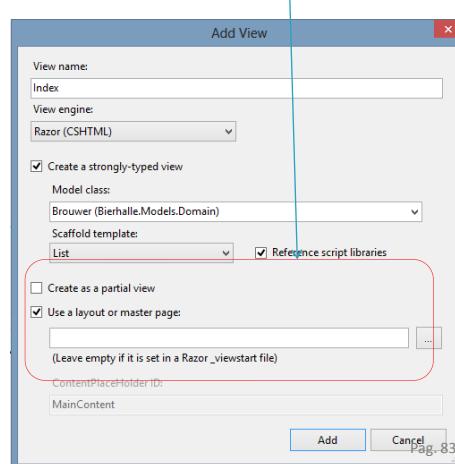
► Aanmaken van de View

- Rechtsklikken in de action method
- Kies Add View...
- Vul in zoals afgebeeld



HoGent

Bepaalt de master layout voor een website (bepaalt de look en feel van de site : header, footer, content gedeelte, css). Meer hierover op het einde hoofdstuk.



Pag. 83

7C. Index : View

► Scaffolding – List : VS genereert een pagina met een tabel

```
@foreach (var item in Model) {  
    <tr>  
        <td>  
            @Html.DisplayFor(modelItem => item.Naam)  
        </td>  
        <td>  
            @Html.DisplayFor(modelItem => item.Straat)  
        </td>  
        <td>  
            @Html.DisplayFor(modelItem => item.Omzet)  
        </td>  
        <td>  
            @Html.DisplayFor(modelItem => item.AantalBieren)  
        </td>  
        <td>  
            @Html.ActionLink("Edit", "Edit", new { id=item.BrouwerId }) |  
            @Html.ActionLink("Details", "Details", new { id=item.BrouwerId }) |  
            @Html.ActionLink("Delete", "Delete", new { id=item.BrouwerId })  
        </td>  
    </tr>
```

Generatie van rows in de table

Links voor editeren, ...

HoGent

Pag. 84

7C. Index : View



Scaffolding is a temporary structure used to support people and material in the [construction](#) or [repair](#) of [buildings](#) and other large structures. It is usually a modular system of [metal pipes](#) or tubes, although it can be made out of other materials

Scaffolding is a [meta-programming](#) method of building [database](#)-backend [software applications](#). It is a technique supported by some [model-view-controller frameworks](#), in which the programmer may write a specification that describes how the application database may be used. The [compiler](#) uses this specification to generate code that the application can use to [create, read, update and delete](#) database entries, effectively treating the template as a "[scaffold](#)" on which to build a more powerful application.

Scaffolding is an evolution of database code generators from earlier development environments, such as Oracle's CASE Generator, and many other 4GL client-server software development products.

Scaffolding was popularized by the [Ruby on Rails](#) framework. It has been adapted to other software frameworks, including [Django](#), [Monorail \(.Net\)](#), [Symfony](#), [Yii](#), [CakePHP](#), [Model-Glue](#), [Grails](#), [Catalyst](#), [Seam Framework](#), [ASP.NET Dynamic Data](#) and [ASP.NET MVC](#) Framework's Metadata Template Helpers.

7C. Index : View

- ▶ De View is van type `ViewPage<T>` en is strongly typed
 - het type van Model = het type dat de Controller doorgeeft aan de View

```
@model IEnumerable<BierHalle.Models.Domain.Brouwer>
```

- Kan je ook schrijven als
 @using Bierhalle.Models
 @model IEnumerable<Brouwer>
- Je krijgt IntelliSense in de editor. In inline code kan je Model gebruiken. Afhankelijk van het type van Model
 - Model is een object : `@Model.Title`
 - Model is een collectie dan kan je de collectie overlopen :

```
@foreach (var item in Model) {  
    @item.Gemeente  
}
```

Late binding => Type inference, wordt Brouwer type

7C. Index : View

► Run de applicatie

- De flow
 - Controller maakt ViewResult aan. De ViewResult roept de ViewEngine aan om de view te renderen



- De opmaak wordt gedefinieerd door
 - Master page or layout page : Views/shared/_layout.cshtml (zie later)
 - Css : Contents/site.css

7C. Index : View

► Razor

- Transition character van markup naar C# code : @
 - Code expressions : @, gevolgd door 1 C# instructie (geen ; op einde)
 - Code block : @{ }

```
@foreach (var item in Model) {  
    <tr>  
        <td>  
            @Html.DisplayFor(modelItem => item.Naam)  
        </td>  
    </tr>  
}
```

```
@{  
    string rootNamespace = "MyApp";  
}
```

- Wat met @ in tekst ? Escapen, gebruik @@, behalve voor email. Razor herkent email patroon
- In commentaar zetten van blok code : @* *@
- Code expressions mogen geen void retourneren, anders runtime fout. Gebruik hier dan een code block

7C. Index : View

► Razor

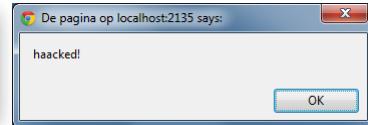
- @ : zorgt voor html encoding
 - -> geen cross-site script injection attack

```
@{  
    string message = "<script>alert('haacked!');</script>";  
}  
<div>@message</div>
```

```
<script>alert('haacked!');</script>
```

- Gebruik @Html.Raw => geen encoding

```
@{  
    string message = "<script>alert('haacked!');</script>";  
}  
<div>@Html.Raw(message)</div>
```



7C. Index : View

► Razor

- Als in een code blok een lijn begint met tekst (geen C# code, geen html) => tekst laten voorafgaan door @: of <text> tag

```
@if (item.Omzet > 1000)  
{  
    @:Omzet : @item.Omzet <span style="font-weight:bold">Splash!</span>  
}  
else  
{  
    <h5>@item.Omzet</h5>  
}  
  
@if (item.Omzet > 1000)  
{  
    <text>Omzet : </text>@item.Omzet <span style="font-weight:bold">Splash!</span>  
}  
else  
{  
    <h5>@item.Omzet</h5>  
}
```

7C. Index : View

► Razor

- De automatische overgang van code naar markup is een + van Razor, maar soms moet je Razor een handje helpen door () te gebruiken

```
@{  
    string rootNamespace = "MyApp";  
}  
  
<span>@rootNamespace.Models</span>
```

Runtime fout. Reden : transitie van code naar markup pas na .Model.
Dus Models wordt hier gezien als property van MyApp.

: 'string' does not contain a definition for 'Models' and no extension method 'Models' accepting a first argument of type 'string' could be found (are you missing a using directive or an assembly reference?)

```
@{  
    string rootNamespace = "MyApp";  
}  
  
<div>@(rootNamespace).Models</div>
```

MyApp.Models

Nu wordt .Models niet als code
gezien maar als literal tekst

7C. Index : View

► Html helpers

Van type System.Web.Mvc.HtmlHelper<T>, met T is Model van de view

- Extension methods die helpen bij de generatie van html

- Het **Html** property van **ViewPage** biedt toegang tot deze methods

- `Html.ActionLink (tekst : string, actionnaam : string, routevalues : object)`

```
@Html.ActionLink("Edit", "Edit", new { id=item.BrouwerId })
```

De routing definitie in RouteConfig.cs legt op dat de naam van de 3de par "id" is

- genereert bvb `Edit`
- `routeValues` = anonymous object. MVC framework beschouwt dit als naam - waarde paar, en voegt de key en waarde toe aan de `RouteData` (zie ook volgende slide)

7C. Index : View

▶ Html helpers

◦ Html.DisplayFor

- ASP.NET MVC bevat built-in display en editor templates.
 - bool => checkbox
 - Email address => mailto: anchor tag
 - Password => <input type="password" /> textbox
 - Number => <input type="number"> textbox
 - Je kan zelf ook templates aanmaken, zie later
- DisplayFor zal pogingen voor die bepaalde property de juiste template toe te passen
- Properties kan je hiervoor opmaken met data annotaties ([DataType(DataType.Password)]), ook Currency, Date, Time, MultiLineText,... mogelijk, zie later)

HoGent

```
[DataType(DataType.Currency)]
public decimal? Omzet
```

```
Omzet A
€ 20.000.000,00 6
```

