Marcel de Vries
Info Support -Nederland

Hét alternatief voor de uitrol van een .NET-applicatie

Installatie met Setup- & Deployment-projecten

Het op eenvoudige wijze overdragen van applicaties naar een test- of productieomgeving vergt aardig wat werk. Natuurlijk is het mogelijk zelf een installatie te maken met behulp van batchfiles en scripts, maar er is een betere manier: het uitrollen met behulp van de Microsoft Installer-technologie die in Visual Studio.NET beschikbaar is in de vorm van "Setup & Deployment"-projecten.

.NET is nu een tijdje uit en de eerste projecten zijn toe aan hun uitrol naar een test- of productieomgeving. Dit is het moment waarop men zich gaat afvragen hoe dit moet worden gedaan. Is hiervoor XCopy te gebruiken of is dat niet voldoende? Afhankelijk van het type applicatie zal dit moeten worden bekeken. Het uitrollen van applicaties met behulp van XCopy is alleen mogelijk voor Windows-applicaties (winforms) en voor het updaten van al bestaande webapplicaties. Voor een groot aantal scenario's zal XCopy niet werken. Neem bijvoorbeeld de uitrol van een multi-tier webapplicatie. Hierbij moet onder meer de "Virtual Directory" voor de webapplicatie worden aangemaakt, en moeten eventuele EnterpriseServices-componenten[1] worden geregistreerd in "component services" en geplaatst worden in de GAC (Global Assembly Cache). Vaak moeten dan ook nog eventlogs, performance counters en Windows Services geregistreerd worden, hetgeen veel meer eist dan een simpel Xcopy-commando. Visual Studio.NET biedt hiervoor een betere oplossing.

Visual Studio.NET "Setup & Deployment"-projecten

Met "Setup & Deployment"-projecten kun je installaties maken waarbij de omgeving een heleboel standaard installatiezaken regelt. Dit is uit te breiden met eigen scripts, classes en externe executables. Er zijn vijf projecttypen waaruit je kunt kiezen. De drie belangrijkste hiervan zijn:

- het Setup Project;
- het Web Setup Project;
- en het Merge Module Project.

Setup-projecten worden gebruikt voor de uitrol van Windows-applicaties, Web Setup-projecten voor de uitrol van webapplicaties en Merge Modules voor het uitleveren van bibliotheken en andere gedeelde componenten zoals het .NET Framework.

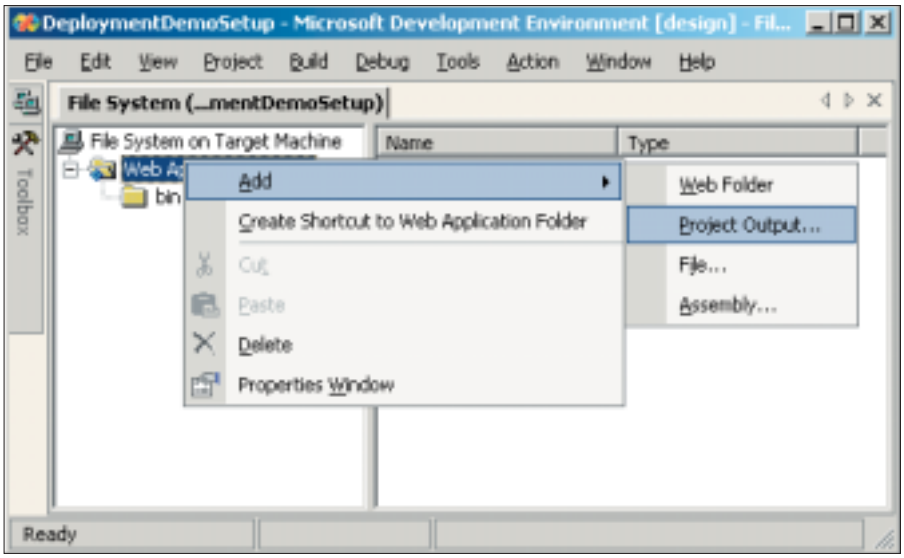
Een Deployment-project bestaat uit een aantal verschillende onderdelen waarvoor editors beschikbaar zijn, te weten:

- Het filesysteem van de doelmachine waarop de verschillende bestanden moeten worden geïnstalleerd.
- De registry van de doelmachine waar eventueel keys moeten worden toegevoegd voor de applicatie.
- De gebruikersinterface die tijdens de installatie wordt gebruikt voor interactie met de beheerder.
- Startcondities waaraan moet worden voldaan om de installatie te laten slagen. Bijvoorbeeld de conditie dat het .NET Framework aanwezig moet zijn op de doelmachine.
- Filetypes die op de doelmachine moeten worden geregistreerd, zodanig dat ze in de file-explorer worden geopend met het juiste programma.
- "Custom actions" die zelf kunnen worden geprogrammeerd en die niet standaard in het project aanwezig zijn.

