Jelle Druyts is .NET consultant

De Guidance Automation Toolkit

DE AUTOMATISERING VAN COMPLEXE PROCESSEN

Met behulp van de Guidance Automation Toolkit kun je herbruikbare code en patterns rechtstreeks in Visual Studio ter beschikking stellen aan ontwikkelaars. De toolkit is ontworpen om de integratie van herbruikbare stukken code in businessapplicaties te vergemakkelijken, en het stelt je in staat processen te automatiseren en ontwikkelaars door moeilijke procedures te leiden die anders tientallen bladzijden documentatie zouden vergen. In dit artikel leg ik uit hoe je zelf aan de slag kunt met deze technologie.

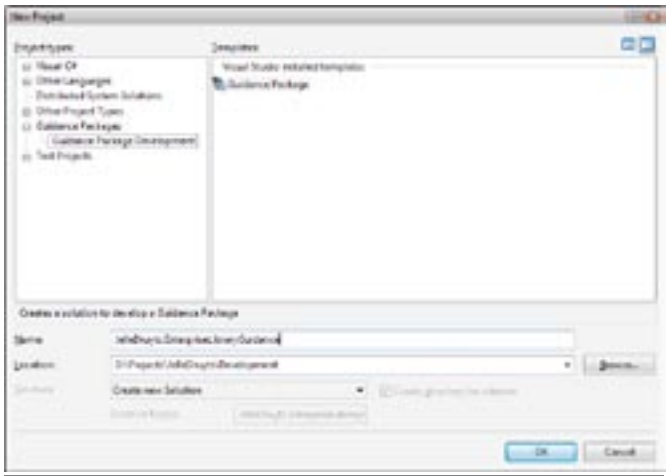
In nummer #13 van .NET Magazine kon je al kennismaken met de 'Web Service Software Factory', of ook 'Service Factory'. Dit is een toolkit die naadloos integreert met Visual Studio 2005 om je te helpen bij het ontwikkelen van servicegeoriënteerde applicaties, zowel door het gebruik van projecttemplates, wizards en codegeneratie, maar ook doordat het een hele hoop best practices 'ingebakken' heeft en ter beschikking stelt aan de ontwikkelaar. Dit soort guidance kan je ook gemakkelijk zelf automatiseren en publiceren met behulp van de Guidance Automation Toolkit, de software die aan de basis ligt van deze 'Service Factory' en alle andere softwarefactories (zoals de Smart Client Software Factory) die de Patterns & Practices-groep van Microsoft uitbrengt.

De bouwblokken

Een Guidance Automation Package of Guidance Package is een pakket van herbruikbare code, wizards, best practices en procesautomatisering dat je ter beschikking stelt aan ontwikkelaars. Om dit pakket te kunnen gebruiken, moeten zij de Guidance Automation Extensions (GAX) geïnstalleerd hebben; dat zijn de runtime-componenten die de integratie met Visual Studio verzorgen. Om zelf zo'n pakket samen te stellen, maak je gebruik van de Guidance Automation Toolkit (GAT), het gedeelte dat je helpt bij het ontwikkelen en installeren van deze Guidance Packages.

Aan de slag

Als je de Guidance Automation Toolkit hebt geïnstalleerd, krijg je een nieuw solution-template in Visual Studio, zoals je kunt zien in afbeelding 1.



Afbeelding 1. De standaard 'Guidance Package'.

Dit helpt je meteen op weg bij het bouwen van je eerste echte Guidance Package, want het genereert de benodigde projectstructuur en een hoop voorbeeldcode. Daarbij genereert het ook een setup-project zodat jij je Guidance Package onmiddellijk kunt installeren bij de ontwikkelaars voor wie het bedoeld is. Natuurlijk wil jij je package niet testen door ze voortdurend te installeren en deïnstalleren op je lokale machine. In plaats daarvan kan je meteen in Visual Studio rechtsklikken op het Guidance Automation-project en kiezen voor 'Register Guidance Package'. Dit zal je package registreren, zodat je het meteen kunt uitproberen in een nieuwe instantie van Visual Studio. Je kunt overigens het beste altijd je eerste instantie van Visual Studio openhouden om je guidance package te ontwikkelen; het testen kan je dan telkens in een nieuwe instantie doen. Het registreren zelf duurt even, maar als je niets veranderd hebt aan de integratie met Visual Studio (als je bijvoorbeeld geen nieuwe 'recipes' of nieuwe onderdelen in toolbars of contextmenu's toegevoegd hebt) kun je ook de optie 'Quick Register' gebruiken, wat aanzienlijk sneller gaat.