Pag. 93

7C. Index : View

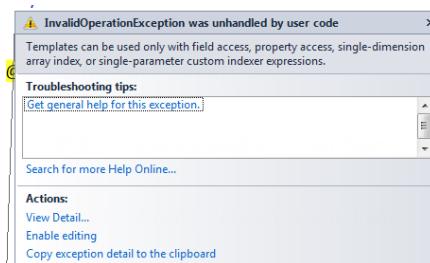
▶ We passen de View nog verder aan

- De tabel moet ook de postcode en gemeente tonen
 - Voeg aan de <table> een kolom toe voor gemeente (th en td).
Plaats in td

```
<td>
    @Html.DisplayFor(modelItem => item.Gemeente.Postcode + " " + item.Gemeente.Naam)
</td>
```

- Runnen geeft een runtimefout :

HoGen



```
@Html.DisplayFor(modelItem => item.Gemeente.Postcode + " " + item.Gemeente.Naam)
```

Pag. 94

7C. Index : View

- ▶ We passen de View nog verder aan
 - De tabel moet ook de postcode en gemeente tonen
 - Oplossing

```
<td>
@if (item.Gemeente!=null) {
    @String.Format("{0} {1}", item.Gemeente.Postcode ,item.Gemeente.Naam);
}
</td>
```
 - ! FindAll bevat een Include(b=>b.Gemeente)!!!!
 - Omzet en aantalBieren rechts aligneren
 - Style attribuut of werken met css class

```
<td style="text-align:right">
```
 - Run applicatie

HoGent

Pag. 95

7D. Index : Extra vraag klant

- ▶ Extra vraag van de klant :
 - Kan je ook de totale omzet tonen?
- ▶ TDD : Dus eerst de test hiervoor aanmaken
 - De Actie moet dus ook de totale omzet doorgeven aan de View.
 - Hoe?
- ▶ ViewData dictionary
 - ViewBag (ViewData) is handig om extra informatie van de Controller naar de View door te geven die niet in het Model vorhanden is
 - Voorbeeld
 - ViewBag.Message= “Hello world”;
 - ViewBag.BrouwerJan = new Brouwer(“Jan”);

HoGent

Pag. 96

7D. Index : Extra vraag klant

- ▶ TDD : We schrijven eerst een test

```
[TestMethod]
public void IndexMustReturnTotaleOmzet()
{
    ViewResult result = controller.Index() as ViewResult;
    Assert.AreEqual(20500000, result.ViewBag.TotaleOmzet);
}
```

7D. Index : Extra vraag klant

- ▶ Pas de Index methode aan

```
public ActionResult Index()
{
    IEnumerable<Brouwer> brouwers = brouwerRepository.FindAll().OrderBy(b=>b.Naam);
    ViewBag.TotaleOmzet = brouwers.Sum(b => b.Omzet);
    return View(brouwers);
}
```

- Run alle testen : Ze slagen ALLEMAAL!



- Pas View aan, onder de end tag table

- Nadeel ViewBag : je hebt geen IntelliSense in de View, let op schrijfwijze (typfouten geven geen compilatiefouten en ook geen runtimefouten. Toont dan niets)
- Let op de @(). Ronde haakjes nodig!

De totale omzet : `@(String.Format ("{0:F} €", ViewBag.TotaleOmzet))`

7E. Create – GET : TDD van Controller

► Create action : GET

- Een inputformulier voor de creatie van een brouwer
- De Url
 - /Brouwer/Create
- Wat?
 - Render de default view.
 - Geef een nieuwe instantie van de Brouwer door aan de View.
 - De pagina moet echter ook een dropdownlijst van de gemeenten tonen zodat de gebruiker ook een gemeente kan kiezen.
 - => naast een nieuwe brouwer dienen we ook de lijst van alle gemeenten door te geven. Maak hiervoor gebruik van een SelectList
 - Hoe? Via ViewBag. (Beter met **ViewModel**, zie later)

HoGent

Pag. 99

7E. Create – C

► Create action : GET

- We schrijven eerst de testen

De testen zijn reeds aangemaakt. Zie BrouwerControllerCDUTest.cs. Plaats region Create uit commentaar.

In [TestInitialize] voeg je toe :

```
mockGemeenteRepository.Setup(m =>
    m.FindAll()).Returns(context.GemeenteLijst {
```

```
#region Create
[TestMethod]
public void CreateMustUseConventionToChooseView()
{
    ViewResult result = controller.Create() as ViewResult;
    Assert.IsNotNull(result);
    Assert.IsTrue(String.IsNullOrEmpty(result.ViewName));
}

[TestMethod]
public void CreateMustReturnNewBrouwer()
{
    ViewResult result = controller.Create() as ViewResult;
    Assert.IsInstanceOfType(result.Model, typeof(Brouwer));
    Assert.AreEqual(0, (result.Model as Brouwer).BrouwerId);
}

[TestMethod]
public void CreateMustReturnSelectListOfGemeenten()
{
    ViewResult result = controller.Create() as ViewResult;
    Assert.IsInstanceOfType(result.ViewBag.Postcode, typeof SelectList);
    mockGemeenteRepository.Verify(m=>m.FindAll(), Times.Once());
}
```

HoGent

F 05. 100

7E. Create – GET : TDD van Controller

► Create action : GET

- Implementeer Create methode

```
public ActionResult Create()
{
    ViewBag.Postcode =
        new SelectList(gemeenteRepository.FindAll().OrderBy(g => g.Naam), "Postcode", "Naam");
    return View(new Brouwer());
}
```

- Testen slagen
- Refactor

7E. Create – GET : TDD van Controller

► SelectList

Object Browser

```
public SelectList(System.Collections.IEnumerable items, System.String dataValueField, System.String dataTextField)
```

Member of [System.Web.Mvc.SelectList](#)

Summary:

Initializes a new instance of the System.Web.Mvc.SelectList class by using the specified items for the list, the data value field, and the data text field.

Parameters:

items: The items.

dataValueField: The data value field.

dataTextField: The data text field.

- *dataValueField* = de naam van de property die de waarde bevat die moet worden gereturneerd bij selectie van een item (moet een unieke waarde bevatten, hier postcode)
- *dataTextField* = de naam van de property die zal worden weergegeven in de ddl (de beschrijvende tekst, hier de gemeentenaam)
- Eventueel kan je als derde parameter de geselecteerde waarde opgeven (nodig bij Edit)

```
ViewBag.Gemeenten =
```

```
    new SelectList(gemeenteRepository.FindAll().OrderBy(g => g.Naam), "Postcode", "Naam");
```

7F. Create – GET : View

► Genereer de View (Scaffolding)

The image shows two windows side-by-side. On the left is the 'Add View' dialog box. It has fields for 'View name:' (set to 'Create'), 'View engine:' (set to 'Razor (CSHTML)'), and 'Model class:' (set to 'Brouwer (BierHalle.Models.Domain)'). The 'Create a strongly-typed view' checkbox is checked. On the right is the generated 'Create' view page. It has input fields for 'Naam' and 'Straat', a dropdown for 'Omzet', and a 'Create' button. Below the form is a link 'Back to List'. At the bottom of the page is a note: '• Gemeente ontbreekt'.

