

Hoofdstuk 4 : Een eerste MVC applicatie

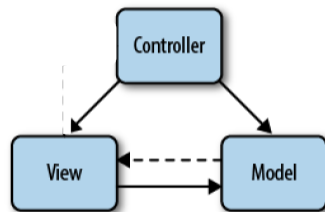
HoGent

Hoofdstuk 4 : Een eerste MVC applicatie

- Inleiding ASP.NET MVC
- Hello MVC
 - ✓ Routing
 - ✓ Controller
 - ✓ View
- SnakeEyes
- MVC Flow
- Bundling en Minification
- Oefening

HoGent

1. Inleiding ASP.NET MVC



Controller

- Intercepts user input
- Coordinates the view and model
- Handles communication between the model and data layer

Model

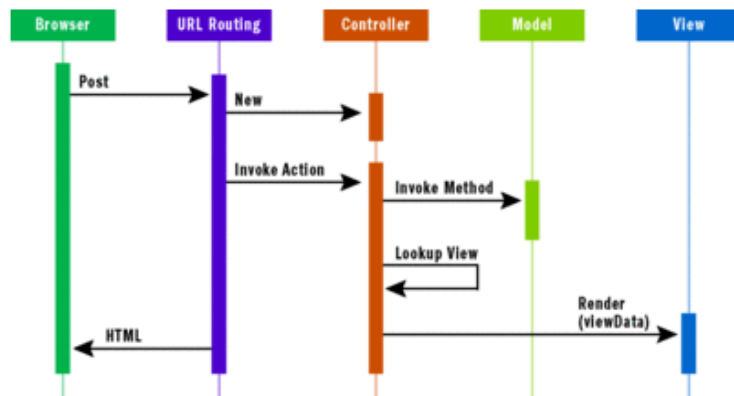
- Data attributes (properties)
- Business logic, behavior, and validation

View

- Renders the UI (HTML, CSS, PDF)
- Binds to the model

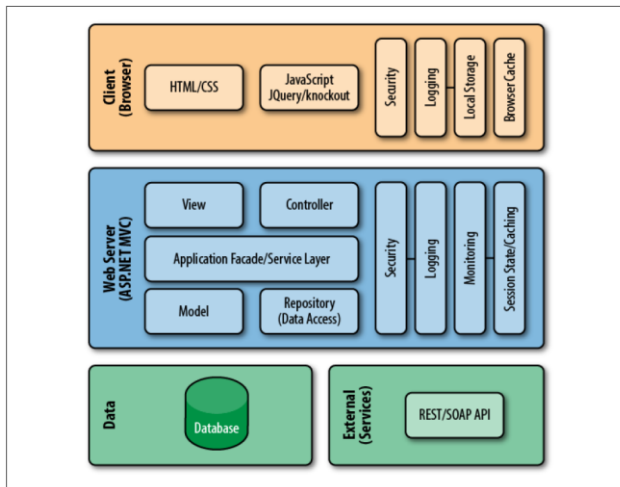
1. Inleiding ASP.NET MVC

- De **web** variant van MVC (Model2)



1. Inleiding ASP.NET MVC

- ▶ De onderdelen van een MVC webapplicatie



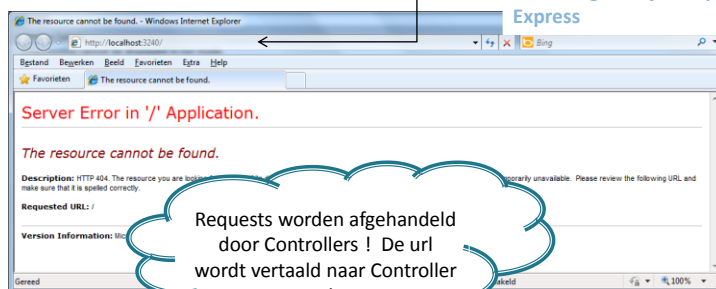
HoGent

Pag. 5

2. Hello MVC

- ▶ Maak een nieuwe MVC4 applicatie – Kies voor een Basic MVC4 project. Naam “HelloMVC”.
- ▶ Run de applicatie. Druk F5.
 - Je krijgt fout.

Dit start de lokale webserver. Het poortnummer is variabel. Binnen de taskmanager zie je het proces IIS Express



HoGent

Pag. 6

2. Hello MVC : Routing

- ▶ We hebben dus een controller nodig.
 - Welke? **Routing** verzorgt de **mapping** van een **URL** naar een **actie** in een **controller**.
 - Global.asax : definieert de routing
 - Roept de methode RegisterRoutes in de klasse RouteConfig in de map App_Start, op die de routing definieert

```
public class MvcApplication : System.Web.HttpApplication
{
    protected void Application_Start()
    {
        AreaRegistration.RegisterAllAreas();

        WebApiConfig.Register(GlobalConfiguration.Configuration);
        FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
        RouteConfig.RegisterRoutes(RouteTable.Routes);
        BundleConfig.RegisterBundles(BundleTable.Bundles);
    }
}
```

HoGent

Pag. 7

2. Hello MVC : Routing

- RegisterRoutes definieert de routing
 - MapRoute definieert URL patroon van een route

```
public class RouteConfig
{
    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 }
        );
    }
}
```

Een naam die je aan een bepaalde route geeft. Willekeurig te kiezen

Het URL patroon

Defaultwaarden voor de onderdelen Van het URL patroon , als die ontbreken

HoGent

Pag. 8

2. Hello MVC : Routing

► REST

◦ Representational State Transfer



Representational state transfer (REST) is a style of [software architecture](#) for [distributed hypermedia](#) systems such as the [World Wide Web](#). The term *representational state transfer* was introduced and defined in 2000 by [Roy Fielding](#) in his doctoral dissertation.^{[1][2]}

REST-style architectures consist of [clients](#) and [servers](#). Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of representations of resources. A [resource](#) can be essentially any coherent and meaningful concept that may be addressed. A [representation](#) of a resource is typically a document that captures the current or intended state of a resource.