Templates & Recipes

Een Guidance Package bestaat doorgaans uit twee grote onderdelen: 'templates' en 'recipes'. Visual Studio Templates is de standaard manier waarop je in Visual Studio aangeeft hoe solutions, projecten en projectitems (bijvoorbeeld klassen of interfaces) aangemaakt moeten worden. Dit zijn de onderdelen die je in de 'Create New Project/Item'-dialoogvensters te zien krijgt. Ze worden gedefinieerd als .vstemplate-files. Guidance Automation Recipes zijn geautomatiseerde activiteiten die bestaan uit een serie instructies om een actie uit te voeren die de ontwikkelaar anders manueel zou moeten uitvoeren (bijvoorbeeld het aanmaken van een aantal projecten en het toevoegen van referenties). Ze worden gedefinieerd in een XML-bestand dat zich in de root van je Guidance Automation-project bevindt. Deze twee onderdelen worden met elkaar verbonden door in een template naar een recipe te verwijzen, zodat dit wordt uitgevoerd op het moment dat een template wordt 'uitgepakt'.

De Visual Studio-template

In codevoorbeeld 1 zie je een typisch voorbeeld van een solution-template, dat één enkel project aanmaakt en daarna de 'CreateApplicationBlock'-recipe uitvoert. Dit voorbeeld is een deel van een groter Guidance Package dat je toelaat om snel een Application Block te maken voor Enterprise Library (bezoek mijn blog voor meer details hierover).

In dit voorbeeld wordt onder meer duidelijk dat we metadata kunnen definiëren voor de template (de naam, het icoon, welk type project, et cetera) en wat er precies in de solution zit. In dit geval is dit één project, gebaseerd op een projecttemplate genaamd 'Runtime.vstemplate', maar de uiteindelijke projectnaam zal een naam krijgen die gebaseerd is op parameters van de recipe. In dit geval zal het project *$ApplicationBlockNamespace$.$ApplicationBlockName$* gaan heten, waarbij de ApplicationBlockNamespace en ApplicationBlockName de parameters zijn die de developer kan invoeren in de wizard die bij de 'CreateApplicationBlock'-recipe hoort. De link tussen beide zit in de Template-sectie op het einde van de solution-template.

De Guidance Automation recipe

De recipe zelf kan er als een eerste eenvoudige versie uitzien als in codevoorbeeld 2.
Zoals gezegd is een recipe een XML-bestand dat beschrijft welke argumenten er zijn en hoe de runtime aan die argumenten kan komen. In dit geval definiëren we de ApplicationBlockName en ApplicationBlockNamespace als vereiste argumenten, die telkens een converter toegewezen krijgen om ervoor te zorgen dat het wel degelijk geldige code- of namespace-identifiers zijn. Dan zien we de definitie van een wizard, waarbij één pagina dient om de twee velden op te vullen. Als we de Guidance Package nu registreren, kunnen we de 'Application Block'-solution kiezen in Visual Studio, zoals te zien is in afbeelding 2. Het wizardscherm dat bij de recipe hoort is te zien in afbeelding 3. Zoals je kunt zien worden verplichte velden aangeduid met een lichtgele achtergrondkleur en komt er een foutboodschap als de invoer ongeldig is (wat bepaald wordt door de geconfigureerde converter).

```
<VSTemplate Version="2.0" Type="ProjectGroup"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>Application Block</Name>
    <Description>
      Guidance Package that creates a new Enterprise Library
      Application Block.
    </Description>
    <ProjectType>CSharp</ProjectType>
    <SortOrder>91</SortOrder>
    <Icon>ApplicationBlock.ico</Icon>
    <CreateNewFolder>false</CreateNewFolder>
    <DefaultName>ApplicationBlock</DefaultName>
    <ProvideDefaultName>true</ProvideDefaultName>
  </TemplateData>
  <TemplateContent>
    <ProjectCollection>
      <ProjectTemplateLink
        ProjectName="$ApplicationBlockNamespace$.$ApplicationBlockName$">
        Projects\Runtime\Runtime.vstemplate
      </ProjectTemplateLink>
    </ProjectCollection>
  </TemplateContent>
  <WizardExtension>
    <Assembly>
      Microsoft.Practices.RecipeFramework.VisualStudio,
      Version=1.0.51206.0,
      Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a
    </Assembly>
    <FullClassName>
      Microsoft.Practices.RecipeFramework.VisualStudio.Templates.
      UnfoldTemplate
    </FullClassName>
  </WizardExtension>
  <WizardData>
    <Template xmlns="http://schemas.microsoft.com/pag/gax-template"
      SchemaVersion="1.0" Recipe="CreateApplicationBlock">
    </Template>
  </WizardData>
</VSTemplate>
```

Codevoorbeeld 1. Een eenvoudige Visual Studio-template.

