Cor Ligthert is Microsoft MVP Visual Developer

Gevolgen internationalisatie onderschat

GLOBALISATIE IN .NET MAAKT ALLES EENVOUDIGER

Een van de meest onderschatte voordelen van het Windows besturingssysteem in combinatie met .NET, is de opgenomen internationalisatie (globalization). Wat we waarschijnlijk niet zouden verwachten van een Amerikaans bedrijf, is dat deze internationalisering krachtig opgenomen is in .NET. Een goed uitgewerkt aspect is de wijze hoe internationaal de datum en de tijd wordt gebruikt. In dit artikel gaat de auteur in op alle aspecten van datum en tijd.

Veel bedrijven denken bij de aanschaf van hun computerapparatuur niet direct aan een aspect als internationalisatie, maar wie zegt dat uw organisatie over een paar jaar niet ook een vestiging in China of de Verenigde Staten zal openen. Door hier vooraf al rekening mee te houden kan in de toekomst behoorlijk wat problemen voorkomen. Vooral omdat met een foutje in een verkeerde omrekening van een datum, de schade heel groot kan zijn. Denk bijvoorbeeld aan aandelen of andere tijdgebonden transacties. De eventuele schade zal ongetwijfeld op de ontwikkelaar worden verhaald. De datum en tijd zitten in de tak globalization. Bovendien zijn hierin de notatie van getallen, de kalender en verdere specifiek cultuureigen eigenschappen opgenomen; zie afbeelding 1.

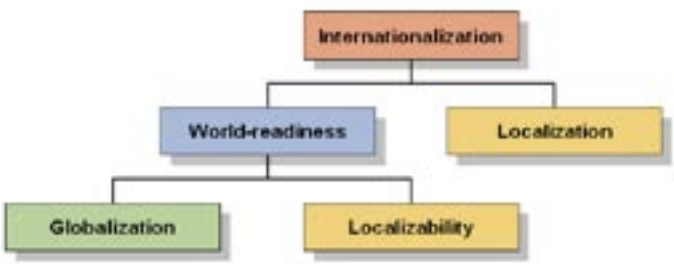
Notaties van de datum en de tijd

Over hoe de datum en de tijd worden genoteerd door mensen en computers bestaan veel misverstanden. Hier enige feiten om dit op te verhelderen:

- De datum en tijd worden niet uniform gebruikt over de gehele wereld
- De datum en tijd worden niet uniform geregistreerd in databases
- In databases van Microsoft worden de datum en de tijd onafhankelijk van de locale versie op een uniforme internationale vorm vastgelegd
- De datum en tijd worden op één wijze uniform geregistreerd in .NET
- De datum en tijd kunnen (zonder extra aanpassingen) maar op één formaat worden genoteerd en doorgegeven aan ASP.NET in webpages.

De vastlegging van de datum en tijd

Elke versie van Windows heeft de mogelijkheid van landinstellingen. Waren deze instellingen in Windows-versies 95 tot en met NT4 vanwege de aanwezigheid van bugs voor veel mensen een reden om eraf te blijven, tegenwoordig zijn ze uitgegroeid tot een krachtige feature. Zie afbeelding 2.
Let er op dat dit bij Windows 98/ME-systemen via de toetsenbordinstelling gaat, in deze systemen doet de landinstelling voor globalisatie niets. Voor dit artikel is alleen de landinstelling van belang. In .NET heet dit de 'taalcultuur'. Deze instelling beschrijft grotendeels hoe de gebruiker de datum, de tijd, de kalender en getallen presenteert. Persoonlijk vind ik het een slechte zaak als de leverancier voorschrijft hoe deze instellingen op een computer ingesteld moeten worden. Het is een voorkeurinstelling van de gebruiker en niet bedoeld voor de softwaremaker om eventuele bugs uit zijn programmatuur te halen. In de praktijk is er buiten de instelling van het land of specifieke gebruikersvoorkeuren, geen aanpassing nodig. Uiteraard kan je bij een Engelstalig besturingssysteem de instelling op Nederlands zetten als je niet de data in een Engelse of Amerikaanse notering wilt gebruiken.



