SharePoint oplossingen die je kunt onderhouden

Nieuwe keuzemogelijkheden in SharePoint 2010

John Sanders

SharePoint 2010 biedt veel nieuwe mogelijkheden voor het upgraden van een solution. Er zijn veel verschillende werkwijzen in gebruik waarvan sommige eigenlijk niet ondersteund worden. In dit artikel wordt gekeken naar de mogelijkheden en onmogelijkheden bij het upgraden van een solution in SharePoint en de nieuwe keuzes die SharePoint 2010 hierbij biedt.

Steeds meer ontwikkelaars bouwen applicaties in SharePoint. Het .Net development deel wordt door nieuwe ontwikkelaars snel opgepakt maar het deployen van een applicatie in SharePoint is een lastiger onderdeel waar vaak wat gewenning en ervaring voor nodig is.
Een SharePoint applicatie kan bestaan uit veel verschillende componenten. Niet alleen gecompileerde code maar ook aspx/ascx files en diverse xml (CAML) bestanden die beschrijven hoe de inhoud van lijsten eruit ziet, hoe een nieuwe site wordt opgebouwd en welke relaties al deze componenten met elkaar hebben. Al deze bestanden worden samengevoegd in een wsp bestand (de 'solution') dat in SharePoint kan worden geïnstalleerd. In Visual studio 2008 was de ondersteuning door Visual Studio nog beperkt en bestonden daarnaast veel verschillende tools en methodieken met als uiteindelijk doel deze wsp te produceren.
Gelukkig is dit in SharePoint 2010 al een stuk eenvoudiger geworden. Hierdoor zijn nieuwe ontwikkelaars sneller productief en worden veel vervelende fouten en extra uitzoekwerk vermeden.
Wanneer een eerste productieversie van de oplossing eenmaal in gebruik is genomen, komen vaak extra wensen of bugs aan het licht. Hierdoor is het vaak onontkoombaar dat er een upgrade nodig is, terwijl er al content aanwezig is.
'Geen punt' zou je zeggen; op de ontwikkelomgeving wordt de applicatie immers meerdere keren per dag uitgerold; bij iedere build/test. Er wordt vaak gedacht dat upgrades in de productieomgeving ook altijd worden ondersteund. Maar pas op. Je mag niet altijd alles upgraden.

Site Definitions

In de SDK wordt beschreven wat er wel (- en niet) ondersteund wordt voor site definitions. De belangrijkste paragraaf daarin is:

…Microsoft does not support changing or removing a site definition after sites have been created from it. Such changes may cause sites created from the definition (or created from Web templates that are based on the site definition) to stop working properly or may prevent the creation of new sites based directly, or indirectly, on the site definition…

De eerste versie van een Site Definition is dus altijd ook de laatste. Het is verstandig om ook in SharePoint 2010 vooraf goed na te denken over de site definition. Aan te raden is de site definition zelf zo leeg mogelijk te laten en zo veel mogelijk aanpassingen via features uit te rollen. Deze zijn namelijk makkelijker te updaten. De features kunnen aan de site definition worden gekoppeld via FeatureSiteTemplateAssociation of de site definition kan een dependency hebben op deze features.
Probeer ook om de site definition en de features in aparte solutionste deployen. Hierdoor kunnen features worden bijgewerkt zonder met de update ook de site definition te raken.

Solutions

De unit of deployment is de solution. Een nieuwe versie van een wsp kan op twee manieren worden uitgerold. Een herinstallatie of een upgrade.
Voor we dieper ingaan op het verschil tussen een upgrade en herinstallatie is het goed om in gedachten te houden dat een wsp geen versienummer heeft. SharePoint ziet daarom iedere aangeboden wsp met een bekende SolutionId als een nieuwe versie van een eerdere package. Wanneer dus per ongeluk een oude versie van de solution wordt geïnstalleerd, zal deze gewoon worden uitgerold.
Dit blijft ook in SharePoint 2010 het geval. Maar er is op Feature niveau nu een goede manier om met versienummers om te gaan. Voor de assemblies in een solution is het mogelijk om zelf bij iedere nieuwe installatie de AssemblyFileVersion aan te passen (in assembly.cs of via properties van Visual Studio solution project). De AssemblyFileVersion is na compileren zichtbaar in de eigenschappen van de dll en is alleen 'ter informatie'. Hij wordt niet gebruikt bij de aanroep van de assembly. Zolang de AssemblyVersion niet veranderd is, blijft wat .NET betreft de versie ongewijzigd en is op de server te zien welke versie van de code is gedeployed. Helaas is de AssemblyFileVersion niet zichtbaar via de Admin UI van SharePoint en is er toegang nodig tot het filesysteem/GAC op de server zelf om deze te kunnen bekijken. Vooral erg handig voor testomgevingen dus, maar ook handig wanneer een externe

beheerder de componenten op een productieserver installeert. Deze kan dan na de uitrol de assembly versies verifiëren.

