

Katholieke Hogeschool Mechelen – Informaticamanagement & Multimedia

ASP.NET

Academiejaar 2009-2010

Docent: J. Hens

Hoofdstuk 1: Starten met ASP.NET	6
Doelstellingen van deze cursus	6
Doelstellingen en Veronderstellingen	6
Welke topics komen aan bod?	6
C# of VB.NET	6
Wat is ASP.NET in vergelijking met ASP?	7
Client versus Server side scripting	11
Wat hebt u nodig om te ontwikkelen in ASP.NET	13
Editor	13
Web server	13
Database server	14
Extra benodigdheden	14
Testen of .NET goed geïnstalleerd is	14
Introductie tot Visual Studio	16
Een nieuwe webapplicatie maken	16
Drie plekken om code te schrijven	19
Wat is de beste plek om uw code te schrijven?	20
Het properties scherm	21
De toolbox	22
Intellisense	24
Webpagina's bekijken en debuggen	25
Programmeren in ASP.NET met VB.NET	27
Variabelen	27
Code in commentaar plaatsen	28
Procedures zonder retourwaarde	28
Procedures mét retourwaarden	28
Conditie's	29
Herhalingen	30
Handboeken	30
Oefeningen bij hoofdstuk 1	31
Hoofdstuk 2: Servercontrols	35
Server controls inleiding	35
Web-controls in meer detail	47
Vallidatie-controls in meer detail	50
Oefeningen bij dit hoofdstuk	51

Hoofdstuk 3: Web forms	54
ASP.NET Web Forms	54
Viewstate	56
Applicatievariabelen, sessievariabelen en cookies	57
Voorbeeld: Status bijhouden van de laatst gezochte films	61
Een cookie plaatsen	61
Een cookie uitlezen	61
Een cookie verwijderen	62
Uitwerking in onze applicatie	62
Oefeningen bij dit hoofdstuk	65
Oefening 1	65
Hoofdstuk 4: Databanken in ASP.NET	66
ADO.NET	66
Een databank aanmaken binnen Visual Studio 2008	66
ASP.NET 2.0 en hoger	71
Databroncontrols	71
Weergavecontrols	72
Werken met SqlDataSource	72
Gegevens toevoegen	78
Een databaselaag opzetten	82
Databank aanmaken en afwerken	82
Wat is een databaselaag en wat zijn de voordelen	82
Opzetten	83
De databanklaag in actie met presentatiecontrols	84
De databanklaag aanspreken via code	95
Data ophalen via het DataReader object	96
Data manipuleren via manueel opgebouwde queries	97
ExecuteNonQuery	98
Extra: toepassing uit de filmapplicatie	99
Alle landen importeren naar SQL Server 2005 Express Edition	99
Relaties tussen de tabellen leggen	101
Hoofdstuk 5: ASP.NET applicaties en beveiliging	105
Opzetten van security	105
Wat gebeurt er achterliggend?	112

Gebruik van custom Membership & Role providers (remote sql server)	114
Extra: Werken met roles	117
De beheerpagina maken enkel voor moderators	121
Toepassing: Een registratieformulier maken	122
Error bij het connecteren naar uw MS SQL Server	123
Hoofdstuk 6: Objectgeoriënteerd programmeren binnen ASP.NET	124
Voordelen van objectoriëntatie	124
Overerving	124
Polymorfisme	125
Inkapseling	125
Abstractie	126
Beginnen met OO in .NET	130
Overerving of inheritance	137
Methoden	140
Codevoorbeelden uit de filmapplicatie	143
De klasse MovieSet	144
De klasse feedback	147
Meer informatie en tutorials om te starten met OOP binnen .NET	148
Hoofdstuk 7: Programmeren volgens een 3-lagenmodel	149
Laag 1: Data –laag (DAL, Data Access Layer)	150
Laag 2: business-laag of logische laag (BLL, Business Logic Layer)	150
Laag 3: presentatielaag (Presentation Layer)	151
Toepassing 1: een timesheet applicatie	153
Laag 1: DAL	153
Toepassing 2: Feedback posten en uitlezen bij elke filmsetlocatie	158
Het formulier toevoegen	168
Verbeteringen aan het feedbackformulier:	180
Workshop 1: “Managing the Styles of a Web Site”	182
Master pages	182
Cascading Style Sheets	186
Workshop 2: Een rijkere gebruikersinterface maken met AJAX	188
Web services	188
Gebruik van AJAX in ASP.NET	188
Voorbeeld1	188
Voorbeeld2	189

Gebruik van de AJAX Control Toolkit	193
Voorbeeld: CalendarExtender	196
Voorbeeld: ModalPopupExtender	197
<i>Workshop 3: Web Services aanmaken en consumeren</i>	202
Een web service aanmaken	202
AJAX en web services	211
Extra documentatie	213
<i>Workshop 4: API's & mashups</i>	214
Google Maps & .NET	215
GeoCoding met web services	221
Extra documentatie	222
<i>Kleine extra's toevoegen aan ons project</i>	223
Een teller voor het aantal toegevoegde filmsets toevoegen	223

Hoofdstuk 1: Starten met ASP.NET

Doelstellingen van deze cursus

Met deze cursus betrachten we dat de student op het einde in staat is om met behulp van ASP.NET een interactieve website te bouwen. De verschillende topics zijn telkens een introductie op een bepaald thema, die de student moet aanzetten tot het verder uitdiepen van zijn kennis en het creatief omgaan met die kennis. Dit kan onder andere op het net: [Microsoft](#)

Doelstellingen en Veronderstellingen

Hoewel de meest recente versie van Visual Studio een uitstekende hulp kan bieden in het snel opbouwen van webpagina's is een grondige kennis van xHTML en CSS nodig. We gaan er in deze cursus van uit dat u xHTML en CSS volledig onder de knie heeft.

Het leggen van connecties naar een databank kan in ASP.NET nieuw zijn voor de meeste studenten. De queries die gebruikt worden om gegevens uit die databank op te halen worden meestal opgebouwd in de taal SQL, waarvan we een basiskennis verwachten.

Welke topics komen aan bod?

- Web forms: events, controls, code-behind, viewstate
- Servercontrols: HTML-controls, web-controls, gebruiker-controls
- Status bijhouden: viewstate, session-state, application-state
- Connecties leggen naar een databank en de gegevens in de databank manipuleren met behulp van een webinterface
- Formulervalidatie, foutafhandeling en beveiliging van applicaties
- Gebruik van master pages om onderhoud aan lay-outs te vereenvoudigen
- Gebruik van AJAX binnen ASP.NET om uw interface levendiger en aantrekkelijker te maken voor uw gebruikers.
- Objectgeoriënteerd programmeren binnen ASP.NET
- Het bouwen en aanspreken van een Data Access Layer (DAL)
- Programmeren in drie lagen (Data Access Layer + Business Logic Layer + Presentation Layer)
- Gebruik van API's: Google Maps als case

C# of VB.NET

Er zijn verschillende talen geschikt voor ontwikkelingen in .NET. De bekendste twee zijn C# (C-sharp) en VB.NET (VB dot net). C# wordt vaak als dé .NET-taal bij uitstek genoemd. VB.NET is echter een taal die sneller door startende ontwikkelaars binnen .NET opgenomen wordt. De meeste boeken en zelfs websites over .NET zullen hun codevoorbeelden zowel in VB.NET als in C# publiceren. Veel mensen die de overstap maken op ASP.NET hebben reeds een achtergrond binnen VB en zullen dus makkelijker met VB.NET kunnen dan met C#. We hebben dan ook gekozen om VB.NET te gebruiken als taal binnen deze cursus.

Wat is ASP.NET in vergelijking met ASP?

Deze cursus handelt over ASP.NET en niet over ASP. De essentie voor beide talen is dezelfde (het maken van dynamische web pagina's). ASP.NET heeft meer weg van het schrijven van een programma dat een HTML pagina zal genereren. Terwijl ASP eerder een HTML pagina is met daartussen scripts geschreven. Het zal u dan ook niet verwonderen dat programmeurs in de wolken zijn van ASP.NET. Dankzij het .NET-framework zijn deze programmeurs nu in staat om met hun bestaande programmeerkennis snel webapplicaties te ontwikkelen.

ASP staat voor Active Server Pages en is een technologie die het mogelijk maakt om dynamische en interactieve webpagina's te maken. ASP code werd vaak in een gewone HTML-pagina geschreven om deze verder dynamisch op te bouwen.

.NET is een framework dat ondertussen haar derde generatie heeft bereikt. Microsoft omschrijft het .NET-framework als volgt: "Een omgeving voor de ontwikkeling en uitvoering die het mogelijk maakt om verschillende talen en bibliotheken naadloos samen te laten werken om op Windows gebaseerde applicaties te creëren die eenvoudig te bouwen, te beheren, uit te rollen en te integreren zijn met andere op het netwerk aangesloten systemen".

Voor ons zou een betere omschrijving zijn dat .NET een omgeving is voor het ontwikkelen en uitvoeren van softwareapplicaties. Wat voor webontwikkelaars interessant is, is dat .NET over veel standaardfunctionaliteiten beschikt die vrij snel en eenvoudig in gebruik genomen kunnen worden. Het .NET-framework stelt ons dus in staat sneller applicaties te bouwen voor het web (en daarbuiten). .NET kan gezien worden als een raamwerk waarbinnen applicaties ontwikkeld kunnen worden.

