Hoofdstuk 18

Debuggen

Elke ontwikkelaar loopt vroeg of laat tegen een fout in zijn eigen of andermans code aan. Debuggen is dan de oplossing. Helaas is dit niet altijd even eenvoudig. Zeker in klassiek ASP, waar geen handige gereedschappen voorhanden waren, kon het debuggen van een applicatie een frustrerende bezigheid zijn. Gelukkig zijn er technieken en – in ASP.NET – hulpmiddelen voorhanden die het debuggen aanzienlijk vereenvoudigen. Dit hoofdstuk behandelt kort de beschikbare mogelijkheden. Debuggen is een vaardigheid die iedere programmeur zal moeten aanleren. Dit hoofdstuk leert u niet hoe u een expert in het debuggen wordt, maar het geeft wel de nodige voorzet om snel aan de slag te kunnen.

U leert in dit hoofdstuk:

Het ontwikkelproces wordt onder de loep genomen. Door een aantal richtlijnen te volgen kunt u solide code schrijven en zult u minder fouten in de code introduceren. Helemaal foutvrij programmeren is praktisch onmogelijk.

Vervolgens wordt het proces van fouten opsporen behandeld. Ook hier zijn bepaalde richtlijnen waar u gebruik van kunt maken om efficiënter te werken en fouten sneller op te sporen.

Tenslotte komen de hulpmiddelen die ASP.NET u biedt aan de orde. Met behulp van deze hulpmiddelen wordt de efficiëntie verder verhoogd en kunt u zich richten op het begrijpen van de code zodat u de fout eerder ontdekt. Enkele voorbeelden laten zien hoe u deze hulpmiddelen kunt inzetten.

Waar gewerkt wordt...

Zoals het spreekwoord al zegt, waar gewerkt wordt vallen spaanders. Die wijsheid geldt ook voor programmeren. En wanneer die spaanders vallen, is het van belang om de problemen zo snel en efficiënt mogelijk op te zoeken en vooral ook op te lossen. Het proces van fouten opsporen en oplossen wordt debuggen genoemd. Zelfs de beste programmeur maakt fouten, ook al werkt hij of zij nog zo nauwkeurig. Gelukkig zijn er methoden beschikbaar om deze fouten op te sporen. Het is echter zeer de moeite waard om eerst te proberen om fouten te voorkomen.

Fouten voorkomen

Ook hier is een spreekwoord van toepassing: voorkomen is beter dan genezen. Bij het ontwikkelen van (web) applicaties is dat zeker waar aangezien het proces van genezen vaak moeizaam en daardoor zeer kostbaar is. Door een aantal eenvoudige richtlijnen aan te houden, kunt u de kans op fouten aanzienlijk verminderen. Uiteraard kunt u nooit helemaal voorkomen dat er fouten gemaakt worden, maar minder fouten is altijd beter.

Een goed ontwerp

Voordat u een (web) applicatie daadwerkelijk gaat implementeren dient u altijd eerst een degelijk technisch ontwerp te (laten) maken. In een technisch ontwerp worden de belangrijkste punten van de applicatie behandeld. De structuur van de database komt aan bod. Daarnaast wordt de interne structuur van de applicatie behandeld. Alle eventuele communicatie met externe systemen moet ook grondig onderzocht en beschreven zijn. Tenslotte zou een technisch ontwerp ook aandacht moeten besteden aan het testen van de te ontwikkelen code.

Het grote voordeel van het maken van een technisch ontwerp, is dat u gedwongen wordt om over de structuur van de applicatie na te denken voordat u een letter geprogrammeerd heeft. Het is aanzienlijk minder werk om tijdens het maken van een technisch ontwerp de structuur van een applicatie aan te passen dan wanneer u reeds begonnen bent met ontwikkelen.

Laat indien mogelijk ook een of meerdere collegae het technisch ontwerp lezen. Een frisse blik op een technisch ontwerp levert meestal verrassend zinnige informatie op. Vooral wanneer de beschreven techniek zeer complex is, loont het zeker de moeite een collega om zijn of haar mening te vragen.

Structuur

In het technisch ontwerp wordt al een voorzet voor een goede structuur gegeven. Een van de belangrijkste methoden om fouten te voorkomen is het aanbrengen van een eenduidige structuur in uw code. Het toepassen van structuur begint bij de naamgeving van diverse onderdelen. Zowel bestanden als klassen, functies, subroutines en tenslotte variabelen dienen een heldere maar vooral consequente naamgeving te hebben. Consequentie is in dit geval belangrijker dan de daadwerkelijke naamgeving.

Een ander aspect van gestructureerd programmeren is de manier waarop u problemen aanpakt. Probeer zoveel gelijksoortige problemen op gelijksoortige manieren aan te pakken. Door zoveel mogelijk dezelfde technieken toe te passen, ziet uw code er beter uit, en is uw code gemakkelijker te lezen. Probeer u eens in te denken hoe uw eigen code over drie of zes maanden te lezen is. Consequentie is zeer belangrijk voor de leesbaarheid van uw code. U hoeft zich immers niet telkens opnieuw af te vragen wat een stukje code doet wanneer u diezelfde techniek al meerdere malen bent tegengekomen.

Documentatie

In het ideale geval wordt het technische ontwerp tijdens het implementeren van de beschreven applicatie bijgewerkt met actuele informatie. In de praktijk gebeurt dat niet altijd. Een goede hulp bij het achteraf bijwerken van het technisch ontwerp is het plaatsen van commentaar in de code. Daarnaast kan commentaar in de code erg nuttig zijn bij het opsporen en oplossen van fouten. Door in commentaar te beschrijven wat een functie of subroutine zou moeten doen, kan een programmeur achteraf eenvoudiger achterhalen hoe de code in elkaar zit. De meeste programmeurs kunnen aan een stukje code wel zien wat het doet, maar de vraag is vaak wat het zou moeten doen. Hoe heeft de programmeur die code tijdens het ontwikkelen ervan bedacht. Die informatie kan van onschatbare waarde zijn tijdens het opsporen en oplossen van fouten.

Commentaar schrijft u niet alleen voor uw collegae. Ook zelf kunt u er voordeel van hebben. Zelfs al enkele dagen later. Neemt u eens de proef op de som en zoekt u eens code op die u zelf, zes maanden of langer geleden, geschreven heeft. U zult er versteld van staan hoe moeilijk het soms kan zijn uw eigen creatie te begrijpen.

Paranoïde

Gebruikers zijn briljant wanneer het erop aankomt om (web) applicaties op de knieën te krijgen. Door een gebrek aan technisch inzicht zien de meeste gebruikers (web) applicaties anders dan programmeurs dat doen. Daardoor gaan gebruikers vaak anders met (web) applicaties om dan programmeurs verwachten. Wat voor een programmeur volledig logisch en verklaarbaar is, kan voor een gebruiker onbegrijpelijk zijn. Houd daar rekening mee. Zowel bij het maken van het

technisch ontwerp als bij het implementeren van de applicatie.

Een gezonde dosis paranoïde kan bij het programmeren van een applicatie geen kwaad. Controleer bijvoorbeeld altijd de invoer van gebruikers. U kunt er niet vanuit gaan dat gebruikers altijd netjes voldoen aan de technische voorwaarden voor gegevensinvoer. Het kan voor een gebruiker volledig logisch zijn om daar waar u om een telefoonnummer vraagt het volgende in te voeren:

```
020-6154234 (alleen tussen 18.00 en 22.30)
```

De bovenstaande waarde past waarschijnlijk niet het databaseveld dat u voor het telefoonnummer gereserveerd heeft, met een lelijke foutmelding als gevolg. U dient de lengte van de invoer altijd te beperken en te controleren zodat u niet voor verrassingen komt te staan.

Controle

Het controleren van uw eigen code is zeer belangrijk, maar tevens erg moeilijk. De code die u zelf geschreven heeft kunt u vaak zelf niet volledig controleren. Vraag – indien mogelijk – regelmatig de hulp van een collega, zodat hij of zij uw code kan bekijken. U kunt hetzelfde voor die collega doen. Uw beider code wordt daar beter van en waarschijnlijk kunt u nog iets van elkaar leren ook.

Een van de belangrijkste regels van het testen, is dat u kleine stukjes code tegelijk test. Iedere functie die u schrijft moet getest worden. Het liefst moet iedere functie afzonderlijk getest worden. Op die manier neemt u factoren weg bij het opsporen van fouten. Wanneer u drie functies schrijft, en die pas test wanneer ze gezamenlijk toegepast worden, kan elk van de drie functies een fout bevatten. Wanneer u de functies afzonderlijk test voordat u ze toepast, is de kans groot dat er ook tijdens het gezamenlijk toepassen van de functies geen fouten optreden. Mocht er

dan toch een fout opduiken, dan is deze meestal gemakkelijker te vinden.

Naast het testen van uw eigen werk, is het ook verstandig een onafhankelijk persoon – iemand die niet bij uw project betrokken is en als zodanig onbevooroordeeld is – uw applicatie te laten testen. Probeer zowel de logische als de onlogische invoer uit. Daarnaast is het verstandig om ook onverwachte routes binnen uw applicatie uit te proberen. Wat gebeurt er bijvoorbeeld als een gebruiker een resultaatpagina vastlegt in zijn of haar favorieten en die pagina later nogmaals opvraagt?

Fouten opsporen

Ondanks alle moeite die u zult doen om fouten te voorkomen is de kans altijd aanwezig dat er toch een of meerdere fouten in uw applicatie sluipen. Als dat het geval is, zult u op zoek moeten naar de oorzaak van de fout.

Zekerheid

Een van de belangrijkste punten bij het opsporen van fouten is zekerheid. Voordat u daadwerkelijk op zoek gaat naar een fout, moet u zeker weten waar u naar kijkt. Regelmatig brengt een fout – of de gebruiker die de fout gemeld heeft – u op een dwaalspoor. Zo kan het voorkomen dat een gebruiker bijvoorbeeld op een andere locatie kijkt dan u denkt. Stel u heeft een testomgeving en een productieomgeving. Het kan gebeuren dat een gebruiker op de testomgeving kijkt, terwijl u verwacht dat hij of zij op de productieomgeving kijkt. In dat geval staat u een lange zoektocht te wachten wanneer u de foutmelding in behandeling neemt. Daarnaast spelen er bij de verwerking van gegevens vaak meerdere aspecten een rol. Zo kan het zijn dat een verzonden bestand niet goed aankomt. In dat geval kan het probleem op verschillende plaatsen zitten. Wordt het bestand überhaupt goed aangemaakt voordat het verzon-

den wordt? Wordt het bestand goed verzonden maar raakt het na ontvangst corrupt? Dit soort vragen dient u zichzelf te stellen voordat u code gaat openbreken. Het kan u een hoop tijd en frustratie schelen.

