Mark Vroom is technisch directeur bij NedFox B.V.

XML-settings in Visual Basic

LEZEN EN SCHRIJVEN VAN EEN XML-CONFIGURATIEBESTAND MET HET .NET FRAMEWORK 2.0

In het .NET Framework 2.0 zijn behoorlijk wat veranderingen doorgevoerd op gebied van het opslaan van instellingen in een XML-configuratiebestand. In dit artikel gaat de auteur in op de vernieuwingen. Hij geeft inzicht in de verschillende manieren waarop er in het .NET Framework 2.0 omgegaan kan worden met het nieuwe ConfigurationManager-object.

Het grootste verschil tussen het .Net Framework 1.1 en 2.0 is dat de instellingen in versie 1.1 untyped werden opgeslagen, terwijl ze in versie 2.0 nu ook typed kunnen worden opgeslagen. Een van de voordelen hiervan is dat een opgeslagen numerieke waarde niet later als string wordt opgevraagd. Ook complexere .NET-objecten kunnen als instelling worden opgeslagen. Bovendien is door de strong typing (en de daarbij behorende collection-definities) Intellisense toegevoegd, zodat alle configuratie-instellingen eenvoudig en zonder fouten in de objectnaam opgevraagd kunnen worden. Ook is het nu mogelijk om via een enumerator alle instellingen te tonen, terwijl je in .NET Framework 1.1 zelf de namen van alle instellingen moest weten. Tot zover de beschrijving van het grootste verschil in settings tussen het .NET Framework 1.1 en 2.0.

AppSettings

In het .NET Framework 2.0 is een aantal 1.1-classes 'obsolete' geworden, waaronder de ConfigurationSettings-klasse uit de namespace System.Configuration. De klasse functioneert nog wel, maar Visual Studio 2005 geeft een 'warning' met het verzoek de ConfigurationManager-klasse te gaan gebruiken. Je kunt de overstap gedeeltelijk maken door alleen de ConfigurationManager-klasse te gebruiken, zonder ook direct over te stappen naar een andere manier van het opslaan van instellingen (via het typed model). Untyped 'AppSettings' worden door de ConfigurationManager-klasse namelijk nog steeds ondersteund. De volgende regel is afkomstig uit het .NET Framework 1.1:

```
System.Configuration.ConfigurationSettings.AppSettings("UserName").ToString()
```

In het .NET Framework 2.0 moet deze regel aangepast worden naar:

```
System.Configuration.ConfigurationManager.AppSettings("UserName").ToString()
```

Deze regel geeft uit een App.Config-bestand het 'Username'-element, maar afgezien van een nieuwe klassenaam is op deze regel geen sprake van nieuwe functionaliteit. Toch biedt het gebruik van de ConfigurationManager-klasse meer voordelen, ook al maak je geen gebruik van een type-collectie. In het .NET Framework 1.1 was het vrij lastig om aanpassingen in de instellingen op te slaan in een App.Config-bestand. De ConfigurationManager-klasse biedt een 'Save'- en een 'SaveAs'-methode waarmee instellingen definitief kunnen worden opgeslagen.

Codevoorbeeld 1 laadt de configuratie die gekoppeld is aan de lopende applicatie. Als parameter van de OpenExeConfiguration-method kan nog ingesteld worden welke soort configuratie moet worden geladen.

- ConfigurationUserLevel.None – Laad de configuratie voor alle gebruikers (application settings)
- ConfigurationUserLevel.PerUserRoamingAndLocal – Laad de lokale configuratie voor de huidige gebruiker (De naam is nog niet consistent, er is geen roaming-ondersteuning bij deze keuze)
- ConfigurationUserLevel.PerUserRoaming – Laad de roaming-configuratie voor de huidige gebruiker

Als er gekozen wordt voor PerUserRoamingAndLocal of PerUserRoaming wordt niet de app.exe.Config-file gebruikt, maar de user.Config. Deze file bevindt zich in principe in de Documents and Settings-map. In dit voorbeeld wordt 'None' gebruikt als ConfigurationUserLevel. Het uitvoeren van de Save()-method resulteert dan in het opslaan van settings naar de app.exe.Config-file. Aan de huidige configuratie wordt een userNameElement 'UserName' toegevoegd met als value 'Administrator'. Hierna worden de instellingen opnieuw opgeslagen. Al bestaande instellingen worden opnieuw opgeslagen, inclusief het nieuwe 'UserName'-element.