HoGent

Pag. 103

7F. Create – GET : View

► Html helpers

- Creatie van input velden
- Creatie van labels
- Creatie van validatie fouten
- Creatie van links
- Creatie van forms

- In Browser > View Source toont wat helpers gegenereerd hebben

HoGent

Pag. 104

7F. Create – GET : View

▶ Html helper Html.BeginForm()

- voor het renderen van een form tag (= container html input elements, die verstuurd worden naar server)

```
@using (Html.BeginForm("MyAction", "MyController")  
{  
}
```

- Wordt

```
<form action="/MyController/MyAction" method="post"></form>
```

- Html.BeginForm() : post naar de huidige request URL
- Html.BeginForm("MyAction", "MyController", new {par="1"})
post naar /MyController/MyAction?par=1
- Html.BeginForm("MyAction", "MyController", new {id="1"})
post naar /MyController/MyAction/1
- Je kan meerdere forms op 1 view plaatsen

7F. Create – GET : View

▶ Html helper Html.BeginForm()

- Form tag heeft 2 belangrijke attributen :
 - action : methode die request moet afhandelen,
 - method=Get of Post

```
@using (Html.BeginForm("Search", "Home", FormMethod.Get))  
{  
    <input type="text" name="q" />  
    <input type="submit" value="Search" />  
}
```



- Get : plaatst de input waarden in een querystring
 - Voordeel : bookmarking mogelijk(want alle params in url), ook te gebruiken als hyperlink
 - GET wordt gebruikt voor read-only operaties, die de state op server NIET veranderen, repeating requests leveren hetzelfde resultaat
- POST : past informatie op server aan, repeating requests geven vaak een ongewenst resultaat (bvb key bestaat reeds)

7F. Create – GET : View

▶ Html helper Html.BeginForm()

- Bij de Html.BeginForm kan je ook htmlAttributes meegeven

```
@using (Html.BeginForm())  
{
```

```
    @using (Html.BeginForm("Search", "Home", FormMethod.Get, new { target = "_blank" }))
```

7F. Create – GET : View

▶ HTML Helpers voor input tags

- Ipv `<input type="text" name="Naam" id="Naam" value="@Model.Naam />`

- Kan dit verkort door gebruik te maken van HtmlHelpers

```
@Html.TextBoxFor(model => model.Naam)
```

```
@Html.EditorFor(model => model.Naam)
```

→ Gebruikt de template horend bij property

public static [System.Web.Mvc.MvcHtmlString TextBoxFor<TModel, TProperty>\(this \[System.Web.Mvc.HtmlHelper<TModel> htmlHelper, \\[System.Linq.Expressions.Expression<Func<TModel, TProperty>> expression\\\)\\]\\(#\\)\]\(#\)](#)

Member of [System.Web.Mvc.HtmlInputExtensions](#)

Summary:

Returns a text input element for each property in the object that is represented by the specified expression.

Type Parameters:

TModel: The type of the model.

TProperty: The type of the value.

Parameters:

htmlHelper: The HTML helper instance that this method extends.

expression: An expression that identifies the object that contains the properties to render.

Returns:

An HTML input element whose type attribute is set to "text" for each property in the object that is represented by the expression.

Exceptions:

[System.ArgumentException](#): The expression parameter is null or empty.

Extension methode

Delegate (werken met lambda's) : input is een Model, output is een waarde van een property

7F. Create – GET : View

- Html helpers : **string based helpers.** Gebruiken voor **ViewBag**
 - System.Web.Mvc.HtmlHelper
 - 2^{de} par is tekst.

Description	Example	
Check box	Html.CheckBox("myCheckbox", false) Output: <input id="myCheckbox" name="myCheckbox" type="checkbox" value="true" /> <input name="myCheckbox" type="hidden" value="false" />	Opm : 2 input controls => wanneer checkbox niet is aangevinkt, submitten browsers hiervoor geen waarde
Hidden field	Html.Hidden("myHidden", "val") Output: <input id="myHidden" name="myHidden" type="hidden" value="val" />	
Radio button	Html.RadioButton("myRadiobutton", "val", true) Output: <input checked="checked" id="myRadiobutton" name="myRadiobutton" type="radio" value="val" />	
Password	Html.Password("myPassword", "val") Output: <input id="myPassword" name="myPassword" type="password" value="val" />	
Text area	Html.TextArea("myTextarea", "val", 5, 20, null) Output: <textarea cols="20" id="myTextarea" name="myTextarea" rows="5"> val </textarea>	
Text box	Html.TextBox("myTextbox", "val") Output: <input id="myTextbox" name="myTextbox" type="text" value="val" />	Pag. 109

7F. Create – GET : View

- Html helpers : **strongly typed.** Gebruiken voor **Model**
 - System.Web.Mvc.HtmlHelper<T>
 - maakt gebruik van lambda's => Intellisense

Label	Html.LabelFor(x=>x.IsApproved) <label for="IsApproved">IsApproved</label>	
Check box	Html.CheckBoxFor(x => x.IsApproved) Output: <input id="IsApproved" name="IsApproved" type="checkbox" value="true" /> <input name="IsApproved" type="hidden" value="false" />	Opm : name en id attribuut van tag is naam property
Hidden field	Html.HiddenFor(x => x.SomeProperty) Output: <input id="SomeProperty" name="SomeProperty" type="hidden" value="value" />	
Radio button	Html.RadioButtonFor(x => x.IsApproved, "val") Output: <input id="IsApproved" name="IsApproved" type="radio" value="val" />	
Password	Html.PasswordFor(x => x.Password) Output: <input id="Password" name="Password" type="password" />	
Text area	Html.TextAreaFor(x => x.Bio, 5, 20, new{}) Output: <textarea cols="20" id="Bio" name="Bio" rows="5"> Bio value </textarea>	
Text box	Html.TextBoxFor(x => x.Name) Output: <input id="Name" name="Name" type="text" value="Name value" />	Pag. 110

7F. Create – GET : View

◦ Html Helpers voor dropdown en multiselect

Description	Example
Drop-down list	Html.DropDownList("myList", new SelectList(new [] {"A", "B"}), "choose") Output: <select id="myList" name="myList"> <option value="">Choose</option> <option>A</option> <option>B</option> </select>
Drop-down list	Html.DropDownListFor(x => x.Gender, new SelectList(new [] {"M", "F"})) Output: <select id="Gender" name="Gender"> <option>M</option> <option>F</option> </select>
Multiselect list	Html.ListBox("myList", new MultiSelectList(new [] {"A", "B"})) Output: <select id="myList" multiple="multiple" name="myList"> <option>A</option> <option>B</option> </select>
Multiselect list	Html.ListBoxFor(x => x.Vals, new MultiSelectList(new [] {"A", "B"})) Output: <select id="Vals" multiple="multiple" name="Vals"> <option>A</option> <option>B</option> </select>

HoG

Pag. 111

7F. Create – GET : View

◦ Html helpers voor input tags

- Html helpers kunnen ook een 2^{de} par bevatten
 - htmlAttributes van het type object

```
@Html.TextAreaFor(m=>m.Naam, new { @class="myTextBox", maxLength=10})
```

- = anonymous object.
- Het MVC framework beschouwt dit als naam, waarde paren en zal via Reflection de properties opzoeken en de waarde
- Opgelet de property class begint met een @, want class is een reserved woord in C#. "class" geeft een compilatie fout :

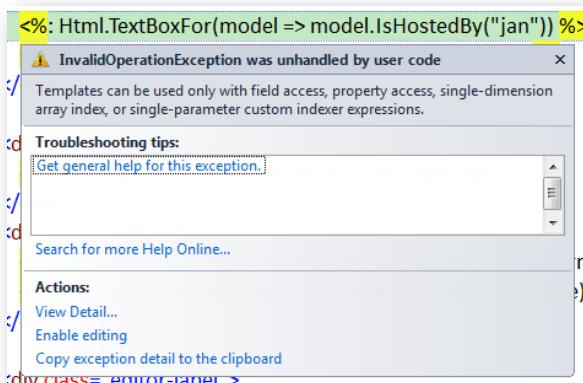
```
new {class= "MyTextBox", MaxLength=10})  
} expected
```

HoGent

Pag. 112

7F. Create – GET : View

- Merk op : In de HtmlHelpers kan je geen methodes van het model aanroepen. Levert een **InvalidOperationException** bij het runnen!!