Zoals reeds gezegd is het ontwikkelen van die applicaties mogelijk in verschillende talen. Niet alleen VB.NET of C# kunnen gebruikt worden binnen .NET, maar ook COBOL, Perl, Python, ... Het is zelfs mogelijk uw applicatie in verschillende talen te schrijven en deze toch te combineren tot één samenhangend geheel.

Microsoft zelf beschrijft het verschil tussen ASP en ASP.NET als volgt:

Whats the difference between ASP and ASP.NET?

ASP (Active Server Pages) and ASP.NET are both server side technologies for building web sites and Web applications, but ASP.NET is not simply a new version of ASP. ASP.NET has been entirely re-architected to provide a highly productive programming experience based on the .NET Framework, and a robust infrastructure for building reliable and scalable web applications.¹

Meer over de verschillen tussen ASP en ASP.NET kunt u vinden op onder andere volgende website:
<http://www.beansoftware.com/ASP.NET-Tutorials/Classic-ASP-vs-ASP.NET.aspx>

Belangrijk om te beseffen is dat we te maken hebben met een "server side" technologie waarmee we een webapplicatie zullen bouwen. Wanneer we gewone webpagina's opbouwen via xHTML en CSS worden deze als volgt aan de webbrowsers van uw bezoekers aangeleverd:

1. URL wordt op de client ingetikt

¹ <http://msdn.microsoft.com/en-us/ASP.NET/aa336670.aspx>

2. De client lokaliseert de webserver waar de informatie over de webpagina is opgeslagen door middel van de URL en vraagt aan de webserver die specifieke pagina op
3. De server zoekt de pagina in zijn bestanden en als de pagina gevonden kan worden wordt deze teruggestuurd naar de client
4. De client ontvangt de html-pagina en door middel van een browser wordt die code omgezet in een leesbare webpagina

Nadat de pagina is doorgestuurd is er geen enkele link meer tussen de client en de server. Dit is het principe van een connectieloos protocol. Wat er op de client met de ontvangen pagina gebeurt, heeft totaal geen invloed op de webserver. Pas bij een volgende vraag naar informatie zal de webserver terug reageren. Zulke aanvraag wordt ook een “request” genoemd.

Daar waar klassieke ASP en PHP *geïnterpreteerd* wordt door een webserver, bestaat een ASP.NET applicatie uit *gecompileerde* code. Bij PHP en ASP bleef de code van de applicatie op de webserver nog steeds perfect leesbaar voor ons (of toch voor diegenen die deze programmeertalen beheersen). ASP.NET wordt *gecompileerd* naar DLL bestanden, waardoor de vaak gevoelige broncode niet meer met het blote oog leesbaar is. Bovendien wordt de uitvoeringssnelheid positief beïnvloed dankzij deze compilatie van code, wat vooral merkbaar wordt bij grote hoeveelheden code die uitgevoerd moet worden.

Het uitvoeren van een ASP.NET applicatie kan systematisch als volgt beschreven worden:

1. URL (met extensie .aspx) wordt op de client ingetikt
2. De client lokaliseert de webserver waar de informatie over de webpagina is opgeslagen d.m.v. de URL en vraagt aan die server een bepaalde pagina op te zoeken.
3. De server zoekt die pagina in zijn bestanden.
4. In plaats van de pagina direct terug te sturen, voert de server het gecompileerde ASP.NET programma uit en genereert een nieuwe pagina die enkel nog maar uit HTML bestaat (en niet meer uit ASP.NET-code). Indien de pagina voor het eerst wordt opgeroepen, wordt de ASP.NET code eerst gecompileerd.
5. Deze HTML-pagina (met .aspx extensie) wordt teruggestuurd naar de client.
6. De client ontvangt de HTML-file en wordt door de browser geïnterpreteerd en op het scherm getoond.

Gezien de webserver de ASP.NET code eerst uitvoert vooraleer er HTML code wordt gegenereerd kan deze HTML-code nog worden gemanipuleerd op het moment van de aanvraag van de pagina. Afhankelijk van variabelen, parameters, databankgegevens, ... kan dezelfde ASP.NET pagina steeds andere HTML genereren en dus een heel andere inhoud of lay-out hebben.

Omdat de uiteindelijke pagina die wordt teruggestuurd enkel nog maar HTML code bevat kunnen ASP pagina's in eender welke browser bekeken worden, zonder dat er extra technologie nodig is aan de client zijde. Wel moet de server waarop de ASP.NET pagina's staan speciaal worden geconfigureerd.

Deze manier om web pagina's dynamisch te maken geeft een hele waaier van mogelijkheden:

- Het is voor een gewone gebruiker eenvoudiger om gegevens te wijzigen/op te slaan in een databank (met de juiste interface), dan dat hij/zij HTML code moet aanpassen en de nieuwe pagina moet uploaden naar de site.
- Het is ook mogelijk om de inhoud/lay-out van een pagina afhankelijk te maken van de gebruiker die is ingelogd.
- Ook het beheer van gegevens kan gebeuren via webpagina's, onafhankelijk van de PC waar we zitten en de software die daarop is geïnstalleerd. Er is enkel een web browser nodig. Dit maakt het onderhoud van een applicatie zeer eenvoudig. Elke wijziging van een applicatie gebeurt centraal op de webserver.

Een voorbeeldje ter verduidelijking:

```
<%@ Page Language="VB" CodeFile="voorbeeld_000.aspx.vb"
Inherits="voorbeeld_000" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>

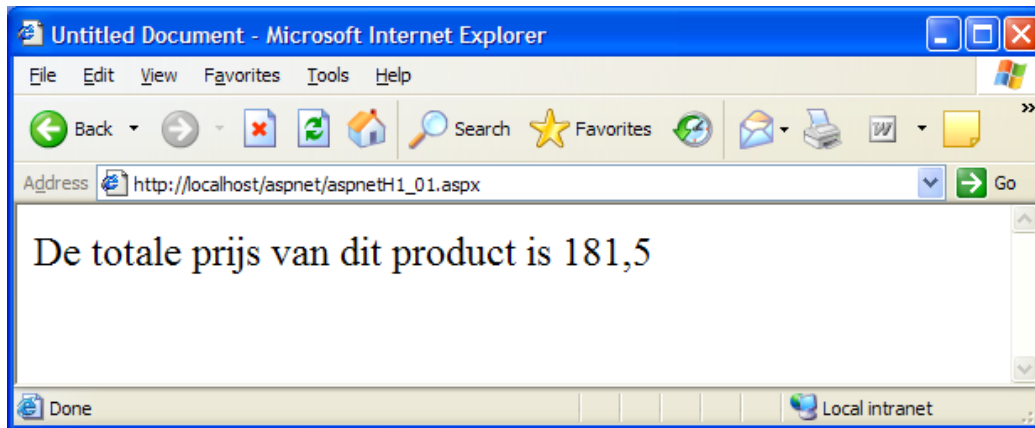
    <script runat="server" language="vb">
        Sub Page_Load(ByVal Src As Object, ByVal E As EventArgs)
            If Not IsPostBack Then
                Dim intPrijs, intBtw, intTotaal
                intPrijs = 150
                intBtw = 21
                intTotaal = intPrijs * (1 + (intBtw / 100))
                resultaat.Text = "De prijs van dit product is " & intTotaal
            End If
        End Sub
    </script>

</head>
<body>

    <asp:Label ID="resultaat" Font-Size="18" runat="server" />

</body>
</html>
```

REFERENTIE: VOORBEELD_000.ASPX



Deze pagina is natuurlijk nog niet dynamisch, maar ze leert ons wel al enkele zaken. Vooreerst volgt nu de HTML code die door deze ASP pagina werd gegenereerd en die u te zien krijgt als u in uw browser View → Source kiest.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>
    Untitled Page
</title></head>
<body>

    <span id="resultaat" style="font-size:18pt;">De totale prijs van dit
product is 182</span>