- ConfigurationSaveMode.Minimal – Alleen opslaan van die properties die verschillen van de originele waarden.
- ConfigurationSaveMode.Modified – Alleen opslaan van die properties die veranderd zijn ten opzichte van de originele waarden, ook als die inmiddels weer gelijk zijn aan de originele waarden.
- ConfigurationSaveMode.Full – Opslaan van alle properties, dit geeft een dump van alle .Net- instellingen en is voor normaal gebruik af te raden, aangezien er een bijzonder groot configurate-file ontstaat.

Het is mogelijk dat bij het opslaan een ConfigurationErrorsException ontstaat. De oorzaak hiervan kan een algemeen schrijfprobleem zijn, maar in veel gevallen is dit het gevolg van de configuratie die al door een andere gebruiker is veranderd. De

```
Dim config As Configuration = _
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None)
Dim userNameElement As KeyValueConfigurationElement = _
New KeyValueConfigurationElement("UserName")
config.AppSettings.Settings.Add(userNameElement)
userNameElement.Value = "Administrator"
config.Save(ConfigurationSaveMode.Minimal, False)
```

Codevoorbeeld 1.

tweede parameter 'ForceUpdateAll' zorgt er voor dat de configuratie ook wordt opgeslagen als er niets gewijzigd is. Na het opslaan is aan de configuratiefile een aantal regels toegevoegd; zie codevoorbeeld 2.

Let op, om gebruik te kunnen maken van de ConfigurationManager-klasse is het noodzakelijk om, behalve een 'Using'-statement, ook een reference in het project te maken naar de System.Configuration-assembly. Door die reference worden er ineens extra eigenschappen van de ConfigurationManager-klasse getoond.

app.Config of app.exe.Config ?

Instellingen in de app.Config-file worden bij het compileren in Visual Studio overgenomen in de app.exe.Config-file. Gebruik je Visual Studio niet (bijvoorbeeld in een deployment-situatie), dan heeft het geen zin om de app.Config-file te gebruiken. Het is daarom aan te raden om deze file bij deployment te verwijderen (dan wel niet mee te sturen), aangezien de aanwezigheid van dat bestand alleen maar tot verwarring leidt en het bestand toch niet uitgelezen wordt. Ik kwam hier achter, omdat het opslaan van instellingen naar een app.Config-file niet mogelijk bleek in het .NET Framework 2.0, ondanks de verschillende enthousiaste blogs waarin stond dat het opslaan van instellingen wel zou functioneren. Het uitvoeren van de eerder in dit artikel beschreven Save-method slaat gegevens automatisch en altijd op in de app.exe.Config-file (of user.Config). Ook bij het opvragen van instellingen wordt altijd diezelfde configuratie-file geraadpleegd. De reden voor de keuze om op te slaan in de hierboven genoemde config-bestanden is na enig onderzoek logisch gebleken. Microsoft heeft het de ontwikkelaar gemakkelijk willen maken door instellingen in de app.Config te kunnen aanbrengen vanuit de VS IDE. Die instellingen komen na het compileren en starten vanuit Visual Studio automatisch in de juiste config-files terecht. Dat kan een 'debug'- of een 'release'-configuratie zijn. Visual Studio zorgt ervoor dat beide config-files telkens automatisch worden bijgewerkt. Verderop in dit artikel bij 'Project settings' kom ik nogmaals terug op het app.Config-bestand.

```
<appSettings>
  <add key="UserName" value="Administrator" />
</appSettings>
```

Codevoorbeeld 2.

```
Imports System.Configuration

Public Class CustomSection : Inherits ConfigurationSection
  <ConfigurationProperty("UserName", DefaultValue:="Administrator", _
  RequiredValue:=True)> _
  Public Property UserName() As String
    Get
      Return CStr(MyBase.Item("UserName"))
    End Get
    Set(ByVal value As String)
      MyBase.Item("UserName") = value
    End Set
  End Property
End Class

' Define the above defined CustomSection in a CustomSectionGroup
Public NotInheritable Class CustomSectionGroup : Inherits _
    ConfigurationSectionGroup
  Public ReadOnly Property Custom() As CustomSection
    Get
      Return CType(Sections.Get("CustomSection"), CustomSection)
    End Get
  End Property
End Class
```

Codevoorbeeld 3.