2. Hello MVC : Routing

► REST (Representational state transfer)

- REST is echter geen standaard, maar een architectuur die gebruik maakt van de volgende standaarden
 - HTTP
 - URL
 - XML (POX (Plain Old XML) zonder SOAP omslag)
 - JSON
 - GIF/JPEG/etc (Resource Representations)
- REST betekent dat elke unieke URL een representatie is van een object of sommige objecten (**resources** genaamd). Opvragen een object(en) dmv HTTP GET. Creatie, wijzigen, verwijderen van object dmv HTTP POST, PUT, of DELETE. (**actions** genaamd)

2. Hello MVC : Routing

► REST : URI's

- Iedere resource wordt geïdentificeerd door een unieke uri
 - Alle gebruikers (bvb in xml formaat)
 - <http://www.hogent.be/users/>
 - 1 gebruiker
 - <http://www.hogent.be/users/1>
- Meest zichtbare aspect van REST : URI ontwerp
 - REST: <http://www.hogent.be/users/1>
 - i.p.v : <http://www.hogent.be/users?id=1&action=detail>

2. Hello MVC : Routing

► Routing

- MVC gebruikt **resource centric view** van URL. Een URL stelt een resource op het web voor.
- De resource is in ASP.NET MVC een stukje code die de request afhandelt.
- Het patroon definieert het URL formaat voor aanroepen resource.

`"{controller}/{action}/{id}"`

- MVC Routing zorgt er voor dat de een request(URL) gemapt wordt met een actie (methode) binnen een Controller, eventueel met parameters.
 - **{controller}** => de naam van de controller klasse die MVC zal instantiëren. MVC voegt het woord Controller toe aan de naam. Bvb Home. De bijhorende controller klasse HomeController.cs.
 - **{action}** => naam van de *publieke* methode binnen de controller die MVC zal aanroepen
 - **{id}** => een (optionele) parameter met naam id die als argument van de methode kan worden doorgegeven. Is dan parameter van de methode

2. Hello MVC : Routing

► Routing

- De routing definieert ook defaultwaarden, als onderdelen van de URL ontbreken

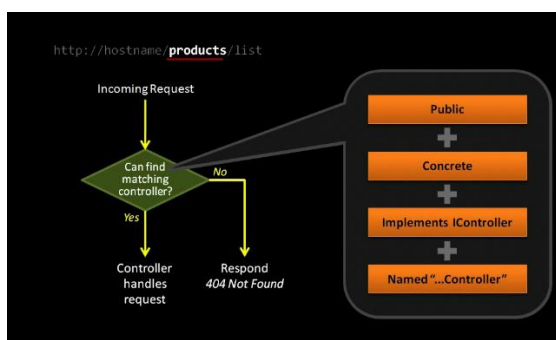
```
new { controller = "Home", action = "Index", id = UrlParameter.Optional }
```

- id is optioneel. Methode hoeft geen parameter id te bevatten
- action : indien url geen action bevat, roep dan methode Index aan
- controller : als url geen controller bevat instantieer dan de Controller met naam Home

URL	CONTROLLER CLASS	ACTION METHOD	PARAMETERS PASSED
/Dinners/Details/2	DinnersController	Details(id)	id=2
/Dinners/Edit/5	DinnersController	Edit(id)	id=5
/Dinners/Create	DinnersController	Create()	N/A
/Dinners	DinnersController	Index()	N/A
/Home	HomeController	Index()	N/A
/	HomeController	Index()	N/A

2. Hello MVC : Routing

► De routing :



"... Controller"
erft van
Controller die
erft van
IController

2. Hello MVC : Controller

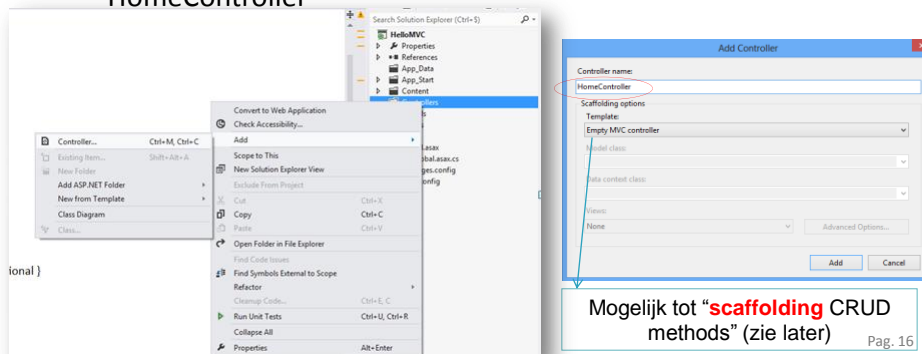
▶ Controller

- Verantwoordelijk voor
 - Verwerken binnenkomende **HTTP requests**
 - Verwerken van de input van de gebruiker, uitvoeren van de applicatielogica. Eventueel ophalen en opslaan van de gegevens.
 - Bepalen van de **HTTP response** (html, downloaden van een bestand, redirect naar een andere url,...).
 - Als de Controller een view result retourneert delegeert het ASP.NET Framework naar de **View Engine**, die de taak heeft de gevraagde view te laden en te renderen. De Controller geeft de nodige data door.
- Is een klasse met de actie methodes
- Meestal 1 controller per use case. Bevat in feite de systeemoperaties uit het SSD vertaald naar de **naming conventions** voor “Action methods” in ASP.NET MVC

2. Hello MVC : Controller

▶ Controller aanmaken.

- Wat? HomeController.cs met methode Index().
 - = controller/actie die gemapt wordt met URL <http://localhost:3240>
- Rechtsklik folder Controllers > Add > Controller. Naam : HomeController



2. Hello MVC : Controller

▶ Controller aanmaken

- Dit genereert :

```
using System.Web.Mvc;

namespace HelloMVC.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

- In folder Controllers : een bestand HomeController.cs met de klasse HomeController
- **Convention over configuration**: de naam van een Controller klasse moet eindigen op Controller.
- Erft van klasse **System.Web.Mvc.Controller**.
- Elke publieke methode = **actie** en kan dus opgeroepen worden adhv url

2. Hello MVC : Controller

▶ Controller aanmaken

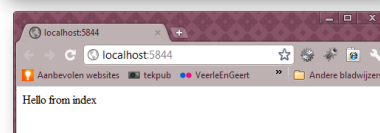
- Routing in action
 - Maak 2 actie methodes aan in HomeController. Beide retourneren tekst :
 - Index
 - About. Heeft 1 parameter. Naam parameter moet id zijn. In global.asax :

"{controller}/{action}/{id}", // URL with parameters

- Run en surf naar (poort varieert, hier 1234)
 - <http://localhost:1234>
 - <http://localhost:1234/Home/Index>
 - <http://localhost:1234/Home/About/1>
 - <http://localhost:1234/Home/About?id=10> -> querystring wordt naar parameters in methode vertaald