Herinstallatie

Een herinstallatie bestaat uit het volledig verwijderen en opnieuw installeren van de solution. Een herinstallatie is ingrijpender dan een upgrade. De oude versie van de solution wordt onder de site vandaan gehaald (Uninstall-SPSolution/Remove-SPSolution) waarna de nieuwe terug wordt geplaatst (Add-SPSolution/Install-SPSolution). Na dit proces moeten alle features die in de database op de site waren geactiveerd weer 'passen' op de nieuw geïnstalleerde solution. Doordat features in de verwijderde solution nog in gebruik zijn, werkt de site tijdens de herinstallatie niet. Een risico hierbij is ook dat wanneer de herinstallatie niet lukt, de solution afwezig is en dat de site niet werkt. Verder worden tijdens dit proces in sommige scopes de events FeatureUninstalling en FeatureInstalled opnieuw aangeroepen. Dit kan gevolgen hebben wanneer maatwerk acties die in het event geprogrammeerd staan opnieuw (en kort na elkaar) worden uitgevoerd. Ook kan het zijn dat je features op de web applicatie scope opnieuw moet activeren.

Upgrade

Een upgrade is de snelste en simpelste optie. Tijdens de upgrade naar een nieuwe versie bewaart SharePoint de oude versie van de wsp. Wanneer de upgrade mislukt, wordt automatisch de oude solution weer teruggeplaatst. In een solution upgrade is echter niet alles toegestaan. In SDK staat het volgende bij topic 'Solution upgrade':
Solution upgrade can only be used to replace files. You can add new files in a solution upgrade and remove old versions of the files, but you cannot install Features or use Feature event handlers to run code for Feature installation and activation. The following operations are not supported in solution upgrade.

- *Removing old Features in a new version of a solution.*
- *Adding new Features in a solution upgrade.*
- *Updating or changing the receiver assembly for existing Features in a new version of a solution.*
- *Adding or changing Feature elements (Element.xml files) in a new version of a solution.*
- *Adding or changing Feature properties in a new version of a solution.*
- *Changing the ID or scope of old Features in a new version of a solution.*
- *Removing Feature elements (Element.xml files) in a new version of a solution.*
- *Removing Feature properties in a new version of a solution.*

De mogelijkheden zijn dus beperkt. Wat blijft er over? Bestanden vervangen, zolang je maar geen nieuwe zaken toevoegt of verwijdert. Assemblies in features mag je ook aanpassen, behalve de assembly die de events van de feature zelf afhandelt.
Dit levert direct een interessant punt op. In SharePoint 2010 bestaat een nieuw FeatureUpgrading event waar we straks in meer detail op ingaan. Dit event moet net als alle feature events geïmplementeerd worden in de feature receiver assembly. Volgens de huidige SDK mag deze assembly echter niet via een solution upgrade vervangen worden. Om de nieuwe upgrade mogelijkheden te gebruiken mag je de nieuwe versie van de wsp dus eigenlijk niet via solution upgrade installeren, herinstallatie is nodig omdat de feature receiver assembly ook moet worden vervangen. Hopelijk verandert dat in een volgende versie van de SDK. Een upgrade voelt toch meer als een upgrade als je de solution ook zo mag uitrollen.