Afbeelding 1. Internationalisatie

Afbeelding 2. Landinstellingen

In Visual Basic code

```
If IsDate("21 november 2005") Is True then
   Dim mijnDatum As DateTime = CDate("21 november 2005")
Else
   'Doe wat je moet doen als de datum fout is geschreven
End if
```

In C# code

```
//In C# kan dit zijn (ook bruikbaar in VB in de gebruikelijke VB vorm)
Try
   {
   DateTime mijnDatum = DateTime.Parse.("21 november 2005")
   }
Catch
   {
   //Doe wat je moet doen als de datum fout is geschreven
   }
```

Codevoorbeeld 1.

In Visual Basic code

```
Dim mijnDatum as DateTime = DateTime.ParseExact("17112005","ddMMyyyy", _
   Nothing)
```

In C# code

```
DateTime mijnDatum = DateTime.ParseExact("17112005","ddMMyyyy",null);
```

Codevoorbeeld 2.

In Visual Basic code

```
Dim myDatum as DateTime = DateTime.Parse("November 21 2005", _
   CultureInfo.CreateSpecificCulture("en-US"))
```

In C# code

```
DateTime mijnDatum = DateTime.Parse("November 21 2005",
   CultureInfo.CreateSpecificCulture("en-US"));
```

Codevoorbeeld 3.

Historisch interne vastlegging van de datum

In dit artikel wordt voor de beschrijving van de elementen de gangbare formaten voor de datum en tijd in .NET-programmatuur gebruikt.

- dd = dag in twee cijfers
- MM = maand in twee cijfers
- yy of yyyy = het jaar in twee of vier cijfers
- hh = uur
- mm = minuten
- ss = seconden
- tt = de Engelse pm- of am-toevoeging

De vastlegging van de datum is aanleiding geweest voor vele problemen, waarvan de ponskaart waarschijnlijk nog steeds een van de grootste veroorzakers is. Voor degene die dit niet weten: de ponskaart in het meest gangbare formaat had 80 posities. Dit betekende dat er zuinig moest worden omgegaan met posities. Vaak werd een datum daarom vastgelegd als MMdd of ddMM. Ging het over het jaar heen dan werd het ddMMy of yMMdd en als het om mensen ging ddMMyy of yyMMdd. De ponskaart is niet lang genoeg meegegaan om problemen op te leveren. Veel systemen zijn rechtstreeks geconverteerd naar tape, vervolgens naar disk en daarna naar databases. Dit heeft onder andere tot de bekende millenniumproblematiek geleid.

De huidige vastlegging van datum en tijd

Een oorzaak van veel misverstanden is vaak de wijze hoe Visual Studio met Visual Basic momenteel in de debugger, IntelliSense, de expressies - en in de tooltips - de datum representeert. Dit gebeurt uitsluitend in de USA-notatie. Waarschijnlijk zal dit in de volgende versie optioneel ook in Visual Basic in ISO-notatie worden weergegeven. In het .NET-systeem zelf is maar één wijze van vastlegging van de datum en tijd. Dit gebeurt in tikken die een tijdlengte van honderd nanoseconden beslaan. De startdatum van deze tikken is de eerste maand van het jaar 1 in de christelijke jaartelling. Dit is echter niet in het vastleggingformaat in databases. In databases worden verschillende formaten gebruikt. In SQL Server gebeurt dit bijvoorbeeld in tikken, maar dit zijn niet dezelfde als intern in .NET. Voor SQL Server bestaan twee formaten: de DateTime en de ShortDateTime. Vooral de notatie van de DateTime in SQL Server is interessant en geeft iets weer over de problematiek van het gebruik van de datum. De DateTime in SQL Server zijn tikken in eenheden van eenderde milliseconde. De startdatum hiervan is 1 januari 1753. Deze datum is ongeveer gelijk aan de invoering van de Gregoriaanse kalender in het Britse Imperium (3 Juliaans en 14 Gregoriaans september 1952). Het Britse Imperium was tegelijkertijd ook de laatste natie met een oorspronkelijk Romaanschristelijke cultuur die dit deed. (Rusland 1919, Griekenland 1923, Turkije 1928, China de officiële kalender 1949). Geen enkele datum geeft daarom enige precisie als we niet weten waar hij werd gebruikt en wat de bijbehorende invoeringsdatum van de Gregoriaanse kalender was. Op zich is 1-1-1753 dus een goede keus, al is het lastig als je bijvoorbeeld gebouwen vóór deze datum wilt dateren, iets wat bij de Amerikanen zelden voorkomt.