</body>
</html>
```

Wat opvalt:

- Alle ASP.NET code is uit de uiteindelijk source die werd doorgestuurd verdwenen. De code zoals de browser die te zien krijgt is dus standaard HTML.
- De variabelen die worden gebruikt zijn van het type variant en nemen subtype aan bij eerste toewijzing van een waarde. Zo is intTotaal blijkbaar geen integer vermits dat dit getal decimalen bevat. Dit maakt een naming convention zeer moeilijk, tenzij u er een gewoonte van maakt om telkens bij de eerste toewijzing de uitkomst te converteren naar het juiste type.
- ASP.NET code is afgeleid van VB. Iemand die goed kan programmeren in VB, die zal niet veel moeite hebben om correcte ASP.NET pagina's te schrijven.

Het is niet nodig dat u alle details van de code begrijpt, deze zullen verder in de cursus wel duidelijk worden.

Een kleine bedenking bij het codevoorbeeld is dat op dit moment nog steeds programmeercode doorheen de HTML geschreven werd. Later in deze cursus zullen we zien dat dit niet de gewenste

manier van programmeren is. Om dit eerste codevoorbeeld eenvoudig te houden werd gekozen om de code in de HTML-pagina zelf te schrijven.

Client versus Server side scripting

Uit de uitleg van het vorige hoofdstuk heeft u kunnen afleiden dat ASP.NET pagina's code bevatten die door de server worden verwerkt. Naast ASP.NET zijn er andere scripting talen die het mogelijk maken om dynamische websites te bouwen. Zo is PHP een volwaardig niet-microsoft alternatief. De PHP code wordt, net als de vroegere ASP code, op de server geïnterpreteerd.

De verwerking van de web pagina's op de server heeft verschillende voordelen:

- De code die in de scripts werd geschreven kan nooit worden gelezen door de gebruiker, vermits dat deze nooit zomaar wordt doorgestuurd. Hierdoor zijn deze scripts zeer veilig.
- Omdat de pagina's volledig worden opgebouwd in HTML voordat ze worden doorgestuurd, zal er enkel die code worden doorgestuurd die strikt noodzakelijk is. Hierdoor is er minder trafiek over het net en heeft men snellere load tijden.
- Het gebruik van server scripts is browseronafhankelijk. Er moet dus geen rekening gehouden worden op welk platform en met welke browser een pagina wordt bekeken. Het is wel server afhankelijk, dus moet de webserver op de juiste manier worden ingesteld.
- Omdat de pagina volledig wordt opgebouwd op de server hebben we de mogelijkheid om data aan te bieden die niet op de client staat. De gegevens zitten dus niet verspreid over het ganse netwerk, maar worden centraal op een server opgeslagen.

De nadelen:

- Eenmaal dat de pagina is opgebouwd, zijn alle scripts eruit. M.a.w. de pagina's zijn omgevormd naar statische pagina's.
- Omdat elke pagina eerst moet worden verwerkt op de server is er een hogere belasting van de webserver ten opzichte van statische HTML pagina's. Door ASP.NET pagina's op de webserver te compileren (i.p.v. interpreteren zoals bij PHP en ASP) tracht Microsoft dit nadeel wat in te perken.

Tegenover server side programmeren staat client side scripting. Dit zijn scripts die mee worden gestuurd met de HTML code en die op de client worden geïnterpreteerd. De meest bekende voorbeelden van client side scripts zijn Javascript en Vbscript. Beide talen zijn gebaseerd op respectievelijk Java en VB maar zijn beperkter en worden niet gecompileerd. Vooral dankzij de voortdurend toenemende populariteit van AJAX op het web, is Javascript de belangrijkste client side scriptingtaal geworden.

Van client side script's is de code leesbaar in de html source en herkenbaar door de <script>-tag. In de openingstag kan ook de gebruikte taal gespecificeerd worden.

```
<html>
<head>
<title>voorbeeld javascript</title>

<script language="javascript">
<!--
  alert("dit is javascript");
//-->
</script>
```

```
</head>
<body>
test pagina
</body>
</html>
```

Client side scripts kunnen gebruikt worden om dynamische client effecten te realiseren: wisselende afbeeldingen, popups, openen van nieuwe browser windows, speciale cursors,... Zij zijn dan ook zeer geschikt om event driven pagina's te maken.



Echter is er bij client side verwerking totaal geen band niet meer tussen de web pagina in de browser, en de site die op de server staat. Ook zijn client side scripts, browserafhankelijk. Meer specifiek zijn deze scripts afhankelijk van de script engine die wordt meegeleverd bij de browser en die de scripttaal moet interpreteren.

Door beide technieken te combineren (client cripting en server scripting) wordt het wel mogelijk om krachtige applicaties te schrijven. Client scripts zorgen voor het opvangen van de events op de webpagina zoals bijvoorbeeld een muisklik op een bepaald element in de pagina. De client stuurt op zijn beurt een request naar de server die aan de serverside wordt verwerkt in serverscripts. Afhankelijk van de request wordt er een nieuwe specifieke pagina opgebouwd die wordt teruggestuurd naar de client. Door deze interactie kunnen we pas echt gaan spreken van een webapplicatie. In die webapplicaties streeft men naar een nauwere koppeling tussen client en server zonder te moeten inboeten op de onafhankelijkheid van beide.

Bij de oudere scriptingtalen (ASP, PHP) moest men de samenwerking tussen client- en server-scripts zelf schrijven. Men moest zelf zorgen voor het bijhouden van de status van een pagina/applicatie. In ASP.NET heeft men allerlei objecten gecreëerd die zelf de nodige scripts genereren en parameters bijhouden om een volwaardige webapplicatie te maken. Dit maakt dat het bouwen van ASP.NET applicaties veel vereenvoudigd is. Maar het heeft als grote nadeel dat ASP.NET meer gaat lijken op een programmeertaal in plaats van een design taal.

Naast het dynamisch maken van webpagina's kan men met ASP.NET van een website, die eigenlijk een losse verzameling web pagina's is, een echte applicatie maken. Met de application en session statussen wordt aan toestandsloze webpagina's een extra dimensie gegeven door meta data bij te houden over de verschillende pagina's heen. Een duidelijk voorbeeld hiervan is het gebruik van een winkelmandje op een website.

In veel webwinkels kan je producten toevoegen aan een winkelmandje. Dit mandje blijft in een goede applicatie opgevuld met de geselecteerde artikelen, ook wanneer we naar andere pagina's surfen.

SHOPPING CART			
Note: You do not have to have a PayPal account to place an order. When you are redirected to the PayPal page, just click the continue link by the credit card logos to the left of the login.			
Item	Qty	Description	Total
	<input type="text" value="1"/>	<u>WILL MIX FOR FOOD - UNISEX S BROWN/WHITE INK</u> \$ 20.00 each	\$ 20.00 remove
	<input type="text" value="1"/>	<u>DROP BEATS - UNISEX S GRAY</u> \$ 22.00 each	\$ 22.00 remove
Subtotal:			\$ 42.00 USD

Wat hebt u nodig om te ontwikkelen in ASP.NET

Editor

Om de code te schrijven voor een .NET-applicatie hebt u vanzelfsprekend een editor nodig. In het eenvoudigste geval zou u gebruik kunnen maken van kladblok/notepad, maar gezien dit een teksteditor is in de simpelste vorm is dat niet de meest gebruiksvriendelijke manier om applicaties te schrijven.

Over de verschillende .NET-versies heen zijn er meerdere editors verschenen en ook weer verdwenen. Op het moment van schrijven is de meest gebruiksvriendelijke editor Visual Studio van Microsoft zelf. De basisversie van dit pakket is trouwens compleet gratis beschikbaar via download.

Op volgende URL kan u de meest recente versie van Visual Studio downloaden:

<http://www.microsoft.com/express/>

Op dit moment is net versie 2008 van de Visual Studio uitgebracht. Gezien we in deze cursus trachten de meest actuele technieken toe te lichten, raden we ontwikkelaars ook aan versie 2008 te downloaden van Visual Studio. Voor deze cursus is het voldoende om Visual Web Developer 2008 te downloaden. Hierin zitten een aantal recente uitbreidingen (onder andere voor AJAX!) reeds ingebouwd, die normaal gezien afzonderlijk geïnstalleerd zouden moeten worden indien u met versie Visual Studio 2005 zou werken.

Web server

Zoals dat voor elke website en webapplicatie geldt, is er een webserver nodig om uw pagina's aan te bieden aan de rest van de wereld .

Internet Information Server (IIS) is de meest geschikte webserver voor het hosten van een .NET-applicatie. Deze webserver werkt naadloos samen met al wat het .NET-framework te bieden heeft. Werkt u met Windows XP Professional, dan wordt versie 5.1 van IIS meegeleverd. Windows Vista is voorzien van versie 7.0 van IIS en biedt standaard ondersteuning voor .NET 3.0.

Apache, de meest gebruikte webserver op UNIX/LINUX systemen is helaas standaard niet geschikt voor het draaien van ASP.NET-applicaties. Met plugins (zie ook www.mono-project.com) is het mogelijk om .NET-applicaties te draaien op de Apache server, maar die configuratie valt buiten de scope van deze cursus. Bovendien is het geen officieel ondersteunde omgeving dus bij problemen is het mogelijk moeilijk te achterhalen of de fout bij uw applicatiecode ligt, dan wel bij de webserver in kwestie.

Visual Studio-webserver. De recente versies van Visual Studio (2005, 2008) beschikken over een ingebouwde webserver, waardoor u geen losse webserver meer nodig hebt om uw applicatie te kunnen testen. Wij maken dan ook graag gebruik van deze zeer handige tool. Als u uw applicatie later in productie gaat nemen heeft u natuurlijk wel nood aan een professionele hosting of webserver.

De keuze van de webserver ligt eigenlijk voor de hand als u in .NET ontwikkelt. We raden iedereen dan ook aan om te werken met de ingebouwde webserver van Visual Studio.

Database server

Het laatste wat u nodig heeft om in .NET te leren ontwikkelen, behalve veel vrije tijd dan, is een databaseomgeving.

Microsoft SQL Server

De professionele databaseserver van Microsoft is SQL Server. Er is een (beperkte) gratis versie beschikbaar van deze server die overigens standaard meegeleverd wordt met Visual Studio Express Edition. Het gebruik van deze Microsoft SQL Server Express Edition is een ideale manier om kleinere tot middelgrote projecten via .NET uit te bouwen. We raden dan ook aan deze server mee te installeren op uw ontwikkelmachine.

MySQL

Voor mensen die eerder met PHP in aanraking zijn gekomen zal MySQL geen onbekende zijn. Deze databankomgeving is zowel in Windows als Linux prima bruikbaar. Binnen .NET is MySQL echter niet standaard aanspreekbaar. Via een extra component is een connectie met MySQL wel mogelijk, maar meestal wordt binnen een Microsoft omgeving geopteerd voor MS SQL Server. In deze cursus zullen we ons dan ook baseren op deze laatste databankomgeving wanneer we met databanken zullen werken.

Extra benodigdheden

Het .NET-framework is niet op alle Windowscomputers standaard aanwezig en dient dan ook los geïnstalleerd te worden. Op www.asp.net kunt u steeds de laatste versie van het framework downloaden indien nodig. Uiteraard wordt het .NET-framework automatisch geïnstalleerd wanneer u Visual Studio installeert.

Testen of .NET goed geïnstalleerd is

Om even te controleren of .NET goed geïnstalleerd staat op onze (lokale) server, open je een nieuwe website in Visual Studio en tik je volgende code in in de default.aspx pagina die reeds werd aangemaakt.