Laten we beginnen met een Web Setup-project voor de uitrol van een simpele webapplicatie. Uitgangspunt is de aan-

1 EnterpriseServices worden gebruikt voor services zoals transacties, queuing, jit-activation en role based security. EnterpriseServices gebruiken hiervoor de services die geleverd worden door COM+.

wezigheid van een webproject in de solution met de naam DeploymentDemo. Na het toevoegen van een Web Setup-project aan de bestaande solution, wordt het project standaard geopend met de “Filesystem Editor”. In deze editor kun je de folders zien die op de doelmachine worden aangemaakt en in deze folders kunnen bestanden worden geplaatst die worden gekopieerd naar de doelmachine. Voor de uitrol van onze webapplicatie moeten we aangeven welke bestanden moeten worden geplaatst in de “Web Application Folder” (dit is de virtual directory op de webserver). Dit is mogelijk door met de rechter muisknop op de “Web Application Folder” te klikken en te kiezen uit het contextmenu voor de optie “Project Output” (zie afbeelding 1).



Afbeelding 1.

Er verschijnt een dialoogvenster met daarin een keuzemogelijkheid voor de te installeren bestanden. We kiezen voor twee mogelijkheden te weten “Primary Output” en “Content Files” voor de active configuratie. De “Primary Output” bevat alle gecompileerde bestanden die in de bin-directory van de webserver staan. De “Content Files” zijn alle bestanden die als build type “content” hebben in het web-project. Dit zijn onder meer .aspx- en .ascx-bestanden.

De instellingen van de webapplicatiefolder zijn ook te configureren. Zoals bijvoordbeeld beveiliging en naam van de virtual root, installation port (default port 80) en indexeren door indexserver.

Na de build, die enige minuten in beslag kan nemen, staan in de debug directory een aantal bestanden dat de installatie vormt. Als je setup.exe start zal er een wizard verschijnen die als leidraad dient voor de installatie.

Gebruik van custom actions

Stel dat de eerder beschreven webapplicatie een eigen eventlog gebruikt om fouten te registreren en componenten bevat die gebruik maken van EnterpriseServices voor transactiebewaking. Tijdens de installatie wil je dit eventlog aanmaken en de componenten registreren in de Component services. In de standaard opties van een deployment-project is er geen optie voor het aanmaken van een eventlog of het registreren van EnterpriseService-componenten. Om deze onderdelen van de applicatie toch te kunnen installeren is er een speciale optie beschikbaar om zelf gemaakte installatie-acties te definiëren. Deze optie is beschikbaar via de “Custom Actions Editor”. Een Custom Action is een extern proces (zoals een command file, vbs script of .exe-bestand) of een managed class die is afgeleid van System.Configuration.Install.Installer. Geregistreerde Custom Actions worden aan het eind van de installatie (na alle standaard stappen van de Microsoft Installer) aangeroepen. Een command file of vbs script werden ook in het verleden al gebruikt, dus laten we eens verder gaan kijken naar het gebruik van Custom Installer Classes.

Een Installer Class wordt door het installatieprogramma aangeroepen. De Installer Class heeft hiervoor virtuele methoden die worden aangeroepen voor de installatie en de de-installatie. De belangrijkste methoden die de Installer Class bevat staan in afbeelding 2.

De Installer Class kent een standaard implementatie van de install- en Uninstall-

```
public virtual void Install(IDictionary mySavedState )
public virtual void Uninstall(IDictionary mySavedState)
```

Afbeelding 2.

Tip voor “multi-homed” webserver

In het geval dat men op een “multi-homed” webserver moet uitrollen, moet men er rekening mee houden dat de website waarin moet worden geïnstalleerd niet aan te geven is tijdens de installatie. Een workaround hiervoor is om de betreffende website naar een extra port te laten luisteren (bijv. 5321) en vervolgens in de eigenschappen van de “virtual directory” ook dit portnummer te gebruiken. Door de verschillende sites een ander nummer te geven kan toch tijdens installatie de juiste site worden gekozen.

Installer class	Beschrijving
System.Diagnostics.EventLogInstaller	Gebruikt voor het installeren van een eventlog en event source
System.Messaging.MessageQueueInstaller	Gebruikt voor het aanmaken van messagequeues
System.Diagnostics.PerformanceCounterInstaller	Gebruikt voor het registreren van performance counters en categorieën
System.ServiceProcess.ServiceInstaller	Gebruikt voor het registreren van een Windows service
System.ServiceProcess.ServiceProcessInstaller	Gebruikt in samenwerking met de ServiceInstaller class voor het installeren van een Windows service

Afbeelding 3.

