HOOFDSTUK 2

Datum en tijd

Een instantie van de klasse *DateTime* kan een 8-byte waarde bevatten, die een datum representeert tussen 1 januari 0001 en 31 december 9999. Daarnaast bevat *DateTime* ook een tijd tussen 0:00:00 (middernacht) tot 23:59:59.999999. Het is dus een datatype dat zeer precies een bepaald moment in de tijd kan opslaan; de precisie is zelfs 100 nanoseconden! In vergelijking met Visual Basic 6 kent Visual Basic 2005, in combinatie met het .NET Framework een groot aantal klassen en methoden waarmee het werken met datums en tijd veel eenvoudiger is geworden. Antwoorden op vragen zoals: 'Wat is de datum tien dagen na nu?' of 'Is dit jaar een schrikkeljaar?' zijn met één regel code te verkrijgen. Om redenen van compatibiliteit met Visual Basic 6 kunt u ook nog gebruikmaken van de klasse *Date*. Beide klassen *Date* en *DateTime* hebben dezelfde functionaliteit. Wij adviseren u echter om binnen Visual Basic 2005 gebruik te maken van de nieuwe klasse DateTime.
In dit hoofdstuk maakt u kennis met de diverse methoden op de klasse *DateTime*. Ook worden deze methoden toegepast om wat meer ingewikkelde programmeervraagstukken op te lossen. In het laatste recept stellen we u de klasse *StopWatch* voor. Hiermee kunt u vaststellen hoeveel tijd een bepaalde procedure kost.

RECEPT 2.1 Bepaal de huidige datum

PROBLEEM U wilt de datum van vandaag bepalen.

OPLOSSING Gebruik de methode *System.DateTime.Now()* of *System.DateTime.Today()*.

UITWERKING

Elke variabele van het type *DateTime* kent de methoden *Now()* en *Today()*. Het zijn echter shared methoden die u ook direct op de klasse kunt aanroepen. Indien u alleen geïnteresseerd bent in het datumgedeelte, dan biedt de methode *Today()* u voldoende informatie. Is daarentegen ook de huidige tijd van belang, dan gebruikt u de methode *Now()*.

```
' Bepaal de huidige datum
Dim datNu1 As Date = DateTime.Now()
Dim datNu2 As DateTime = DateTime.Today()

' Toon de datums
Console.WriteLine("Vandaag is het: " & datNu1.ToString())
Console.WriteLine("Vandaag is het: " & datNu2.ToString())

' Aantal ticks
Console.WriteLine("Aantal ticks: " & datNu1.Ticks.ToString)
Console.WriteLine("Aantal ticks: " & datNu2.Ticks.ToString)
```

Bovenstaande code schrijft het resultaat naar het venster Output van uw Visual Studio 2005-omgeving. Bij ons resulteert dit voorbeeld in het volgende resultaat:

```
Vandaag is het: 27-5-2007 20:27:07
Vandaag is het: 27-5-2007 0:00:00
Aantal ticks: 633158944271950000
Aantal ticks: 633158208000000000
```

Zoals u ziet bevat de variabele `datNu1`, die van een datum is voorzien door de methode *Now()*, wel een tijd, terwijl dat bij `datNu2` niet het geval is. In dit voorbeeld hebben we tevens het aantal *Ticks* laten weergeven. De eigenschap *Ticks* is interessant, omdat het erg dicht komt bij de manier waarop een datum of tijd intern wordt opgeslagen. Deze eigenschap komt overeen met het aantal '100e nanoseconden' die verstreken zijn sinds 1 januari 0001. Hoewel de eigenschap *Ticks* wel een zeer precies tijdstip weergeeft, wordt deze intern niet zo vaak bijgewerkt. Hierdoor is deze eigenschap eigenlijk ongeschikt voor zogenaamde 'high-resolution' timers. Sommige applicaties, met name multimedia- of 3D-toepassingen, geven een beter resultaat wanneer bijvoorbeeld de animatie vaker per tijdseenheid wordt bijgewerkt. Voor dit soort toepassingen is de klasse *StopWatch* beter geschikt.

RECEPT 2.2 Bepaal het verschil tussen twee datums

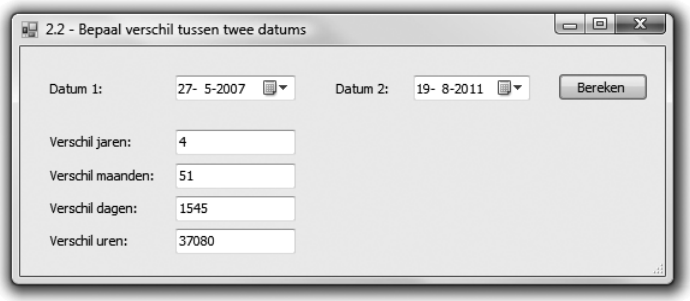
PROBLEEM U wilt het verschil tussen twee datums bepalen. U wilt het verschil weten in jaren, maanden, dagen of uren.

OPLOSSING Gebruik de methode *Microsoft.VisualBasic.DateDiff()*.

UITWERKING

In sommige gevallen biedt de namespace *Microsoft.VisualBasic* functionaliteit waar geen directe .NET-vervanger voor is. De methode *DateDiff()* is hier een voorbeeld

van. Hoewel wij normaal gesproken de functionaliteit uit deze namespace proberen te vermijden, kunnen we *DateDiff()* niet links laten liggen. Het onderscheidt zich met name door de uitgebreide mogelijkheden om de eenheden waarin het resultaat moet worden berekend in te stellen. Standaard kent het .NET Framework ook de klasse *TimeSpan*. De methode *DateTime.Substract()* retourneert namelijk een object van dit type. Op het eerste gezicht lijkt deze methode en klasse de optimale oplossing om het verschil tussen twee datums te berekenen. Feitelijk gezien is het daar ook zeer geschikt voor. Alleen is de grootste uitvoereenheid van een object van het type *TimeSpan*, het aantal dagen en dat is dus niet zo praktisch. Stelt u zich voor dat het verschil in aantal dagen 30 is. Is er dan ook een maand verschil? Een maand kan immers uit 28, 29, 30 of 31 dagen bestaan. Indien u het aantal dagen wilt omrekenen naar maanden, zult u dus rekening moeten houden met de specifieke maanden en eventueel zelfs met het feit of het een schrikkeljaar betreft. Dezelfde problematiek is natuurlijk ook aan de orde bij het berekenen van het verschil in jaren.