```

<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="voorbeeld_001.aspx.vb" Inherits="voorbeeld_001" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>

    <div>

        Datum en tijd op dit moment: <%
Page.Response.Write(System.DateTime.Now.ToString)%>

    </div>

</body>
</html>

```

REFERENTIE: VOORBEELD_001.ASPX

Bewaar het bestand en druk op de “play” knop (groen pijltje naar dat naar rechts wijst) bovenaan je toolbar om je project te compileren en te starten.



Als alles goed zit krijg je ongeveer het volgende te zien in je browser.

het is nu 19/07/200X 17:55:14

Sta ons toe wel op te merken dat deze manier van code schrijven tussen uw html code niet overeenstemt met de filosofie van te programmeren binnen .NET. Dit voorbeeld diende enkel om te zien dat onze .NET code correct wordt uitgevoerd.

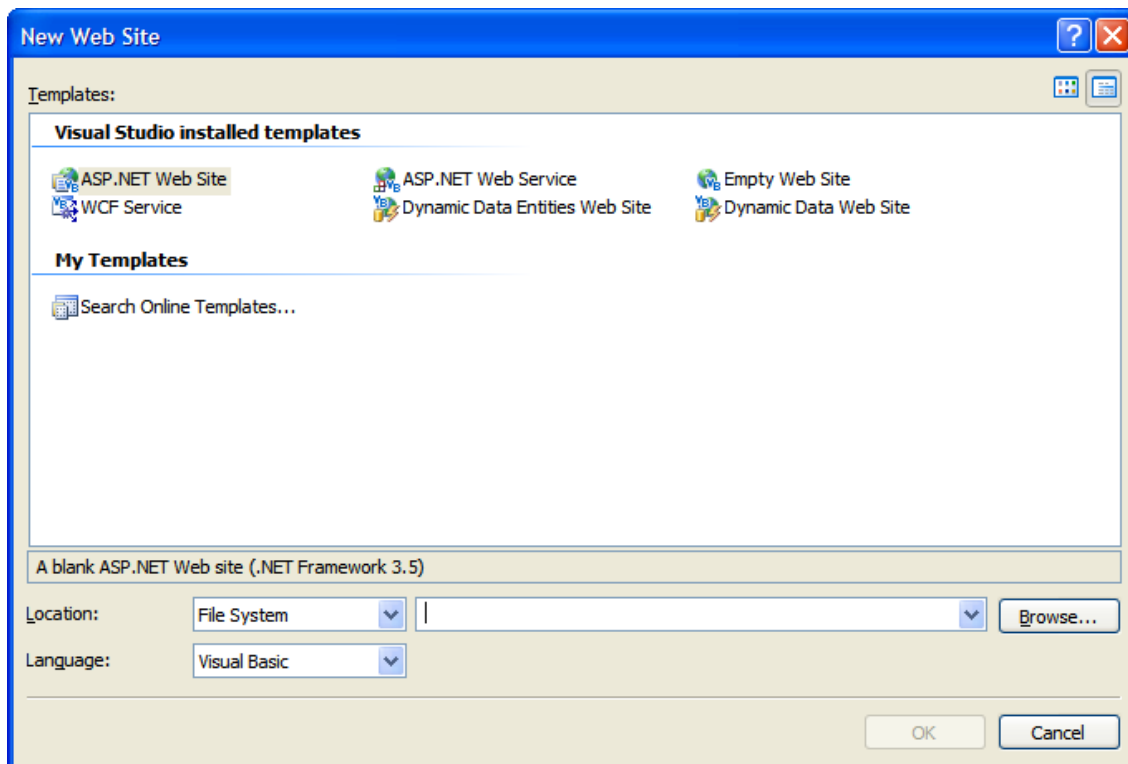
Introductie tot Visual Studio

Om op een efficiënte manier .Net applicaties te ontwikkelen maak je best gebruik van Visual Studio, de ontwikkelomgeving van het .Net Framework.

De gratis versie “Visual Studio Web Developer Express Edition” kan je downloaden van de Microsoft website. Op het moment van schrijven is versie 2008 van Visual Studio Express Edition de meest recente. Deze versie biedt ondermeer standaard ondersteuning voor AJAX-toepassingen zonder hiervoor extra add-ons te moeten installeren. We raden dan ook aan te werken met deze laatste versie van Visual Studio.

Een nieuwe webapplicatie maken

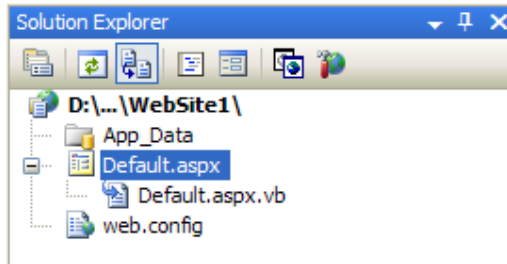
Om een nieuwe webapplicatie te maken selecteren we via het menu bovenaan File – New Website.



Je zult merken dat je onderaan dit eerste scherm een keuze kan maken tussen enkele programmeertalen waarin we onze ASP.NET applicatie zullen schrijven.

Eén van de voordelen van .NET is dat een ontwikkelaar niet meer verplicht is zijn applicatie in één taal te schrijven. De code-behind van onze applicatie kan geschreven worden in C# (lees: C Sharp), Visual Basic .NET (dot net) of Visual J#. In onze lessen zullen wij werken met VB.NET.

Kleine opmerking: het kiezen van een bepaalde taal in dit scherm wil niet zeggen dat het onmogelijk is achteraf extra pagina's in een andere programmeertaal toe te voegen aan ons project.



Nadat een nieuwe website werd aangemaakt, krijgen we in de rechterzijde van onze Visual Studio IDE (Integrated Development Environment) een schermje “Solution Explorer” te zien. Daarin komen al onze bestanden die we tijdens onze ontwikkeling zullen aanmaken terecht.

We zien op dit moment standaard bijvoorbeeld het bestand Default.aspx staan. Daarin komt onze standaard HTML-opbouw. Dat is overigens ook de pagina die meteen geopend wordt na het starten van een nieuw project.

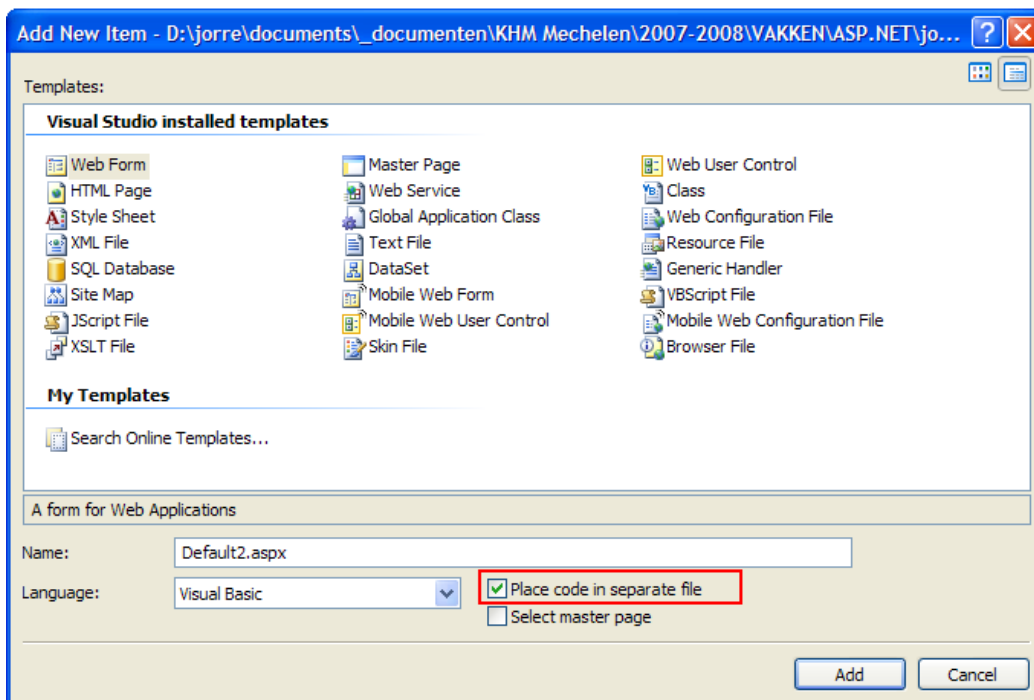
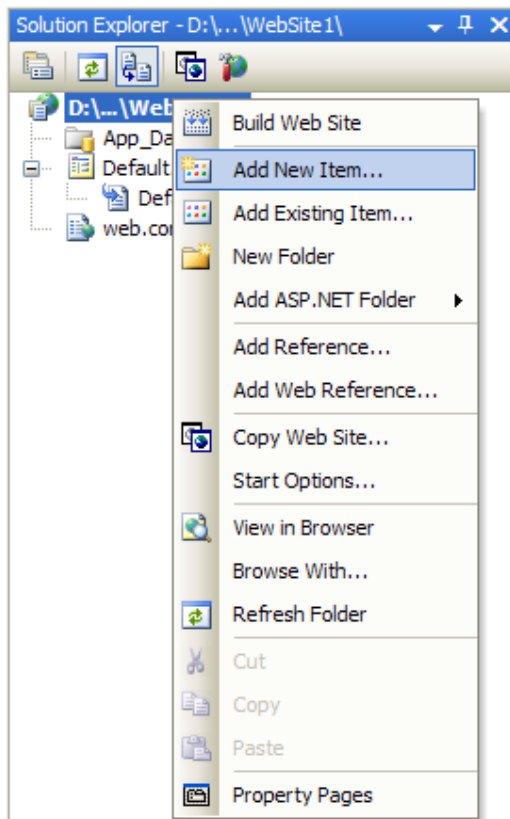
Wat nieuw zal zijn voor mensen die gewoon zijn om met PHP of ‘gewone’ ASP een website uit te werken, is dat we nu de mogelijkheid hebben om een aparte code-file aan te maken. In dit geval gaat het om een file met extensie “.vb”, vermits we gekozen hebben om in Visual Basic onze code te gaan programmeren.

Het voordeel van een aparte code-file is dat we onze programmeercode perfect kunnen gaan scheiden van onze webpagina zelf, iets wat het onderhoud van onze applicatie enorm kan vergemakkelijken.

Let wel op, de code van onze webapplicatie wordt enkel in een aparte file geplaatst indien we dit zelf vragen, namelijk door de optie “Place code in separate file” te kiezen wanneer we een nieuwe pagina aanmaken.

Voegen we een nieuwe pagina toe aan onze website door rechts te klikken op onze projectnaam binnen de Solution Explorer, dan kunnen we kiezen voor “Add New Item”, waarna we via een vinkje kunnen beslissen om onze code in een aparte file te plaatsen.

In de meeste gevallen ga je dit willen doen. Enkel wanneer je enorm weinig code moet schrijven kan het handiger zijn gewoon je code bovenaan je .aspx pagina te schrijven, maar meestal zal de code uitgebreider zijn, waardoor het makkelijker wordt die code in een aparte code-file te plaatsen. In de lessen zullen we vanaf nu steeds werken met een aparte code-file.



Opmerkelijke zielen zullen reeds gemerkt hebben dat er ook een “web.config” bestand geplaatst zal worden bij onze project file. Dit is niets meer dan een XML gebaseerde configuratiefile voor je

webapplicatie. Daarbinnen kunnen allerlei instellingen worden gewijzigd of toegevoegd om bijvoorbeeld de veiligheid van je applicatie te verhogen. Meer daarover later.

Drie plekken om code te schrijven

1. In de pagina's zelf

Je kan serverside code schrijven in je webpagina's zelf. Je moet dan wel je code markeren en onderscheiden van normale xHTML-code. Je kan dit door een script tag te gebruiken samen met het attribuut `runat="server"`

Bovenaan elke pagina zal een directive staan die aangeeft welke taal je zal gebruiken in je pagina:

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Calendar1.SelectedDate = System.DateTime.Now.Date
        txtDatum.Text = Calendar1.SelectedDate
    End Sub
</script>
```