HoGent

Pag. 113

7F. Create – GET : View

- Opm:
 - In Controller

```
public ActionResult Index()
{
    ViewBag.Brouwer = new Brouwer() { Naam = "Bavik" };
    return View();
}
```

- In View

```
<div>
    @Html.TextBox("Brouwer.Naam")
</div>
```

- De gegenereerde html pagina

```
<input id="Brouwer_Naam" name="Brouwer.Naam" type="text" value="Bavik" />
```

Merk op : Het id attribuut gebruikt _ ipv de dot (name daarentegen gebruikt wel de dot. Reden dots worden niet toegestaan in het id attribuut, dus de runtime vervangt dot door _ (d.i. de waarde van de static HtmlHelper.IdAttributeDotReplacement) => naamgeving in id is later belangrijk voor jQuery)

HoGent

Pag. 114

7F. Create – GET : View

► Templatized helpers

- Genereren html gebruik makend van **metadata** en een template. De metadata bevat info over model waarde (naam en **type**) en annotaties (zie later).
- `Html.Display`
- `Html.Editor`
- `Html.DisplayFor`
- `Html.EditorFor`
- Voorbeeld : `EditorFor` genereert
 - Genereert een `TextBox`
`public string Beschrijving { get; set; }`
 - Genereert een `TextArea`
`[DataType(DataType.MultilineText)]`
`public string Beschrijving { get; set; }`

HoGent

Pag. 115

7F. Create – GET : View

► Templatized helpers : `EditorFor` <-> `TextBoxFor`

- `TextBoxFor` `@Html.TextBoxFor(model => model.Omzet)`
 - Genereert een `TextBox` : `<input type="text" ...>`

- `EditorFor` `@Html.EditorFor(model => model.Omzet)`
 - Genereert eveneens een textbox maar nu `<input type="number" ...>` => wordt in de moderne browsers, op mobiele toestellen anders weergegeven. Je krijgt ook aangepast software keyboard. Bevat bovendien validatie.

- Een voorbeeld van de verschillende input types
 - <http://robertnyman.com/html5/forms/input-types.html>
 - Kan je eens uitproberen in verschillende browsers. Niet alle browsers implementeren alles.

HoGent

Pag. 116

7F. Create – GET : View

- ▶ Html Helpers voor renderen van links en urls

Description	Example
App-relative URL	Url.Content("~/my/content.pdf") Output: /my/content.pdf
Link to named action/controller	Html.ActionLink("Hi", "About", "Home") Output: Hi
Link to absolute URL	Html.ActionLink("Hi", "About", "Home", "https", "www.example.com", "anchor", new{}, null) Output: Hi
Raw URL for action	Url.Action("About", "Home") Output: /Home/About
Raw URL for route data	Url.RouteUrl(new { controller = "c", action = "a" }) Output: /c/a
Link to arbitrary route data	Html.RouteLink("Hi", new { controller = "c", action = "a" }, null) Output: Hi
Link to named route	Html.RouteLink("Hi", "myNamedRoute", new {}) Output: Hi

HoGent

Pag. 117

7F. Create – GET : View

- ▶ View verder aanpassen
 - Gemeente dropdown lijst
 - Nu is het niet mogelijk om een gemeente te selecteren
 - Het beste kan dit met een dropdownlijst gebeuren
 - Gebruik Html helper Html.DropDownList/DropDownListFor
 - Verwacht een parameter van type SelectList met per SelectListItem een naam en een waarde
 - We kunnen dit via ViewBag doorgeven. (Beter alternatief is ViewModel, zie later)

```
public ActionResult Create()
{
    ViewBag.Postcode =
        new SelectList(gemeenteRepository.FindAll().OrderBy(g => g.Naam), "Postcode", "Naam");
    return View(new Brouwer());
}
```

Kies willekeurige naam, maar geen naam van een property uit de klasse brouwer. (zie later Model binding)

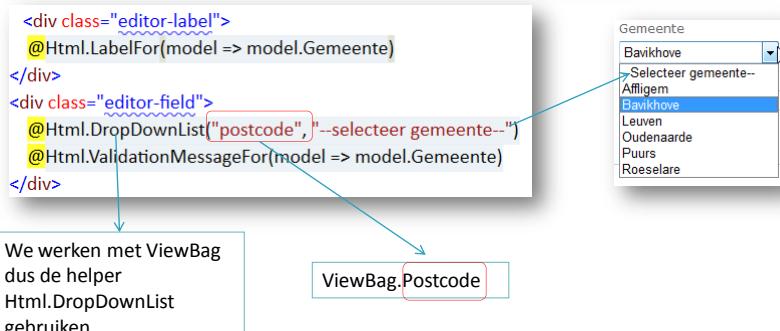
HoGent

Pag. 118

7F. Create – GET : View

► Dropdownlist in de View

```
Drop-down list   Html.DropDownList("myList", new SelectList(new [] {"A", "B"}), "Choose")  
Output:  
<select id="myList" name="myList">  
    <option value="">Choose</option>  
    <option>A</option>  
    <option>B</option>  
</select>
```



Pag. 119

7G. Create – POST : TDD van Controller

► Toevoegen brouwer

- Als de gebruiker de gegevens heeft ingevoerd en op submit klikt worden de formulier gegevens met de **HTTP Post** meegestuurd
 - Het brouwer object moet worden aangemaakt en opgeslaan in de repository (validatie zie later)
 - De gebruiker wordt omgeleid naar de overzichtspagina
- Hiervoor nieuwe action method nodig om de HTTP POST te verwerken
- MVC beschikt over "**magic**" **model binding features**.

Dit betekent dat het MVC framework de form gegevens (is een dictionair van naam – waarde paren) zal omvormen naar een brouwer object. MVC probeert de namen van form gegevenen te mappen met de properties van het object.

```
[HttpPost]  
public ActionResult Create(Brouwer brouwer, string postcode)  
{  
    throw new NotImplementedException();  
}
```

HoGent

Pag. 120

7G. Create – POST : TDD van Controller

► De routing

- Create-GET url : /Create : invulformulier
- Create-Post url : /Create : creëren van brouwer op server
- Waarom dezelfde Url (en dus dezelfde actienaam)?
 - Als de gebruiker niet alle gegevens correct heeft ingevuld, moet de Create View opnieuw getoond worden met de gegevens die de gebruiker reeds heeft ingevuld.
 - Bookmarking (een brouwer mag hierdoor NIET gecreëerd worden)
- Hoe weet MVC welk actie moet worden uitgevoerd :
 - Het MVC framework gaat zelf de juiste action method zoeken a.d.h.v. de naam, de parameters en attributen zoals [HttpPost].

HoGent

Pag. 121

7G. Create – POST : TDD van Controller

- Wat heeft de server gegenereerd => de html pagina?

Create

Fields

Naam	Jan
Straat	Teststraat 10
Gemeente	Bavikhove
Omzet	2000

Create

[Back to List](#)

Create

```
<form action="/Brouwer/Create" method="post">
```

Brouwer

Naam

Straat

Gemeente
--Selecteer gemeente--

Omzet

Create

Google Chrome : met de Web developer addon :
Display form details

HoGent

Pag. 122

7G. Create – POST : TDD van Controller

- Wat stuurt browser naar server na Submit van de form ?
 - Firebug in FireFox (of via Chrome > Developers tools > Network)

The screenshot shows the Firebug Network tab with a 302 Found status for the 'POST Create' request. The 'Post' tab is active, showing the following parameters:

- Naam: Jan
- Omzet: 2000
- Straat: Teststraat 10
- postcode: 8531

Below the parameters, the raw HTML of the form is displayed, with the 'Naam' and 'Straat' input fields highlighted with red boxes.

Create – POST : TDD van de Controller

- MVC zonder model binding
 - Request.Form biedt toegang tot alle parameters meegestuurd met http post

```
[HttpPost, ActionName("Create")]
public ActionResult CreatePost()
{
    Brouwer brouwer = new Brouwer();
    brouwer.Naam = Request.Form["naam"];
    brouwer.Straat = Request.Form["straat"];
    brouwer.Omzet = int.Parse(Request.Form["omzet"]);
    string postcode = Request.Form["postcode"];
    brouwer.Gemeente = (String.IsNullOrEmpty(postcode) ? null : gemeenteRepository.FindBy(postcode));
    brouwerRepository.Add(brouwer);
    brouwerRepository.SaveChanges();
    return RedirectToAction("Index");
}
```

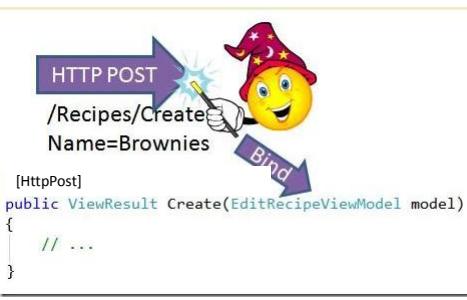
The screenshot shows the Firebug Network tab with a 302 F status for the 'POST Create' request. The 'Post' tab is active, showing the following parameters:

- Naam: Jan
- Omzet: 2000
- Straat: Teststraat 10
- postcode: 8531

A red box highlights the 'Naam' parameter.