Afbeelding 2.1 Bereken het verschil tussen twee datums.

In onderstaand voorbeeld wordt het verschil berekend tussen de twee datums die in de twee *DateTimePickers* zijn ingevoerd. We rekenen met de eigenschap *Date* van de eigenschap *Value* omdat we alleen met het datumgedeelte willen rekenen. De eigenschap *Value* bevat namelijk (al is het nu niet zichtbaar in het *besturingselement*) ook de huidige tijd. Het eerste argument van de methode *DateDiff()* is een element uit de enumeration *Microsoft.VisualBasic.DateInterval*. Een enumeration is een lijst met relevante mogelijkheden. Hierdoor is het mogelijk om het verschil te berekenen in seconden, minuten, uren, dagen, maanden, kwartalen of jaren. U kunt zelfs het aantal weken tussen twee datums berekenen. Het is belangrijk te weten dat standaard de zondag als eerste dag van de week wordt gezien. U kunt deze echter ook op bijvoorbeeld maandag instellen. Dit doet u door het derde en optionele argument van deze functie te voorzien van de waarde *FirstDayOfWeek.Monday*.

```
' Dimensioneer de variabelen
Dim datDatum1 As DateTime = dtpDatum1.Value.Date
Dim datDatum2 As DateTime = dtpDatum2.Value.Date
Dim intVerschilDagen As Integer = 0
Dim intVerschilJaren As Integer = 0
Dim intVerschilMaanden As Integer = 0
```

```
Dim intVerschilUren As Integer = 0

' Bepaal de het verschil tussen twee datums
intVerschilDagen = DateDiff(DateInterval.Day, datDatum1, datDatum2)
intVerschilJaren = DateDiff(DateInterval.Year, datDatum1, datDatum2)
intVerschilMaanden = DateDiff(DateInterval.Month, datDatum1, datDatum2)
intVerschilUren = DateDiff(DateInterval.Hour, datDatum1, datDatum2)

' Toon het resultaat
txtVerschilDagen.Text = intVerschilDagen.ToString()
txtVerschilJaren.Text = intVerschilJaren.ToString()
txtVerschilMaanden.Text = intVerschilMaanden.ToString()
txtVerschilUren.Text = intVerschilUren.ToString()
```

Indien u het verschil tussen twee datums in dagen of kleinere eenheden wilt weten, dan is de methode *DateTime.Substract()* te prefereren. U schrijft dan code die consistenter is met het .NET Framework en daardoor ook beter is te lezen door bijvoorbeeld een C#-ontwikkelaar. Daarom ziet u voor de volledigheid ook de volledige .NET-implementatie.

```
' Bepaal het verschil tussen de twee datums
Dim objTimeSpan As TimeSpan = datDatum2.Subtract(datDatum1)

' Bepaal het verschil in dagen/uren
intVerschilDagen = objTimeSpan.TotalDays
intVerschilUren = objTimeSpan.TotalHours

' Toon het resultaat
txtVerschilDagen.Text = intVerschilDagen.ToString()
txtVerschilUren.Text = intVerschilUren.ToString()
```

RECEPT 2.3 Tel bij een datum een bepaalde tijdseenheid op

PROBLEEM U wilt weten wat de nieuwe datum is na het optellen van een aantal dagen, maanden of jaren.

OPLOSSING Gebruik de methoden *DateTime.AddDays()*, *DateTime.AddMonths()* of *DateTime.AddYears()*.

UITWERKING

In tegenstelling tot het vorige recept, biedt het .NET Framework voor het berekenen van een nieuwe datum, wel een zeer krachtige en intuïtieve oplossing. De

klasse *DateTime* beschikt namelijk over de methoden *AddDays()*, *AddMonths()* en *AddYears()* waarmee ons probleem makkelijk valt op te lossen. Het biedt zelfs methoden om ook snel (milli)seconden, minuten of uren op te tellen.

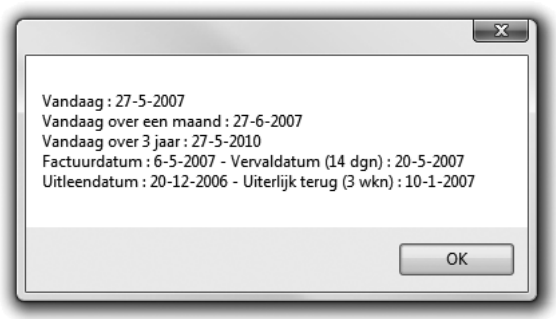
```
' Bereken de datum van vandaag over een maand
Dim dteVandaagOverEenMaand As DateTime = Today.AddMonths(1)

' Bereken de datum van vandaag over drie jaar
Dim dteVandaagOverDrieJaar As DateTime = Today.AddYears(3)

' Bereken de vervaldatum van een factuur
Dim dteFactuurDatum As DateTime = #5/6/2007#
Dim dteVervalDatum As DateTime = dteFactuurDatum.AddDays(14)

' Bereken datum boeken terugbrengen (3 weken)
Dim dteUitleenDatum As DateTime = #12/20/2006#
Dim dteUiterlijkTerugDatum As DateTime = dteUitleenDatum.AddDays(3 * 7)
```