Structuur

Net als bij het schrijven van code geldt ook bij het opsporen en oplossen van fouten dat een gestructureerde werkwijze vruchten af zal werpen. Zoals uit de vorige paragraaf al blijkt dient u altijd eerst het probleem te analyseren voordat u daadwerkelijk gaat opsporen. Pas wanneer u weet wat het probleem is, kunt u op zoek gaan naar de oorzaak. Pas wanneer u de oorzaak achterhaald heeft, kunt u op zoek gaan naar de oplossing.

Wanneer u een lijst met foutmeldingen krijgt, is het van groot belang om een foutmelding tegelijk te verwerken. Het is altijd verstandig om eerst de lijst eens door te nemen zodat u misschien overeenkomsten of juist vreemde afwijkingen tegenkomt, maar bij het daadwerkelijk opsporen van fouten verwerkt u een fout tegelijk. Pas nadat u een fout hebt opgelost en de oplossing ook getest is, kunt u verder gaan met de volgende fout. Wanneer alle fouten uit de lijst verwerkt zijn, test u alle oplossingen nogmaals. Het kan immers voorkomen dat de oplossing voor de ene fout de oorzaak van een andere fout is.

Wanneer u de fout aan het opsporen of aan het oplossen bent, brengt u een wijziging tegelijk aan. Alleen dan kunt u snel en efficiënt werken. Brengt u meer dan een wijziging tegelijk aan, dan kunt u nooit met volledige zekerheid weten welke wijziging effect gehad heeft. Houd ook bij welke wijzigingen u aanbrengt. Mocht u de wijzigingen weer ongedaan moeten maken, dan is het handig als u weet welke wijzigingen u heeft aangebracht. Als u onverwacht in uw werk gestoord wordt, kunnen aantekeningen over aangebrachte wijzigingen en hun effect u hel-

pen snel weer op gang te komen wanneer u uw werk weer kunt hervatten.

Consequentie

Bij het opsporen en oplossen van fouten is consequentie een belangrijke factor. Wanneer u op zoek bent naar een fout gebruikt u waarschijnlijk testgegevens om een foutmelding te reproduceren. Als dat het geval is, dient u consequent te zijn in de toepassing van testgegevens. Controleer uw aanpassingen ook met dezelfde testgegevens. Pas als blijkt dat uw wijzigingen de fout verholpen hebben kunt u andere testgegevens gebruiken om te zien of de fout ook in andere gevallen de kop opsteekt.

Houdt bij het aanbrengen van wijzigingen rekening met de geldende regels voor onder andere naamgeving. Het kan zijn dat het project waarin u een wijziging aanbrengt andere regels voor naamgeving aanhoudt dan recentere projecten waar u aan gewerkt heeft.

Documentatie

Wanneer u wijzigingen in code aanbrengt, of de structuur van de database aanpast om een fout op te lossen, dient u deze wijzigingen te documenteren. Wijzigingen in code dienen altijd met behulp van commentaar vermeld te worden bij de betreffende code. In het ideale geval worden alle wijzigingen ook in de technische documentatie verwerkt.

Door de wijzigingen bij te houden, voorkomt u dat uzelf of uw collegae voor verrassingen komen te staan bij een volgende keer dat er wijzigingen in de code moeten worden aangebracht.

Soorten fouten

Bij het opsporen van fouten is het zeer nuttig een onderscheid te maken tussen de verschillende soorten fouten die kunnen voorkomen.

Grofweg zijn alle fouten in drie categorieën onder te brengen.

- **Syntaxisfouten** Deze categorie fouten zal het eerst opvallen. Bij syntaxisfouten is er een probleem met de spelling of grammatica van de broncode. De compiler kan daardoor de broncode niet begrijpen en geeft een foutmelding. Deze fouten zijn over het algemeen het eenvoudigst op te lossen.
- **Semantische fouten** Fouten in deze categorie komen tijdens het uitvoeren van de applicatie aan het licht. Het gaat hier om fouten waar zowel de spelling als de grammatica van de broncode correct zijn, echter de combinatie van objecten niet toegestaan of mogelijk is. Zo kan het bijvoorbeeld zijn dat u gebruik maakt van een DataSet object. Wanneer u dit object niet initialiseert voordat u het gebruikt, zal er syntactisch niets aan de hand zijn. Wanneer u de code wilt uit voeren krijgt u echter een foutmelding omdat u een object wilt gebruiken dat niet geïnitialiseerd is. Deze categorie fouten is meestal relatief gemakkelijk te vinden en op te lossen.
- **Logische fouten** De logische fouten zijn verreweg het moeilijkst om op te sporen en op te lossen. Het gaat hier om fouten in de logica van de code. De betreffende code is zowel syntactisch als semantisch in orde. De combinatie van objecten is toegestaan en mogelijk. Toch gaat er iets fout. Uw code levert een ongewenst resultaat op. Over het algemeen zal de applicatie niet gestopt worden wanneer er een logische fout optreedt. Dat is een van de redenen waardoor het erg moeilijk kan zijn deze fouten op te sporen. Het reproduceren van de fout kan erg lastig zijn. De oorzaak van logische fouten ligt meestal niet op de plaats waar deze tot uiting komt.

Gereedschappen

In klassiek ASP was het opsporen van fouten een zeer ongemakkelijke taak. De foutmeldingen waren zeer onduidelijk en als programmeur moest u zelf code ontwikkelen om de status van bijvoorbeeld het HTTP Request object of de huidige sessie variabelen te kunnen zien. Al met al was het zeer omslachtig om efficiënt op zoek te gaan naar fouten in de code. ASP.NET heeft daar gelukkig een aantal oplossingen voor. In de komende paragrafen worden deze mogelijkheden besproken.

Code volgen

Een aanzienlijke verbetering in ASP.NET ten opzichte van klassiek ASP is de mogelijkheid om de uitvoer van code te volgen (in het Engels: *Tracing*). Wanneer u van deze mogelijkheid gebruik maakt, krijgt u onderaan de pagina een aanzienlijke hoeveelheid informatie over de uitvoer van de pagina. Bovendien kunt u uw eigen informatie aan deze volginformatie toevoegen.

U kunt zowel code per pagina als per applicatie volgen. De volgende paragrafen laten dit zien aan de hand van enkele voorbeelden.

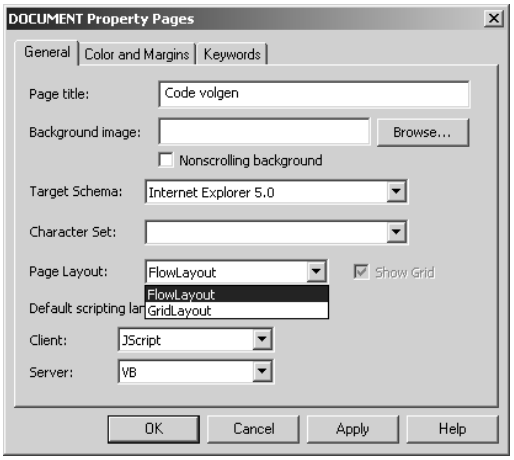
Code volgen per pagina

In het onderstaande voorbeeld ziet u hoe u uw pagina zo kunt configureren dat de code gevolgd wordt. Hiervoor wordt het *Trace* attribuut toegevoegd aan de *@ Page* opdracht.

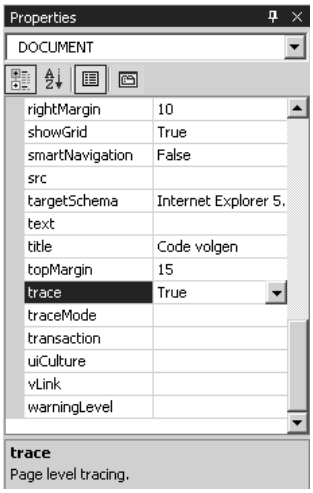
1. Open Visual Studio.NET en maak een nieuwe webapplicatie aan. Noem deze Hoofdstuk18.
2. Hernoem de standaard WebForm1.aspx pagina naar CodeVolgen.aspx.
3. Rechtsklik in de pagina – in de ontwerpweergave – en kies de optie **Properties**.

FlowLayout

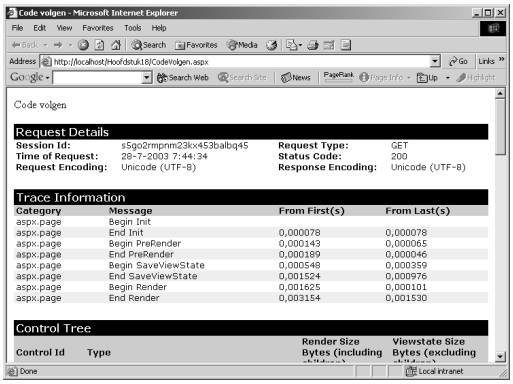
Wanneer uw pagina in GridLayout modus staat – dit is de standaard modus in Visual Studio.NET – kan de uitvoer van code volgen over de elementen van uw pagina heen vallen. Het is uiteraard niet altijd mogelijk, maar in het ideale geval staat uw pagina in FlowLayout wanneer u code wilt volgen.



Afbeelding 18.1 *De dialoogvenster DOCUMENT Property Pages.*



Afbeelding 18.2 *De eigenschap Trace van het huidige document.*



Afbeelding 18.3 *De pagina CodeVolgen.aspx met daarin een overvloed aan informatie.*

Het dialoogvenster **DOCUMENT Property Pages** verschijnt. Selecteer de waarde **FlowLayout** in het **PageLayout** uitklapmenu. Tevens kunt u de titel van de pagina veranderen in `Code volgen`.

4. Sleep een Label control naar de pagina en verander de tekst in `Code volgen`.
5. Klik in een leeg gedeelte van de pagina zodat u de eigenschappen van het document kunt aanpassen. Selecteer de waarde **True** bij de eigenschap **Trace**.
6. Sla de pagina op en bekijk deze in een browser. U kunt de optie **Build en Browse** uit het contextmenu van de CodeVolgen.aspx pagina gebruiken.

Gevoelig

De uitvoer van code volgen kan gevoelige informatie aan het licht brengen. Bovendien heeft het volgen van code gevolgen voor de prestaties van de webapplicatie. Daarom is het aan te raden alleen code te volgen op ontwikkelmachines, tenzij het echt niet anders kan.

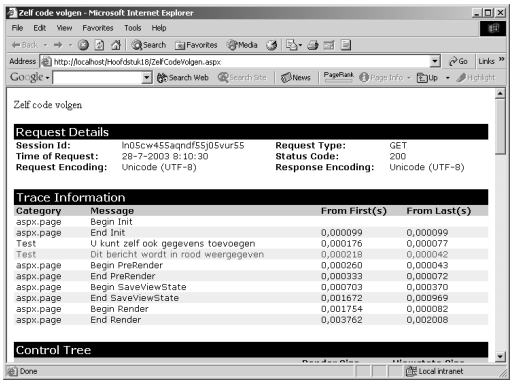
Zelf gegevens volgen

Naast de standaard beschikbare informatie, kunt u ook uw eigen gegevens toevoegen aan de uitvoer van *code volgen*. In klassiek ASP – evenals in andere ontwikkelomgevingen – is een veel gebruikte methode van fouten opsporen om markeringen in de code en uitvoer van bepaalde functies naar de webpagina te schrijven, zodat u het verloop van de code kunt volgen. In ASP.NET kunt u deze methode nog steeds toepassen, het kan immers een zeer succesvolle methode zijn. In dat geval voegt u de uitvoer toe aan de resultaten van code volgen. Op die manier blijft uw pagina overzichtelijk, en kunt u de resultaten ook verbergen voor bezoekers.