Je zult al meteen merken dat dit een vrij slordige manier van werken is waarbij de programmeercode weer in de pagina's zelf belandt. Dit willen we zoveel mogelijk vermijden, maar we gebruiken dit voorbeeld enkel en alleen om aan te tonen dat het effectief mogelijk is uw code in de pagina's zelf te gaan schrijven.

2. Inline codeblokken

Het is zoals in ASP en PHP perfect mogelijk om je code inline te gaan schrijven, bijvoorbeeld als volgt:

```
<%@ Page Language="VB" AutoEventWireup="true" CodeFile="Default.aspx.vb"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

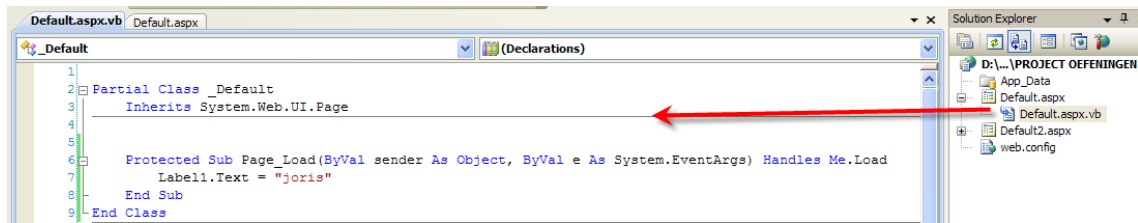
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
```

Mijn naam is <% Response.Write("joris") %>

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

3. Code-behind

Binnen .NET beschikken we over een beter manier om te programmeren. We kunnen onze programmalogica scheiden van de inhoud van de pagina door te werken met aparte codebestanden of de zogenaamde code-behind.



We kunnen zo een label plaatsen in onze pagina zelf:

```
<%@ Page Language="VB" AutoEventWireup="true" CodeFile="Default.aspx.vb"
Inherits="_Default" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
```

Mijn naam is

```
<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>

</body>
</html>
```

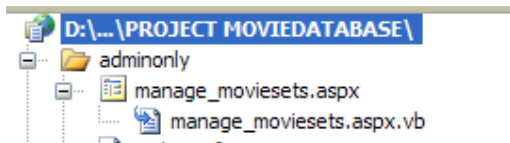
Vervolgens kunnen we dit label in de achterliggende code-behind gaan aanspreken en opvullen.