In bovenstaande code wordt in de eerste coderegel een datum berekent die precies een maand verder ligt dan vandaag. Bij het bereken van de datum die drie jaar verder ligt wordt ook gebruikgemaakt van de methode *Today()*. U kunt echter ook de berekeningsmethoden aanroepen op elke variabele van het type *DateTime*. In dit geval berekenen we de vervaldatum van een factuur met de factuurdatum van 6 mei 2007. In dit voorbeeld moet de factuur namelijk binnen veertien dagen betaald worden. Hoewel er standaard geen methode *AddWeeks()* is geïmplementeerd, kunt u hiervoor natuurlijk ook de methode *AddDays()* aanroepen. U moet alleen het aantal gewenste weken vermenigvuldigen met zeven. Deze truc is hier toegepast om de uiterlijke retourdatum van een bibliotheekboek te berekenen. Het boek mag maximaal drie weken worden geleend.
Omdat de diverse *Add...()* methoden een waarde teruggeven die ook van het type *DateTime* is, kunt u direct op het resultaat wederom een (andere) methode aanroepen. Dit kan in sommige gevallen erg handig zijn.



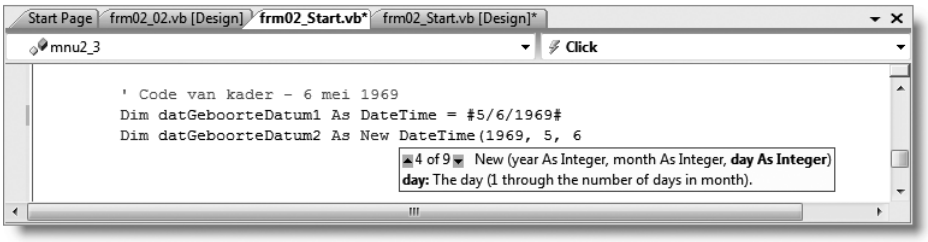
Afbeelding 2.2 Resultaat van enkele datumberekeningen.

Hardgecodeerde datums

In veel gevallen zullen *DateTime* variabelen worden gevuld door gegevens uit een database of bijvoorbeeld door invoer van een gebruiker. Soms kan het echter voorkomen dat u een datum toch in uw code wilt opnemen. Hoe wijst u die datum toe aan een variabele? Het gevaar is namelijk dat u bij de toewijzing een verkeerd datumformaat gebruikt. Een mogelijkheid is om de datum tussen hekjes (#) te zetten. Omdat u niet uw code wilt aanpassen wanneer uw software op systemen moet draaien waarbij gebruik wordt gemaakt van niet-Nederlandse datuminstellingen, is hiervoor slechts één formaat geldig: de Amerikaanse datumnotering 'MM/DD/YYYY'. In de praktijk zult u zien dat hierin nog wel eens fouten worden gemaakt. Dit komt mede door het feit dat de *Intellisense* van Visual Studio 2005 u hierbij niet de helpende hand biedt. Er zullen ontwikkelaars zijn die de eerste coderegel zullen interpreteren als 5 juni 1969 in plaats van 6 mei 1969!

```
Dim datGeboorteDatum1 As DateTime = #5/6/1969#
Dim datGeboorteDatum2 As New DateTime(1969, 5, 6)
```

Om dit soort problemen te voorkomen kunt u daarom beter gebruikmaken van de constructor van de klasse *DateTime*. Deze constructor bestaat uit maar liefst negen overloaded methoden.



Afbeelding 2.3 Gebruik de constructor van de klasse DateTime om een datum in te stellen.

Het is nu haast onmogelijk om een verkeerde datum op te geven. Er kan immers geen verwarring bestaan over de volgorde of het formaat van de desbetreffende datum.

Stel bijvoorbeeld dat u van uw software een evaluatieversie wilt uitbrengen en dat u gebruikers een testperiode van een maand wil geven. De software moet dan eigenlijk vandaag over een maand nog wel goed functioneren. De dag daarna is de evaluatieperiode verlopen en moet de software stoppen met werken. U kunt deze expiratiedatum op de volgende manier berekenen:

```
' Bereken einddatum evaluatieperiode
Dim datExpiratieDatum As DateTime = Today.AddMonths(1).AddDays(1)
```

Naast het optellen kunt u natuurlijk ook vanaf een datum terug rekenen. Welke datum was het vandaag minus 45 dagen? Roep in dit geval *DateTime.AddDays()* aan met een negatief getal.

```
' Bereken datum in het verleden
Dim datDatumInVerleden As DateTime = Today.AddDays(-45)
```

RECEPT 2.4 Bepaal het jaar, de maand en de dag van een datum

PROBLEEM U wilt van een gegeven datum jaartal, maand en dagnummer bepalen.

OPLOSSING Roep de eigenschappen *Year*, *Month* en *Day* aan.

UITWERKING

De klasse *DateTime* beschikt over de eigenschappen *Year*, *Month* en *Day* waarmee u gewenste gegevens gemakkelijk uit kunt lezen. Deze eigenschappen zijn van het type *Integer*, waardoor u ze eenvoudig kunt gebruiken in andere berekeningen.

```
' Bepaal datum van vandaag
Dim datVandaag As DateTime = DateTime.Today

' Bepaal het jaartal, de maand en de dag
Dim intJaar As Integer = datVandaag.Year
Dim intMaand As Integer = datVandaag.Month
Dim intDag As Integer = datVandaag.Day

' Bouw de tekst om te tonen op
Dim objStringBuilder As New System.Text.StringBuilder
objStringBuilder.AppendLine("Datum vandaag: " & datVandaag)
objStringBuilder.AppendLine("Jaar: " & intJaar)
objStringBuilder.AppendLine("Maand: " & intMaand)
objStringBuilder.AppendLine("Dag: " & intDag)
objStringBuilder.AppendLine(New String("-"c, 40))
objStringBuilder.AppendLine("Dag van de week: " & datVandaag.DayOfWeek.ToString())
objStringBuilder.AppendLine("Dag van het jaar: " & datVandaag.DayOfYear)

' Toon het resultaat
MessageBox.Show(objStringBuilder.ToString(), "Eigenschappen klasse DateTime")
```

Zoals u wellicht hebt gezien in bovenstaand codevoorbeeld worden er ook nog twee extra eigenschappen uitgevraagd, namelijk *DayOfWeek* en *DayOfYear*. De eerste eigenschap geeft het dagnummer. Dit is een waarde uit de enumeration *System.*

DayOfWeek, waarbij de zondag de waarde 0 heeft, de maandag een waarde 1 tot en met uiteindelijk de waarde 6 voor zaterdag. De eigenschap *DayOfYear* retourneert het dagnummer in het jaar, dus de hoeveelste dag vandaag in het jaar is.