```
namespace HelloMVC.Controllers
{
    public class HomeController : Controller
    {
        public string Index()
        {
            return "Hello from index";
        }

        public string About(int id)
        {
            return "Hello from about " + id.ToString();
        }
    }
}
```



2. Hello MVC : Controller

- ▶ Controller aanmaken
 - Welke van onderstaande url's falen? Waarom?
 1. <http://localhost:1234/Home>
 2. <http://localhost:1234/Index>
 3. <http://localhost:1234/Index/1>
 4. <http://localhost:1234/Home/About>
 5. <http://localhost:1234/Home/About/abc>
 6. <http://localhost:1234/Home/About?id=abc>
 7. <http://localhost:1234/Home/Contact>
- ▶ Pas signatuur About aan : `public string About(int id=0)`. Welke van bovenstaande url's werken nu wel?
- ▶ Wat moet er worden aangepast aan de signatuur van About zodat 5 en 6 werken?

2. Hello MVC : Controller

- ▶ Controller
 - Een Controller actie methode handelt een request af en stuurt een resultaat (meestal html) terug naar de browser.
 - Indien we HTML wensen te retourneren naar de browser moet de actie methode een **ViewResult** object retourneren. Dit erft van de basisklasse **ActionResult**.
 - **ActionResult** is de abstracte basisklasse voor alles wat een action methode naar de browser kan retourneren (Er zijn meerdere resulttypes, zie verder). Een ActionResult stelt een **Command** voor, die het resultaat van een action methode bevat en dat het framework zal uitvoeren in naam van de action methode.
 - Voor het aanmaken van een ViewResult roept een actie methode de **View()** methode aan.
 - **View()** maakt een instantie van **ViewResult** (subklasse van ActionResult) aan en voert deze uit. Hierdoor wordt een HTML pagina gegenereerd o.b.v. een View template (.cshtml) en wordt deze naar de browser gestuurd

2. Hello MVC : Controller

► Controller aanmaken

- Pas de controller als volgt aan
 - View() : vraagt aan MVC framework om de default View te renderen. MVC maakt hiervoor gebruik van **naming conventions**
 - Run de applicatie

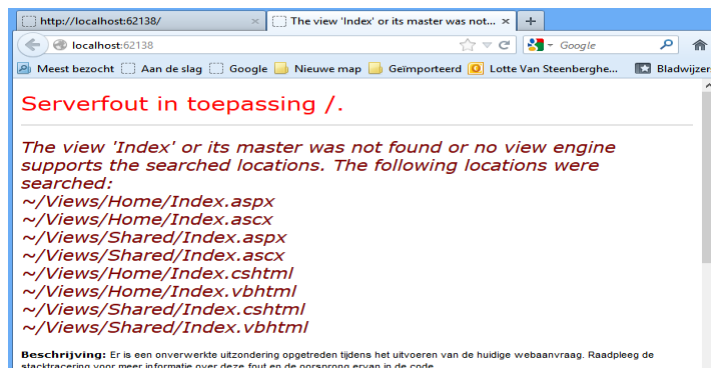
```
using System.Web.Mvc;

namespace HelloMVC.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public string About(int id)
        {
            return "Hello from about " + id.ToString();
        }
    }
}
```

2. Hello MVC : Controller

► Als we nu de applicatie runnen



return View() geeft de instructie aan MVC framework om de **default view** van de action methode te renderen. Views die bij een actie horen worden automatisch gevonden als ze in de directory gezet worden met de naam van de Controller of in de Shared folder geplaatst worden binnen de Views folder. (=Convention over configuration)

2. Hello MVC : View

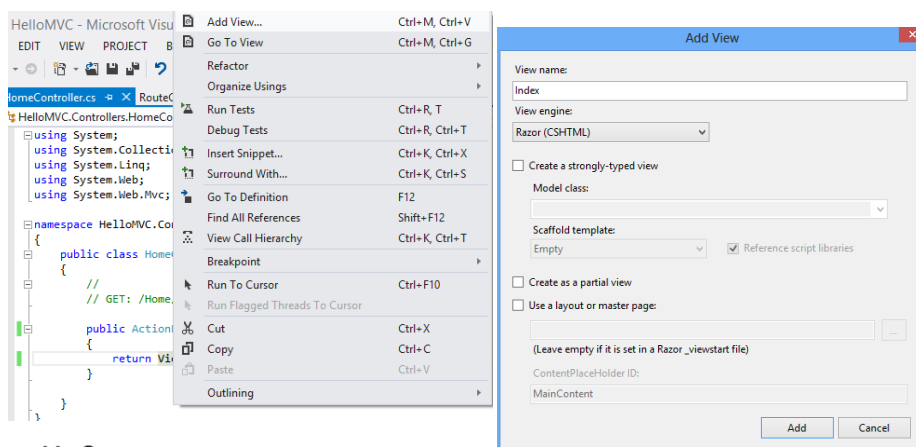
► View

- Bevat **alleen presentatieloga**
- De UI van de applicatie
- Een view maakt een HTML pagina gebruik makend van de data (model) die het van de Controller ontvangen heeft. Een action methode geeft deze data door bij het aanroepen van de View methode
- De View methode creëert ViewResult, die roept de View Engine van het MVC Framework op, die de View zal renderen en het resultaat naar de browser zal sturen.
- De default view engine is the Razor View Engine. Er bestaan nog andere : aspx, Spark, Nhaml.



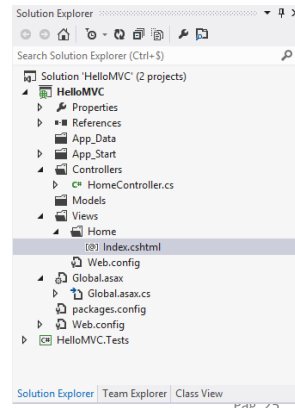
2. Hello MVC : View

- Creatie van de view page door rechtsklik op action methode. Vink use a layout or master page uit.



2. Hello MVC : View

- ▶ Convention over configuration : De View Index.cshtml wordt aangemaakt in
 - folder Views
 - subfolder
 - Home = naam Controller of
 - subfolder Shared : bevat view pages die door meerdere Controllers aangeroepen worden
 - pagina Index.cshtml = naam view is de naam van action methode
- ▶ Dus Views folder bevat folder per Controller, die op zijn beurt een .cshtml bevat per actie.



HoGent

2. Hello MVC : View

- ▶ View = (gegenereerde) HTML pagina