Typed settings

Behalve het eenvoudig kunnen opslaan van untyped 'appSettings' voorziet het .NET Framework 2.0 in typed settings. Bovendien kunnen verschillende sectiegroepen worden aangemaakt, zodat instellingen te splitsen zijn in logische secties. Om te kunnen werken met typed instellingen moet eerst een of meer klassen zijn ontworpen waarin het settingsmodel wordt gedefinieerd. Optioneel kan gebruik worden gemaakt van een op ConfigurationSectionGroup gebaseerde klasse, verplicht is in ieder geval de op ConfigurationSection gebaseerde klasse. In die laatste bestaat een configuratiesectie met eventuele attributen. Onder deze sectie kunnen desgewenst nog configuratie-properties worden toegevoegd. Codevoorbeeld 3 beschrijft een ConfigurationSectionGroup en een onderliggende ConfigurationSection-klasse.

In codevoorbeeld 3 wordt een CustomSection gedefinieerd met een eigen attribuut 'UserName' van het type 'string'. Nu de CustomSection-klasse is gedefinieerd kan een UserName worden toegevoegd aan de actieve configuratie. In codevoorbeeld 4 wordt een nieuwe SectionGroup aangemaakt die is gebaseerd op de CustomSectionGroup-klasse zoals eerder weergegeven. De sectie krijgt als naam 'CustomGroup', maar mag natuurlijk elke naam hebben.

In codevoorbeeld 5 wordt een Section aangemaakt, gebaseerd op de CustomSection-klasse zoals eerder weergegeven. De sectie krijgt als naam 'CustomSection' maar mag ook in dit geval weer elke naam hebben. Het laatste blok code (codevoorbeeld 6) instantieert de CustomSection-klasse en vult die met de 'CustomSection'-section uit de configuratie-file via een cast. In het customSectionObj wordt de UserName-property vervolgens aangepast naar 'TestUser' en wordt de totaal aangemaakte configuratie opgeslagen. Het resultaat is een SettingsDemo.exe.config-file zoals weergegeven in codevoorbeeld 7 (gedeelte van bestand). In de SettingsDemo.exe.config is duidelijk te zien hoe de implementatie van 'strong typing' vertaald wordt naar een <section>-groep waar verwezen wordt naar het type 'System.Configuration.CustomSection'.

Mapped configurations

Er bestaan verschillende methods in de ConfigurationManager-klasse om configuraties te openen, waaronder de OpenExe-

```
Dim config As Configuration = _
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None)

Dim customSectionGroupObj As CustomSectionGroup
If config.SectionGroups("CustomGroup") Is Nothing Then
  customSectionGroupObj = New CustomSectionGroup()
  config.SectionGroups.Add("CustomGroup", customSectionGroupObj)
  customSectionGroupObj.ForceDeclaration(True)
End If
```

Codevoorbeeld 4.

```
Dim customGroup As ConfigurationSectionGroup
customGroup = config.SectionGroups.Get("CustomGroup")
Dim customSection As CustomSection
If customGroup.Sections.Get("CustomSection") Is Nothing Then
  customSection = New CustomSection()
  customGroup.Sections.Add("CustomSection", customSection)
  customSection.SectionInformation.ForceSave = True
End If
```

Codevoorbeeld 5.

```
Dim customSectionObj As CustomSection
customSectionObj = customGroup.Sections.Get("CustomSection") as _
  CustomSection
customSectionObj.UserName = "TestUser";
config.Save(ConfigurationSaveMode.Minimal)
```