ActivationDependencies

Wanneer een applicatie bestaat uit meerdere SharePoint solutions (wsp) is het bij het uitrollen altijd belangrijk te controleren of de juiste solutions zijn geactiveerd. SharePoint 2010 biedt nu de mogelijkheid ook op solution niveau een ActivationDependency aan te brengen.

```
Solution 1
<?xml version="1.0" encoding="utf-8"?>
<Solution xmlns="http://schemas.microsoft.com/sharepoint/"
SolutionId="b5f8ce94-3708-425a-9552-0564dd758b81" Title="Article
ContentType V1" SharePointProductVersion="14.0">
  ... </Solution>

Solution 2 is afhankelijk van solution 1
<?xml version="1.0" encoding="utf-8" ?>
<Solution xmlns="http://schemas.microsoft.com/sharepoint/"
SolutionId="62e0dab4-c263-4742-b85b-6a8f24a1bbf7" Title="Article
WebPart V1" SharePointProductVersion="14.0">
 <ActivationDependencies>
<!-- This webpart soln depends on the ContentType soln -->
<ActivationDependency SolutionId="b5f8ce94-3708-425a-9552-
0564dd758b81" SolutionTitle="ContentType Solution" />
</ActivationDependencies>
 ...</Solution>
```

Hou er wel rekening mee dat de ActivationDependency alleen wordt gecontroleerd bij het activeren- en niet bij het retracten van een solution. Je kunt dus zonder waarschuwing een solution retracten waar nog een andere solution afhankelijk van is.

BindingRedirect in manifest.xml

In de huidige praktijk blijven veel in SharePoint uitgerolde assemblies eeuwig op versie 1.0.0.0 staan. Een van de redenen hiervoor is dat het redirecten van oude versies vaak fout gaat. Wanneer een nieuwe versie van een assembly is uitgerold wordt de oude versie overschreven. Bestaande componenten (zoals webparts op een pagina) kunnen de oude assembly niet vinden via de strong name. Een BindingRedirect in web.config zorgt ervoor dat calls naar assemblies van versie 'OldVersion' worden omgeleid naar de in de solution uitgerolde assembly. In SharePoint 2010 is het mogelijk in de solution manifest een BindingRedirect op te nemen.

```
<Solution...
<Assemblies>
<Assembly Location="Article.Display.dll" DeploymentTarget="Global-
AssemblyCache">
<BindingRedirects>
<BindingRedirect OldVersion = 1.0.0.0/>
</BindingRedirects>
</Assembly>
</Assemblies>
... </Solution>
```

De BindingRedirect past bij het activeren van de solution de web.config aan zonder dat hier extra acties voor nodig zijn. Hierdoor wordt het makkelijker om de assemblyversie op te hogen wanneer een solution wordt aangepast. Houd er wel rekening mee dat dit alleen voor de assembly geldt, andere bestanden worden niet geredirect. Als je meerdere versies van webparts of aspx pagina's tegelijk wil laten bestaan moet je dat zelf regelen.

Features

Eerder is al gebleken dat Features de belangrijkste bouwstenen van een solution zijn. Het kunnen upgraden van features is dus ook erg belangrijk want één ding weet je als ontwikkelaar zeker: 'Things Change'. In een nieuwe versie van een feature is het moge-

lijk om bestanden en assemblies in de feature te vervangen, te verwijderen of toe te voegen en om feature properties toe te voegen of te wijzigen. Dat is heel wat, maar ook in features mag niet alles. In de SDK staat bij topic 'Upgrading Features' dat de volgende acties niet ondersteund worden:

Removing old Features in a new version of the solution
Changing the ID and/or scope of old Features in a new version of the solution.
Removing Feature elements or element XML files in a new version of the solution.
Removing Feature properties in a new version of the solution.

Nieuw in 2010 features - Versies

In het feature schema zijn nieuwe elementen beschikbaar die specifiek voor upgrades zijn bedoeld. Naast het feature schema is ook het objectmodel voor features uitgebreid en zijn er nieuwe eventhandlers toegevoegd, waardoor tijdens de upgrade meer mogelijkheden ontstaan. In het feature schema wordt nu het version attribuut gebruikt om aan te geven wat het versienummer van een feature is. Dit is eenvoudig aan te passen in de feature properties. Leeg laten betekent versie 1.0.0.0. Dit feature versienummer heeft overigens niets te maken met de assembly versie. Een assembly kan immers voor meerdere features worden gebruikt.