```
@{  
    Layout = null;  
}
```

Codeblok geïnterpreteerd door Razor View Engine.

Betekent dat Razor geen layout of master page moet gebruiken

```
<!DOCTYPE html>
```

De html van de pagina.

Voeg tussen div tags "Hello MVC" toe

```
<html>  
<head>  
    <meta name="viewport" content="width=device-width" />  
    <title>Index</title>  
</head>  
<body>  
    <div>  
  
    </div>  
</body>  
</html>
```

DOCTYPE: HTML5

Zie bovenaan toobar : laat toe om HTML versie te kiezen en WCAG

HoGent

2. Hello MVC : View

▸ Razor View Engine

- Lightweight, simple View Engine om aan HTML C# code toe te voegen
- Rendert html
- Start karakter van C# code : @.
 - Code nugget : @ gevolgd door 1 C# instructie die inline geëvalueerd en gerenderd wordt. Razor herkent zelf einde van instructie
 - Code blok : @{...}. bevat C# code
- Voorbeeldjes
 - <h1>Listing @stuff.Length items</h1>
 - Razor weet hier dat een spatie na de Length expression geen geldige identifier is, dus schakelt hij terug over naar HTML
 - @{Layout=null}

HoGent

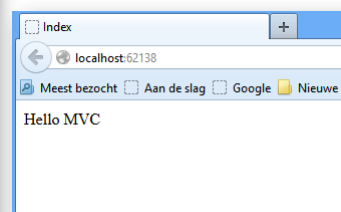
2. Hello MVC : View

▸ View = (gegenereerde) HTML pagina. Run

```
Index.cshtml
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <div>
        Hello MVC
    </div>
</body>
</html>
```



HoGent

2. Hello MVC : View

- ▶ View en dynamische output
 - Taak van controller om de data voor de View te genereren
 - Taak van View om die data in de html pagina weer te geven
 - Controller bevat property ViewBag van type dynamic.
“Dynamic means you can dynamically get/set values and add any number of additional fields, properties, methods without need of strongly-typed classes. Properties on a dynamic object are defined at run-time, so to add a new property you just use it”
 - In de ViewBag kan je informatie plaatsen die zal doorgegeven worden aan de View page bij aanroep van de View() methode.
Maak gewoon een property aan en plaats er de waarde in

2. Hello MVC : View

- ▶ View en dynamische output
 - Controller genereert de data die de View zal moeten renderen

```
public ActionResult Index()
{
    ViewBag.Greeting = DateTime.Now.Hour < 12 ? "Good moring" : "Good afternoon";
    return View();
}
```

2. Hello MVC : View

► View en dynamische output

- De View kan de ViewBag lezen (is property van View)
- @ : Razor code nugget. Start met een @ teken gevolgd door de C# statement die html/tekst retourneert
- @ zal C# instructie uitvoeren en het resultaat sturen naar de output en bovendien html encoden (hiermee wordt XSS vermeden).
 - < wordt vertaald naar < > naar > ...

```
@{
    Layout = null;
}

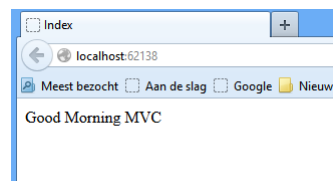
<!DOCTYPE html>

<html>
<head>
    <title>Index</title>
</head>
<body>
    <div>
        @ViewBag.Greeting MVC
    </div>
</body>
</html>
```

2. Hello MVC : View

► View en dynamische output

- Run de applicatie.
- Bekijk eens de broncode (Rechtsklik > Bron weergeven) (bevat geen inline code meer! Deze werd uitgevoerd op de server, server retourneert de gegenereerde html)

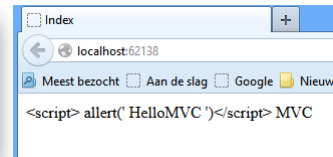


- Opm : ViewBag is nieuw in MVC3 en is een wrapper rond ViewData : een dictionary met key/value paren.
 - In MVC2 : ViewData["Greeting"] = "Hello world";
 - In MVC3 en MVC4 : ViewBag.Greeting = "Hello World";

2. Hello MVC : View

- ▶ View en dynamische output
 - Vb htмлencoding :

```
public ActionResult Index()
{
    ViewBag.Greeting = "<script>alert('Hello MVC ')</script>";
    return View();
}
```



- Ipv ViewBag kan ook het model doorgegeven worden aan de View.
=> Zie verder, SnakeEyes voorbeeld

3. SnakeEyes

- ▶ Demo SNAKE Eyes GAME

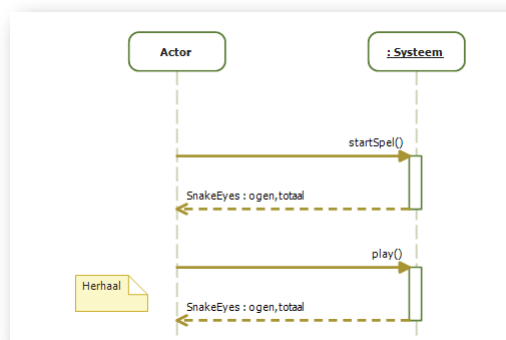


3. SnakeEyes

- ▶ Demo SNAKE Eyes GAME
 - Ontwerp
 - De starter MVC web applicatie
 - Model
 - Controller
 - View

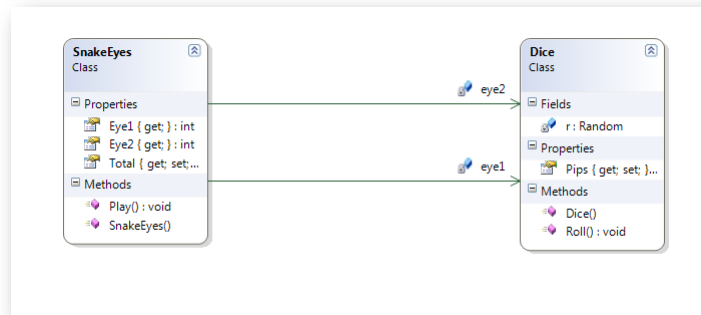
3. SnakeEyes : Ontwerp

- ▶ Ontwerp



3. SnakeEyes : Ontwerp

► Ontwerp domain



HoGent

3. SnakeEyes : Model

► Starter applicatie

- Bevat reeds Model : SnakeEyes en Dice.
- Klasse Dice

```
using System;

namespace SnakeEyesGame.Models
{
    public class Dice
    {
        private Random r;
        public int Pips { get; private set; }

        public Dice()
        {
        }

        public void Roll()
        {
        }
    }
}
```

Namespaces die binnen de code gebruikt worden..

Namespace van de klasse Dice

Automatic property

HoGent

3. SnakeEyes : Model

- Methode Roll()
 - Maakt gebruik van een Random generator.
 - Zoek klasse Random op in Help/Object Browser.
 - Hoe genereer je een getal tussen 1 en 6?
 - Declareer attribuut van type Random en maak instantie aan
 - Bij declaratie

```
private Random r = new Random();
```

- Of in de constructor. Initialiseer ook Pips op 6.