1. Dit voorbeeld borduurt voort op het voorgaande voorbeeld. Open indien nodig het project Hoofdstuk18 in Visual Studio.NET.
2. Voeg een nieuwe pagina aan het project toe. Noem deze pagina `ZelfCodeVolgen.aspx`.
3. Rechtsklik in de pagina – in de ontwerpweergave – en kies de optie **Properties**. Het dialoogvenster **DOCUMENT Property Pages** verschijnt. Selecteer de waarde **FlowLayout** in het **PageLayout** uitklapmenu. Tevens kunt u de titel van de pagina veranderen in `Zelf Code volgen`.
4. Sleep een Label control naar de pagina en verander de tekst in `Zelf code volgen`.
5. Klik in een leeg gedeelte van de pagina zodat u de eigenschappen van het document kunt aanpassen. Selecteer de waarde **True** bij de eigenschap **Trace**.
6. Schakel over naar de codeweergave om zelf informatie toe te voegen aan de uitvoer van code volgen.
7. Om te voorkomen dat u gegevens tevergeefs toevoegt, controleert u eerst of code volgen wel aan staat. Vervolgens voegt u twee berichten toe. Met behulp van de *Write* methode kunt u een gewoon bericht toevoegen, wanneer u gebruik maakt van de *Warn* methode wordt het betreffende bericht in rood weergegeven. Plaats de onderstaande code in de Page_Load subroutine.

```
If Trace.IsEnabled Then
Trace.Write("Test", "U kunt zelf ook
    gegevens toevoegen")
Trace.Warn("Test", "Dit bericht wordt
    in rood weergegeven")
End If
```

8. Sla de pagina op en bekijk deze in een browser. U kunt de optie **Build en Browse** uit het contextmenu van de ZelfCodeVolgen.aspx pagina gebruiken.



Afbeelding 18.4 *De pagina ZelfCodeVolgen.aspx met daarin uw eigen twee berichten.*

De methodes *Write* en *Warn* van het *Trace* object kennen drie varianten. U kunt alleen een bericht als parameter meegeven, of een categorie en een bericht – zoals u in het bovenstaande voorbeeld gedaan heeft – of een categorie, een bericht en uitzonderingsinformatie. De eerste twee parameters zijn altijd in de vorm van een string. De uitzonderingsinformatie moet in de vorm van een *Exception* object worden aangeleverd.

```
Trace.Write(bericht)
Trace.Write(categorie, bericht)
Trace.Write(categorie, bericht,
            uitzonderingsinformatie)
```

Code volgen per applicatie

In plaats van per pagina, kunt u ook code volgen per applicatie. Dit heeft een aantal voordelen ten opzichte van code volgen per pagina. Ten eerste wordt de uitvoer niet aan de gebruiker getoond. Daardoor kunt u code volgen per applicatie wel toepassen op productieservers. Daarnaast kunt u de uitvoer van een gehele website volgen. Dit is vooral gunstig indien u niet zeker weet op welke pagina er een fout optreedt. Tenslotte kunt u de uitvoer van de betreffende pagina's op een later tijdstip bekijken.

Om deze mogelijkheid aan te zetten, moeten er enkele wijzigingen in het web.config bestand worden doorgevoerd.

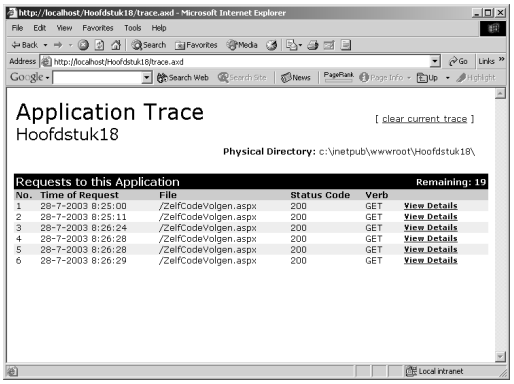
1. Open nogmaals het Hoofdstuk18 project in Visual Studio.NET.
2. Open het web.config bestand.
3. Zoek de <trace>-tag op.
4. Pas de attributen van de <*trace*>-tag als aan. Om het volgen per applicatie in te schakelen stelt u de waarde van de eigenschap **enabled** op *True*. U kunt het aantal gevolgde pagina's instellen met behulp van de eigenschap **requestLimit**. Door de eigenschap **pageOutput** de waarde *False*

te geven, voorkomt u dat de uitvoer zichtbaar is op de pagina zelf.

```
<trace enabled="true"
  requestLimit="25"
  pageOutput="false"
  traceMode="SortByTime"
  localOnly="true" />
```

5 Sla de wijzigingen op.

Om de opgeslagen uitvoer te bekijken, kunt u een speciale pagina met de naam trace.axd opvragen. Wanneer u deze pagina in de hoofddirectory van uw webapplicatie opvraagt, krijgt u een overzicht van alle gevolgde pagina's. U kunt in dat overzicht ook de resultaten verwijderen. Dat dient u regelmatig te doen aangezien ASP.NET geen resultaten meer opslaat zodra het maximum bereikt is. De resultaten worden dus niet automatisch verwijderd.



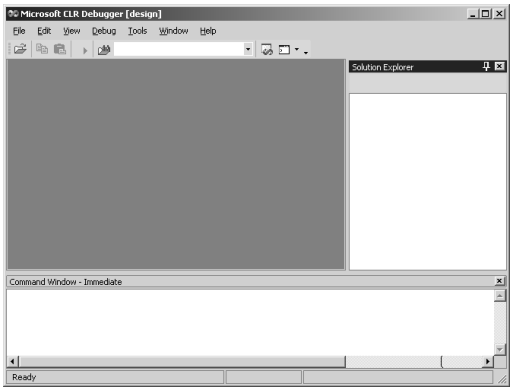
Afbeelding 18.5 *De trace.axd pagina toont een overzicht van gevolgde code.*

Pagina

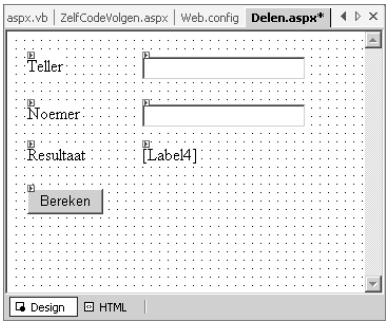
Het *Trace* attribuut in de *@ Page* opdracht overschrijft de waarde van het web.config bestand. Wanneer u *Trace=True* in uw *@ Page* opdracht heeft staan, zal de uitvoer altijd in de pagina getoond worden, ook als u expliciet heeft opgegeven dat u per applicatie wilt volgen en dat u geen uitvoer in de pagina wilt. In dat geval kunt u het beste alle *Trace* attributen uit de *@ Page* opdrachten verwijderen.

Debugger

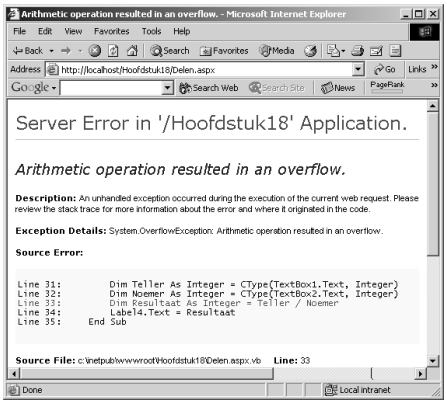
Het laatste gereedschap is de zogenaamde debugger. Een debugger is een stuk gereedschap dat u in staat stelt om regel voor regel door een programma – of in dit geval een webpagina – te lopen. Per regel kunt u bijvoorbeeld zien wat de waarden van de verschillende variabelen zijn. Dit stuk gereedschap was niet beschikbaar voor klassiek ASP en is van onschatbare waarde bij het opspo-



Afbeelding 18.6 *De standaard debugger van het .NET Framework.*



Afbeelding 18.7 *De ontwerpweergave van de delen.aspx pagina.*



Afbeelding 18.8 *U krijgt een foutmelding wanneer u door nul wilt delen.*

ren en oplossen van lastige fouten in ASP.NET.

Bij het .NET Framework wordt een volledige debugger geleverd. Deze heet DbgCLR en is te vinden in *C:\Program Files\Microsoft.NET\FrameworkSDK\GuiDebug* of *C:\Program Files\Microsoft Visual Studio .NET\FrameworkSDK\GuiDebug*, afhankelijk van uw installatie kan deze debugger ook op een andere drive staan.

In de dagelijkse praktijk is de debugger in Visual Studio.NET echter een stuk eenvoudiger in gebruik. Het onderstaande voorbeeld toont deze debugger in de praktijk.

Het klassieke voorbeeld

1. Het is in dit boek al eerder ter sprake gekomen, het klassieke voorbeeld van een fout is delen door nul. In het onderstaande voorbeeld maakt u een pagina waarop u twee getallen kunt invullen. Deze worden op elkaar gedeeld en het resultaat wordt getoond. Nu kan het echter voorkomen dat een gebruiker een verkeerde waarde invult, zodat er een 'delen door nul' fout ontstaat.

2. Open indien nodig het project Hoofdstuk18 in Visual Studio.NET.

3. Voeg een nieuwe pagina aan het project toe. Noem deze pagina `Delen.aspx`.

4. Sleep vier Label controls, twee TextBox controls en een Button control naar de pagina en maak deze op zoals in afbeelding 18.7 te zien is.

5. Dubbelklik op het Button control om de Button1_Click subroutine aan te maken en naar de codeweergave over te schakelen.

6. Plaats de code om de berekening uit te voeren en het resultaat te tonen in deze subroutine:

```
Dim Teller As Integer =
 CType(TextBox1.Text, Integer)
Dim Noemer As Integer =
 CType(TextBox2.Text, Integer)
Dim Resultaat As Integer = Teller / Noemer
Label4.Text = Resultaat
```

7 De testpagina is hiermee klaar. Sla deze op en bekijk deze in een browser door de optie **Build and Browse** te kiezen in het contextmenu van deze pagina.