Create – POST : TDD van de Controller

- ▶ MVC en model binding
 - Model binding zorgt voor mapping request data naar parameters in een methode.
 - de **model binding features** van het MVC framework



A **model binder** in MVC provides a *simple way to map posted form values to a .NET Framework type and pass the type to an action method as a parameter*. Model binders are like type converters, because they convert HTTP requests into objects that are passed to an action method. As you'd expect from ASP.NET MVC, it defines certain *naming conventions* to let you quickly map complex data structures without having to specify all the mapping rules manually.

Create – POST : TDD van de Controller

- ▶ MVC en Model binding
 - Stel je hebt een actie `public ActionResult RegisterMember(string email)`
 - De Action Invoker, de component die een actie methode aanroeft, is verantwoordelijk voor het toekennen van waarden aan de parameters van de methode alvorens het de methode aanroeft. Hier voor zoekt hij
 1. Request.Form["email"] als bestaat
 - => input velden in formulier ige POST, zoekt naar key "email"
 2. Anders RouteData.Values["email"], als bestaat
 - => .../RegisterMember/jan@hogent.be (Maar hier voor dient de default routing te worden aangepast, want parameter moet naam id hebben)
 3. Anders Request.QueryString["email"], als bestaat
 - => .../RegisterMember?email=jan@hogent.be&x=2
 4. Anders Request.Files["email"], als bestaat
 - Kan enkel indien type HttpPostFileBase is ipv string (= uploaded file)
 5. Anders null, of als het DataType (bvb Date, int,...) geen null kan bevatten dan wordt een InvalidOperationException gethrowdHet zoeken van waarde stopt van zodra een match gevonden

Create – POST : TDD van de Controller

- ▶ Mapping request data naar parameters in method.
 - De Action Invoker maakt gebruik van value providers :

Value Provider	Retrieves Data From	How It Interprets String Values
FormValueProvider	Request.Form (i.e., POST parameters)	Culture sensitive (CultureInfo.CurrentCulture)
RouteDataValueProvider	RouteData.Values (i.e., curly brace routing parameters plus defaults)	Culture insensitive (CultureInfo.InvariantCulture)
QueryStringValueProvider	Request.QueryString (i.e., query string parameters)	Culture insensitive (CultureInfo.InvariantCulture)
HttpFileCollectionValueProvider	Request.Files (i.e., uploaded files)	n/a

Create – POST : TDD van de Controller

- ▶ Mapping request data naar parameters in method:
 - Voor elke input tag => een parameter in de action methode.
Naam parameter moet overeenkomen met id van input tag

```
[HttpPost]
public ActionResult Create(string naam, string straat, decimal omzet , string postcode)
{
    Brouwer brouwer = new Brouwer{naam}{Straat = straat, Omzet = omzet};
    brouwer.Gemeente = (String.IsNullOrEmpty(postcode) ? null : gemeenteRepository.FindBy(postcode));
    brouwerRepository.Add(brouwer);
    brouwerRepository.SaveChanges();
    return RedirectToAction("Index");
}
```

- De model binder zal dan pogingen te maken om de parameternamen te mappen met keynamen gevonden in Request.Form, Request.QueryString,...
- Run

Create – POST : TDD van de Controller

- ▶ Mapping request data naar parameters in method.
 - De gegenereerde html pagina -> mappen naar methode pars

```

<form action="/Brouwer/Create" method="post"> <fieldset>
<div class="editor-label">Naam</div>
<div class="editor-field">
<input id="Naam" name="Naam" type="text" value="" />
<span class="field-validation-valid" data-valmsg-for="Naam" data-valmsg-replace="true"></span>
</div>

<div class="editor-label">Straat</div>
<div class="editor-field">
<input class="text-box single-line" id="Straat" name="Straat" type="text" value="" />
<span class="field-validation-valid" data-valmsg-for="Straat" data-valmsg-replace="true"></span>
</div>

<div class="editor-label">Gemeente</div>
<div class="editor-field">
<select class="editor-select" data-val="true" data-val-number="The field Gemeente must be a number." data-valmsg-for="Gemeente" data-valmsg-replace="true">
<option value="1790">&gt;Aalst</option>
<option value="8531">&gt;Bavikhove</option>
<option value="3000">&gt;Leuven</option>
<option value="9700">&gt;Oudenaarde</option>
<option value="2870">&gt;Puurs</option>
<option value="8800">&gt;Roeselare</option>
</select>
<span class="field-validation-valid" data-valmsg-for="Gemeente" data-valmsg-replace="true"></span>
</div>

<div class="editor-label">Omzet</div>
<div class="editor-field">
<input class="text-box single-line" data-val="true" data-val-number="The field Omzet must be a number." id="Omzet" name="Omzet" type="text" value="" />
<span class="field-validation-valid" data-valmsg-for="Omzet" data-valmsg-replace="true"></span>
</div>

```

POST Create

Headers	Post	Antwoord
Parameters	application/x-www-form-urlencoded	
Naam	Jan	
Omzet	2000	
Straat	Teststraat 10	
postcode	8531	

Pag. 129

Create – POST : TDD van de Controller

- ▶ Mapping request met parameters :
 - Value providers kunnen ook mappen naar complexe structuren

```

[HttpPost]
public ActionResult Create(Brouwer brouwer, string postcode)

```

- De model binder zal een nieuw Brouwer object aanmaken. Via Reflection haalt hij de public properties van de klasse op en zoekt dan voor elke property de waarde op in Request.Form, Request.QueryString,... o.b.v. de *naam van de property*. Als de property ook een object is, bvb Gemeente, zoekt de binder Gemeente.Postcode en Gemeente.naam op.
- Als de setters bij het mappen fouten throwen, wordt de ModelState op false geplaatst. (zie later validatie)

HoGent

Pag. 130

7G. Create – POST : TDD van Controller

- ▶ Aanmaken van de testen
 - Moet brouwer toevoegen aan repository
 - Moet redirecten naar index
 - Wat bij foutieve input ? zie hoofdstuk 8 validatie.
 - Foutmelding geven
 - View opnieuw tonen

HoGent

Pag. 131

7G. Create – POST : TDD van Controller

- ▶ Aanmaken van de testen

```
[TestMethod]
public void CreatePostMustReturnToIndexWhenUpdateSuccessfull()
{
    RedirectToRouteResult result = controller.Create(new Brouwer("Jan"), "3000") as RedirectToRouteResult;
    Assert.AreEqual("Index", result.RouteValues["action"]);
}

[TestMethod]
public void CreatePostMustAddBrouwer()
{
    Brouwer newBrouwer = new Brouwer("Jan");
    mockBrouwerRepository.Setup(m => m.Add(newBrouwer));
    controller.Create(newBrouwer, "3000");
    mockBrouwerRepository.Verify(m => m.Add(newBrouwer), Times.Once());
    mockBrouwerRepository.Verify(m => m.SaveChanges(), Times.Once());
}
```

HoGent

Pag. 132

7G. Create – POST : TDD van Controller

▶ Coderen Create – Post

```
[HttpPost]
public ActionResult Create(Brouwer brouwer, string postcode)
{
    brouwerRepository.Add(brouwer);
    brouwer.Gemeente = (String.IsNullOrEmpty(postcode) ? null : gemeenteRepository.FindBy(postcode));
    brouwerRepository.SaveChanges();
    return RedirectToAction("Index");
}
```

- View : moeten we niet aanmaken want er wordt geredirect naar de Index pagina
- Run de applicatie en voeg brouwer toe
- Opmerking : voeg brouwer instantie zo snel mogelijk toe aan Repository. Is belangrijk als ORM tool gebruikt wordt, want dan pas is het gebonden aan de context en werkt lazy loading,...!! (zie later)

7G. Create – POST : TDD van Controller

▶ Wat doet model binder?