Afbeelding 2.4 De eigenschappen Year, Month en Day in actie.

RECEPT 2.5 Bepaal de uren, de minuten en seconden van een tijd

PROBLEEM U wilt van een gegeven datum of tijd het aantal uren, minuten en seconden bepalen.

OPLOSSING Roep de eigenschappen *Hour*, *Minute* en *Second* aan.

UITWERKING

De klasse *DateTime* beschikt over een groot aantal eigenschappen waarmee u informatie kunt verkrijgen over een bepaalde datum en/of tijd. Zoals u reeds in recept 2.4 hebt kunnen lezen kunt u onder meer het jaartal, het maandnummer en het dagnummer eenvoudig bepalen. Naast deze eigenschappen kent deze klasse ook nog eigenschappen om meer informatie van een specifieke tijd te bepalen. Gebruik hiervoor de eigenschappen *Hour*, *Minute* en *Second*. U kunt zelfs het aantal milliseconden bepalen, maar de vraag is in hoeverre die informatie nuttig voor u is.

```
' Bepaal datum/tijd van vandaag
Dim datNu As DateTime = DateTime.Now

' Bepaal de uren, minuten en seconden
Dim intUren As Integer = datNu.Hour
Dim intMinuten As Integer = datNu.Minute
Dim intSeconden As Integer = datNu.Second

' Bouw de tekst om te tonen op
Dim objStringBuilder As New System.Text.StringBuilder
```

```
objStringBuilder.AppendLine("Datum/tijd nu: " & datNu)
objStringBuilder.AppendLine("Uren: " & intUren)
objStringBuilder.AppendLine("Minuten: " & intMinuten)
objStringBuilder.AppendLine("Seconden: " & intSeconden)
objStringBuilder.AppendLine(New String("-"c, 40))
objStringBuilder.AppendLine("Milliseconden: " & datNu.Millisecond)

' Toon het resultaat
MessageBox.Show(objStringBuilder.ToString(), "Eigenschappen klasse DateTime")
```

RECEPT 2.6 Bepaal de tijdzone van het systeem

PROBLEEM U wilt vanuit uw code bepalen op welke tijdzone het huidige systeem is ingesteld. Daarnaast wilt u weten of de zomertijd actief is.

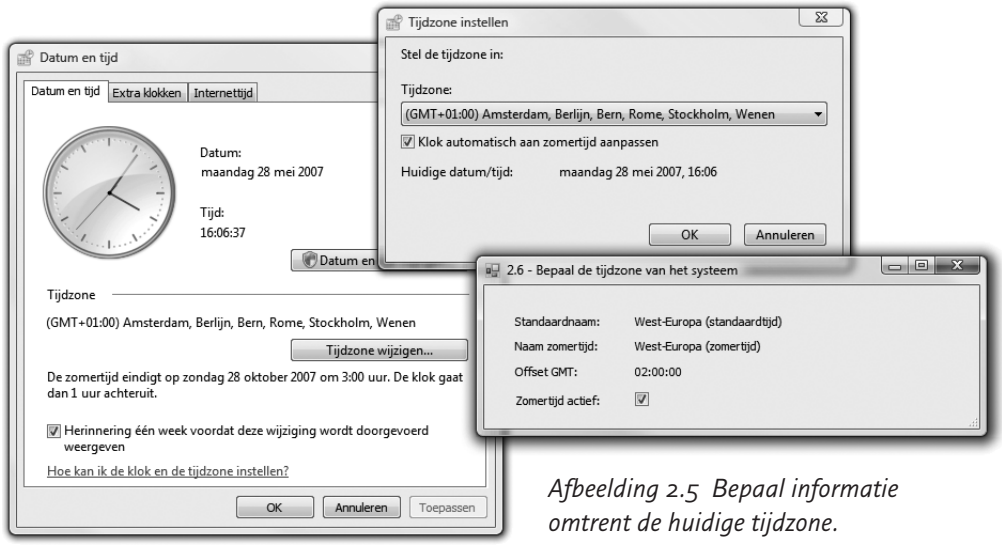
OPLOSSING Gebruik de klasse *System.TimeZone* en vraag de eigenschappen op een objectinstantie van deze klasse uit.

UITWERKING

Het gebruik van de klasse *TimeZone* is niet ingewikkeld. U maakt eerst een variabele aan van dit type en kent er vervolgens de huidige tijdzone aan toe. Dit doet u door de shared methode *TimeZone.CurrentTimeZone()* aan te roepen. U kunt nu de eigenschappen *StandardName* en *DaylightName* uitlezen. Om te bepalen of de zomertijd nu actief is roept u de methode *IsDaylightSavingTime()* aan en geeft de huidige datum mee. Indien u wilt weten of op een datum in de toekomst de zomertijd actief is, geeft u die gewenste datum mee. Dit geldt natuurlijk ook voor een datum in het verleden. Wanneer u wilt weten wat het tijdverschil is ten opzichte van de Greenwich Mean Time (GMT), dan kunt de methode *GetUtcOffset()* aanroepen. Deze aanroep en de eerder beschreven theorie over datums in de toekomst of in het verleden is analoog aan de methode *IsDaylightSavingTime()*.

```
' Maak TijdZone object aan
Dim objTijdZone As TimeZone = TimeZone.CurrentTimeZone
With objTijdZone
    ' Lees eigenschappen aan
    lblNaamStandaard.Text = .StandardName
    lblNaamZomerTijd.Text = .DaylightName
    ' Roep de methoden aan
    lblOffsetGMT.Text = .GetUtcOffset(Now).ToString
    chkZomertijdActief.Checked = .IsDaylightSavingTime(Now)
End With
```

Zoals u kunt zien in afbeelding 2.5, is het erg belangrijk om te weten of de zomertijd actief is, omdat op dat moment het tijdverschil GMT +2:00 uur is. Onze tijdzone staat ingesteld op GMT +1:00!