```
private Random r;  
public Dice()  
{  
    r = new Random();  
}
```

- Codeer de methode Roll.

HoGent

3. SnakeEyes : Model

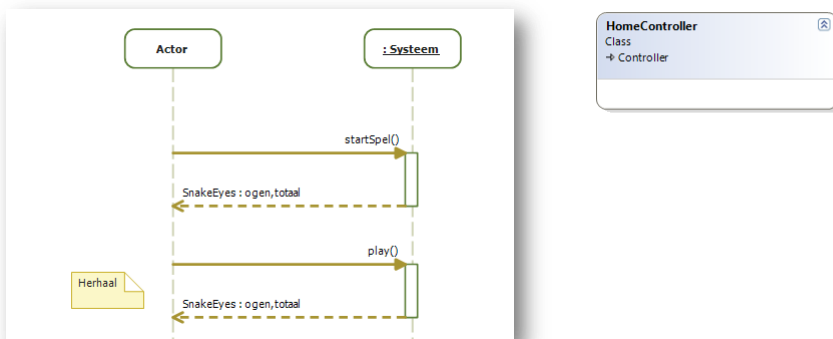
- Klasse SnakeEyes
- Vervolledig de methode Play.
 - Merk op :
 - Properties hebben geen setter
 - Regions

```
namespace SnakeEyesGame.Models  
{  
    public class SnakeEyes  
    {  
        #region Attributes  
        private Dice eye1;  
        private Dice eye2;  
        #endregion  
  
        #region Properties  
        public int Total { get; private set; }  
        public int Eye1 { get { return eye1.Pips; } }  
        public int Eye2 { get { return eye2.Pips; } }  
        #endregion  
  
        #region Constructor  
        public SnakeEyes()  
        {  
            eye1 = new Dice();  
            eye2 = new Dice();  
            Total = 0;  
        }  
        #endregion  
  
        #region Methods  
        public void Play()  
        {  
            // ...  
        }  
        #endregion  
    }  
}
```

HoGent

3. SnakeEyes : Controller

► Ontwerp ?



HoGent

3. SnakeEyes : Controller

- Maak een nieuwe controller aan : HomeController
 - Declareer een private attribuut snakeEyes van type SnakeEyes
 - Actie methode "Index" :
 - Maak een instantie aan van SnakeEyes
 - Roep methode View aan, en geef de instantie van SnakeEyes door
 - Ipv ViewBag kunnen we ook een object mee doorgeven aan de View, maar hierdoor wordt View Strongly typed en krijg je IntelliSense

HoGent

3. SnakeEyes : Controller

▶ HomeController

◦ Actie methode Index :

- Declareer een variabele van type SnakeEyes.

```
public class HomeController : Controller
{
    [HttpGet]
    private SnakeEyes snakeEyes;
    public
    {
        Cannot resolve symbol 'SnakeEyes'
```

- *using namespace* toevoegen. In Visual Studio Plaats cursor juist voor SnakeEyes. Een blauw vierkantje verschijnt. Ga over vierkantje met muis :

```
public class HomeController : Controller
{
    [HttpGet]
    private SnakeEyes snakeEyes;
    public
    {
        using SnakeEyesGame.Models;
        return
    }
}
```

- Of Resharper : doet een voorstel in het blauwe kader

```
public class HomeController : Controller
{
    [HttpGet]
    private SnakeEyesGame.Models.SnakeEyes? snakeEyes;
}
```

HoGent

3. SnakeEyes : Controller

▶ HomeController

◦ Index :

- Maak een instantie aan van SnakeEyes
- Roep View() aan en geef instantie van SnakeEyes door.
 - Strongly typed view : je geeft model mee als parameter van de view methode.
 - (<-> HelloMVC : daar maakten we gebruik van ViewBag, dynamic View)

```
public class HomeController : Controller
{
    private SnakeEyes snakeEyes;
    public ActionResult Index()
    {
        snakeEyes = new SnakeEyes();
        return View(snakeEyes);
    }
}
```

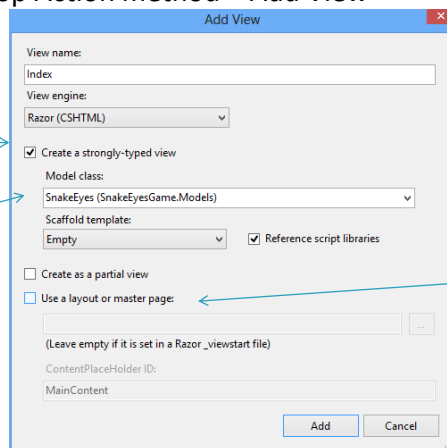
HoGent

3. SnakeEyes : View

- ▶ Maak Strongly typed view aan
 - Rechtsklik op Action method > Add View

Kies Strongly typed View

Kies Model : Als lijst leeg, Build solution : Menu > Build Solution. Je mag geen errors (zie ErrorList) hebben, anders dien je die eerst te verbeteren.



Uitvinken

HoGent

3. SnakeEyes : View

- ▶ Maak Strongly typed view aan

`@model SnakeEyesGame.Models.SnakeEyes`

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <title>Index</title>
</head>
<body>
    <div>

    </div>
</body>
</html>
```

↓

@model = strongly typed versie van ViewPage (generic class). De ViewPage bevat nu een Model. Merk op