- We kijken via Firebug in FireFox wat er verstuurd wordt

The screenshot shows the Firebug Network tab with a POST request to 'Create'. The 'Parameters' section lists four fields: 'Naam' (Janssens), 'Omzet' (200), 'Straat' (Teststraat 10), and 'postcode' (8531). A red arrow points from the 'Naam' field to a callout box titled 'Wat Model binder ervan maakt'. This box contains a 'QuickWatch' window with an expression 'brouwer'. The 'Value' table shows the deserialized object:

Name	Value	Type
brouwer	(BierHalle.Models.Brouwer)	BierHalle.Models.Brouwer
AantalBieren	0	int
bieren	Count = 0	System.Collections.Generic.List<BierHalle.Models.Bieren>
Bieren	Count = 0	System.Collections.Generic.IEnumerable<BierHalle.Models.Bieren>
brouwerveld	0	int
Gemeente	null	BierHalle.Models.Gemeente
Naam	"Jan"	string
naam	"Jan"	string
Omzet	2000	decimal?
omzet	2000	decimal?
Straat	"Teststraat 10"	string

7H. Edit – GET : TDD van Controller

► Edit action : GET

- Toon de details van een brouwer gegeven de id
- De id-parameter is een onderdeel van de Url
 - /Home/Edit/[id]

```
@Html.ActionLink("Edit", "Edit", new { id = item.BrouwerId }) |
```

```
<a href="/Brouwer/Edit/1">Edit</a>
```

- [id] wordt een parameter van de Edit methode
 - De routing bepaalt dat de 3^{de} waarde in de url gemapt wordt naar een parameter id van de action methode

```
public ActionResult Edit(int id)
{
    throw new NotImplementedException();
}
```

7H. Edit – GET : TDD van Controller

► Edit action : GET

- We schrijven eerst de testen
 - De methode moet de default View retourneren => Maar eigenlijk verschilt die niet van de Create. Dus we gebruiken de Create View
 - Het brouwer object doorgeven als Model
 - De lijst van gemeenten moet worden doorgegeven, evenals de geselecteerde gemeente

7H. Edit

► Edit action

- Testen :

```
#region Edit
[TestMethod]
public void EditMustUseCreateView()
{
    ViewResult result = controller.Edit(1) as ViewResult;
    Assert.IsNotNull(result);
    Assert.AreEqual("Create", result.ViewName);
}

[TestMethod]
public void EditMustReturnBrouwer()
{
    ViewResult result = controller.Edit(1) as ViewResult;
    Assert.IsInstanceOfType(result.Model, typeof(Brouwer));
    Assert.AreEqual(brouwer1, (result.Model as Brouwer));
    mockBrouwerRepository.Verify(m => m.FindBy(1), Times.Once());
}

[TestMethod]
public void EditMustReturnSelectListOfGemeentenAndSelectedValue()
{
    ViewResult result = controller.Edit(1) as ViewResult;
    Assert.IsInstanceOfType(result.ViewBag.Postcode, typeof(SelectList));
    Assert.AreEqual("8531", (result.ViewBag.Postcode as SelectList).SelectedValue);
    mockBrouwerRepository.Verify(m => m.FindBy(1), Times.Once());
    mockGemeenteRepository.Verify(m => m.FindAll(), Times.Once());
}
```

HoGent

7H. Edit – GET : TDD van Controller

► Edit action : GET

- Implementeer Edit methode

```
public ActionResult Edit(int id)
{
    Brouwer brouwer = brouwerRepository.FindBy(id);
    ViewBag.Postcode =
        new SelectList(gemeenteRepository.FindAll().OrderBy(g => g.Naam), "Postcode", "Naam",
            brouwer.Gemeente == null ? "" : brouwer.Gemeente.Postcode);
    return View("Create", brouwer);
}
```

- Testen slagen
- Refactor

Maakt gebruik van de View Create.

Geselecteerde waarde in de dropdownlijst

Daar de Gemeente null mag zijn dan moet er staan :

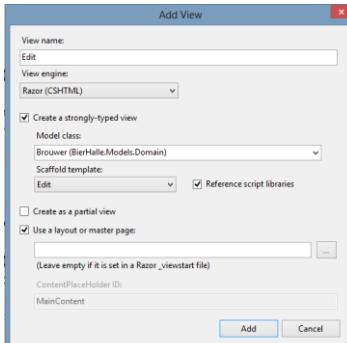
(brouwer.Gemeente==null) ? "" : brouwer.Gemeente.Postcode)

HoGent

7I. Edit – GET : View

► Genereer de View.

- Indien gewenst kan een nieuwe View aanmaken. Hier herbruiken we de Create view. In de View de tekst Create in hoofding vervangen door Brouwer. De submit knop door Save.
 - I.g.v. een Edit View



HoGent

Pag. 139

7J. Edit – POST : TDD van Controller

► Aangepaste brouwer aanpassen

- HTTP POST verwerken (HomeController.cs)

```
[HttpPost]
public ActionResult Edit(int id, FormCollection formValues)
{
    throw new NotImplementedException();
}
```

- FormCollection : NameValueCollection (key (string, is de naam van de property van een input tag) – waarde (string))
 - [Meer op : http://msdn.microsoft.com/en-us/library/system.web.mvc.formcollection.aspx](http://msdn.microsoft.com/en-us/library/system.web.mvc.formcollection.aspx)
- Let op, beter niet binden met een model object
 - Eigenschappen die niet op het formulier getoond worden zouden verloren gaan. Als je dit wel doet, moet je Attach methode voorzien in Repository :
 - context.Attach(brouwer)
 - context.Entry(brouwer).State = EntityState.Modified;

HoGent

Pag. 140

7J. Edit – POST : TDD van Controller

► HTTP-Post Edit actie

- We schrijven eerst de testen
 - De aangepaste brouwer is opgeslaan na succesvolle update
 - Redirect naar index pagina na succesvolle update
 - Igv fouten : zie later
- In de testen moeten we de FormCollection aanmaken

```
FormCollection formValues = new FormCollection()  
{  
    {"BrouwerId", "1"},  
    {"Naam", "Bavikske"},  
    {"postcode", "3000"},  
    {"Straat", "test"}  
};  
controller.ControllerContext = new ControllerContext();
```

7J. Edit – POST : TDD van Controller

► HTTP-Post Edit actie

- De testen

```
[TestMethod]  
public void EditMustReturnToIndexWhenUpdateSuccessfull()  
{  
    FormCollection formValues = new FormCollection()  
    {  
        {"BrouwerId", "1"},  
        {"Naam", "Bavikske"},  
        {"postcode", "3000"},  
        {"Straat", "test"}  
    };  
    controller.ControllerContext = new ControllerContext();  
    RedirectToRouteResult result = controller.Edit(1, formValues) as RedirectToRouteResult;  
    Assert.AreEqual("Index", result.RouteValues["action"]);  
}
```

7J. Edit – POST : TDD van Controller

► HTTP-Post Edit actie

- De testen

```
[TestMethod]
public void EditMustChangeBrouwer()
{
    FormCollection formValues = new FormCollection()
    {
        {"BrouwerId", "1"},
        {"Naam", "Bavikske"},
        {"postcode", "3000"},
        {"Straat", "test"}
    };
    controller.ControllerContext = new ControllerContext();
    ViewResult result = controller.Edit(1, formValues) as ViewResult;
    Assert.AreEqual("Bavikske", brouwer1.Naam);
    Assert.AreEqual("3000", brouwer1.Gemeente.Postcode);
    Assert.AreEqual("test", brouwer1.Straat);
    mockBrouwerRepository.Verify(m => m.FindBy(1), Times.Once());
    mockGemeenteRepository.Verify(m => m.FindBy("3000"));
    mockBrouwerRepository.Verify(m => m.SaveChanges(), Times.Once());
}
```

HoGent

Pag. 143

7J. Edit – POST : TDD van Controller

► HTTP-Post Edit actie

- Schrijven van de code
 - The hard way :

```
[HttpPost]
public ActionResult Edit(int id, FormCollection formValues)
{
    Brouwer brouwer = brouwerRepository.FindBy(id);
    brouwer.Naam = formValues["naam"];
    brouwer.Straat = formValues["straat"];
    ...
    brouwerRepository.SaveChanges();
    return RedirectToAction("Index");
}
```

HoGent

Pag. 144

7J. Edit – POST : TDD van Controller

► HTTP-Post Edit actie

- De built-in **UpdateModel** helper methode
 - Zal de properties van een object aanpassen, gegeven de FormCollection
 - Zal ook de ModelState aanpassen. Zie verder Validatie

```
[HttpPost]
public ActionResult Edit(int id, FormCollection formValues)
{
    Brouwer brouwer = brouwerRepository.FindBy(id);
    UpdateModel(brouwer, formValues.ToValueProvider());
    string postcode = formValues["postcode"];
    brouwer.Gemeente = (String.IsNullOrEmpty(postcode)?null:gemeenteRepository.FindBy(postcode));
    brouwerRepository.SaveChanges();
    return RedirectToAction("Index");
}
```

Betekent dat enkel de formValues gebruikt worden om de brouwer aan te passen, en niet de querystring,....