Actions, value providers & code generation

Natuurlijk kun je met de Guidance Automation Toolkit veel meer dan alleen maar projecten genereren en wizardpagina's laten zien. Nadat een template 'uitgepakt' is kun je namelijk nog willekeurig veel actions uitvoeren, wat gewone .NET-klassen zijn die je toestaan om allerlei onderdelen van je proces te automatiseren. Hierin

```
<?xml version="1.0" encoding="utf-8" ?>
<GuidancePackage xmlns="http://schemas.microsoft.com/pag/gax-core"
  Name="JelleDruyts.EnterpriseLibraryGuidance"
  Caption="Enterprise Library Guidance"
  Description="Provides guidance around the creation of Enterprise
               Library Application Blocks"
  Guid="2cac5b9c-a04f-4a49-8a56-3ee5d63bd83f"
  SchemaVersion="1.0">
  <Recipes>
    <Recipe Name="CreateApplicationBlock">
      <Caption>Create a new Enterprise Library Application Block</Caption>
      <Arguments>
        <Argument Name="ApplicationBlockName" Required="true">
          <Converter Type="Microsoft.Practices.RecipeFramework.
            Library.Converters.CodeIdentifierStringConverter,
            Microsoft.Practices.RecipeFramework.Library" />
        </Argument>
        <Argument Name="ApplicationBlockNamespace" Required="true">
          <Converter Type="Microsoft.Practices.RecipeFramework.
            Library.Converters.NamespaceStringConverter,
            Microsoft.Practices.RecipeFramework.Library" />
        </Argument>
      </Arguments>
      <GatheringServiceData>
        <Wizard xmlns="http://schemas.microsoft.com/pag/gax-wizards"
          SchemaVersion="1.0">
          <Pages>
            <Page>
              <Title>Application Block Information</Title>
              <LinkTitle>Application Block</LinkTitle>
              <Help>
                Enter the Application Block name and namespace.
              </Help>
              <Fields>
                <Field ValueName="ApplicationBlockName"
                  Label="Application Block Name"
                  InvalidValueMessage="Must be a valid .NET
                  identifier (e.g. it shouldn't contain spaces
                  or special characters)." />
                <Field ValueName="ApplicationBlockNamespace"
                  Label="Application Block Namespace"
                  InvalidValueMessage="Must be a valid .NET
                  namespace Identifier (e.g. it shouldn't contain
                  spaces or special characters)." />
              </Fields>
            </Page>
          </Pages>
        </Wizard>
      </GatheringServiceData>
    </Recipe>
  </Recipes>
</GuidancePackage>
```

Codevoorbeeld 2. Een eenvoudige Guidance Automation Recipe.

kun je het objectmodel van Visual Studio ('EnvDTE') aanspreken om rechtstreeks op de projecten en de projectitems in te werken. Denk hierbij aan het leggen van projectreferenties, het genereren van klassen, het dynamisch toevoegen van projectitems, enzovoort. Verder kun je value providers aanspreken, waarmee je bepaalde argumenten van je recipe een waarde kunt geven. Dit zijn ook .NET-klassen, maar ze zijn er speciaal op gericht om waarden te berekenen die gebruikt moeten worden in de templates (bijvoorbeeld een post-build command of de naam van één of ander project), eventueel zelfs nog afhankelijk van de waarde van een ander argument.

Een andere belangrijke pilaar bij het schrijven van je eigen Guidance Automation Packages is de T4-engine (waarbij T4 staat voor Text Templates Transformation Toolkit). Deze engine stelt je in staat om met behulp van een ASP.NET-achtige syntax code te genereren, afhankelijk van argumenten in je recipe en ongeacht welke andere logica die je wilt implementeren. Een voorbeeld van zo'n T4-template zie je in codevoorbeeld 3, dat dient om een basis te leggen voor een Enterprise Library ConfigurationNode, gebaseerd op TargetNamespace- en NodeName-argumenten uit de recipe.

```
<#@ template language="C#" #>
<#@ assembly name="System.dll" #>
<#@ property processor="PropertyProcessor" name="TargetNamespace" #>
<#@ property processor="PropertyProcessor" name="NodeName" #>
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Text;

using Microsoft.Practices.EnterpriseLibrary.Common.Configuration;
using Microsoft.Practices.EnterpriseLibrary.Configuration.Design;

namespace <#= this.TargetNamespace #>
{
   internal sealed class <#= this.NodeName #> : ConfigurationNode
   {
   }
}
```