- Gebruik de fully qualified name

OF voeg using toe

@using SnakeEyesGame.Models

@model SnakeEyes

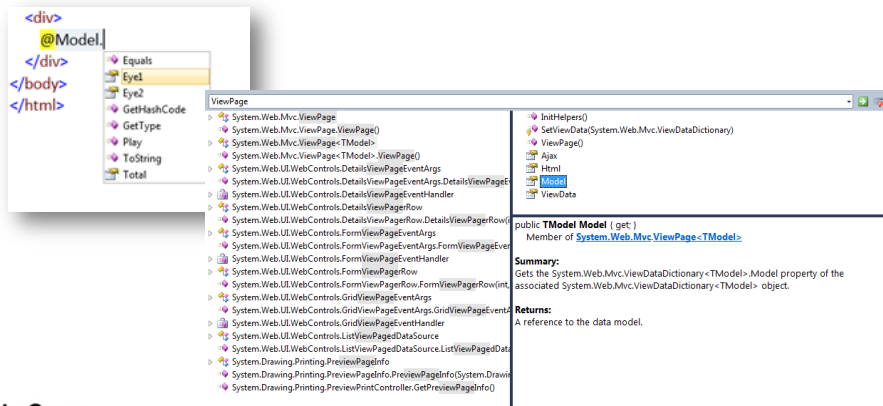
OF in web.config, onder

```
<system.web><pages><namespaces>
<add
namespace="SnakeEyesGame.Models"/>
```

HoGent

3. SnakeEyes : View

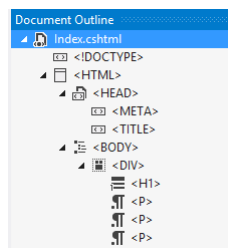
- ▶ Maak Strongly typed view aan
 - model toegankelijk via Model property van ViewPage
 - Intellisense



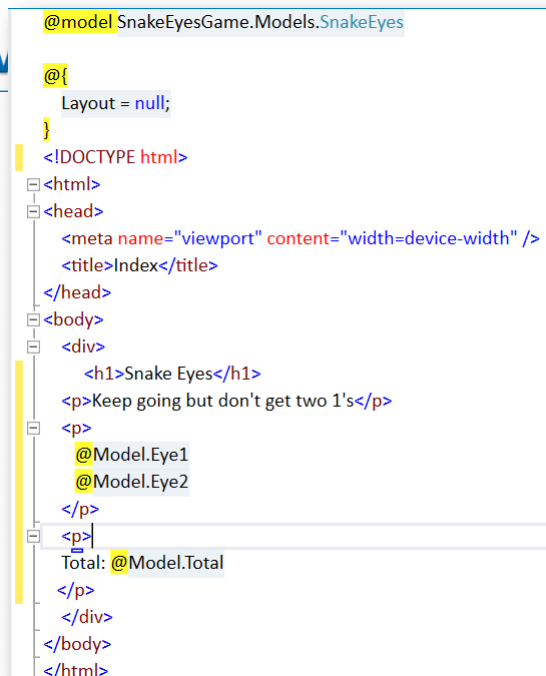
HoGent

3. SnakeEyes : View

- ▶ Maak Strongly typed view aan en run
- ▶ Open eventueel Document Outline venster



HoGent



3. SnakeEyes : View

► Toevoegen van stijl

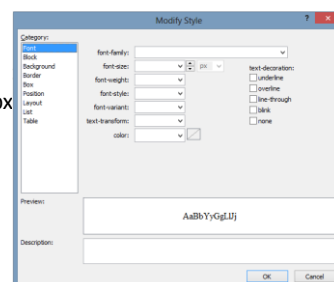
- De Content folder bevat reeds een stylesheet site.css. Die kan je verder aanvullen.
- Of je kan een nieuwe stylesheet aanmaken : In Solution Explorer > rechtermuis op Content folder > Add New Item. Selecteer Style Sheet. Geef een naam in en klik Add.
- Gebruiken van een stylesheet in een View:
 - Selecteer stylesheet Site.css in de solution explorer. Drop stylesheet op view na de title tag. De <link> tag wordt toegevoegd. Sla op!
 - Merk op : onderstaande wordt toegevoegd
 - ~ : virtual path, vertrekkende van de root

```
<link href="~/Content/Site.css" rel="stylesheet" />
```

3. SnakeEyes : View

► Css aanpassen

- Creatie nieuwe CSS rule. Open Site.css. Geef onderaan in .SnakeEye{
- plaats cursor binnen .SnakeEye > Rechtsklik Build Style laat toe om de rest van de rules in te geven
 - .SnakeEye
 - Font : Family : Arial en Size : 30px
 - Box : padding same for all en Top : 5px
 - Border : Borderstyle : Same for all en Solid
 - Dit zal de css genereren. Onderaan in de Preview zie je al een voorbeeld, Description bevat de css



3. SnakeEyes : View

- ▶ Toevoegen van een formulier
 - Van `Html.BeginForm` bestaan 13 overloads

Intellisense

```
@using (Html.BeginForm() {  
    ▲ 1 of 13 ▼ (extension) MvcForm HtmlHelper.BeginForm()  
    Writes an opening <form> tag to the response. When the user submits the form, the request will be processed by an action method.  
})
```

- In 1 van de overloads kan je de controllernaam en actionnaam ingeven (definieert de target URL bij klikken op Submit)

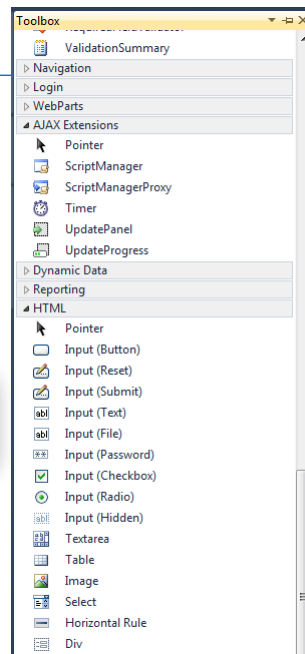
```
@using (Html.BeginForm("Play", "Home") {  
}
```

HoGent

3. SnakeEyes : View

- ▶ Toevoegen van een formulier
 - Voeg ook een submit knop toe.
 - Zelf code ingeven
 - Of Selecteer in Toolbox (View > ToolBox > HTML tab) een Input (Submit) en sleep op view page.

```
@using (Html.BeginForm("Play", "Home") {  
    <input id="buttonPlay" type="submit" value="Play" />  
}
```



HoGent

3. SnakeEyes : Controller

► Afhandelen van submitten van de form

- Formulier wordt verstuurd met http post

```
<% using (Html.BeginForm("Play", "Home")) { %>
```

- De actie die request ontvangt : **Play** in de HomeController
 - Moet de business operatie Play aanroepen van SnakeEyes
 - Kiest dan de view die gerenderd moet worden (mag hier ook de Index zijn) en geeft het model door.

```
public ActionResult Play()  
{  
    snakeEyes.Play();  
    return View("Index", snakeEyes);  
}
```