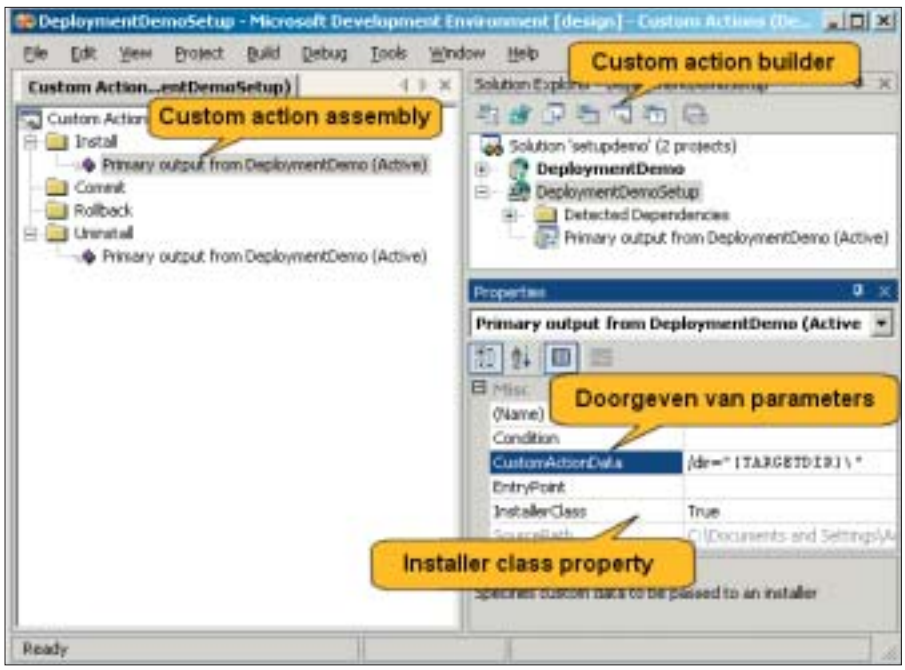
```
[RunInstaller(true)]
public class ProjectInstaller : System.Configuration.Install.
  Installer
{
  private System.Diagnostics.EventLogInstaller eventLogInstaller;
  public ProjectInstaller()
  {
     this.eventLogInstaller = new System.Diagnostics.
         EventLogInstaller();
     this.eventLogInstaller.Log = "DemoEventLog";
     this.eventLogInstaller.Source = "Hello world app2";
     this.Installers.AddRange( new
         System.Configuration.Install.Installer[] {eventLogInstaller});
  }
}
```

Afbeelding 4.

methoden. Deze implementatie bestaat uit het itereren en aanroepen van de beschikbare installers die zijn geregistreerd in de property installers. Bij het bouwen van een Installer Class kun je gebruik maken van deze standaard implementatie of je kunt een eigen implementatie van de Install- en UnInstall-methoden gaan schrijven. Bij gebruik van de standaard installer-implementatie zul je in de Constructor van de Installer Class één of meerdere andere Installer Classes registreren in de installers-collectie. Deze zullen dan vervolgens automatisch worden aangeroepen in de install van de baseclass.
Er is in het .NET Framework een aantal standaard Installer Classes beschikbaar die hierbij bruikbaar zijn (zie afbeelding 3).

Zoals uit afbeelding 3 blijkt is een Installer Class aanwezig voor het aanmaken van een eventlog. Voor de registratie van het eventlog moeten we dus een class afleiden van de Installer base class en de eventlog Installer registreren in de Constructor (zie het codevoorbeeld in afbeelding 4).

In het project kunnen we nu de “Custom Action” registreren met behulp van de “custom action builder”. (zie afbeelding 5) In de Custom Action Builder kan worden aangegeven welke assembly moet worden aangeroepen tijdens installatie.



Afbeelding 5.

In de custom action builder registreren we voor de Install en Uninstall acties de primary output van het Web project. In de eigenschappen van de Custom Action moet “InstallerClass” op “true” staan (zie afbeelding 5), zodat het setup-programma weet dat het gaat om een Installer Class in plaats van een executable of script. Het installatieprogramma zal hierdoor in de assembly zoeken naar Installer Classes (via reflection) en deze één voor één aanroepen.