Startpagina

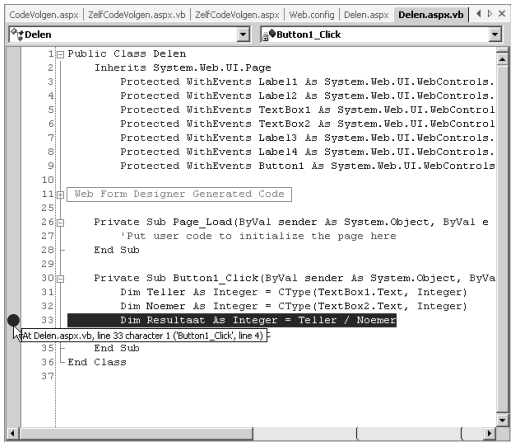
Wanneer u de optie Build and Browse gebruikt, wordt de startpagina van het project uitgevoerd. Indien de infrastructuur van uw pagina's dat toestaat, kunt u via de startpagina naar de pagina die u wilt debuggen. Als dat niet kan, zult u de betreffende pagina als startpagina moeten instellen. Dit kunt u doen met het context menu van de betreffende pagina.

8 Wanneer u geldige getallen invoert, zult u een normaal resultaat krijgen. Voert u echter nul in als noemer, dan krijgt u de bekende foutmelding..

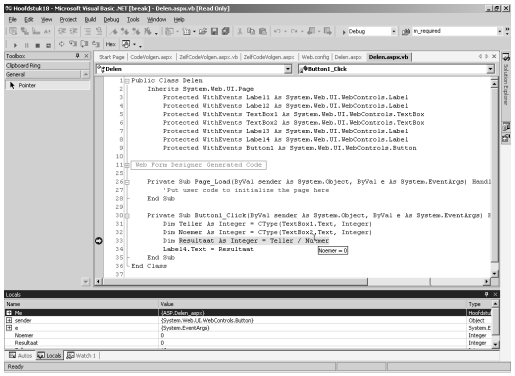
9 Sluit de pagina af en keer terug naar Visual Studio.NET. Door nu in de linker kantlijn naast een regel code te klikken, kunt u de debugger opdracht geven om op die regel te stoppen voordat de regel is uitgevoerd. Klik naast regel 33 om een breekpunt te zetten.

10 Wanneer u nu dezelfde pagina nogmaals uitvoert en u voert een ongeldige noemer in, krijgt u niet de foutmelding te zien, maar wordt de geselecteerde regel getoond in de debugger. Door uw muis over een variabele te houden, krijgt u de waarde van die variabele te zien.

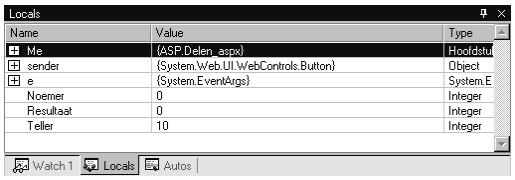
11 Er zijn verschillende manieren om te achterhalen wat de waarde van een variabele is op het moment dat de code onderbro-



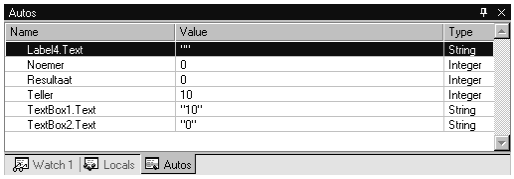
Afbeelding 18.9 *Een breekpunt op regel 33.*



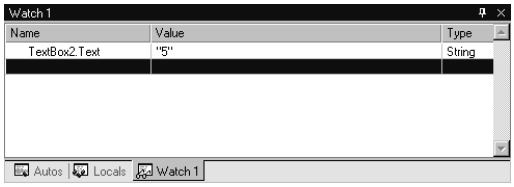
Afbeelding 18.10 *De geselecteerde regel in de debugger.*



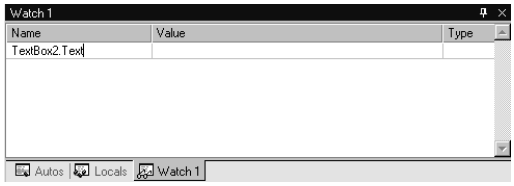
Afbeelding 18.11 *Het Locals paneel.*



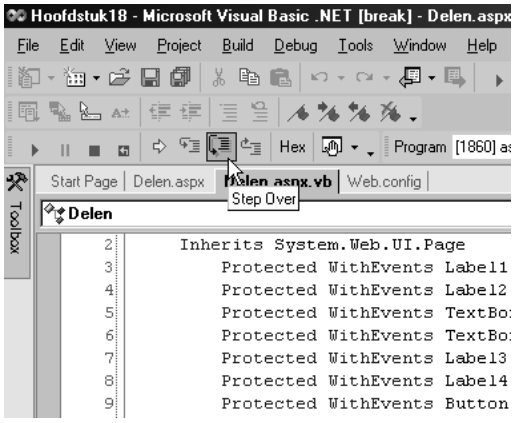
Afbeelding 18.12 *Het Autos paneel.*



Afbeelding 18.13 *Geef de naam van een variabele op in het Watch paneel.*



Afbeelding 18.14 *De opgegeven variabele wordt getoond in het Watch paneel.*



Afbeelding 18.15 *Voer de huidige regel code uit.*

ken wordt. Zo kunt u het *Locals* paneel gebruiken. In het *Locals* paneel staan alle variabelen die het huidige code blok gebruikt worden.

Het *Autos* paneel laat alle variabelen zien die in het huidige code blok gedeclareerd zijn.

Tenslotte kunt u ook zelf bepalen welke variabelen u wilt zien. Door in het *Watch* paneel de naam van een variabele in te voeren wordt de waarde van die variabele getoond.

12 Om de huidige regel code uit te voeren kunt u gebruik maken van de iconen in de knoppenbalk zoals getoond in afbeelding 18.13. Als alternatief kunt u de F10 toets gebruiken om de huidige regel code uit te voeren.

Met behulp van de debugger kunt u de verwerking van uw code stap voor stap volgen terwijl u de waarden van de verschillende variabelen in de gaten houdt. Op deze manier kunt u zeer efficiënt op zoek gaan naar – al dan niet hardnekkige – fouten. Dankzij deze functionaliteit wordt het ontwikkelen van robuuste webapplicaties met ASP.NET aanzienlijk vereenvoudigd. Uiteraard zult u zelf de (on)logica achter eventuele fouten moeten ontdekken, maar de beschikbare gereedschappen zullen u helpen beter te begrijpen wat er daadwerkelijk in de code gebeurt – in tegenstelling tot wat er zou moeten gebeuren.

Appendix C

Web Matrix installeren

U leert in dit hoofdstuk:

Om te beginnen leert u hoe u kunt controleren of het .NET Framework op uw machine geïnstalleerd is. Indien die niet het geval is, kunt u het installatiepakket downloaden en installeren.

Vervolgens zult u Web Matrix downloaden en installeren. Daarna kunt u in principe beginnen met het ontwikkelen van webapplicaties.

Om krachtigere webapplicaties te kunnen ontwikkelen, zult u een database server nodig hebben. De MSDE is daarvoor prima geschikt en gratis te downloaden. U leert hoe u de MSDE kunt installeren.

Tenslotte wordt kort het installeren van de voorbeelddatabases behandeld.

Een van de gereedschappen waarmee u ASP.NET pagina's kunt ontwikkelen is Web Matrix. Het is gratis te downloaden van de website http://www.asp.net. Naast Web Matrix kunt u De MSDE van Microsoft gebruiken, een gratis database server. Om het werken met de voorbeelden in dit boek eenvoudiger te maken, worden de gebruikte voorbeelddatabases ter download aangeboden. Deze databases kunt u in de MSDE installeren.

Low budget

Er zijn diverse gereedschappen beschikbaar waarmee u ASP.NET webapplicaties kunt ontwikkelen. Deze gereedschappen zijn er in verschillende vormen en vooral ook in verschillende prijsklassen. Het gereedschap Web Matrix is geheel gratis en stelt u in staat om bijna alle mogelijkheden van ASP.NET te gebruiken. In deze appendix wordt het downloaden en installeren van Web Matrix behandeld.

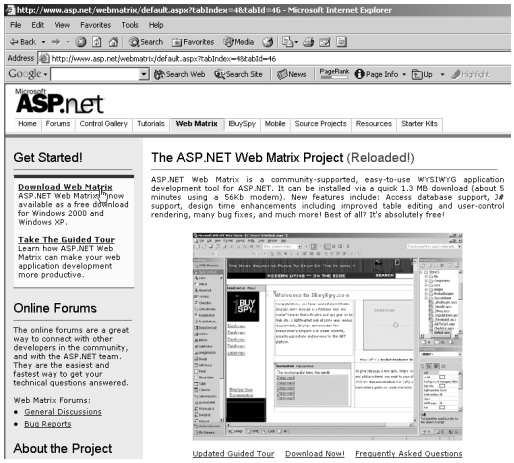
Bij de meeste webapplicaties zult u gebruik willen maken van een database. Gelukkig heeft Microsoft de SQL Server Desktop Engine ontwikkeld. Deze lichte versie van SQL server wordt gratis door Microsoft beschikbaar gesteld bij Web Matrix. Ook de installatie van dit pakket wordt behandeld.

Web Matrix

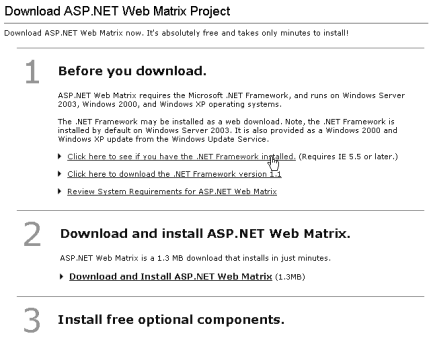
Web Matrix is een gratis ontwikkelomgeving die geheel in .NET ontwikkeld is. In deze omgeving kunt u gebruik maken van de meeste eigenschappen van ASP.NET.

1. Open een browser en surf naar http://www.asp.net. Op de startpagina van deze website wordt direct melding gemaakt van de ontwikkelomgeving Web Matrix. In de hoofdnavigatie staat ook een verwijzing naar de specifieke pagina's voor Web Matrix.
2. Kies in de hoofdnavigatie voor Web Matrix zodat u op de specifieke pagina voor Web Matrix terecht komt. Op deze pagina staat aan de linkerkant een tekstblok met daarin een hyperlink naar de downloadpagina.
3. Klik op de hyperlink naar de downloadpagina. U krijgt nu een pagina te zien met daarop een stappenplan.