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Label1.Text = "joris"
    End Sub
End Class
```

Wat is de beste plek om uw code te schrijven?

De beste plek om uw code te groeperen in een .NET-applicatie is zonder twijfel de code-behind van uw pagina's. De code-behind is een bestand met extensie ".VB" (als u met VB.NET werkt) waarin u uw code kan gaan plaatsen voor de bijhorende pagina. In Visual Studio ziet dit bestand er als volgt uit:

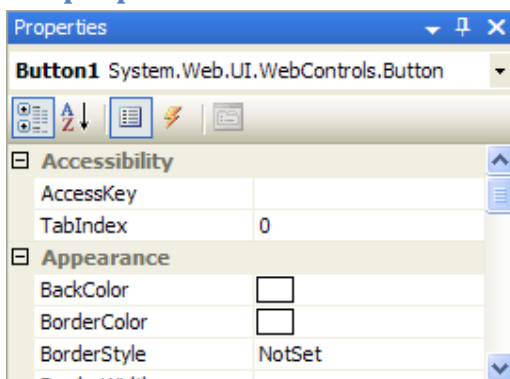


In bovenstaande afbeelding ziet u een het bestand `manage_moviesets.aspx` waarin bijvoorbeeld een formulier zit verwerkt om filmsets te verwijderen. Hieraan is een bestand gekoppeld met extensie `“.vb”`. Dit is de codefile waarin alle programmeercode verwerkt zit om het formulier in kwestie aan te sturen. Op die manier wordt het eenvoudiger om uw code te onderhouden en uit te breiden aangezien deze steeds op dezelfde plaats terug te vinden is.

Het werken met aparte code-files heeft duidelijke voordelen. We zetten de belangrijkste voordelen even op een rijtje:

- Vormgevers en programmeurs kunnen onafhankelijk van elkaar hun werk doen. De kans op fouten wordt hierdoor kleiner en de efficiëntie van een team wordt groter. Uiteraard dient uw applicatie op voorhand grondig geanalyseerd te zijn. De analyse vormt de blauwdruk van uw applicatie en is daardoor minstens even belangrijk als de code die je schrijft.
- Een tweede voordeel is dat de applicatie makkelijker onderhouden kan worden. Mensen die de lay-out bijvoorbeeld moeten aanpassen zullen de broncode niet moeten of kunnen aanpassen omdat deze in aparte bestanden is opgeslagen.
- De broncode kan makkelijker herbruikt worden, omdat deze niet “vervuild” is met HTML-code.
- De broncode kan indien gewenst gecompileerd worden, waardoor het onmogelijk wordt de geschreven broncode te gaan inkijken of te copieren. Dit zal vooral nuttig zijn wanneer intellectueel eigendom van de code belangrijk wordt.

Het properties scherm



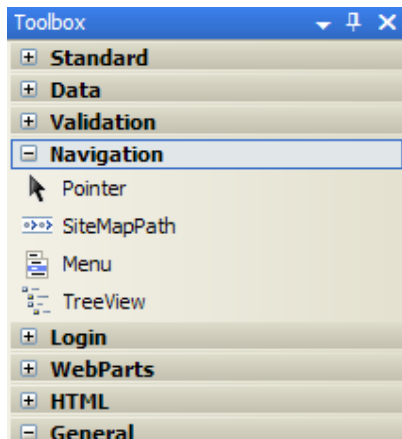
Rechts onderaan wordt binnen Visual Studio meestal het properties scherm getoond waarin heel eenvoudig de eigenschappen van elementen binnen je webapplicatie gewijzigd kunnen worden. Binnen Visual Studio trachten we liefst niet onnodig manueel onze code aan te passen, maar te werken met de properties om zo bijvoorbeeld de kans op typefouten te verkleinen.

Deze manier van werken zal heel herkenbaar zijn voor diegenen die reeds ervaring hebben met het bouwen van bijvoorbeeld Visual Basic applicaties met Visual Studio.

De toolbox

Microsoft heeft een hele set tools voorzien die het ontwikkelen van een webapplicatie aanzienlijk kunnen versnellen. Al deze standaard elementen kunnen teruggevonden worden in de “toolbox”.

Denk maar aan een calendar tool waarmee je binnen enkele seconden een calendar op je webapplicatie tovert die interageert met andere elementen van je pagina. Binnen PHP kon dit al eens langer duren.



De tools uit de Toolbox kunnen simpelweg door drag & drop op je webapplicatie gesleept en gepositioneerd worden. Er bestaan een aantal soorten controls die in onze Visual Studio als volgt worden opgedeeld:

Standard: deze controls vormen het hart van het .NET web form model

Data: hiermee kunnen connecties gelegd worden naar een database

Validation: invoer verifiëren kan in een mum van tijd met deze controls

Navigation: om bijvoorbeeld een navigatiemenu op te bouwen binnen een applicatie

Login: voorgedefinieerde veilige login mogelijkheden

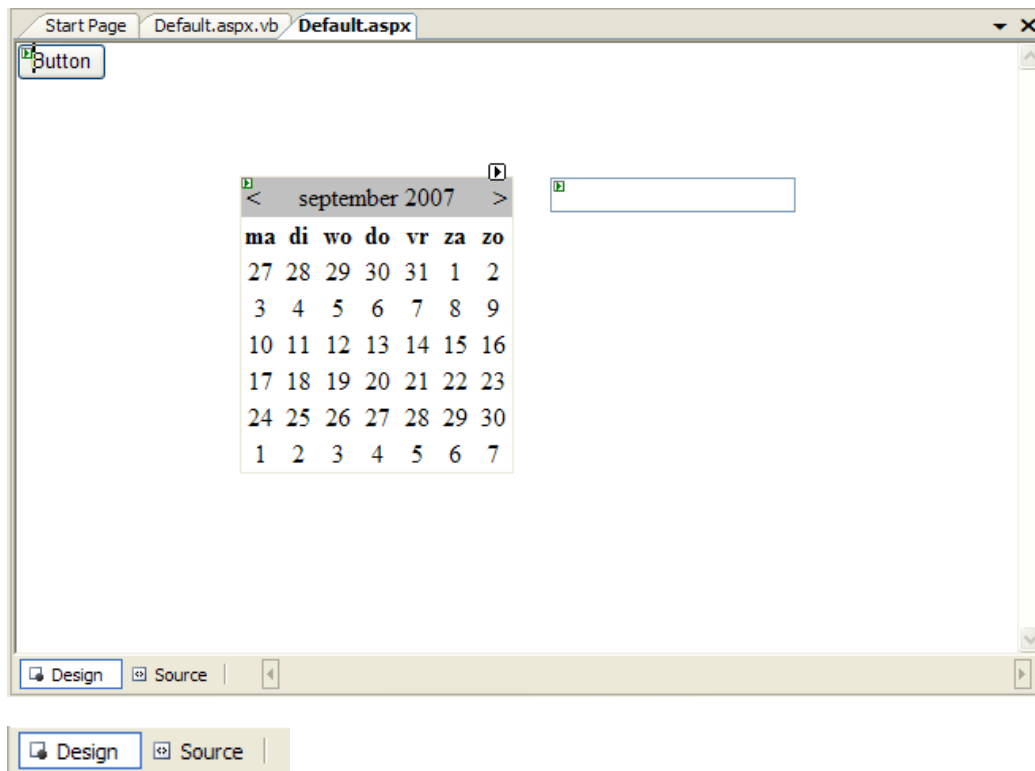
HTML: statische HTML-controls zoals we ze eigenlijk kennen uit gewone HTML. Het is beter om deze controls niet meer te gebruiken, maar gebruik te maken van de standard controls die over een

grotere functionaliteit beschikken.

Design view en source view (en split view)

Het middelste en grootste gedeelte van onze Visual Studio bestaat uit de code editor. Hier gaan we onze webapplicatie opbouwen en bewerken. Dit kan zowel in “Design view” als in “Source view”, iets wat voor onder andere Dreamweaver gebruikers niet vreemd zal zijn.

In Design view kunnen we grafisch onze webapplicatie gaan opbouwen door bijvoorbeeld extra controls vanuit onze Toolbox op de applicatie te slepen. Willen we toch in onze code gaan duiken om enkele parameters handmatig te gaan tunen, dan klikken we op source view.



Intellisense

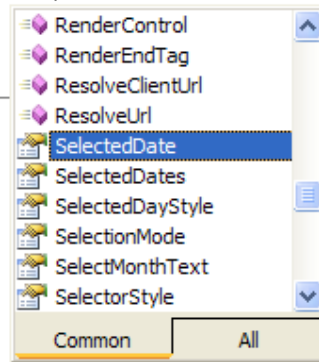
Voor we in onze Visual Basic code gaan duiken is het nuttig er even op te wijzen dat Microsoft het ons graag gemakkelijk probeert te maken. Wanneer we code schrijven zal Visual Studio ons geregeld een handje helpen door namen van variabelen, tags, enzovoort automatisch aan te vullen.

Dit versnelt niet alleen het typen, het zorgt er ook vaak voor dat typefouten minder snel gemaakt worden.

Waar we ook enorm veel gebruik van zullen maken bij het schrijven van code, is de automatische suggestie van bruikbare methoden, properties, ... die worden gesuggereerd. Dit gebeurt in regel meestal na het typen van bijvoorbeeld de naam van onze calendar tool. Zoals op onderstaande schermafdruck te zien valt, worden er automatisch een aantal mogelijkheden gesuggereerd waaruit we kunnen kiezen.

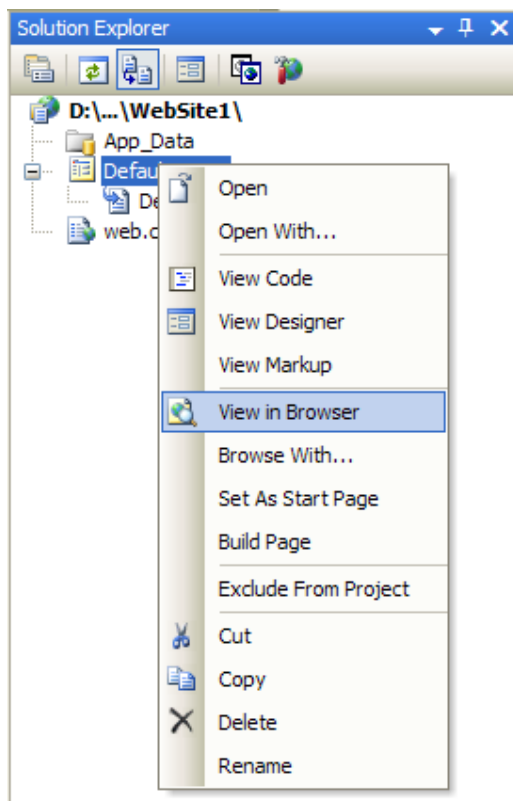
Het gemak van deze feature zal snel duidelijk worden wanneer we zelf code beginnen schrijven voor onze applicatie.