De SmallDateTime heeft een heel ander formaat. De datum en tijd zijn gescheiden terwijl de start 1 januari 1900 is. Het prettige is dat we in .NET normaliter - zolang we bezig zijn met DateTime-formaten naar DateTime-formaten - geen rekening hoeven te houden met de interne formaten. Bij string- naar DateTime-formaten en vice versa hoeven we er praktisch uitsluitend bij externe documenten of harde bestandsformaten rekening mee te houden.

De internationale notatie van de datum en tijd

Dit leidt tot ongelofelijk veel misverstanden, misschien niet direct bij mensen uit de Benelux, maar vooral bij inwoners uit de grotere landen is het vaak vermakelijk om te zien hoe men denkt dat de eigen notatie over de gehele wereld wordt gebruikt.

Er zijn 4 hoofdformaten. In volgorde van gebruik zijn dat:

- Het ISO-formaat
- Het formaat in landen waar een Europese taal wordt gesproken met uitzondering van onderstaande
- Het formaat van Engelstalige landen met uitzondering van de Verenigde Staten
- Het formaat van de VS en enige omringende landen, maar Canada weer niet.

Het ISO-formaat is het meest logische. Het is yyyy-MM-dd hh:mm:ss. We zien dit terug als een datum wordt weergegeven in (West)Arabische karakters in landen als China en India en in het verleden in veel communistische landen. De uren worden hier weergegeven met een klok van 24 uur. Het formaat dat wij en de meeste landen met Europese talen (met uitzondering van Engels) gebruiken is dd-MM-yyyy dd:hh:ss. Ook hier wordt de klok weergeven in 24 uur. Het formaat dat praktisch alle Engelstalige landen met uitzondering van de USA gebruiken, is dd-MM-yyyy dd:hh:ss tt. Bijna hetzelfde dus dat in alle landen met Europese talen wordt gebruikt met uitzondering van de tt die wordt weergegeven als pm of am, voor (post) middernacht of na (after) middernacht. De klok heeft hier dus 12 uur.

In de USA wordt een totaal ander systeem gebruikt. Niemand weet meer waarom het zo is ingevoerd en veel Amerikanen zien er eigenlijk ook het nut niet van in. Het formaat is MM-dd-yyyy dd:hh:ss tt. Waarbij pm of am wordt gehanteerd zoals het overal in de Engelstalige landen. Het systeem is in de USA ook onderhevig aan kritiek. De staat Nevada heeft bijvoorbeeld verordend dat op zijn officiële webpages alleen ISO-notaties mogen worden gebruikt. In Canada worden twee notaties gebruikt. Geen USA en Frans, maar standaard Engels en Frans.

Wat betekent dit voor .NET?

Als mensen er niet over zouden nadenken, dan zou dit met de juiste ingestelde landinstelling en het juiste gebruik van datumnotaties voor de programmatuur van string naar DateTime geen gevolgen hebben. De conversieformaten tussen een DateTime en een string en vice versa gaat via .NET probleemloos. Maanden worden zelfs in woorden vertaald naar digitale formaten en ook weer terug, zie codevoorbeeld 1. Zolang de string maar een datum bevat die voor de gebruiker en zijn landinstelling (culture) geldig is, dan zal dit naadloos worden omgezet naar het DateTime-formaat. Of dat een database-item is of een interne DateTime-waarde is daarbij niet van belang. Het resultaat in de processing vanuit de ILS is in beide programmeertalen gelijk voor alle drie getoonde methoden. (De Visual Basic 'IsDate' gebruikt intern ook de Try Catch.) Voor Visual Basic is als derde en voor C# als tweede optie de onderstaande methode nog beschikbaar:

```
mijnDatum = Convert.ToDateTime("21 november 2005")  (;)
```

Wat betekent dit voor .NET voor string naar DateTime vanaf documenten of databestanden?