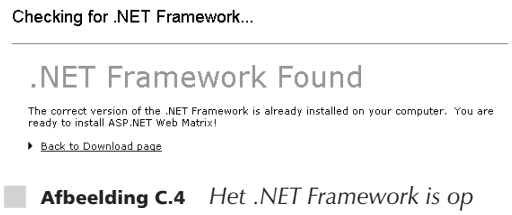
Afbeelding C.1 *De startpagina van www.asp.net.*



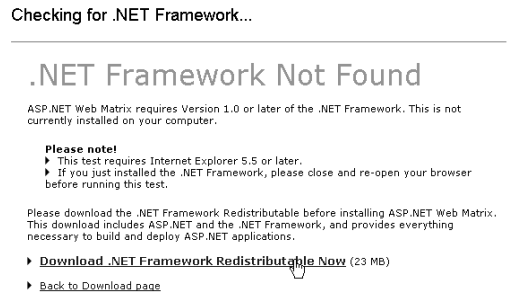
Afbeelding C.2 *De specifieke pagina voor Web Matrix.*



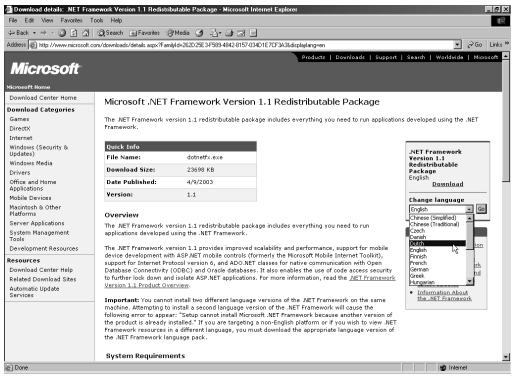
Afbeelding C.3 *De downloadpagina voor Web Matrix*



Afbeelding C.4 *Het .NET Framework is op uw machine geïnstalleerd.*



Afbeelding C.5 *Het .NET Framework is niet op uw machine geïnstalleerd.*



Afbeelding C.6 *U kunt de taal selecteren waarin u het Framework wilt downloaden.*

Voorbereidingen

Voordat u Web Matrix kunt installeren dient uw machine aan een aantal voorwaarden te voldoen. Het kan u veel tijd en moeite besparen om vooraf deze voorwaarden te controleren.

1. De eerste stap op de downloadpagina van Web Matrix wordt gevormd door enkele voorbereidingen. Klik op de bovenste link, u wordt doorverwezen naar een pagina waar gecontroleerd wordt of u het .NET Framework reeds op uw machine geïnstalleerd heeft.
2. Indien uw machine reeds voorzien is van het .NET Framework hoeft u verder geen actie te ondernemen, u kunt doorgaan met het downloaden en installeren van Web Matrix. Is dat echter niet het geval, dan dient u het .NET Framework te downloaden en te installeren.
3. Wanneer het .NET Framework niet op uw machine is aangetroffen, dient u het te downloaden en vervolgens te installeren. In afbeelding 5 ziet u dat er een link aanwezig is naar de downloadpagina voor de huidige versie van het .NET Framework. Klik op deze link.
4. Op de download pagina van het .NET Framework staat aan de rechterkant van de pagina een tekstblok waarin u de taal van het Framework kunt selecteren. Het Framework is ook beschikbaar in het Nederlands.

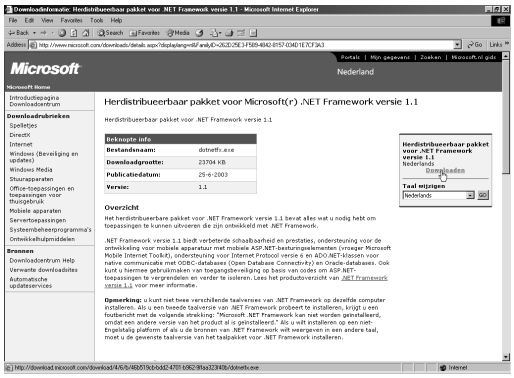
Versies

De huidige versie van het .NET Framework kan hoger zijn dan in deze appendix vermeld staat, tegen de tijd dat u dit leest. Tijdens het schrijven van dit boek is de versie al verhoogd van 1.0 naar 1.1. Wanneer u deze appendix leest, zou de versie inmiddels nog een of meerdere malen verhoogd kunnen zijn.

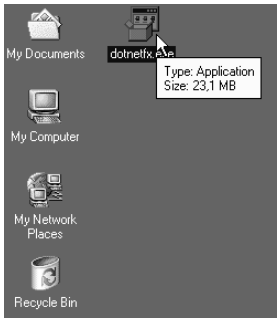
5 Nadat u de taal van uw keuze geselecteerd heeft – in dit geval Nederlands – kunt u het Framework downloaden door op de link **Downloaden** te klikken. De browser zal u vragen of het bestand opgeslagen moet worden. Op deze vraag antwoordt u dat het bestand inderdaad opgeslagen moet worden. Vervolgens zal u gevraagd worden waar het bestand opgeslagen moet worden. Veel mensen vinden het gemakkelijk om dit soort bestanden op hun bureaublad op te slaan. Dat is dan ook wat er in dit voorbeeld gedaan wordt.

6 Wanneer u het installatieprogramma voor het .NET Framework eenmaal op uw computer heeft staan, kunt u het Framework installeren. Dubbelklik op het icoon voor het Framework, op uw bureaublad.

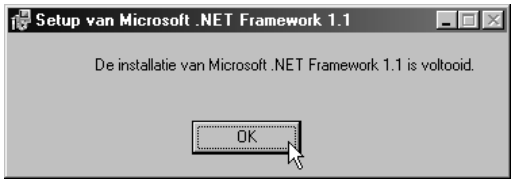
7 Tijdens het installeren van het .NET Framework worden u twee vragen gesteld. Ten eerste de vraag of u het Framework wilt installeren. Daarna krijgt u de vraag of u akkoord gaat het de voorwaarden voor installatie. Indien u op beide vragen een bevestigend antwoord geeft, zal het Framework geïnstalleerd worden. Tenslotte krijgt u de melding dat het Framework op uw machine geïnstalleerd is. U kunt daarna doorgaan met het downloaden en installeren van Web Matrix.



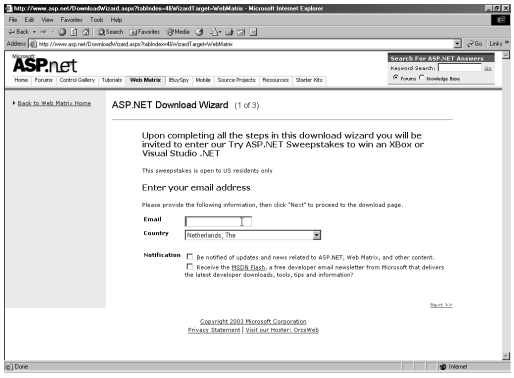
Afbeelding C.7 *U kunt het .NET Framework downloaden nadat u de taal geselecteerd heeft.*



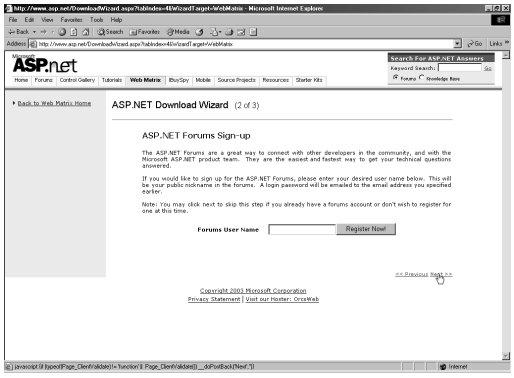
Afbeelding C.8 *Het installatieprogramma voor het .NET Framework op uw bureaublad.*



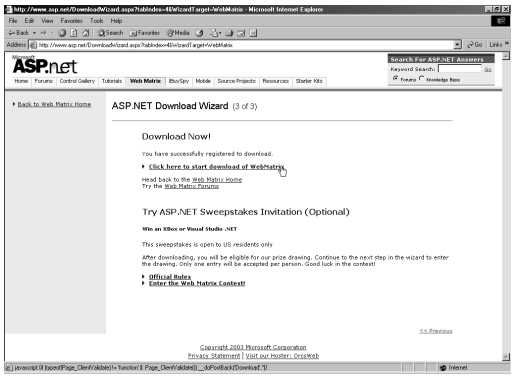
Afbeelding C.9 *Het .NET Framework is geïnstalleerd.*



Afbeelding C.10 *De eerste stap van het downloaden van Web Matrix.*



Afbeelding C.11 *De tweede stap van het downloaden van Web Matrix.*



Afbeelding C.12 *De derde en laatste stap van het downloaden van Web Matrix.*

Downloaden

Wanneer het .NET Framework eenmaal op uw machine geïnstalleerd is, kunt u Web Matrix downloaden en installeren.

1. Kies de optie **Download and Install ASP.NET Web Matrix** op de download pagina voor Web Matrix.
2. Voordat u Web Matrix mag downloaden dient u eerst een e-mail adres en het land waarin u woont op te geven. Met behulp van de optie **Next** (rechtsonder in de pagina) kunt u naar de volgende stap gaan.
3. In de tweede stap kunt u zich opgeven voor de Forums op de website . Dit is echter niet verplicht. Indien u geen interesse heeft in de forums, of wanneer u zich later wilt wilt opgeven, kunt u ook op deze pagina met de **Next** optie naar de volgende stap doorgaan.
4. Op de derde pagina kunt u tenslotte Web Matrix daadwerkelijk downloaden. Klik op de link met de tekst **Click here to start download of WebMatrix** om het installatieprogramma naar uw computer te halen.
5. Wederom zal de browser u vragen of het installatieprogramma opgeslagen moet worden, en zo ja waar dan. Sla het bestand wederom op uw bureaublad op.

Installeren

De laatste stap voordat u Web Matrix kunt gebruiken, is het installeren van Web Matrix.

1. Nu u het installatieprogramma voor Web Matrix op uw computer heeft staan, kunt u Web Matrix installeren. Dubbelklik op het icoon voor Web Matrix, op uw bureaublad.
2. Tijdens het installeren van Web Matrix wordt u een groot aantal schermen getoond, zeven om precies te zijn. Het eer-

ste scherm vertelt u dat Web Matrix geïnstalleerd zal worden. Klik hier op de knop **Next** om door te gaan met de installatie.

3 Vervolgens krijgt u de bekende licentieovereenkomst van Microsoft te zien. Indien u Web Matrix wilt installeren, zult u akkoord moeten gaan met deze overeenkomst. Met de knop **Next** gaat u door naar het volgende scherm.

4 In dit derde scherm wordt u om uw naam en de naam van uw organisatie gevraagd. Tevens kunt u aangeven of Web Matrix alleen voor uzelf of voor alle gebruikers van de computer wordt geïnstalleerd. De knop Next brengt u weer naar het volgende scherm.