Afbeelding 2.5 Bepaal informatie omtrent de huidige tijdzone.

RECEPT 2.7 Bepaal de dag- en maandnamen in een andere taal

PROBLEEM U wilt de naam van een dag of maand in een andere taal opvragen.

OPLOSSING Gebruik de klassen *System.Globalization.CultureInfo* en *System.Globalization.DateTimeFormatInfo*.

UITWERKING

Het .NET Framework kent de namespace *System.Globalization*. Deze namespace bevat diverse typen en klassen waarmee u uw applicatie kunt internationaliseren. De klasse *CultureInfo* representeert een landinstelling. Om de gewenste informatie te verkrijgen dient u de eigenschap *DateTimeFormat* te gebruiken. Deze eigenschap is van het type *DateTimeFormatInfo* en dit type bevat de methoden *GetDayName()* en *GetMonthName()*. Dit klinkt misschien allemaal ingewikkeld, maar het valt reuze mee. Een codevoorbeeld zal een en ander verduidelijken.

```
Imports System.Globalization

' Dimensioneer de DateTimeFormatInfo objecten
Dim objFormatInfoDE As DateTimeFormatInfo
Dim objFormatInfoFR As DateTimeFormatInfo
```

```
' Dimensioneer en instantieer de CulterInfo objecten
Dim objCultureInfoDE = New CultureInfo("de-DE")
Dim objCultureInfoFR = New CultureInfo("fr-FR")

' Bepaal de FormatInfo objecten
objFormatInfoDE = objCultureInfoDE.DateTimeFormat
objFormatInfoFR = objCultureInfoFR.DateTimeFormat
Dim objStringBuilder As New System.Text.StringBuilder

' Doorloop alle maanden
For intMaandTeller As Integer = 1 To 12
    objStringBuilder.Append(intMaandTeller.ToString("00") & " - ") _
                    .Append(objFormatInfoDE.GetMonthName(intMaandTeller) & " - ") _
                    .AppendLine(objFormatInfoFR.GetMonthName(intMaandTeller))
Next

' Voeg scheidingslijn toe
objStringBuilder.AppendLine(New String("-", 25))

' Doorloop alle dagen
For intDagTeller As Integer = 0 To 6
    objStringBuilder.Append(intDagTeller.ToString("00") & " - ") _
                    .Append(objFormatInfoDE.GetDayName(intDagTeller) & " - ") _
                    .AppendLine(objFormatInfoFR.GetDayName(intDagTeller))
Next

' Toon resultaat in Output window
Console.WriteLine(objStringBuilder.ToString())
```

Deze code geeft het onderstaande resultaat. Het toont een lijst met het maand- of dagnummer en de daarbij behorende omschrijving. Er worden twee objecten aangemaakt van het type *CultureInfo*. De eerste voor de Duitse taal en de ander voor de Franse taal. Daarna wordt voor beide objecten de *DateTimeFormat* bepaald door het uitlezen van de gelijknamige eigenschappen. In een lus worden eerst de maandnamen bepaald. Dit wordt gedaan door het aanroepen van de methode *GetMonthName()*. Deze methode verwacht één argument, namelijk het gewenste maandnummer. In de volgende lus wordt eigenlijk hetzelfde gedaan, alleen wordt nu de *GetDayName()* aangeroepen met het dagnummer. Op het eerste gezicht lijkt het verwarrend dat de maandnummers van 1 tot 12 lopen, terwijl de dagnummers binnen een reeks van 0 to 6 vallen. Dit heeft te maken met het feit dat het argument van de methode *GetMonthName()* een integer is, terwijl het argument van *GetDayName()* een item is uit de enumeration *System.DayOfWeek*. In praktijk zult u hier dus niet zo snel mee in de fout gaan, omdat de Intellisense van Visual Studio u de helpende hand biedt.

```
01 - Januar - janvier
02 - Februar - février
03 - März - mars
04 - April - avril
05 - Mai - mai
06 - Juni - juin
07 - Juli - juillet
08 - August - août
09 - September - septembre
10 - Oktober - octobre
11 - November - novembre
12 - Dezember - décembre
-------------------------
00 - Sonntag - dimanche
01 - Montag - lundi
02 - Dienstag - mardi
03 - Mittwoch - mercredi
04 - Donnerstag - jeudi
05 - Freitag - vendredi
06 - Samstag - samedi
```

RECEPT 2.8 Formatteer een datum

PROBLEEM U wilt een datum formatteren naar een string.

OPLOSSING Gebruik een van de voorgedefinieerde variaties van de methode *DateTime.ToXXXString()*, of roep de methode *ToString()* aan met een zogenaamde 'Standard Date Format Specifier' of met een eigen 'Custom Date Format Specifier'.