Wordt een datumveld niet direct van een scherm, maar bijvoorbeeld als document aangeleverd, dan hebben we een ander probleem. We moeten dan het formaat weten waarin de datum is geschreven of weten in welke culture het is opgegeven. (Dit geldt dus ook voor aanlevering vanuit ouderwetse datastructuren die zijn afgeleid van de ponskaart.) Stel dat we weten dat het aangeleverde formaat 'ddMMyyyy' is, dan kan dit worden omgezet naar DateTime via DateTime.ParseExact; zie codevoorbeeld 2. Dit gaat goed zolang we digits gebruiken om de datum aan te geven. In het niet zo veelvoorkomende geval dat maandnamen gebruikt zijn, moet natuurlijk bekend zijn hoe deze naam wordt geschreven. Dit wordt verteld via de culture-parameter. Wordt een correcte datum (en tijd) doorgegeven volgens de aangegeven cultuur, dan is het eigenlijk weer heel eenvoudig. We dienen de cultuur op een of andere manier te weten te komen. Dit kan bijvoorbeeld via de database die aangeeft van welke cultuur de leverancier van de data gebruikmaakt. De cultuur kan ook zijn opgenomen in het document of via welke andere methode dan ook.

In Visual Basic code

```
Dim mijnDatum As DateTime = DateTime.ParseExact ("17 May 2005", _
   "dd MMMM yyyy", Globalization.CultureInfo.CreateSpecificCulture("en-Us"))
```

In C# code

```
DateTime mijnDatum =
DateTime.ParseExact("17 May 2005","ddMMyyyy",
Globalization.CultureInfo.CreateSpecificCulture("en-Us"));
```

Codevoorbeeld 4.

In Visual Basic code

```
Dim mijnDatum As DateTime = DateTime.Parse _
   ("17 May 2005", Globalization.CultureInfo.InvariantCulture)
```

In C# code

```
DateTime mijnDatum =
DateTime.Parse("17 May 2005", Globalization.CultureInfo.InvariantCulture);
```

Codevoorbeeld 5.

In Visual Basic code

```
Dim mijnDatum As New DateTime(2005, 11, 15)
Dim mynDatumString As String = mijnDatum.ToString("ddMMyyyy")
```

In C# code

```
DateTime mijnDatum = new DateTime(2005,11,15);
string mynDatumString = mijnDatum.ToString("ddMMyyyy");
```

Codevoorbeeld 6.

Voor het aangeven van de culture zijn verschillende systemen. In dit artikel gaan we uit van de standaard language-culture-notatie, zoals we die in .NET gebruiken . Hierbij betekent 'nl-NL' de Nederlandse (en Friese) cultuur 'nl-BE' Nederlands-Belgisch, 'en-UK' Verenigd Koninkrijk en 'en-US' Engelstalig Verenigde Staten; zie codevoorbeeld 3. In het geval van een geschreven maand en een correcte notatie van de datum kan de DateTime.Parse met als extra parameter de CultureInfo worden gebruikt. In deze situaties gaat het verder weer heel simpel. Het commando DateTime.Parse doet alles voor ons; zie codevoorbeeld 3. Staat het echter niet correct, dan komt weer de DateTime.ParseExact om de hoek kijken; zie codevoorbeeld 4. Een heel bijzondere CultureInfo is de InvariantCulture, die ik graag omschrijf als 'en-*'. Deze culture kan worden gebruikt bij Engelse maandnamen. Of de dag dan voor of achter de maand staat is niet van belang; zie codevoorbeeld 5. Voor het gebruik van culture-informatie direct in een programma kan een import/using van de namespace System.Globalization worden gebruikt.

Wat betekent dit voor .NET voor string naar DateTime vanaf de webbrowser in ASP.NET?