HoGent

Pag. 145

7K. Delete – GET : TDD van Controller

► Brouwer verwijderen

- De delete ActionLinks verwijzen naar /Brouwer/Delete/id

```
<a href="/Brouwer/Edit/1">Edit</a> |
<a href="/Brouwer/Delete/1">Delete</a>
```

- Nieuwe action method nodig

```
public ActionResult Delete(int id)
{
    throw new NotImplementedException();
}
```

HoGent

Pag. 146

7K. Delete – GET : TDD van Controller

► Brouwer verwijderen

- Ook een Get en Post versie
- Get : vraagt om bevestiging. Dit gebeurt via een formulier.
- Post : na bevestiging wordt brouwer verwijderd.

- Waarom Get en Post?
 - Beveiligen tegen web crawlers en SEO die een brouwer zouden kunnen verwijderen door een link te volgen.
 - GOLDEN RULE : Aanpassen/verwijderen van gegevens steeds via POST

HoGent

Pag. 147

7K. Delete – GET : TDD van Controller

► Brouwer verwijderen

- De testen
 - Opent default view
 - Geeft de brouwer door

result.Model
Of
result.ViewData.Model

```
#region Delete
[TestMethod]
public void DeleteMustUseConventionToChooseView()
{
    ViewResult result = controller.Delete(1) as ViewResult;
    Assert.IsNotNull(result);
    Assert.IsTrue(String.IsNullOrEmpty(result.ViewName));
}

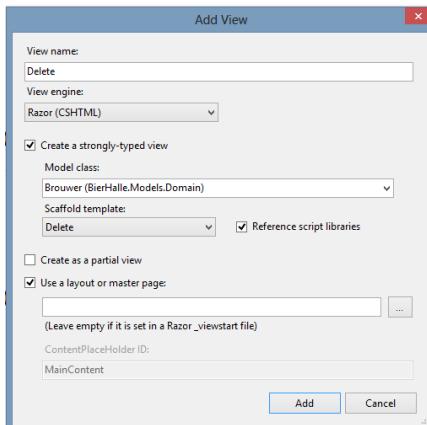
[TestMethod]
public void DeleteMustReturnBrouwer()
{
    ViewResult result = controller.Delete(1) as ViewResult;
    Assert.IsInstanceOfType(result.Model, typeof(Brouwer));
    Assert.AreEqual(brouwer1, (result.Model as Brouwer));
    mockBrouwerRepository.Verify(m => m.FindBy(1), Times.Once());
}
#endregion
```

HoGent

Pag. 148

7L. Delete – GET : View

- ▶ Brouwer verwijderen
 - Nieuwe View aanmaken voor de Delete methode



HoGent

Pag. 149

7L. Delete – GET : View

- ▶ Brouwer verwijderen
 - Template aanpassen (Delete.cshtml)

```
@model BierHalle.Models.Domain.Brouwer

@{
    ViewBag.Title = "Verwijderen brouwer";
}

<h2>@ViewBag.Title</h2>

<h3>Weet je zeker dat je de brouwer @Model.Naam wil verwijderen?</h3>

@using (Html.BeginForm()) {
    <p>
        <input type="submit" value="Delete" /> |
        @Html.ActionLink("Back to List", "Index")
    </p>
}
```

HoGent

Pag. 150

7M. Delete – POST : TDD van Controller

► Brouwer verwijderen

- Afhandelen van de confirmatie POST (BrouwerController.cs)

```
[HttpPost, ActionName("Delete")]
public ActionResult DeleteConfirmed(int id)
{
    throw new NotImplementedException();
}
```

2 action methodes met dezelfde naam is niet mogelijk. Vandaar DeleteConfirmed, maar wordt gemapped naar Action met name Delete

7M. Delete – POST : TDD van Controller

► Brouwer verwijderen

- De testen
 - Brouwer moet verwijderd zijn
 - Redirect naar index

```
#region Delete Post
[TestMethod]
public void DeletePostMustReturnToIndexWhenDeleteSuccessful()
{
    RedirectToRouteResult result = controller.DeleteConfirmed(1) as RedirectToRouteResult;
    Assert.AreEqual("Index", result.RouteValues["action"]);
}

[TestMethod]
public void DeletePostMustRemoveBrouwer()
{
    mockBrouwerRepository.Setup(m => m.Delete(brouwer1));
    controller.DeleteConfirmed(1);
    mockBrouwerRepository.Verify(m => m.Delete(brouwer1), Times.Once());
    mockBrouwerRepository.Verify(m => m.SaveChanges(), Times.Once());
}
```

7M. Delete – POST : TDD van Controller

► Brouwer verwijderen

- De code

```
[HttpPost, ActionName("Delete")]
public ActionResult DeleteConfirmed(int id)
{
    Brouwer brouwer = brouwerRepository.FindBy(id);
    brouwerRepository.Delete(brouwer);
    brouwerRepository.SaveChanges();
    return RedirectToAction("Index");
}
```

HoGent

Pag. 153

7. Controller



► Controller bevat GEEN businesslogica

- Hiervoor deleert de controller naar het domein
- Stel in Controller : methode voor het ophalen van de bieren van een bepaald type van de opgegeven brouwer
 - Slechte oplossing

```
public ActionResult Bieren(int brouwerid, string type) {
    Brouwer b = brouwerRepository.FindBy(brouwerid);
    return View(b.Bieren.Where(b=>b.Type==type).ToList());}
```

- Goeie oplossing

```
public ActionResult Bieren(int brouwerid, string type) {
    Brouwer b = brouwerRepository.FindBy(brouwerid);
    return View(b.GetBieren(type));}
```

- **Pas Law of Demeter toe in MVC**
 - Vermijd . . notatie (is in Linq heel eenvoudig)

- **Ook Views bevatten GEEN businesslogica, presenteren enkel data**
- **Pas design patterns toe in je ontwerp**

HoGent

Pag. 154

8. Layouts

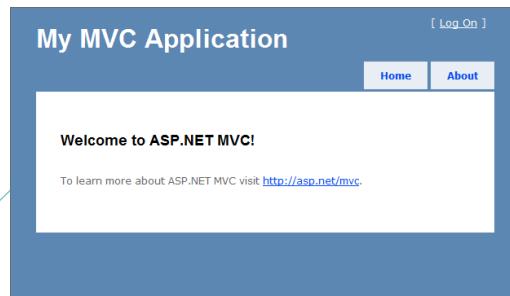
- ▶ Standaardiseren van
 - Web Site Look and Feel
 - CSS
 - Web Site Layout
 - Layout (of master page)

8. Layouts

- ▶ Layouts
 - Bieden sjablonen voor een uniforme indeling van pagina's en het delen van gemeenschappelijke functionaliteit
 - De web pagina's maken gebruik van dit sjabloon en voegen hier hun inhoud aan toe. "Visual inheritance" voor Web pagina's (~abstracte base class voor views)
- ▶ Navigatie
 - Menu