UITWERKING

De klasse *System.DateTime* kent een aantal methoden waarmee u het erg gemakkelijk wordt gemaakt om een datum en of tijd te formatteren naar een bepaald datumformaat. Het zijn stuk voor stuk implementaties van de methode *ToString()*, maar hebben allemaal wel een eigen methodenaam:

```
' Bepaal de datum/tijd van nu
Dim datNu As DateTime = Now()

' Bouw een string op met de verschillende mogelijkheden
Dim objStringBuilder As New System.Text.StringBuilder
objStringBuilder.Append("ToString - ").AppendLine(datNu.ToString)
```

```
objStringBuilder.Append("ToLongDateString - ").AppendLine(datNu.ToLongDateString)
objStringBuilder.Append("ToLongTimeString - ").AppendLine(datNu.ToLongTimeString)
objStringBuilder.Append("ToShortDateString - ").AppendLine(datNu.ToShortDateString)
objStringBuilder.Append("ToShortTimeString - ").AppendLine(datNu.ToShortTimeString)

' Toon resultaat in Output window
Console.WriteLine(objStringBuilder.ToString())
```

Deze code geeft het volgende resultaat. Deze methoden houden rekening met de landafhankelijk datuminstellingen op het systeem waarop ze worden uitgevoerd.

```
ToString - 29-5-2007 5:58:43
ToLongDateString - dinsdag 29 mei 2007
ToLongTimeString - 5:58:43
ToShortDateString - 29-5-2007
ToShortTimeString - 5:58
```

In veel gevallen zult u voldoende hebben aan de functionaliteit van een van deze implementaties van *ToString()*. Wilt u echter toch meer invloed hebben op de geformatteerde string, dan kunt u de methode *ToString()* aanroepen en het gewenste formaat als argument meegeven. Hierin onderscheiden we eigenlijk twee mogelijkheden. U geeft een standaard argument mee, of u maakt uw eigen string om de datum te formatteren. In hoofdstuk 1, om precies te zijn in recept 1.18, hebt u reeds kennis kunnen maken met deze standaardargumenten. Elk argument is nader beschreven in tabel 1.2.

```
' Bepaal de datum/tijd van nu
Dim datNu As DateTime = Now()

' Bouw een string op met de verschillende mogelijkheden
Dim objStringBuilder As New System.Text.StringBuilder
objStringBuilder.Append("d      - ").AppendLine(datNu.ToString("d"))
objStringBuilder.Append("D      - ").AppendLine(datNu.ToString("D"))
objStringBuilder.Append("t      - ").AppendLine(datNu.ToString("t"))
objStringBuilder.Append("T      - ").AppendLine(datNu.ToString("T"))
objStringBuilder.Append("f      - ").AppendLine(datNu.ToString("f"))
objStringBuilder.Append("F      - ").AppendLine(datNu.ToString("F"))
objStringBuilder.Append("g      - ").AppendLine(datNu.ToString("g"))
objStringBuilder.Append("G      - ").AppendLine(datNu.ToString("G"))
objStringBuilder.Append("m of M - ").AppendLine(datNu.ToString("m"))
objStringBuilder.Append("r of R - ").AppendLine(datNu.ToString("r"))
objStringBuilder.Append("s      - ").AppendLine(datNu.ToString("s"))
objStringBuilder.Append("u      - ").AppendLine(datNu.ToString("u"))
objStringBuilder.Append("U      - ").AppendLine(datNu.ToString("U"))
objStringBuilder.Append("y of Y - ").AppendLine(datNu.ToString("U"))
```

```
' Toon resultaat in Output window
Console.WriteLine(objStringBuilder.ToString())
```

De uitvoer van deze code op onze computer geeft het volgende resultaat. Net zoals bij de eerder besproken variaties van *ToXXXString()* is deze uitvoer landspecifiek. Er wordt dus rekening gehouden met de huidige regionale instellingen binnen Windows.

```
d      - 29-5-2007
D      - dinsdag 29 mei 2007
t      - 6:14
T      - 6:14:51
f      - dinsdag 29 mei 2007 6:14
F      - dinsdag 29 mei 2007 6:14:51
g      - 29-5-2007 6:14
G      - 29-5-2007 6:14:51
m of M - 29 mei
r of R - Tue, 29 May 2007 06:14:51 GMT
s      - 2007-05-29T06:14:51
u      - 2007-05-29 06:14:51Z
U      - dinsdag 29 mei 2007 4:14:51
y of Y - dinsdag 29 mei 2007 4:14:51
```

Wanneer u nog meer controle wenst uit te oefenen op de uitvoer van *DateTime.ToString()*, dan kunt u ook uw eigen string opbouwen. Hiervoor zijn een aantal letters gereserveerd, waarbij de combinatie waarin ze worden aangeboden van belang is. Enkele voorbeelden:

```
' Bepaal de datum/tijd van nu
Dim datNu As DateTime = Now

' Bouw een string op met de verschillende mogelijkheden
Dim o As New System.Text.StringBuilder
o.Append("dd-mm-yyyy hh:mm:ss - ").AppendLine(datNu.ToString("dd-mm-yyyy hh:mm:ss"))
o.Append("dd-mm-yyyy HH:mm:ss - ").AppendLine(datNu.ToString("dd-mm-yyyy HH:mm:ss"))
o.Append("ddd dd MMM yy       - ").AppendLine(datNu.ToString("ddd dd MMM yy"))
o.Append("hh:mm:ss:fff        - ").AppendLine(datNu.ToString("hh:mm:ss:fff"))
```

In de eerste regel wordt de uurwaarde van het tijdstip weergeven in een reeks van 1-12, terwijl de regel waarbij gebruikt gemaakt wordt van de hoofdletters 'HH' een reeks hanteert van 0-23. Ook kunt u dagen of maanden weergeven als afkortingen.

```
dd-mm-yyyy hh:mm:ss - 29-37-2007 04:37:28
dd-mm-yyyy HH:mm:ss - 29-37-2007 16:37:28
ddd dd MMM yy       - di 29 mei 07
dd-mm-yyyy hh:mm:ss - 29-37-2007 04:37:28:267
```

In tabel 2.1 ziet u een overzicht van de zogenaamde 'Custom Date Format Specifiers'.