Codevoorbeeld 6.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="SettingsDemo.Settings"
             type="System.Configuration.CustomSection,
             System, Version=2.0.0.0, Culture=neutral,
             PublicKeyToken=b77a5c561934e089" />
    </sectionGroup>
  </configSections>
  <userSettings>
  </userSettings>
  <applicationSettings>
  </applicationSettings>
  <CustomGroup>
    <CustomSection UserName="TestUser" />
  </CustomGroup>
</configuration>
```

Configuration, maar ook een OpenMachineConfiguration om een machine.config-file te openen. Bovendien bestaan beide methods ook in een 'Mapped' uitvoering. In de mapped versie van deze methods kan (moet) een pad en bestandsnaam worden opgegeven naar een .config-bestand dat moet worden ingelezen. Een mogelijke toepassing daarvan is een universele tool om configuraties uit te lezen. Via de OpenMappedExeConfiguration-method kan vervolgens elke configuratie ingelezen worden, en zelfs aangepast worden opgeslagen; zie codevoorbeeld 8.

Projectsettings

In de projecteigenschappen in Visual Studio zijn via een user-interface ook 'Project settings' op te geven. Die worden standaard niet opgeslagen in een configuratiebestand, tot het moment dat de Save-method wordt aangeroepen. Een setting 'WordOfWelcome' met als waarde 'HelloWorld', resulteert in extra regels in het configuratiebestand na een Save()-opdracht. Codevoorbeeld 9 illustreert het XML config-bestand na een Save-opdracht. Instellingen zoals de originele URL van webreferences worden door Visual Studio ook opgeslagen in de projectsettings en dus in de App.Config-file. Hetzelfde geldt voor 'Class only'-projecten waarbij nooit een app.exe.Config bestaat. Op basis van deze instellingen kan later voor 'Update Web Reference' gekozen worden.

ClickOnce

Wie al serieus gewerkt heeft met ClickOnce-applicaties heeft gemerkt dat de app.exe.Config-file bij veranderingen op de server altijd meegestuurd wordt naar de client. Het heeft dus geen zin om instellingen op te slaan in dit bestand vanuit ClickOnce-applicaties. Microsoft kiest er bij ClickOnce voor om dit bestand te laten gebruiken voor applicatie-'defaults'. Aangepaste instellingen moeten worden opgeslagen in het user.Config-bestand. Op het moment dat een applicatie-update wordt geïnstalleerd, zal het ClickOnce-systeem een merge uitvoeren op de oude config-files (zowel user als application) en het resultaat omzetten naar een nieuwe applicatie-directory. Door ConfigurationUserLevel.PerUserRoamingAndLocal of ConfigurationUserLevel.PerUserRoaming als parameter te gebruiken bij het openen van een configuratie, zal user.Config geopend worden en aanpassingen daarin worden opgeslagen na een Save()-commando. Meer informatie over ClickOnce in combinatie met applicationsetting-file is te vinden op msdn2.microsoft.com/en-us/library/ms135488.

Vooruitgang, mits voorbereid

Visual Studio 2005 in combinatie met het .NET Framework 2.0 biedt een behoorlijke vooruitgang op het gebied van het ophalen, aanpassen en weer wegschrijven van instellingen. Er dient wel het nodige aan voorbereiding te gebeuren in de vorm van het schrijven van een eigen klasse waarin de sectionGroups en sections worden gedefinieerd, maar het toepassen van deze technologie leidt

```
Dim config As Configuration = _
   ConfigurationManager.OpenMappedExeConfiguration("TestApp.exe.config", _
   ConfigurationUserLevel.None)
```

Codevoorbeeld 8.

```
<userSettings>
  <WindowsApplication1.Settings />
</userSettings>
<applicationSettings>
  <WindowsApplication1.Settings>
    <setting name="WordOfWelcome" serializeAs="String">
      <value>HelloWorld</value>
    </setting>
  </WindowsApplication1.Settings>
</applicationSettings>
```

Codevoorbeeld 9.

vervolgens wel tot een vermindering van potentiële bugs in je end-user-applicaties. Dit artikel is gebaseerd op een Win32-applicatie. Het gebruikmaken van settings vanuit een webapplicatie op basis van dit artikel is mogelijk, maar dan moet het ASP.NET process account wel voldoende rechten hebben op de verschillende config-files. Impersonation is de eenvoudigste manier om dat te regelen. Activeer daarvoor 'Integrated Windows Authentication' en geef in de Web.Config de mogelijkheid vrij om impersonation toe te passen.

```
<identity impersonate="true"/>
```

Mijn eigen code had ik in C# ontwikkeld, maar voor dit artikel was Visual Basic gewenst. Ik heb gebruikgemaakt van een online converter waarmee je C#-code kunt omzetten naar Visual Basic. Deze converter vind je op www.kamalpatel.net/ConvertCSharp2VB.aspx.

Mark Vroom is technisch directeur bij NedFox BV dat is gespecialiseerd in retail-automatisering. Mark is regelmatig spreker en schrijver voor Software Developer Network en enkele andere magazines. Sinds kort is hij bij SDN benoemd als groupleader van het VB.Net network. Zijn blog is te vinden op www.sdn.nl/cs/blogs/mark_vroom, e-mail mark@nedfox.nl

Referenties

Software Developer Network
www.sdn.nl

Retrieving configuration settings
quickstart.developerfusion.co.uk/QuickStart/aspnet/doc/management/retrieve.aspx

System.Configuration
msdn2.microsoft.com/en-us/library/2a1tyt9s(en-us,VS.80).aspx

Fredrik Normen's Blog
fredrik.nsquared2.com/viewpost.aspx?PostID=252