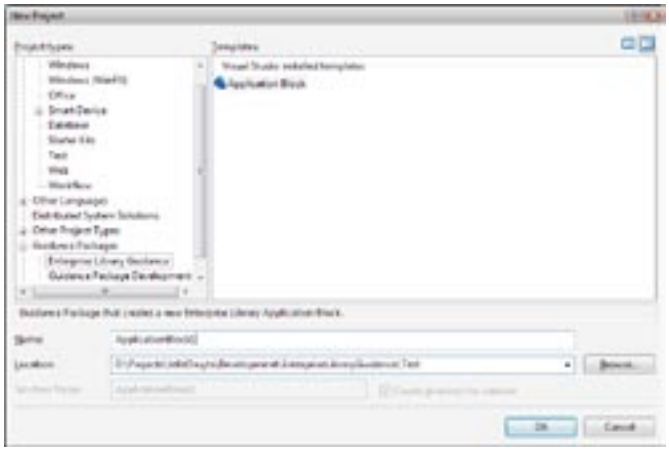
Codevoorbeeld 3. Een eenvoudige T4 template.

En daarna

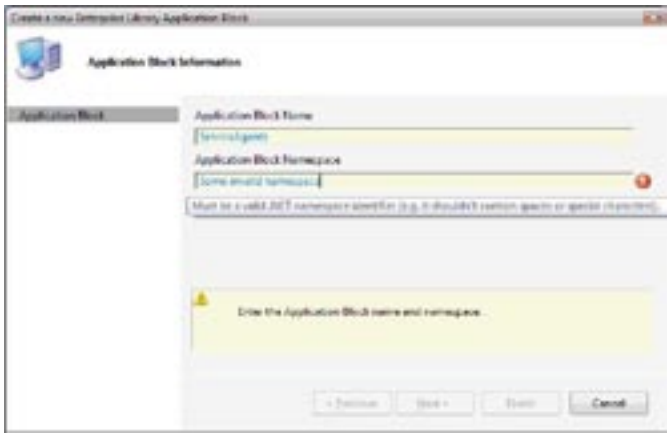
Ook nadat de solution of het project gecreëerd is, kun je nog steeds gebruikmaken van de Guidance Automation Toolkit: je kunt namelijk ook recipes associëren met bepaalde onderdelen van je gecreëerde project of met Visual Studio contextmenu's. Zo kun je bijvoorbeeld nog andere wizards uitvoeren door extra icoontjes te plaatsen in het 'Add New Item'-menu, waarmee telkens weer een recipe geassocieerd is. Hierdoor blijft de Guidance Package actief meewerken aan de taken van de ontwikkelaar.

Ook kun je Visual Studio-templates definiëren die daarna nog aan de gegenereerde solutions of projecten kunnen worden toegevoegd. Denk daarbij aan specifieke projecttypes (een smart client-project, een data access-project, een design time-project voor een Enterprise Library Application Block, et cetera) of gespecialiseerde klassen (een businesscomponent, een Enterprise Library ConfigurationNode, et cetera).

De beschikbare recipes en de geschiedenis van de uitgevoerde recipes worden ook getoond in de Guidance Navigator, een venster dat je een klare kijk geeft op de Guidance Package en wat er juist allemaal mee mogelijk is. Hierin kun je ook links verwerken naar guidelines en andere documenten, zodat de ontwikkelaar steeds de belangrijkste informatie binnen handbereik heeft.



Afbeelding 2. De Application Block Guidance Package.



Afbeelding 3. De wizard voor de Application Block Guidance Package.

Conclusie

De Guidance Automation Toolkit is een uitgebreid pakket dat je toelaat om complexe processen te automatiseren en om ervoor te zorgen dat guidelines en best practices gevolgd worden in je organisatie. Eén van de grote voordelen is dat je met het automatiseren van eenvoudige scenario's kunt beginnen en daarna incrementeel de complexiteit kunt opdrijven door steeds meer templates, logica en guidance toe te voegen. Bovendien vergt het creëren van zo'n guidance package niet eens zoveel tijd, maar het bespaart je gegarandeerd wel een hele hoop werk achteraf.

Referenties

Guidance Automation Toolkit: http://msdn.microsoft.com/vstudio/teamsystem/workshop/gat/

Web Service Software Factory: http://practices.gotdotnet.com/svcfactory

Smart Client Software Factory: http://practices.gotdotnet.com/scbat

Microsoft Patterns & Practices: http://msdn.microsoft.com/practices/

Enterprise Library 2.0: http://msdn.microsoft.com/library/en-us/dnpag2/html/EntLib2.asp

Jelle Druyts' Guidance Automation Series: http://jelle.druyts.net/CategoryView.aspx?category=Blog|Programming|.NET|GuidanceAutomation

Jelle Druyts is een .NET consultant die zich graag bezighoudt met de nieuwste Microsoft-technologieën en met het zoeken naar obscure bugs in Notepad. Hij schrijft over zijn ontdekkingen op zijn blog op http://jelle.druyts.net.