Format	*Beschrijving*
d of dd	Toont de dag van de maand als een nummer tussen 1 en 31.
ddd	Toont de afkorting van de dag van de maand, bijvoorbeeld 'di' voor dinsdag.
dddd	Toont de volledige naam van de dag van de maand.
f, ff, fff tot fffffff	Toont de fracties van seconden (milliseconden). Van 1 tot 7 fracties.
g of gg	Toont de eeuw, bijvoorbeeld A.D.
h of hh	Toont het uur in een bereik van 1-12.
H of HH	Toont het uur in een bereik van 0-23.
m of mm	Toont de minuten in een bereik van 0-59.
M of MM	Toont de maand als een nummer in een bereik van 1-12.
MMM	Toont de afkorting van een maand, bijvoorbeeld 'jan' of 'feb'.
MMMM	Toont de volledige naam van de maand.
s of ss	Toont de seconden in een bereik van 0-59.
t	Toont de eerste letter van de AM/PM markering.
tt	Toont de AM/PM markering.
y of yy	Toont het jaartal in maximaal 2 cijfers. De eerste 2 getallen van het jaartal worden achterwege gelaten.
yyyy	Toont het jaartal in vier cijfers. Indien nodig worden er een of meerdere voorloopnullen toegevoegd.
z of zz	Toont de offset van tijdzone ten opzichte van GMT in hele uren.
zzz	Toont de offset van tijdzone ten opzichte van GMT in uren en minuten.
:	Scheidingsteken voor tijd.
/	Scheidingsteken voor datum.
'	Een teken tussen twee enkele aanhalingstekens wordt als teken weergeven. Dit geldt alleen voor de tekens die een speciale betekenis hebben, zoals bijvoorbeeld de slash (/). Andere letters of karakters kunnen gewoon gebruikt worden.

Tabel 2.1 Overzicht Custom Date Format Specifiers.

RECEPT
2.9 Bepaal het verschil tussen twee datums in werkdagen

PROBLEEM U wilt het aantal werkdagen berekenen tussen twee datums.

OPLOSSING Doorloop alle dagen tussen deze twee datums en bepaal of het een werkdag is. Tel alle werkdagen bij elkaar op.

UITWERKING
Voor sommige toepassingen moet u het aantal werkdagen berekenen die er sinds een bepaalde datum zijn geweest. Het bepalen van dit aantal is op zich niet heel erg ingewikkeld; het is vooral een kwestie van (juist) tellen. Wanneer u echter ook rekening met bepaalde feestdagen wilt houden, zoals Pasen, Pinksteren of Hemelvaartsdag, dan zult u hier extra code voor moeten schrijven.

```
Public Function BepaalVerschilWerkdagen(ByVal datBeginDatum As DateTime, _
                  ByVal datEindDatum As DateTime, _
                  Optional ByVal blnZaterdagGeenWerkdag As Boolean = True) As Integer
    Dim intVerschil As Integer = 0
    Dim intVerschilWerkdagen As Integer = 0
    ' Het verschil kan zowel +1 als -1 zijn
    If datBeginDatum < datEindDatum Then intVerschil = 1 Else intVerschil = -1
    Do Until datBeginDatum = datEindDatum
        ' Ga naar de volgende of vorige dag
        datBeginDatum = datBeginDatum.AddDays(intVerschil)
        ' Is deze dag een werkdag?
        If datBeginDatum.DayOfWeek <> DayOfWeek.Sunday AndAlso _
          (datBeginDatum.DayOfWeek <> DayOfWeek.Saturday Or Not _
           blnZaterdagGeenWerkdag) Then
            ' Werkdag... Verhoog of verlaag het totaal
            intVerschilWerkdagen += intVerschil
        End If
    Loop
    ' Geef het aantal dagen terug
    Return intVerschilWerkdagen
End Function
```

Omdat in sommige branches ook de zaterdag als werkdag wordt aangemerkt, is deze in de functie `BepaalVerschilWerkdagen()` variabel gemaakt. Het is als optioneel derde argument te gebruiken. Indien u dit argument achterwege laat wordt de zaterdag, net als de zondag, als vrije dag gezien.

```
Dim intAantalWerkdagenJaar As Integer = 0
Dim intAantalWerkdagenMaand1 As Integer = 0
Dim intAantalWerkdagenMaand2 As Integer = 0
Dim intAantalWerkdagenWeek As Integer = 0
Dim strBoodschap As String = ""

' Bepaal het aantal werkdagen
intAantalWerkdagenJaar = BepaalVerschilWerkdagen(#1/1/2007#, DateTime.Today)
intAantalWerkdagenMaand1 = BepaalVerschilWerkdagen(DateTime.Today.AddMonths(-1), _
                                                   DateTime.Today)
intAantalWerkdagenMaand2 = BepaalVerschilWerkdagen(DateTime.Today.AddMonths(-1), _
                                                   DateTime.Today, False)
intAantalWerkdagenWeek = BepaalVerschilWerkdagen(DateTime.Today.AddDays(-7), _
                                                 DateTime.Today)
' Toon het resultaat
Console.WriteLine("Vandaag: " & DateTime.Today.ToShortDateString)
Console.WriteLine("Jaar: " & intAantalWerkdagenJaar.ToString)
Console.WriteLine("Maand: " & intAantalWerkdagenMaand1.ToString)
Console.WriteLine("Maand + za: " & intAantalWerkdagenMaand2.ToString)
Console.WriteLine("Week: " & intAantalWerkdagenWeek.ToString)
```

Deze code resulteert in de volgende uitvoer:

```
Vandaag: 3-6-2007
Jaar: 109
Maand: 21
Maand + za: 26
Week: 5
```

RECEPT 2.10 Bepaal einddatum na aantal werkdagen

PROBLEEM U wilt de nieuwe datum bepalen op basis van een aantal werkdagen na de opgegeven datum.

OPLOSSING Tel alleen de dag erbij op indien die dag een werkdag is.