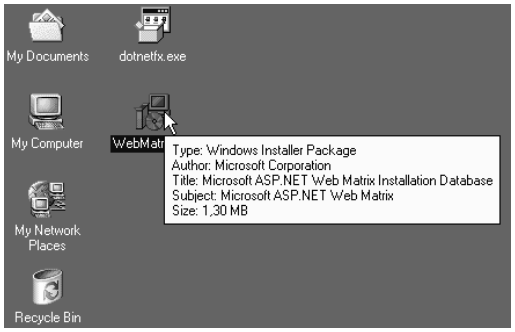
5 Het vierde scherm biedt u in theorie de mogelijkheid om te selecteren welke onderdelen er wel en welke niet geïnstalleerd moeten worden. In de prakrijk heeft u die keuze niet. Web Matrix bestaat uit een enkel onderdeel dat derhalve altijd geïnstalleerd moet worden. U kunt wel aangeven waar het pakket geïnstalleerd moet worden. Klik op de knop **Next** om door te gaan naar het volgende scherm.

6 Het vijfde scherm is weinig zinvol. U kunt hier de gemaakte keuzes bekijken voordat u de installatie start. Helaas valt er op dit scherm niets te zien. U kunt wederom de knop **Next** gebruiken om een scherm verder te gaan.

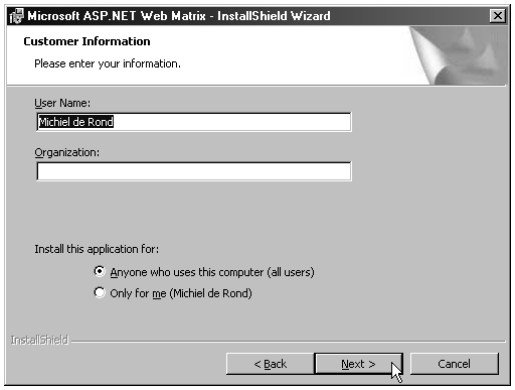
7 Nadat u vijf schermen heeft doorgewerkt, wordt Web Matrix nu uiteindelijk op uw systeem geïnstalleerd. Wanneer de installatie klaar is, wordt de knop **Next** beschikbaar, en kunt u die gebruiken om naar het zevende en laatste scherm te gaan.

8 Het laatste scherm geeft u de melding dat de installatie voltooid is. U kunt op de knop **Finish** klikken om het installatieprogramma af te sluiten.

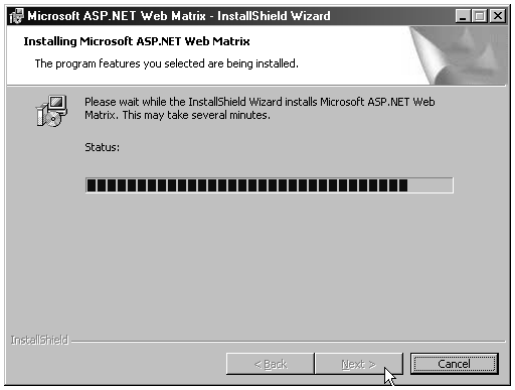
U kunt nu gebruik maken van Web Matrix.



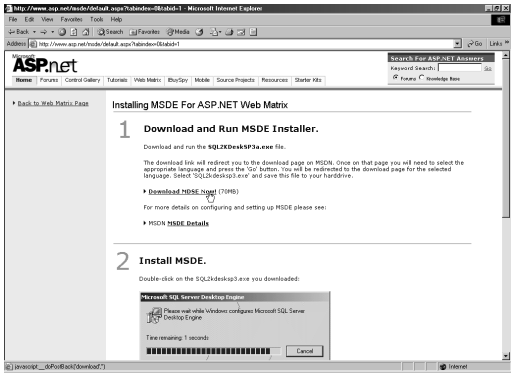
Afbeelding C.13 *Het installatieprogramma voor Web Matrix op uw bureaublad.*



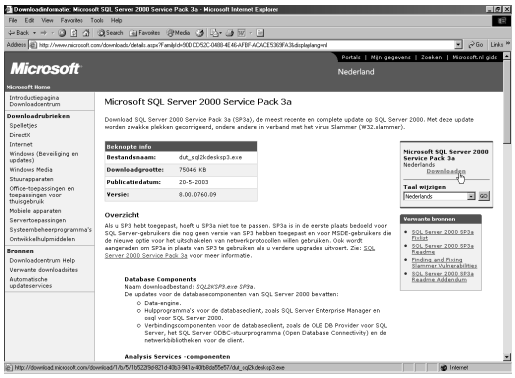
Afbeelding C.14 *U kunt uw naam en de naam van uw organisatie opgeven.*



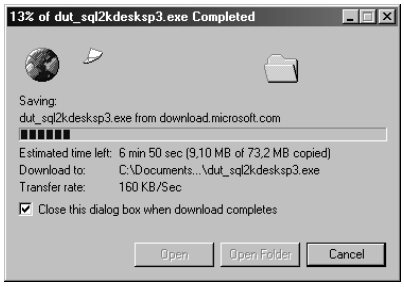
Afbeelding C.15 *Web Matrix wordt geïnstalleerd.*



Afbeelding C.16 *De downloadpagina van de MSDE.*



Afbeelding C.17 *De Nederlandstalige downloadpagina.*



Afbeelding C.18 *Het downloaden van het grote installatiepakket van de MSDE.*

Microsoft SQL Server Desktop Engine

De Microsoft SQL Server Desktop Engine is een eenvoudige versie van Microsoft SQL Server 2000. Deze eenvoudige versie is gratis verkrijgbaar op de website van Microsoft.

MSDE

De Microsoft SQL Server Desktop Engine wordt meestal afgekort tot MSDE. Zo'n afkorting is immers veel bruikbaarder dan de lange volledige naam.

1. Nadat u het .NET Framework en Web Matrix geïnstalleerd heeft, kunt u op de downloadpagina van Web Matrix de link volgen naar de download pagina van de MSDE. Klik op de link met de tekst **Download MSDE**.
2. Net als bij het .NET Framework, kunt u ook bij de MSDE de gewenste taal kiezen. Selecteer in het tekstvak rechts bovenaan de pagina de Nederlandse taal en druk op de knop Go. Op de volgende, Nederlandstalige, pagina kunt u de link Downloaden gebruiken om de MSDE naar uw computer te halen.
3. Het MSDE installatiepakket is redelijk groot, het kan dus even duren voordat het geheel binnen is. Afhankelijk van uw verbinding kan het downloaden tussen enkele minuten en enkele uren duren.
4. Nadat u het bestand heeft gedownload, en het zoals gewoonlijk op uw bureaublad heeft opgeslagen, kunt u de MSDE gaan installeren. Door te dubbelklikken op het installatieprogramma, wordt dit geopend.
5. Het installatiepakket zal de MSDE niet daadwerkelijk installeren. In plaats daarvan zal het alle benodigde bestanden uitpakken en ergens op uw computer plaatsen. U kunt zelf aangeven waar de

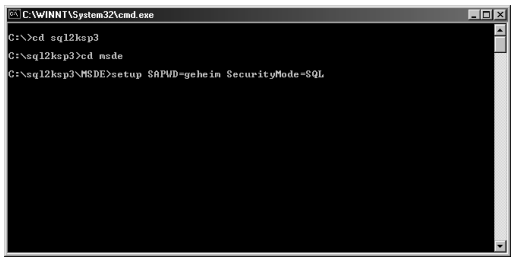
bestanden geplaatst worden. Vanaf die locatie kunt u dadelijk de MSDE installeren. Indien de map die u aangeeft niet bestaat, wordt deze – na uw goedkeuring – aangemaakt.

6 Om de MSDE daadwerkelijk te installeren, opent u een commandovenster door in het start menu de optie **Uitvoeren** of **Run** te kiezen. In het dialoogvenster dat verschijnt voert u `cmd` in en drukt u vervolgens op de Enter-toets.

7 In het commandovenster verandert u eerst de huidige map met behulp van het CD commando. Daarna voert u het setup commando uit met daarachter twee parameters. De eerste parameter geeft het gewenste wachtwoord voor het sa account weer, de tweede parameter geeft aan dat u SQL Server beveiliging wilt gebruiken, in plaats van Windows beveiliging. In afbeelding 20 wordt de opdrachtregel getoond.

8 Nadat de installatie voltooid is, dient de machine opnieuw opgestart te worden.

Afbeelding C.19 *U kunt zelf bepalen waar de installatiebestanden geplaatst moeten worden.*



Afbeelding C.20 *Het installeren van de MSDE via een opdrachtregel.*

Voorbeelddatabases

In dit boek wordt een drietal voorbeelddatabases gebruikt. Evenals alle broncode voor dit boek, zijn ook die databases te downloaden vanaf http://www.de-rond.nl/boeken. De databases worden aangeboden in de vorm van SQL scripts. Deze scripts kunt u eenvoudig installeren via een opdrachtregel.

1 Om een voorbeelddatabase te installeren, opent u een commandovenster door in het start menu de optie **Uitvoeren** of **Run** te kiezen. In het dialoogvenster dat verschijnt voert u `cmd` in en drukt u vervolgens op de Enter-toets.

2 In het commandovenster verandert u eerst de huidige map met behulp van het CD commando naar de map waar u het SQL script heeft opgeslagen. In dit voorbeeld is

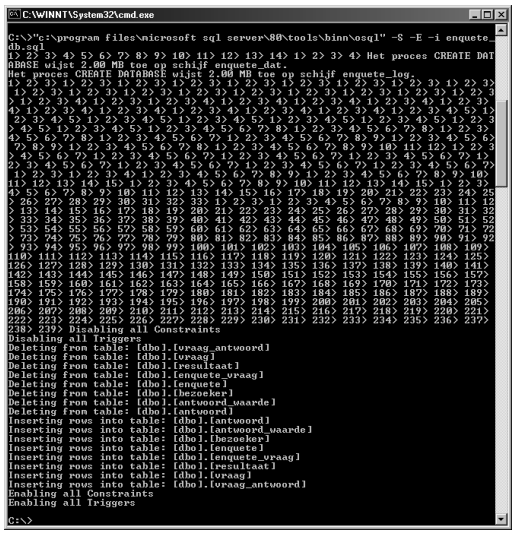
dat de hoofdmap van de C: drive. In dat geval hoeft u de map niet te veranderen.

3 Op de commandoregel voert u de opdracht osql uit, met een aantal parameters. De eerste parameter geeft aan welke server gebruikt wordt. Doordat er aan deze parameter geen extra waarde wordt meegegeven, wordt de lokale server gebruikt. De tweede parameter geeft aan dat er een vertrouwde verbinding gebruikt wordt. De derde en laatste parameter geeft aan dat er een invoerbestand gebruikt wordt.

```
osql –S –E –i enquete_db.sql
```

Het osql commando vindt u in de Binn submap van de Tools map van de MSDE:

```
C:\Program Files\Microsoft SQL Server\80\
Tools\Binn
```



Afbeelding C.21 *Het installeren van de voorbeelddatabase via een opdrachtregel.*

Appendix D

Opdrachten

U leert in dit hoofdstuk:

U leert welke opdrachten er beschikbaar zijn in ASP.NET. Per opdracht wordt er een korte beschrijving van het gebruik gegeven.