Registratie van Enterprise-Services componenten

Misschien is het al opgevallen dat er voor EnterpriseServices geen standaard Installer Class aanwezig is. We zullen hiervoor dus zelf iets moeten maken. In het .NET framework is wel een class aanwezig voor het registreren van een EnterpriseService component. Dit is de System.EnterpriseServices.Registration-Helper class. Om deze te kunnen aanroepen, moet er opnieuw een Installer Class worden gemaakt en geregistreerd als custom action. In dit geval wordt dat een eigen implementatie van de Install- en Uninstall-methoden. De Registration-Helper heeft voor de registratie informatie nodig over de naam en locatie van de assembly die het component bevat. Een Setup-project bevat diverse properties die informatie bevatten over de huidige installatie. Deze properties worden gebruikt voor het overdragen van informatie binnen de Windows Installer. De locatie van de assembly is bekend binnen het Setup-project als een property met de naam [TARGETDIR].

Om de property door te geven aan de Installer Class kunnen we in de Custom Action Builder gebruik maken van de CustomActionData eigenschap van een custom action (zie afbeelding 5). Voor het overdragen van CustomActionData moet een argumentenlijst worden doorgeven volgens het volgende formaat: /<ParamName>=<Value>. In het geval van meerdere argumenten moeten deze gescheiden worden door een spatie. In het geval dat een argument spaties bevat (zoals in het geval van [TARGETDIR] kan optreden), moet de waarde wor-

```
public override void Install(System.Collections.IDictionary
  stateSaver)
{
string AppID = null; string TypeLib = null;
// ophalen van de doorgegeven CustomActionData
string Assembly = Context.Params["dir"]+"MyAssembly.dll";
RegistrationHelper rh = new RegistrationHelper ();
rh.InstallAssembly (Assembly, ref AppID, ref TypeLib,
          InstallationFlags.FindOrCreateTargetApplication);
    // Opslaan van de state voor gebruik in de-installatie
stateSaver.Add ("AppID", AppID);
stateSaver.Add ("Assembly", Assembly);
}

public override void Uninstall( System.Collections.IDictionary
  savedState)
{
try
{
  // Haal de state op die bij installatie is opgeslagen
  string AppID = (string)savedState["AppID"];
  string Assembly = (string)savedState["Assembly"];
  // Uninstall de app
  RegistrationHelper rh = new RegistrationHelper ();
  rh.UninstallAssembly (Assembly, AppID);
}
catch
{
// Onderdruk Eventuele excepties om te voorkomen dat de applicatie
// niet meer te de-installeren is
}
}
```

Afbeelding 6.

den geplaatst tussen dubbele quotes. Het doorgeven van de target directory doe je als volgt: `/dir="[TARGETDIR]\"`. De backslash is een escape-karakter zodat de laatste quote ook wordt meegenomen en dit niet wordt gezien als einde van de string.

De customActionData kan in de Installer Class worden uitgevraagd op een "Context Member"-variabele uit de Base Class.

Naast het registreren van het component bij installatie is het ook wenselijk deze te de-registreren tijdens de de-installatie. Hiervoor is informatie nodig die beschikbaar is in de install-methode. Voor het overdragen van een waarde tussen de install- en uninstall-methode kunnen we echter geen gebruik maken van een instance variabele in de Installer Class. Dit komt doordat tussen de installatie en de-installatie het proces wordt beëindigd. Bij de de-installatie wordt een nieuwe instantie gemaakt van de Installer Class waarop de uninstall-methode wordt aangeroepen.
In de install-methode wordt een parameter meegegeven die een implementatie van IDictonary bevat. Dit is de plaats waarin we variabelen kunnen opslaan die na de install-methode persistent worden opgeslagen op disk. Deze waarden worden weer gelezen bij het aanroepen van de uninstall-methode en meegegeven als parameter.

In het codevoorbeeld in afbeelding 6 is een implementatie weergegeven voor het registeren en de-registreren van een EnterpriseService-component.

De kracht van "Setup- en Deployment"-projecten

Voor het uitrollen van een .NET-applicatie voldoet in veel gevallen de Xcopy-methode niet. "Setup en Deployment"-projecten bieden hiervoor een alternatief. In dit artikel zijn maar een paar van de mogelijkheden uit "Setup en Deployment"-projecten beschreven. Er zijn nog veel meer mogelijkheden, waaronder het aanpassen van de gebruikersinterface van de installatie en het definiëren van startcondities. "Setup en Deployment"-projecten zijn bijzonder krachtig en bieden een prima middel om applicaties uit te rollen naar de test- en productie-omgeving.

Nuttige internetadressen

- Algemene informatie over setup & deployment
  http://msdn.microsoft.com/library/en-us/vsintro7/html/vbconIntroductionToDeployment.asp
- Informatie over custom actions
  http://msdn.microsoft.com/library/en-us/vsintro7/html/vbconTheCustomActionsEditor.asp
- Informatie over custom launch conditions
  http://msdn.microsoft.com/library/en-us/vsintro7/html/vxconLaunchConditionManagementInDeployment.asp