UITWERKING

In recept 2.9 hebt u kunnen lezen hoe we het verschil tussen twee datums in werkdagen kunnen bepalen. Het komt echter ook voor dat u een nieuwe datum wilt bepalen aan de hand van een begindatum en een aantal werkdagen. Om deze datum te berekenen moet u alleen de dag erbij optellen als het dus een werkdag is.

```
Public Function BepaalDatumWerkdagen(ByVal datBeginDatum As DateTime, _
                                     ByVal intAantalDagen As Integer, _
                                     Optional ByVal blnZaterdagGeenWerkdag As _
                                                                  Boolean = True) As DateTime
    Do While intAantalDagen
        ' Bepaal de 'nieuwe' begindatum
        datBeginDatum = datBeginDatum.AddDays(Math.Sign(intAantalDagen))
        ' Bepaal of het een werkdag is
        If datBeginDatum.DayOfWeek <> DayOfWeek.Sunday AndAlso _
          (datBeginDatum.DayOfWeek <> DayOfWeek.Saturday Or Not _
          blnZaterdagGeenWerkdag) Then
            ' Bepaal het 'nieuwe' aantal dagen
            intAantalDagen -= Math.Sign(intAantalDagen)
        End If
    Loop
    ' Geef de nieuwe datum terug
    Return datBeginDatum
End Function
```

RECEPT
2.11 Bepaal de leeftijd

PROBLEEM U wilt de leeftijd van iemand bepalen op basis van de geboortedatum.

OPLOSSING Bereken het verschil in jaren en houdt rekening met het feit of de persoon het huidige jaar al jarig is geweest.

UITWERKING

Op zich is het niet lastig om te bepalen wat de leeftijd is van een persoon. U kunt immers het verschil tussen twee datums met één regel code in jaren vaststellen. Wat u echter niet moet vergeten, is het feit of de persoon het huidige jaar al jarig is geweest. Wanneer dit namelijk niet het geval is, moet u de eerder uitgerekende leeftijd met een jaar verminderen.

```
Public Function BepaalLeeftijd(ByVal datGeboorteDatum As DateTime, _
                               ByVal datHuidigeDatum As DateTime) As Integer
    Dim intLeeftijd As Integer = 0
    ' Indien er geen andere datum is opgegeven, reken met vandaag
    If datHuidigeDatum = #1/1/1800# Then datHuidigeDatum = DateTime.Today
    ' Bepaal de leeftijd
    intLeeftijd = datHuidigeDatum.Year - datGeboorteDatum.Year
    ' Bepaal of persoon dit jaar al jarig is geweest
    If New Date(datHuidigeDatum.Year, datGeboorteDatum.Month, _
```

```
                datGeboorteDatum.Day) > datHuidigeDatum Then
        ' Nog niet jarig geweest
        intLeeftijd -= 1
    End If
    ' Retourneer de leeftijd
    Return intLeeftijd
End Function

Public Function BepaalLeeftijd(ByVal datGeboorteDatum As DateTime) As Integer
   ' Roep de overloaded versie aan
    Return BepaalLeeftijd(datGeboorteDatum, New DateTime(1800, 1, 1))
End Function
```

Omdat het in de praktijk ook nog wel eens voorkomt dat u de leeftijd van een persoon op een bepaalde datum in het verleden wilt weten, hebben we in bovenstaande functie een extra mogelijkheid geïmplementeerd. U kunt hiervoor optioneel een tweede argument meegeven, namelijk de datum waarop u de leeftijd wilt berekenen. Indien u dit argument achterwege laat, zal de leeftijd zoals die vandaag is worden berekend.

```
Dim intLeeftijd As Integer = 0

' U kunt het tweede argument achterwege laten --> Vandaag
'intLeeftijd = BepaalLeeftijd(dtpGeboorteDatum.Value)

' Bereken de leeftijd
intLeeftijd = BepaalLeeftijd(dtpGeboorteDatum.Value, dtpRekenDatum.Value)

' Toon het resultaat
lblLeeftijd.Text = "Leeftijd: " & intLeeftijd.ToString & " jaar."
```

Afbeelding 2.6 Bepaal de leeftijd.

RECEPT
2.12 Bepaal hoeveel tijd een procedure in beslag neemt

PROBLEEM U wilt de performance van een procedure of een deel ervan verbeteren en meten hoelang het uitvoeren van de bewuste code in beslag neemt.

OPLOSSING Gebruik de klasse *System.Diagnostics.StopWatch*.

UITWERKING
De klasse *StopWatch* is een beetje een vreemde eend in de bijt in dit hoofdstuk. Het heeft zeker met tijd te maken, maar het wijkt sterk af van alle andere klassen die de revue zijn gepasseerd. Met de klasse *StopWatch* kunt u meten hoeveel tijd een bepaalde procedure of een deel ervan in beslag neemt. Vooral bij langdurende processen kan het soms een groot verschil maken wanneer u een tiende van een seconde per aanroep van de procedure weet te winnen. Om de beste keuze uit het beschikbare programmeerarsenaal te kunnen maken, wilt u de tijdsduur heel precies kunnen vaststellen. Immers: 'meten is weten, gissen is missen'. De klasse *StopWatch* is hiervoor het perfecte gereedschap. In onderstaande code kunt u zien hoe makkelijk u deze klasse kunt toepassen. Het meet hoeveel milliseconden er nodig zijn om een ListBox te vullen met tienduizend regels.

```
Dim objStopWatch As New Stopwatch

' Start de StopWatch
objStopWatch.Start()

' Voer een langdurige actie uit
For intTeller As Integer = 1 To 10000
    lstTest.Items.Add(intTeller.ToString("00000"))
Next

' Stop de StopWatch
objStopWatch.Stop()

' Toon resultaat
lblResultaat.Text = objStopWatch.ElapsedMilliseconds.ToString()
```

U maakt een object aan van het type *StopWatch*. Vervolgens roept u de methode *Start()* aan om het 'timen' te beginnen. Na de procedure roept u de methode *Stop()* aan. Vervolgens kunt de tijd die verstreken is uitlezen met behulp van onder meer de eigenschap *ElapsedMilliseconds*. Wanneer u dezelfde *StopWatch* binnen een procedure vaker wilt gebruiken, kunt u door het aanroepen van de methode *Reset()* de begintijd weer op nul zetten.