8. Layouts

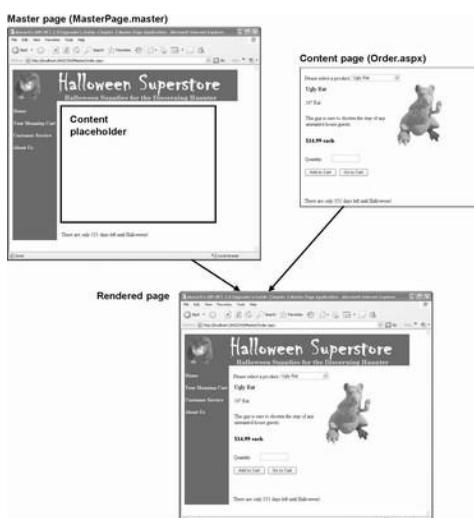
- ▶ Definitie van het paginasjabloon
 - Op elke pagina hebben we eenzelfde layout : header, footer en navigatie
 - Elke pagina heeft een eigen inhoud (content gedeelte genaamd)



HoGent

Pag. 157

8. Layouts



HoGent

- ▶ Het concept Layout kent 2 onderdelen

- Enerzijds zijn er de paginasjablonen (_layout) waarin de layout website bepaald wordt
- In de pagina's gebaseerd op een layout dien je enkel nog de inhoud te plaatsen (= content page).
- => De daadwerkelijke inhoud van een pagina wordt gecombineerd met het geselecteerde sjabloon.
- => Hierdoor bepaal je op 1 plaats hoe de indeling van de website is en alle web pagina's krijgen dezelfde indeling

Pag. 158

8. Layouts

► De Layout page

- In de folder Views/Shared/_layout.cshtml
- Deze pagina bevat
 - Html markup(<html>, <head>, <body>)
 - De gemeenschappelijke layout en inhoud (Header, Footer, Menu)
 - placeholders (@RenderBody) : de plaats waarbinnen de inhoud van de web pagina komt
- Voordelen
 - Consistente, gestandaardiseerd pagina layout
 - Shared UI & Code Elementen
 - Kan programmatorisch en declaratief gedefinieerd worden
 - Op 1 plaats de layout wijzigen, op alle pagina's zichtbaar

8. Layouts

► Creatie van een Layout Page

- Maak je een nieuw MVC4 Project aan
 - Deze bevat reeds een layout page
- OF maak zelf een Layout page aan
- Binnen de Views Folder > Shared > rechtsklik > Add New Item > C# > Web > MVC4 > MVC4 Layout Page (Razor) en geef de page een naam
- Voeg nu HTML, toe

```
<!DOCTYPE html>
]<html>
]<head>
]<title>@ViewBag.Title</title>
]</head>
]<body>
]<div>
]<div>
    @RenderBody()
]</div>
]</body>
]</html>
```

8. Layouts

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>@ViewBag.Title - My ASP.NET MVC Application</title>
    <link href="~/favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <meta name="viewport" content="width=device-width" />
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <header>
        <div class="content-wrapper">
            <div class="float-left">
                <p class="site-title">@Html.ActionLink("Bierhalle", "Index", "Home")</p>
            </div>
            <div class="float-right">
                <section id="login">
                    @Html.Partial("_LoginPartial")
                </section>
                <nav>
                    <ul id="menu">
                        <li>@Html.ActionLink("Home", "Index", "Home")</li>
                        <li>@Html.ActionLink("About", "About", "Home")</li>
                        <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                    </ul>
                </nav>
            </div>
        </div>
    </header>

```

Pagina bevat header, main en footer

ViewBag.Title voor de inhoud van de title tag

Styles.Render laadt de css, zie BundleConfig.cs

Scripts.Render laadt modernizr, zie BundleConfig.cs

p class=site-title: bevat de naam site of logo,...

De link naar login page

Nav : Pas aan waar nodig.

ActionLink :

1^e par : tekst op link

2^e : action naam

3^e : controller naam

HoGent

Pag. 161

8. Layouts

```
<div id="body">
    @RenderSection("featured", required: false)
    <section class="content-wrapper main-content clear-fix">
        @RenderBody()
    </div>
</div>
<footer>
    <div class="content-wrapper">
        <div class="float-left">
            <p>&copy; @DateTime.Now.Year - Hogeschool Gent</p>
        </div>
    </div>
</footer>

@Scripts.Render("~/bundles/jquery")
@RenderSection("scripts", required: false)
</body>
</html>
```

RenderSection : laadt toe om extra inhoud toe te voegen

@RenderBody : voor de inhoud van de pagina

Footer met copyright

Scripts.Render laadt jquery, zie BundleConfig.cs

RenderSection : laadt toe om in View scripts toe te voegen

HoGent

Pag. 162

8. Layouts

► Placeholders

- @RenderBody = placeholder die de plaats markeert waar views die van deze layout gebruik maken hun inhoud plaatsen
- @RenderSection : Een layout kan daarnaast nog andere secties hebben waar views hun inhoud kunnen plaatsen.
- Required : false : betekent dat de view de sectie niet hoeft te bevatten

```
<div id="body">
    @RenderSection("featured", required: false)
    <section class="content-wrapper main-content clear-fix">
        @RenderBody()
    </section>
</div>
```

- In View `@section scripts`
- ```
{
 <script>
 alert('Hello');
 </script>
}
```

HoGent

Pag. 163

## 8. Layouts

### ► Aanmaken View gebaseerd op een Layout page

- Bij aanmaken van een View vink je “Use a Layout or Master Page” checkbox aan. Vervolgens kies je de Layout Page of gebruik je de Layout page gedefinieerd in \_viewStart



#### ◦ De View

```
@{
 ViewBag.Title = "ViewPage2";
 Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>ViewPage2</h2>
```

- @{} wordt eerst uitgevoerd  
View bepaalt zijn layout via  
Layout property

De rest wordt geplaatst op de locatie  
@RenderBody

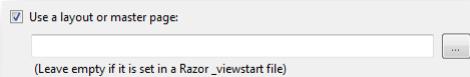
HoGent

Pag. 164

## 8. Layouts

► Aanmaken View gebaseerd op een Layout page

- Werken met \_ViewStart



- Views\Shared\\_Viewstart.cshtml

- Bepaalt de default Layout page voor de views

```
@{
 Layout = "~/Views/Shared/_Layout.cshtml";
}
```

- De View

```
@{
 ViewBag.Title = "ViewPage1";
}

<h2>ViewPage1</h2>
```

## 8. Layouts

► Meerdere sections

- In de layout page

```
<div id="footer">@RenderSection("Footer")</div>
```

- In de view

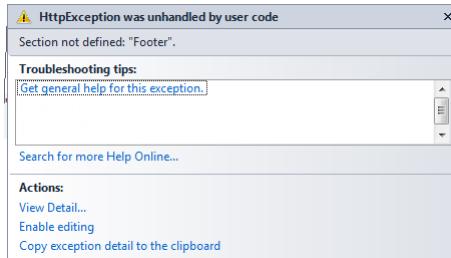
```
@{
 ViewBag.Title = "Home Page";
}

<h2>@ViewBag.Message</h2>
|<p>
 To learn more about ASP.NET MVC visit
 http://asp.net/mvc.
|</p>
@section Footer {
 This is the footer.
}
```

## 8. Layouts

### ► Meerdere sections

- Als je echter een section gedefinieerd in de layout page niet opneemt in de view



## 8. Layouts

### ► Meerdere sections

- Als je echter een section gedefinieerd in de layout page niet opneemt in de view

### ◦ Oplossing in de Layout page

- Als sectie niet verplicht is

```
<div id="footer">@RenderSection("Footer", false)</div>
```

- Je kan ook default content voorzien

```
<div id="footer">
@if (!IsSectionDefined("Footer"))
{
 @RenderSection("Footer")
}
else
{
 This is the default footer.
}
</div>
```

## 8. Layouts

► CSS

- Reeds toegevoegd aan site. Onderdeel van Contents folder
- Bevat oa. de opmaak van de site, en wordt naar verwezen in de master page

## 9. Oefening 1

► Opgave: werk BierHalle verder uit zodat

- Je voor een brouwer de lijst van bieren kunt tonen
- Je een nieuw bier aan een brouwer kunt toevoegen
- Je een bier van een brouwer kunt verwijderen

## 10. Referenties

- ▶ Tutorial : <http://www.asp.net/mvc/tutorials/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application>