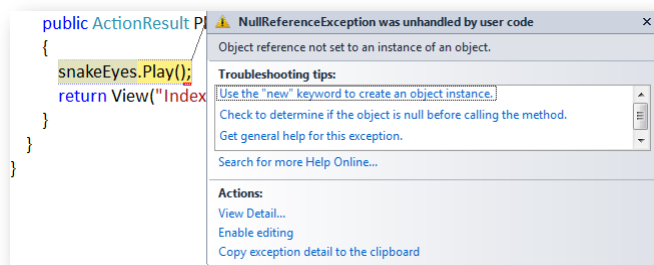
Als niet automatisch de default view van de actie gerenderd moet worden, kan je de naam van de te renderen view doorgeven

HoGent

3. SnakeEyes : Controller

► Afhandelen van submitten van de form

- Run de applicatie



snakeEyes bestaat niet meer???

HoGent

3. SnakeEyes : Controller

► State bijhouden

- Reden fout : HTTP is een **stateless protocol**
 - De toestand van een object wordt NIET bijgehouden tussen opeenvolgende requests. Er wordt telkens een nieuw object aangemaakt.
 - Oplossing : Session variabelen
 - Session
 - Resource op de server voor 1 gebruiker. Hierin kan je gegevens nodig voor die gebruiker tijdens 1 sessie (= 1 visit van de gebruiker aan de site) bijhouden => DUS iedere bezoeker heeft zijn eigen sessie object.
 - Die gegevens zijn toegankelijk vanuit alle pagina's binnen die webapplicatie en bevatten de info van die specifieke bezoeker
 - Hoe weet server nu welk sessie object tot welke bezoeker behoort? Bij het eerste verzoek van 1 gebruiker, maakt de server 1 sessie object aan met een uniek gegenereerd userID. De server stuurt een cookie met de userID naar de browser. Telkens de bezoeker dan een pagina opvraagt binnen de webapplicatie wordt cookie meegezonden.

3. SnakeEyes : Controller

► State bijhouden

- Sessie gegevens lezen en schrijven
 - Verzameling van naam/waarde paren.
 - Schrijven

```
HttpContext.Session["SnakeEyes"] = snakeEyes;
```
 - Lezen

```
snakeEyes= (SnakeEyes) HttpContext.Session["SnakeEyes"];  
OF snakeEyes = HttpContext.Session[" SnakeEyes "] as SnakeEyes;
```
- Het session object wordt vernietigd igv
 - De instructie `HttpContext.Session.Abandon` (gebruiken bij logout)
 - Of bij time out van de sessie : default 20 min na laatste request (staat ingesteld in `Web.config`)

3. SnakeEyes : Controller

- ▶ State bijhouden
 - De code in de controller wordt

```
public class HomeController : Controller
{
    private SnakeEyes snakeEyes;

    public ActionResult Index()
    {
        snakeEyes = new SnakeEyes();
        HttpContext.Session["SnakeEyes"] = snakeEyes;
        return View(snakeEyes);
    }

    public ActionResult Play()
    {
        snakeEyes = HttpContext.Session["SnakeEyes"] as SnakeEyes;
        snakeEyes.Play();
        return View("Index", snakeEyes);
    }
}
```

3. SnakeEyes : Controller

- Run.
- Er worden steeds 2 dezelfde random nummers gegenereerd.

Number Random

Random number generators cannot return truly random numbers. They return instead sufficiently random numbers. With the **Random** type in the .NET Framework, we generate in our C# programs numbers that appear very random. This satisfies almost all algorithms and requirements that involve random numbers.

Key points:

You can use the Random class as a member variable. If you create local Random instances, they may repeat; try instead to reuse the same Random instance.

3. SnakeEyes : Controller

- ▶ Random? Hoe oplossen?

```
public SnakeEyes()
{
    Random r = new Random();
    eye1 = new Dice(r);
    eye2 = new Dice(r);
    Total = 0;
}
```

```
public class Dice
{
    private Random r;
    public int Pips { get; private set; }

    public Dice() : this(new Random())
    {
    }

    public Dice(Random r)
    {
        this.r = r;
        Pips = 6;
    }
}
```

4. MVC Flow

- ▶ MVC Flow
 - Step 1
 - Binnenkomende request wordt verstuurd naar Controller. Daar wordt de bijhorende actie uitgevoerd.
 - De controller en actie wordt bepaald door de routing



4. MVC Flow

► MVC Flow : Step 2

- **Controller** verwerkt request, communiceert met Domein en maakt **Model** (de gegevens die gerenderd moeten worden. Kan een ViewBag(ViewData) of domein of ViewModel object zijn)

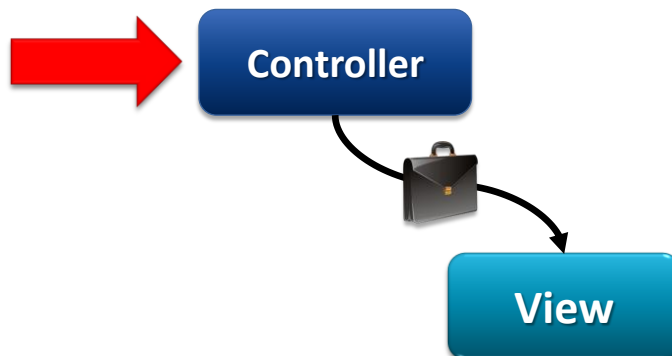


HoGent

4. MVC Flow

► MVC Flow : Step 3

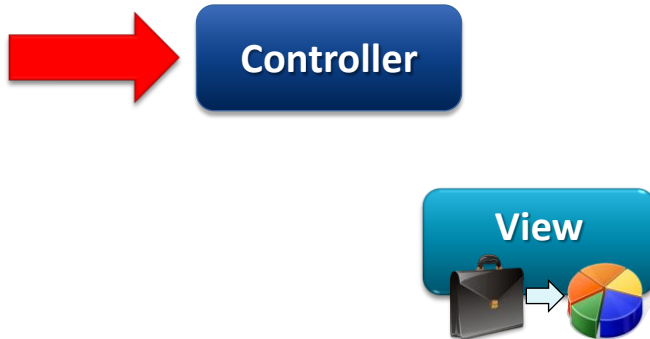
- **ViewBag en Model** wordt doorgegeven aan de **View**