Van iedere opdracht worden de beschikbare attributen behandeld. Per attribuut worden de mogelijke waarden en eventuele uitzonderingen behandeld.

ASP.NET kent een aantal opdrachten welke u bovenaan de pagina kunt plaatsen. Met enkele opdrachten heeft u in de praktijk reeds kennis gemaakt. Er zijn er echter meer. Deze appendix somt de beschikbare opdrachten op en behandelt kort de attributen bij iedere opdracht..

@ Page

Deze opdracht definieert paginaspecifieke eigenschappen die door de ASP.NET compiler en paginaverwerker gebruikt worden. Deze opdracht kan alleen in Web Forms pagina's worden toegepast en per .aspx bestand kan er slechts een @ *Page* opdracht voorkomen. De attributen worden door spaties gescheiden terwijl de attribuut met de bijbehorende waarde juist niet door spaties gescheiden mogen worden; er wordt altijd de vorm `attribuut="waarde"` gebruikt.

Het onderstaande voorbeeld geeft ASP.NET de opdracht om Visual Basic als taal te gebruiken en de resulterende pagina te verzenden met het HTTP MIME type `text/xml`.

```
<%@ Page Language="VB" ContentType="text/xml"
%>
```

De onderstaande paragrafen behandelen de attributen bij deze opdracht.

AspCompat

Wanneer dit attribuut de waarde *True* heeft, wordt de pagina zodanig uitgevoerd, dat oudere componenten, zoals die gemaakt in Visual Basic 6.0, vanuit de pagina aangeroepen kunnen worden. Ook wordt het dan mogelijk om COM+ 1.0 componenten die onbeheerde ASP objecten aanspreken te gebruiken.

Prestaties

Let op, door dit attribuut de waarde *True* te geven, kunnen de prestaties van uw ASP.NET pagina verminderen.

AutoEventWireup

Dit attribuut geeft aan of de gebeurtenissen op deze pagina automatisch gekoppeld moeten worden. Indien de waarde *True* is, worden de gebeurtenissen automatisch gekoppeld. De standaardwaarde voor dit attribuut is *True*.

Buffer

De waarde van dit attribuut bepaalt of de HTTP uitvoer buffer gebruikt wordt. Wanneer de waarde *True* is, wordt deze buffer gebruikt. De standaard waarde is *True*.

ClassName

Dit attribuut bevat de naam van een klasse die bij deze pagina hoort. De klasse zal automatisch dynamisch gecompileerd worden door ASP.NET, zodra de pagina wordt opgevraagd. De waarde kan elke geldige naam van een klasse zijn, maar moet geen namespace bevatten.

ClientTarget

Met behulp van dit attribuut kunt u aangeven voor welke browser de server controls opgemaakt moeten worden. De waarde kan iedere geldige browser of alias zijn.

CodePage

Dit attribuut geeft de code pagina aan waarin het resultaat verzonden zal worden. Indien u de pagina in een andere codepagina ontwikkeld heeft dan de standaardcodepagina van de webserver, zult u dit attribuut moeten gebruiken.

CompilerOptions

Dit attribuut biedt u de mogelijkheid om commandoregel opties voor de C# of Visual Basic.NET compiler mee te geven.

ContentType

Dit attribuut bepaalt het contenttype van de uitvoer. Dit type wordt opgegeven als een standaard MIME type zoals text/html of image/jpeg.

Culture

De waarde van dit attribuut geeft de cultuurinstellingen voor de pagina weer. De CultureInfo klasse bevat informatie over cultuur instellingen.

Debug

Met behulp van dit attribuut geeft u aan of de pagina gecompileerd moet worden met debuggegevens. Indien de waarde *True* is, wordt de pagina met deze debuggegevens gecompileerd.

Description

Met dit attribuut kunt u een beschrijving van de pagina opgeven. Deze beschrijving wordt door de ASP.NET paginaverwerker genegeerd.

EnableSessionState

Dit attribuut definieert de vereisten voor de sessiestatus van deze pagina. De drie mogelijke waarden zijn *True*, *ReadOnly* en *False*. Indien de waarde *True* is, wordt de sessiestatus volledig beschikbaar gesteld. Bij de waarde *ReadOnly* kan de sessiestatus wel uitgelezen maar niet aangepast worden. Bij de waarde *False* is de sessiestatus niet beschikbaar.

EnableViewState

Met dit attribuut kunt u aangeven of de viewstatus bijgehouden moet worden. Indien de waarde *True* is, wordt de viewstatus bijgehouden. De standaardwaarde is *True*.

EnableViewStateMac

Dit attribuut geeft aan of er een controle op de viewstatus moet worden uitgevoerd. Deze controle, de Machine Authentication Check

of MAC, moet voorkomen dat er door de gebruiker wijzigingen in de viewstatus zijn aangebracht. De MAC is een gecodeerde versie van de view status.

ErrorPage

Dit attribuut bevat de URL waarde gebruiker naartoe verwezen moet worden wanneer er een fout plaatsvindt die niet verwerkt kan worden.

Explicit

Dit attribuut is specifiek voor Visual Basic.NET bedoeld en zal genegeerd worden wanneer er een andere taal gebruikt wordt. Indien de waarde op *True* staat, wordt de pagina gecompileerd in de Visual Basic Option Explicit modus. Alle variabelen moeten in dat geval expliciet gedeclareerd worden met behulp van Dim, Private, Public of Redim. De standaardwaarde voor dit attribuut is *False*.

Inherits

De eventuele code behind klasse bij een pagina wordt met behulp van dit attribuut gedefinieerd. De waarde kan elke geldige naam van een klasse zijn. Deze klasse dient echter wel afgeleid te zijn van de Page klasse.

Language

De taal waarin een pagina is ontwikkeld wordt opgegeven via dit attribuut. De waarde is de naam van een taal welke door .NET ondersteund wordt. De voor de hand liggende waarden zijn VB en C#, maar ook andere talen zijn mogelijk.

LCID

U kunt de locatie aanduiding instellen via dit attribuut. Standaard gebruikt ASP.NET de lo-

catie aanduiding die door de webserver gebruikt wordt..

ResponseEncoding

Dit attribuut geeft aan welke codering er gebruikt moet worden voor het resultaat van de pagina. Mogelijke waarden zijn onder andere ASCII en Unicode.

Src

De bestandsnaam van een eventueel code behind bestand wordt met behulp van dit attribuut opgegeven.

SmartNavigation

Dit attribuut geeft aan of de pagina gebruik maakt van de smart navigation mogelijkheid van Internet Explorer 5 en hoger. Smart navigation is een nieuwe mogelijkheid van de moderne Internet Explorer browsers waar een pagina ververst kan worden terwijl zowel de huidige positie binnen de pagina als de focus van het huidige control intact blijven. Met de toepassing van smart navigation kunt u een pagina meerdere malen verversen zonder dat er een hinderlijke flikkering in de pagina waarneembaar is en zonder dat er meerdere pagina's in de geschiedenis van de browser terecht komen. Deze mogelijkheid is vooral geschikt voor pagina's waar weinig visueel verschil tussen de verschillende verversingen zit.

Strict

Dit attribuut is specifiek voor Visual Basic.NET bedoeld en zal genegeerd worden wanneer er een andere taal gebruikt wordt. Indien de waarde op *True* staat, wordt de pagina gecompileerd in de Visual Basic Option Strict modus. De standaardwaarde voor dit attribuut is *False*.

Trace

Dit attribuut geeft aan of het volgen van code actief is. Wanneer de waarde *True* is, wordt de code gevolgd. De standaardwaarde is *False*.

TraceMode

In het verlengde van het *Trace* attribuut, geeft het *TraceMode* attribuut aan hoe de uitvoer van code volgen getoond moet worden. U kunt de uitvoer sorteren op tijd door de waarde *SortByTime* te gebruiken. Wanneer u de waarde *SortByCategory* gebruikt, wordt de uitvoer gesorteerd op categorie. De standaardwaarde, wanneer code volgen actief is, is *SortByTime*.

Transaction

Dit attribuut geeft aan of transacties op de huidige pagina ondersteund worden. Mogelijke waarden zijn *Disabled*, *NotSupported*, *Supported*, *Required*, en *RequiresNew*. De standaardwaarde is *Disabled*.

UICulture

Dit attribuut geeft de cultuurinstelling voor de gebruikersinterface weer. De CultureInfo klasse bevat informatie over cultuur instellingen.

WarningLevel

Dit attribuut geeft het waarschuwingsniveau van de compiler weer. Dit niveau geeft aan wanneer de compiler het compileren van de pagina afbreekt. Mogelijke waarden zijn 0 tot en met 4.

@ Control

Deze opdracht definieert specifieke eigenschappen, voor gebruikerscontrols, die door de ASP.NET compiler en paginaverwerker gebruikt worden. Deze opdracht kan alleen in gebruikers controls worden toegepast en per .ascx bestand kan er slechts een @ *Control* opdracht voorkomen. De attributen worden door spaties gescheiden terwijl de attribuut met de bijbehorende waarde juist niet door spaties gescheiden mogen worden, er wordt altijd de vorm `attribuut="waarde"` gebruikt.

Het onderstaande voorbeeld geeft ASP.NET de opdracht om Visual Basic als taal te gebruiken en de resulterende pagina te verzenden met het HTTP MIME type `text/xml`.

```
<%@ Control Language="VB" Explicit="True" %>
```

De onderstaande paragrafen behandelen de attributen bij deze opdracht.

AutoEventWireup

Dit attribuut geeft aan of de gebeurtenissen op deze pagina automatisch gekoppeld moeten worden. Indien de waarde *True* is, worden de gebeurtenissen automatisch gekoppeld. De standaardwaarde voor dit attribuut is *True*.

ClassName

Dit attribuut bevat de naam van een klasse die bij deze pagina hoort. De klasse zal automatisch dynamisch gecompileerd worden door ASP.NET, zodra de pagina wordt opgevraagd. De waarde kan elke geldige naam van een klasse zijn, maar moet geen namespace bevatten.

CompilerOptions

Dit attribuut biedt u de mogelijkheid om commandoregel opties voor de C# of Visual Basic.NET compiler mee te geven.

Debug

Met behulp van dit attribuut geeft u aan of de pagina gecompileerd moet worden met debuggegevens. Indien de waarde *True* is, wordt de pagina met deze debuggegevens gecompileerd.

Description

Met dit attribuut kunt u een beschrijving van de pagina opgeven. Deze beschrijving wordt door de ASP.NET paginaverwerker genegeerd.

EnableViewState

Met dit attribuut kunt u aangeven of de viewstatus bijgehouden moet worden. Indien de waarde *True* is, wordt de viewstatus bijgehouden. De standaardwaarde is *True*.

Explicit

Dit attribuut is specifiek voor Visual Basic.NET bedoeld en zal genegeerd worden wanneer er een andere taal gebruikt wordt. Indien de waarde op *True* staat, wordt de pagina gecompileerd in de Visual Basic Option Explicit modus. Alle variabelen moeten in dat geval expliciet gedeclareerd worden met behulp van Dim, Private, Public of Redim. De standaardwaarde voor dit attribuut is *False*.