Jammer genoeg heeft de servervariabele geen aanduiding voor de door de browser gebruikte cultuur. Er is een attribuut voor de taalinstelling, maar dit kan bij het gebruik van data problemen geven. Neem bijvoorbeeld de datum 11-11-2005, dit kan in het Engels zowel 11 November als January 11 betekenen. Waarschijnlijk is de beste methode nog het voorschrijven van de notatie in de pagina. Vervolgens controleer je op de clientside in de browser of bij de Nederlandse notatie of de dag tussen 00 en 32 is, de maand niet tussen 00 en 13 en het jaar uit 4 cijfers bestaat. Of elke betere controle op grond van de maand. Voor het converteren van de gebruikte string kunnen de methoden worden gebruikt voor documenten en databestanden.

Vanaf een DateTime naar een string

Het is leuk dat de computer de datum en tijd op een uniforme wijze vastlegt, computers zijn geen mensen. Mensen zijn gewend aan al hun vormen die ze al jaren in hun omgeving gebruiken om te communiceren. Ook hier heeft .NET een hulpmiddel in de vorm van de IFormatProvider. Deze werkt zowel voor getallen, cultuur als voor de datum en tijd. In dit artikel houden we ons bij de datum en de tijd (DateTimeFormatIfo). Enkele voorbeelden uit het grote aantal dat beschikbaar is.

```
mijnDatumString = mijnDateTime.ToString(d)
```

resulteert bij de OS-setting in Nederlands tot een string als '15-01-2005'. Bij USA-setting zal dit '01-15-2005' zijn. Of

```
mynDatumString = mijnDateTime.ToString(D)
```

resulteert bij de OS-setting in Nederlands tot een string als '15 januari 2005'. Bij USA- setting zal dit 'January 15 2005' zijn.

Dit zou meer dan voldoende moeten zijn. Willen we bijvoorbeeld weer een datum exact in een document plaatsen, dan komt de pattern om de hoek kijken en kunnen we het volgende doen; zie codevoorbeeld 6. Het resultaat van codevoorbeeld 6 is de string '15112005'. De mogelijkheden met deze IFormatProvider schijnen eindeloos te zijn, maar probeer de vorm met het pattern te voorkomen. Dit kan leiden tot ongewenste misverstanden. De volgende truc kun je toepassen als je bij de DateTimePicker, waar uitsluitend de pattern kan worden opgegeven, een IProvider-presentatie nodig hebt zoals datum en tijd. Zet de culture-info eerst om naar string en gebruik die dan; zie codevoorbeeld 7.

```
mijnStringPattern = CultureInfo.CurrentCulture.DateTimeFormat.FullDateTimePattern (;)
```

Codevoorbeeld 7.

Eenvoudiger

Het gebruik van de globalisatiemogelijkheden in .NET maakt een aantal dingen veel eenvoudiger dan ze in het verleden waren en levert programmatuur van een hoogwaardiger niveau op. Twee zaken maken dat het nog niet is zoals het zou kunnen zijn. Computers en databases leggen hun tijden nog vast conform de klok van de tijdzone waarin ze staan. In bijvoorbeeld Nederland in CET. De webbrowser geeft geen cultuur af waarin de computer staat. Het zou prettig zijn als onder andere Microsoft hiervoor in de toekomst aandacht zou hebben, maar Microsoft kan dit niet alleen. Voor het overige zijn er geen argumenten meer te vinden om hard-coded data en tijden in programmatuur te gebruiken. De invulling van globalisatie staat nog zichtbaar in de kinderschoenen. Het zijn echter geen babyschoentjes meer, het zijn wel degelijk schoenen in de tienertijd. Tieners weten goed met de datum om te gaan. Via de juiste invulling van globalisatie kan de computer dit nu dus ook.

Cor Ligthert is waarschijnlijk een van de personen, die in Nederland het langst actief is in de ICT. Dit laatste in zowel de technische als de organisatorische aspecten. Vooral het zo goed doordacht zijn van .NET en de mogelijkheden hiermee een stap te maken, die alle beperkingen van de vorige eeuw achter ons laat, heeft zijn bijzondere interesse hiervoor gewekt. Samen met zijn "fellow" Ken Tucker (MVP) onderhoudt Cor een website. www.vb-tips.com gericht op het gebruik van Visual Basic .NET (ADO.NET, ASP.NET).