```
Protected Sub Calendar1_SelectionChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles Calendar1.SelectionChanged
    calTextBox.Text = Calendar1.SelectedDate.ToString()
End Sub
```



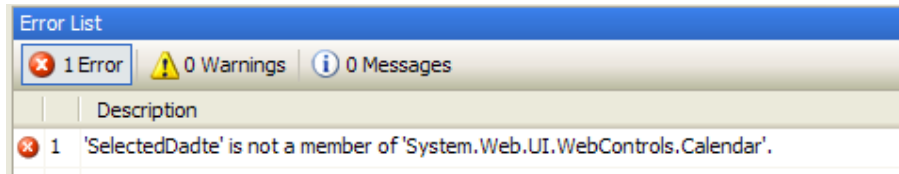
Webpagina's bekijken en debuggen

Om een webpagina uit een project te bekijken, kan je rechtsklikken op de gewenste pagina en voor "View in Browser" kiezen. De geselecteerde pagina wordt dan (als alles goed gaat) getoond in uw browser.

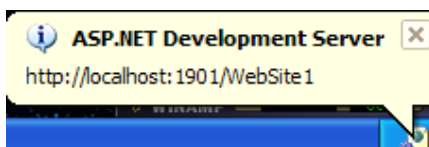
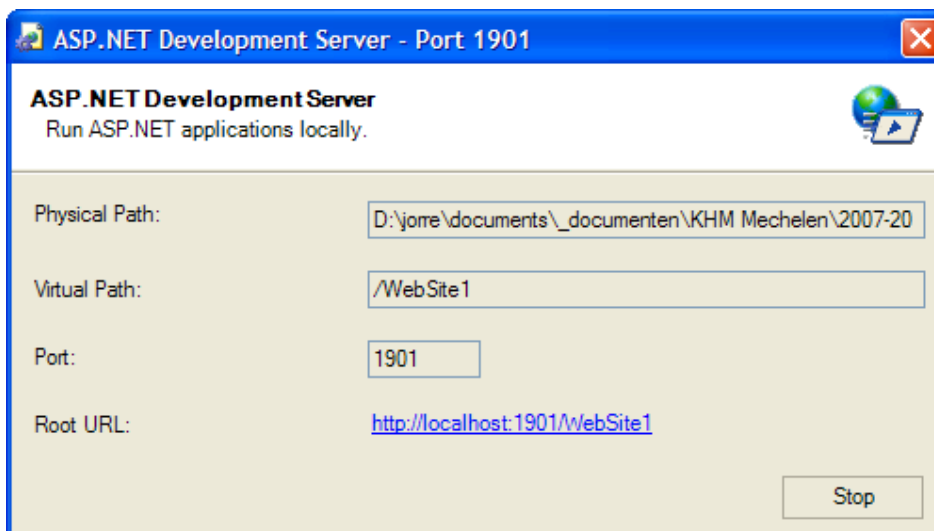


Indien er fouten in de code zijn gesloten zal er een foutmelding getoond worden en komen we automatisch in "debugging mode" terecht.

We krijgen dan automatisch een lijst met errors voorgeschoteld onderaan onze Visual Studio, waarop we kunnen gaan dubbelklikken. We komen dan automatisch terecht op de regel in onze code waar zich een fout heeft voorgedaan.



TIP: Er is geen aparte webserver installatie nodig om onze ASP.NET applicaties te gaan testen. Visual Studio wordt standaard geleverd met een eigen internet webserver die enkel en alleen gebruikt wordt voor het testen van onze webapplicaties binnen Visual Studio.



Programmeren in ASP.NET met VB.NET

Zoals eerder reeds beschreven kan er in ASP.NET met verschillende programmeertalen gewerkt worden. Het is zelfs mogelijk om één applicatie voor een stuk in VB.NET te schrijven en een ander stukje in C# (C-Sharp). In deze cursus maken wij gebruik van de programmeertaal VB.NET om een volwaardige ASP.NET applicatie te ontwikkelen.

Dit kort hoofdstuk is vooral gericht aan studenten voor wie VB.NET een nieuwe programmeertaal is of voor wie hierin nog niet veel geprogrammeerd heeft. Hoewel we er van uit gaan dat de basisbeginselen van programmeren gekend zijn, zijn er een aantal specifieke zaken waar men rekening mee moet houden wanneer men in VB.NET – en meer bepaald binnen ASP.NET – een applicatie ontwikkelt.

Voor studenten met programmeerervaring in VB.NET zal dit hoofdstuk eerder overbodig zijn.

In Visual Basic .NET wordt elke regel code als een nieuwe opdracht aanzien. Begin dus elke programmeeropdracht met een nieuwe regel. Sommige programmeertalen zoals PHP en C# vereisen aan het einde van elke opdracht een kommapunt (;) om een opdracht af te sluiten. In Visual Basic .NET is dit niet nodig. In principe zijn de verschillen tussen VB.NET en C# (de twee meest gebruikte talen binnen ASP.NET) redelijk beperkt. Zo hebben onderstaande codevoorbeelden exact hetzelfde resultaat tot gevolg:

Voorbeeld in C#

```
String Naam;  
Naam = "Bill";  
Naam = Naam + " Gates";
```

Voorbeeld in VB.NET

```
Dim Naam as String  
Naam = "Bill"  
Naam = Naam & " Gates"
```

Zoals u ziet is de opbouw van beide programmeertalen grotendeels hetzelfde.

Variabelen

In elke programmeertaal is de beschikbaarheid en het gebruik van variabelen één van de belangrijkste aspecten. Zonder variabelen zou uw applicatie niet met gegevens kunnen werken.

De standaardnotatie om variabelen te declareren in VB.NET is de volgende:

```
Dim VariabeleNaam As Type = Standaardwaarde
```

In mensentaal staat hierboven het volgende: ‘maak een variabele met de naam VariabeleNaam aan van het type Type (dit kan bijvoorbeeld een integer zijn, een string, ...) en geef deze standaard de waarde Standaardwaarde mee.’

Een belangrijk van VB.NET is dat het niet verplicht is een type mee te geven wanneer u een variabele declareert. Volgend voorbeeld maakt dit verder duidelijk:

```
Dim strNaam1 As String = "Microsoft"
```

```
Dim strNaam2 = "Microsoft"

Dim intGetall As Integer = 4
Dim inGetal2 = 4
```

In het codevoorbeeld hierboven declareren we variabele strNaam1 expliciet met het type String. Zo maken we ons programma duidelijk dat deze variabele tekst zal bevatten. Bij strNaam2 doen we dit echter niet, maar VB.NET voelt automatisch aan welk type inhoud er in de variabele wordt gestopt: in dit geval de string "Microsoft". VB.NET wordt door dit principe ook wel een losse programmeertaal genoemd. Wij raden u echter aan uw tijdens de declaratie van uw variabelen altijd een type mee te geven om latere verwarring te vermijden. Maak er een gewoonte van het type steeds bij uw declaratie mee te geven.

Code in commentaar plaatsen

Als u duidelijke code wil schrijven – en vooral dan wanneer u met meerdere mensen aan één applicatie werkt – zal u uw code willen documenteren aan de hand van commentaar. Het is mogelijk een enkel aanhalingsteken te gebruiken voor een regel die u in commentaar wil plaatsen.

```
' in onderstaande regel declareren we strNaam als type String
Dim strNaam As String = "Microsoft"
```

Via toetsencombinaties of door gebruik van commentaarknoppen kan u in Visual Studio hele blokken code in of uit commentaar zetten. Deze functionaliteit zal u in de loop der lessen automatisch leren kennen.



Procedures zonder retourwaarde

Een procedure die geen waarde teruggeeft wordt in VB.NET een subroutine genoemd. Ze worden aangeduid door het woord *Sub*. Deze subroutines kunnen wel parameters ontvangen, ondanks het feit dat ze geen waarden kunnen teruggeven.

De notatie van een subroutine in VB.NET is de volgende:

```
Sub BewaarBestand(ByVal Bestandsnaam As String, ByVal Extensie As String)
Dim VolledigeNaam As String
VolledigeNaam = Bestandsnaam & "." & Extensie
End Sub
```

De bovenstaande subroutine stelt de bestandsnaam samen aan de hand van een parameter naam en extensie. De procedure kan opgeroepen worden als volgt:

```
BewaarBestand("testje", "txt")
| BewaarBestand (Bestandsnaam As String, Extensie As String)
```

Procedures mét retourwaarden

Indien u wel een waarde wil retourneren dien u gebruik te maken van een functie. Deze worden in VB.NET aangeduid met het sleutelwoord *function*.

Een voorbeeld:

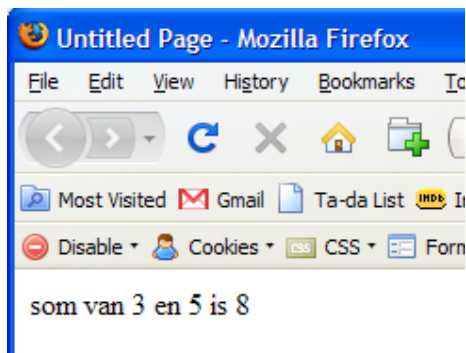
```
Function Som(ByVal getal1 As Integer, ByVal getal2 As Integer) As Integer
    Dim resultaat As Integer = getal1 + getal2
    Return (resultaat)
End Function
```

Bovenstaande functie neemt twee parameters aan (getal1 en getal2) en retourneert een waarde *resultaat*.

Deze functie kan aangeroepen worden als volgt:

```
Dim resultaat = Som(3, 5)
Page.Response.Write("som van 3 en 5 is " & resultaat)
```

Het resultaat van deze functieaanroep is:



Condities

Om uw programma's dynamisch en rekening te houden met verschillende omstandigheden kunnen we net zoals in alle programmeertalen gebruik maken van condities. De meest bekende conditie is de if-else structuur die er in VB.NET als volgt uit ziet:

```
If conditie Then
    'code uitvoeren indien conditie waar is
Else
    'code uitvoeren indien conditie niet waar is
End If
```

Bijvoorbeeld:

```
If not teller = 1 Then
    'code uitvoeren indien conditie waar is
End If
```

Merk op dat in bovenstaand voorbeeldje het woord *not* gebruikt wordt. In vele programmeertalen (zoals PHP) zou deze conditie als volgt geschreven worden: `if ($teller != 1)`. Deze schrijfwijze kan verwarrend zijn voor programmeurs die vaak met PHP werken. Het uitroepteken wordt dus niet gebruikt binnen VB.NET, maar wel het woord *not*.

Herhalingen

Vaak zal u gebruik willen maken van herhalingen of lussen om stukken code meermaals na elkaar uit te voeren. De bekendste herhalingsconstructie is de lus For ... Next. Binnen VB.NET wordt deze als volgt geschreven:

```
For Tellernaam As Type = Start To Einde Step Stapgrootte
    ' uw code om te herhalen
Next
```

Bijvoorbeeld:

```
For Teller As Integer = 2 To 8 Step 2
    ' uw code om te herhalen
Next
```

Merk op dat in de code hierboven een lus wordt uitgevoerd met een teller startend bij 2 tot 8, met een stapgrootte van 2. De teller wordt dus steeds met 2 verhoogd in plaats van 1. Meestal zal u echter een stapgrootte van 1 gebruiken. U hoeft een stapgrootte van 1 niet expliciet te programmeren. Wanneer u de optie *Step* weglaat uit uw code veronderstelt VB.NET vanzelf dat u met een stapgrootte van 1 wil werken.

Handboeken

Aangezien het Internet vol codevoorbeelden staat en er een prima forum bestaat waar u terecht kan met problemen binnen ASP.NET (<http://forums.ASP.NET/>), willen we niemand verplichten een handboek te kopen. Een handboek dat voor de ene persoon prima werkt, is voor een ander persoon vaak veel minder geschikt. Daarom laten we die keuze aan u over.

Wel raden we aan zo snel mogelijk een gratis account aan te maken op <http://forums.ASP.NET/> om eventuele vragen te stellen aan een enorm grote en behulpzame ASP.NET-community.

Eén van onze favoriete boeken voor beginnende programmeurs in ASP.NET is het volgende:

- **Het Complete Boek: ASP.NET 3.0**
ISBN: 978-90-5940-328-4 (9789059403284)



Oefeningen bij hoofdstuk 1

Codevoorbeeld: dynamisch een tabel opbouwen via ASP.NET

Bekijk onderstaand codevoorbeeld en tracht te begrijpen wat het resultaat is van de code. Merk op dat er twee aparte bestanden aangemaakt zijn, namelijk een bestand met extensie .aspx en een ander bestand met extensie .aspx.vb. Het eerste bestand bevat de structuur van uw pagina, met daarin een Placeholder (hierover komen we later op terug). Het tweede bestand bevat de programmeercode die de html-pagina zal manipuleren.

Het is niet erg wanneer u alle code nog niet helemaal begrijpt. Zo is het gebruik van een Placeholder en bepaalde syntax waarschijnlijk nieuw voor u. Tracht wel de logica in de code reeds te begrijpen.

VOORBEELD 002.ASPX:

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="voorbeeld_002.aspx.vb" Inherits="voorbeeld_002" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
<form runat="server">
    <asp:Placeholder ID="Placeholder1" runat="server"></asp:Placeholder>
</form>
</body>
</html>
```

VOORBEELD 002.ASPX.VB

```
Partial Class voorbeeld_002
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim intCols As Integer = 4
        Dim intRows As Integer = 3

        Dim tbl As New Table
        tbl.GridLines = GridLines.Both
        Placeholder1.Controls.Add(tbl)

        For i As Integer = 0 To intRows - 1 'loop over rows
            Dim tr As TableRow = New TableRow
            For j As Integer = 1 To intCols - 1 ' loop over cells
                Dim td As TableCell = New TableCell
                'Dim txtBox As TextBox = New TextBox
                'txtBox.Text = "RowNo:" & i & " " & "ColumnNo:" & " " & j
                'td.Controls.Add(txtBox)

                td.Text = "RowNo:" & i & " " & "ColumnNo:" & " " & j
            End For
            tr.Cells.Add(td)
        End For
        tbl.Rows.Add(tr)
    End Sub
```

```

        tr.Cells.Add(td)
    Next
    tbl.Rows.Add(tr)
Next
End Sub
End Class

```

[REFERENTIE: VOORBEELD_002.ASPX](#)

Oefening 1

Schrijf een ASP.NET pagina die volgende HTML-table genereert. Baseer je hiervoor op bovenstaand codevoorbeeld.

P	Pe	Pep	Pepe	Peper
M	Mo	Mol	Mole	Molen
G	Ge	Gev	Geve	Gevel
B	Bi	Bil	Bill	Billy

De woorden “Peper”, “Molen” en “Gevel” zijn opgeslagen in een ArrayList. Om de woorddelen te bepalen maken we gebruik van de left-functie. Let er ook op dat de woorden in de eerste rij van de tabel vet worden afgedrukt.

Tip: je kan CSS-eigenschappen rechtstreeks aanspreken vanuit ASP.NET code via `.Style.Add()`. Probeer dit uit om uw bovenste rij in vet te zetten.

[REFERENTIE: VOORBEELD_003.ASPX](#)

Oefening 2

Maak een nieuwe pagina aan en zorg voor volgende functionaliteit:

Sorry, u bent te jong

Een gebruiker dient via een tekstveld zijn leeftijd in te geven (vb: “25”). Daarna moet de gebruiker op een knop kunnen klikken die het volgende mogelijk moet maken:

- Als de ingegeven leeftijd groter dan of gelijk is aan 18, dient in een label de tekst “ok, u bent oud genoeg” verschijnen

Ok, u bent oud genoeg

- Als de ingegeven leeftijd kleiner is dan 18 dient de tekst “Sorry, u bent te jong” in een label te verschijnen.

Check leeftijd Sorry, u bent te jong

- Bovendien dient de tekst van uw label rood te worden indien een gebruiker jonger is dan 18. Is de gebruiker minstens 18 jaar oud, dan dient de tekst groen te worden nadat op de knop gedrukt is. Gebruik slechts één label.
- Verberg het tekstveld wanneer op de knop gedrukt is

Zorg voor een naamconventie die je in al je oefeningen zal gebruiken. Zo kan je best de naam van een tekstveld met 'txt' laten beginnen (vb: txtLeeftijd). Laat de naam van labels altijd met 'lbl' beginnen, knoppen met 'btn', enzovoort...

Welke conventie u gebruikt is niet meteen van groot belang, wél dat u EEN conventie gebruikt en blijft gebruiken doorheen uw oefeningen en project.

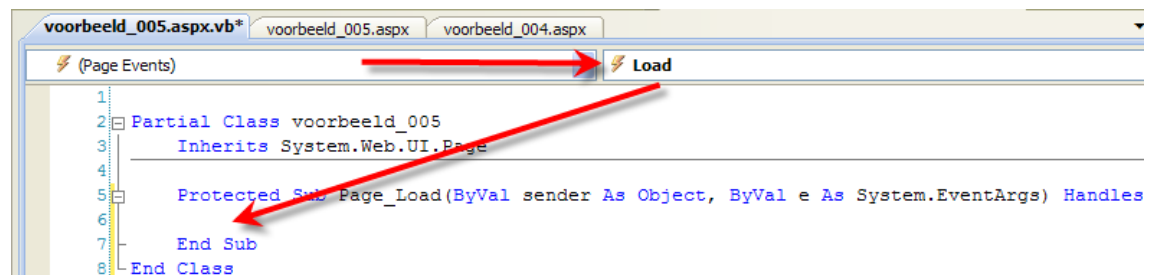
[REFERENTIE: VOORBEELD_004.ASPX](#)

Oefening 3

Schrijf in een nieuwe pagina volgende situatie uit in code:

- Definieer een variabele strWoord met de waarde "a es pee dot net"
- Zorg voor een lus die even vaak wordt uitgevoerd als dat er letters in de variabele strWoord zitten.
- Druk in uw lus telkens de teller af, behalve als de teller 3 of 7 is, dan drukt u niets af.
- Werk voor dit alles in het event "page load" van uw pagina, om uw code te laten uitvoeren wanneer uw pagina geladen werd door de browser.

Tip: via commando *response.write()* kan u tekst naar de body van uw webpagina schrijven.



Het resultaat zou er als volgt moeten uitzien:

1 2 4 5 6 8 9 10 11 12 13 14 15 16

Verandert u de waarde van strWoord in "homer", dan moet u volgend resultaat bekomen:

1 2 4 5

[REFERENTIE: VOORBEELD_005.ASPX](#)