Inherits

De eventuele code behind klasse bij een pagina wordt met behulp van dit attribuut gedefinieerd. De waarde kan elke geldige naam

van een klasse zijn. Deze klasse dient echter wel afgeleid te zijn van de Page klasse.

Language

De taal waarin een pagina is ontwikkeld wordt opgegeven via dit attribuut. De waarde is de naam van een taal welke door .NET ondersteund wordt. De voor de hand liggende waarden zijn VB en C#, maar ook andere talen zijn mogelijk.

Strict

Dit attribuut is specifiek voor Visual Basic.NET bedoeld en zal genegeerd worden wanneer er een andere taal gebruikt wordt. Indien de waarde op *True* staat, wordt de pagina gecompileerd in de Visual Basic Option Strict modus. De standaardwaarde voor dit attribuut is *False*.

Src

De bestandsnaam van een eventueel code behind bestand wordt met behulp van dit attribuut opgegeven.

WarningLevel

Dit attribuut geeft het waarschuwingsniveau van de compiler weer. Dit niveau geeft aan wanneer de compiler het compileren van de pagina afbreekt. Mogelijke waarden zijn 0 tot en met 4.

@ Import

Met de @ *Import* opdracht kunt u expliciet een namespace in een pagina importeren. Nadat u een namespace geïmporteerd heeft, zijn alle klassen en interfaces binnen die namespace beschikbaar in de pagina. De geïmporteerde namespace kan een onderdeel van het .NET Framework uitmaken, maar het

kan ook een namespace zijn die door de gebruiker zelf is gedefinieerd.

Om bijvoorbeeld de System.Data namespace te importeren kunt u de onderstaande opdracht gebruiken.

```
<%@ Import Namespace="System.Data" %>
```

Per @ *Import* opdracht kunt u slechts een namespace importeren. Indien u meerdere namespaces wilt importeren, kunt u meerdere @ *Import* opdrachten gebruiken.

Namespace

Het enige attribuut van de @ *Import* opdracht is het attribuut Namespace. Dit attribuut geeft de naam van de te importeren namespace weer. Standaard worden de onderstaande namespaces altijd in een pagina geïmporteerd.

- System
- System.Collections
- System.Collections.Specialized
- System.Configuration
- System.IO
- System.Text
- System.Text.RegularExpressions
- System.Web
- System.Web.Caching
- System.Web.Security
- System.Web.SessionState
- System.Web.UI
- System.Web.UI.HtmlControls
- System.Web.UI.WebControls

@ Implements

Deze opdracht geeft op een declaratieve manier – in tegenstelling tot een programmatische manier – aan dat de betreffende pagina of het betreffende gebruikerscontrol de opgegeven .NET Framework interface implementeert. In een Web Forms pagina moeten de gebeurtenissen, methoden en eigenschappen van de interface binnen een *<script>* tag gedeclareerd worden. De interface kan niet in een code behind bestand geïmplementeerd worden wanneer u deze opdracht gebruikt.

Interface

Het enige attribuut van de *@ Implements* opdracht is het attribuut Interface. Dit attribuut geeft de naam van de te implementeren interface weer.

@ Register

Deze opdracht biedt u de mogelijkheid om aliassen te associëren met namespaces en klasse namen. Daardoor ontstaat de mogelijkheid om gebruikers controls en op maat gemaakte server controls in een pagina op te nemen. Wanneer de pagina opgevraagd wordt, kunnen deze controls opgemaakt worden. U kunt de controls declaratief – in tegenstelling tot programmatisch – aan uw pagina toevoegen.

Wanneer u een gebruikers control registreert, dient u de *TagName* en *TagPrefix* attributen te gebruiken. Deze twee gebruikt u, gescheiden door een dubbele punt, wanneer u het control in de pagina wilt declareren.

```
TagPrefix:TagName
```

Bij het gebruik van op maat gemaakte server controls die tot een .dll bestand zijn gecompileerd, gebruikt u de TagPrefix met de Assembly en Namespace attributen. Wanneer u het Namespace attribuut niet gebruikt, of

het leeg laat zal de paginaverwerker een foutmelding genereren.

De vijf genoemde attributen worden in de onderstaande paragrafen besproken.

TagPrefix

Dit attribuut wordt gebruikt om een namespace met een alias te associëren.

TagName

Dit attribuut wordt gebruikt om de naam van een klasse met een alias te associëren.

Namespace

Dit attribuut bevat de namespace die met de TagPrefix geassocieerd moet worden.

Src

Dit attribuut bevat de locatie van het gebruikerscontrol dat met het TagPrefix:TagName paar geassocieerd moet worden. Deze locatie kan zowel absoluut als relatief zijn.

Assembly

Dit attribuut bevat de naam van de assembly waarin de namespace die met de TagPrefix geassocieerd moet worden zich bevindt. Deze naam bevat geen extentie.

@ Assembly

Met behulp van de @ *Assembly* opdracht kunt u assemblies declaratief aan uw pagina toevoegen zodat alle namespaces en klassen binnen die assembly beschikbaar zijn binnen uw pagina. Tijdens het compileren wordt er door de compiler rekening gehouden met de genoemde assemblies.

Assemblies die in de \Bin map binnen uw webapplicatie staan, worden automatisch beschikbaar gesteld aan pagina's binnen die applicatie. Voor deze assemblies hoeft u geen *@ Assembly* opdracht te gebruiken. Om deze functionaliteit uit te schakelen, kunt u de onderstaande regel verwijderen uit het web.config bestand van uw applicatie.

```
<add assembly="*"/>
```

Deze opdracht kent twee attributen.

Name

Dit attribuut bevat de naam van de assembly die aan de pagina gekoppeld moet worden. Deze naam dient geen extensie te bevatten.

Src

Het Src attribuut bevat de naam van een bronbestand dat dynamisch gecompileerd moet worden. Na compilatie zal de resulterende assembly aan uw pagina gekoppeld worden.

Kiezen

U kunt niet zowel een *Name* als een *Src* attribuut gebruiken in dezelfde *@ Assembly* opdracht. Indien u beide attributen wilt gebruiken, zult u twee *@ Assembly* opdrachten moeten gebruiken.

@ OutputCache

Met behulp van de *@ OutputCache* opdracht, kunt u op een declaratieve manier de gebruikte regels voor het cachen van de uitvoer van een pagina of gebruikerscontrol beïnvloeden.

U kunt de regels voor het cachen van uitvoer ook beïnvloeden door middel van de HttpCachePolicy.SetExpires en HttpCachePolicy.SetCacheability methoden van de HttpResponse.Cache eigenschap.

De onderstaande opdracht toont hoe u de tijdsduur waarin de uitvoer van een pagina of gebruikerscontrol gecached wordt, kunt beïnvloeden.

```
<%@ OutputCache Duration="25"
    VaryByParam="none" %>
```

Duration

Dit attribuut bepaalt de tijd in seconden waarin de uitvoer van een pagina of gebruikerscontrol gecached wordt. Door dit attribuut in een pagina of gebruikerscontrol te gebruiken, wordt er een verval regel bepaald voor het betreffende object. Tevens wordt de uitvoer van de pagina of het gebruikerscontrol automatisch gecached. Dit attribuut is verplicht binnen deze opdracht.

Location

Met behulp van dit attribuut kunt u bepalen waar de uitvoer gecached moet worden. De mogelijke waarden zijn als volgt:

- **Any** Het cachen van de uitvoer kan op verschillende plaatsen gebeuren: in de browser waar het verzoek vandaan komt, in een proxy server of op de server waar het verzoek verwerkt is.
- **Cliënt** Het cachen van de uitvoer vindt plaats in de browser waar het verzoek vandaan komt.
- **Downstream** Het cachen van de uitvoer kan op verschillende plaatsen gebeuren: in de browser waar het verzoek vandaan komt of in een proxy server, maar niet op de server waar het verzoek verwerkt is.
- **None** De uitvoer wordt niet gecached.
- **Server** De uitvoer wordt gecached op de server waar het verzoek verwerkt is.

De standaardwaarde voor dit attribuut is *Any*. Dit attribuut wordt niet ondersteund in gebruikerscontrols.

VaryByCustom

In dit attribuut kunt u een waarde opgeven die de variatie van het cachen bepaalt. Wanneer de waarde van dit attribuut browser is,

wordt de variatie bepaald door de naam en versie-informatie van de browser. U kunt ook uw eigen specificatie bepalen, in dat geval dient u de HttpApplication.GetVaryByCustomString methode te overschrijven in het global.asax bestand binnen uw webapplicatie. Dit attribuut wordt niet ondersteund in gebruikerscontrols.

VaryByHeader

Ook dit attribuut bepaalt de variatie van het cachen, op basis van de header informatie. U kunt een puntkomma gescheiden lijst van headers opgeven. Voor iedere opgegeven header wordt er een versie van het opgevraagde document bewaard. Dit attribuut wordt niet ondersteund in gebruikers controls.

VaryByParam

Ook dit attribuut bepaalt de variatie van het cachen, op basis van de parameter informatie. U kunt een puntkomma gescheiden lijst van parameters opgeven. Voor iedere opgegeven parameter wordt er een versie van het opgevraagde document bewaard. De parameters kunnen zowel GET als POST parameters zijn. De mogelijke waarden voor dit attribuut zijn `none`, * of een geldige parameternaam.

Dit attribuut is verplicht wanneer u de uitvoer van ASP.NET pagina's wilt cachen. Ook voor gebruikerscontrols is het verplicht, tenzij u gebruik maakt van het *VaryByControl* attribuut.

VaryByControl

Ook dit attribuut bepaalt de variatie van het cachen, op basis van de eigenschappen van gebruikerscontrols. U kunt een puntkomma gescheiden lijst van namen van eigenschappen opgeven. Deze eigenschappen dienen wel volledig uitgeschreven te zijn. De uitvoer

van het betreffende gebruikers control wordt gecached op basis van de waarden van de opgegeven eigenschappen.

Dit attribuut is verplicht wanneer u de uitvoer van gebruikerscontrols wilt cachen, tenzij u gebruik maakt van het *VaryByParam* attribuut. Dit attribuut kan niet gebruikt worden in ASP.NET pagina's.

@ Reference

Deze opdracht biedt u de mogelijkheid om een pagina of gebruikerscontrol declaratief aan de huidige pagina of gebruikerscontrol toe te voegen. Het opgegeven gebruikers control – of de pagina – wordt dynamisch gecompileerd en aan het ControlCollection toegevoegd.

Page

Het attribuut Page bevat de naam van een Web Forms pagina die dynamisch gecompileerd en gekoppeld moet worden.

Control

Het attribuut Control bevat de naam van een gebruikers control dat dynamisch gecompileerd en gekoppeld moet worden.