HoGent

4. MVC Flow

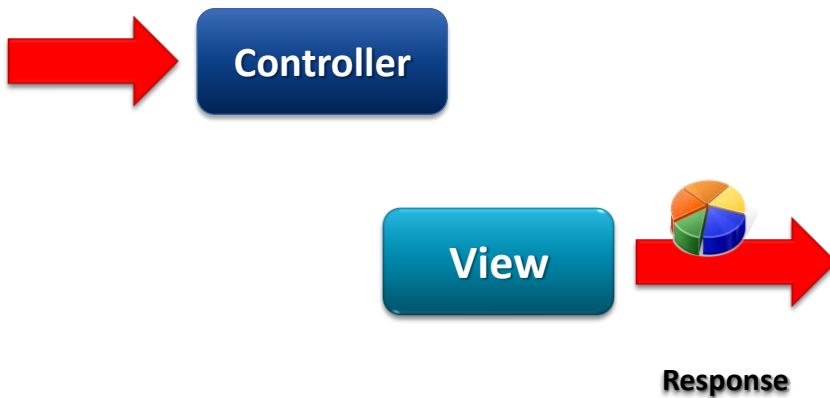
- ▶ MVC Flow : Step 4
 - View transformeert Model naar juiste output formaat



HoGent

4. MVC Flow

- ▶ MVC Flow : Step 5
 - Response is aangemaakt en wordt naar de browser verstuurd als antwoord op de request



HoGent

5. Bundling en Minification

- ▶ Wat als je site meerdere css en js files bevat?
- ▶ Bundling and minification reduceert file size en maakt site performanter
 - Verpakt een verzameling van JS of CSS files in 1 element, en verkleint grootte door minification (verwijderen van blank spaces, commentaar, reducing identifiers).
 - Gebeurt at runtime, zodat process de user agent kan identificeren (bvb IE, Mozilla, etc.), en zo de compressie kan verbeteren door af te stemmen op de browser (bvb verwijderen van Mozilla specifieke onderdelen als request komt van IE)

HoGent

5. Bundling en Minification

- ▶ De configuratie
 - Global.asax, Application_Start

```
BundleConfig.RegisterBundles(BundleTable.Bundles);
```
 - In App_Start folder, BundleConfig.cs
 - Namespace System.Web.Optimization
 - Een script bundle is een combinatie van een virtual path (bvb ~/bundles/jquery, d.i. de eerste parameter in de constructor) en een lijst van files die de bundle zal bevatten. (via de Includes). Include kan van filename of filename met wildcard of {version}
 - In view

```
@Scripts.Render("~/bundles/jquery")
@Styles.Render("~/Content/css")
```

HoGent

5. Bundling en Minification

► De

```
public static void RegisterBundles(BundleCollection bundles)
{
    bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
        "~/Scripts/jquery-{version}.js"));

    bundles.Add(new ScriptBundle("~/bundles/jqueryui").Include(
        "~/Scripts/jquery-ui-{version}.js"));

    bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
        "~/Scripts/jquery.unobtrusive*",
        "~/Scripts/jquery.validate*"));

    bundles.Add(new StyleBundle("~/Content/css").Include("~/Content/site.css"));

    bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
        "~/Content/themes/base/jquery.ui.core.css",
        "~/Content/themes/base/jquery.ui.resizable.css",
        "~/Content/themes/base/jquery.ui.selectable.css",
        "~/Content/themes/base/jquery.ui.accordion.css",
        "~/Content/themes/base/jquery.ui.autocomplete.css"));
}
```

HoGent

5. Bundling en Minification

► De configuratie

- Maak een 2de css file aan SnakeEyes.css en verplaats .SnakeEyes
- Pas index.cshtml aan en run

```
<meta name="viewport" content="width=device-width" />
<link href="~/Content/Site.css" rel="stylesheet" />
<link href="~/Content/SnakeEyes.css" rel="stylesheet" />
<title>SnakeEyes</title>
</head>
```

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency
localhost	GET	200 OK	text/html	Other	832B 618B	13 12
Site.css /Content	GET	304 Not Modified	text/css	localhost:8 Parser	278B 1.00KB	7 6
SnakeEyes.css /Content	GET	304 Not Modified	text/css	localhost:9 Parser	282B 176B	3 2

HoGent

5. Bundling en Minification

► De configuratie

- Pas bundleconfig aan

```
bundles.Add(new StyleBundle("~/Content/css").Include("~/Content/site.css",  
    "~/Content/SnakeEyes.css"));
```

- Pas index.cshtml aan

```
]<head>  
    <meta name="viewport" content="width=device-width" />  
    <title>SnakeEyes</title>  
    @Styles.Render("~/Content/css")  
</head>
```

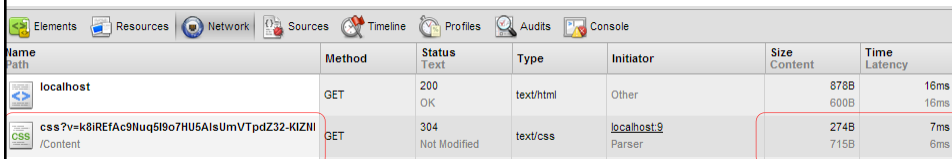
- Run. Nog steeds hetzelfde resultaat. In Debug mode wordt dit niet toegepast. Moeilijker om js te debuggen, css te bekijken
- Als je het toch wenst toe te passen in Debug mode. Pas global.asax aan `BundleTable.EnableOptimizations = true;`
 - Of web.config `<compilation debug="false" .../>`

HoGent

5. Bundling en Minification

► De configuratie

- Run



Name	Path	Method	Status	Type	Initiator	Size	Content	Time
			Text					Latency
	localhost	GET	200 OK	text/html	Other		878B 600B	16ms 16ms
CSS	css?v=k8IREfAc9Nuq5I9o7HU5AIsUmVTpdZ32-KIZNI /Content	GET	304 Not Modified	text/css	localhost:9 Parser	274B 715B		7ms 6ms



Name	Path	Headers	Preview	Response	Cookies	Timing
	localhost		1 body{font-size:.85em;font-family:"Segoe UI", Verdana,Helvetica,Sans-Serif;col			
CSS	css?v=k8IREfAc9Nuq5I9o7HU /Content					

- Bekijk ook eens bron weergeven. Maar 1 css

HoGent

6. Oefening

- ▶ SnakeEyes : Voeg de opmerking “Oeps you did it again!” toe als $2 * 1$ gegoid werd.