ActivationDependency – minimale versie

Het versie attribuut van een al geïnstalleerde feature kan worden gebruikt in de ActivationDependency node van een andere feature. Er kon bij activeren van een nieuwe feature al worden aangegeven welke features er minimaal geactiveerd moeten zijn. Nu kan ook worden aangegeven welke versie van deze features minimaal vereist is.

```
<?xml version="1.0" encoding="utf-8"?>
<Feature xmlns="http://schemas.microsoft.com/sharepoint/"
Title="Article.Display.Feature" Description="Article WebPart V1"
Id="641b0701-b490-48fb-a567-c42fbecf294b"
ReceiverAssembly="Arcticle.Display, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=a76443cf0aaa6faf"
ReceiverClass="Arcticle.Display.Features.Article.Display.Feature.
ArticleDisplayEventReceiver" Scope="Site" Version="1.0.0.0">
<ActivationDependencies>
<!-- Depends on Content type feature-->
<ActivationDependency FeatureId="fe6d7f5c-1416-4ecd-bcd0-
a9359f3c6264" MinimumVersion="1.0.0.0" />
</ActivationDependencies>
...
```

De feature xml wordt grotendeels door Visual Studio ingevuld. Wanneer je deze xml wilt uitbreiden kan dat door in de feature de gegenereerde manifest te bekijken. Via 'edit option' kan je de xml in de xml editor aanpassen.

Het upgraden van Features – UpgradeActions

Een nieuw element in het feature schema is UpgradeActions. Hierin kunnen acties worden toegevoegd die worden uitgevoerd wanneer je een feature upgrade naar een nieuwe versie. Deze acties kunnen per versie worden gegroepeerd in een VersionRange element. Dit element heeft attributen BeginVersion en EndVersion. De feature upgrade acties worden alleen uitgevoerd wanneer de bestaande versie van de feature gelijk of hoger is dan BeginVersion en lager dan EndVersion. Na het uitvoeren van de upgrade actions is de versie van de feature gelijk aan de versie die je hebt uitgerold. De waarde voor EndVersion mag dus de versie zijn van de Feature die wordt uitgerold.
Een van de acties is die tijdens feature upgrade kunnen worden uitgevoerd is ApplyElementManifests. Hiermee kunnen tijdens de upgrade nieuwe bestanden of elementen aan de feature worden toegevoegd zonder dat de feature opnieuw geactiveerd hoeft te worden.

CustomUpgradeAction en het FeatureUpgrading event

In het CustomUpgradeActionelement kan worden aangegeven welke maatwerk acties er tijdens de feature upgrade moeten worden uitgevoerd.

```
<?xml version="1.0" encoding="utf-8"?>
<Feature xmlns="http://schemas.microsoft.com/sharepoint/"
Title="Article.Display.Feature" Description="Article WebPart V2"
Id="641b0701-b490-48fb-a567-c42fbecf294b"
ReceiverAssembly="Arcticle.Display, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=a76443cf0aaa6faf"
ReceiverClass="Arcticle.Display.Features.Article.Display.Feature.
ArticleDisplayEventReceiver" Scope="Site" Version="2.0.0.0">
<UpgradeActions ReceiverAssembly="Arcticle.Display, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=a76443cf0aaa6faf"
ReceiverClass="Arcticle.Display.Features.Article.Display.Feature.
ArticleDisplayEventReceiver">
<VersionRange BeginVersion="1.0.0.0" EndVersion="2.0.0.0">
<CustomUpgradeAction Name="UpgradeWebToV2">
<Parameters>
<Parameter Name="AppendWebTitle">Upgraded to V2</Parameter>
</Parameters>
</CustomUpgradeAction>
</VersionRange>
</UpgradeActions>
...
```

De inhoud van het CustomUpgradeAction element wordt gebruikt in het nieuwe FeatureUpgrading event. In dit event kan code worden uitgevoerd om oude versies van de feature te upgraden naar de huidige versie. In de handler voor dit event kan je extra upgrade acties uitvoeren. Welke acties dat moeten zijn, kan je per VersionRange beschrijven in hetUpgradeActions element. (De handler kan je toevoegen door in Visual Studio rechts te klikken op de feature.)

```
public override void FeatureUpgrading(SPFeatureReceiverProperties
properties, string upgradeActionName, System.Collections.Generic.
IDictionary<string, string> parameters)
{
switch (upgradeActionName)
 {
  case "UpgradeWebToV2":
   SPSite site = (SPSite)(properties.Feature.Parent);
   SPWeb web = site.RootWeb;
   string appendText = parameters["AppendWebTitle"];
   web.Title = web.Title + appendText;
   web.Update();
   break;
  default:
   break;
 }
}
```

Dit event kan meerdere keren worden aangeroepen, één keer voor ieder CustomUpgradeAction element in de xml. Het argument upgradeActionName krijgt de waarde uit het Name attribuut van het CustomUpgradeAction element en het parameters argument wordt gevuld met de waarden uit het Parameterselementen in CustomUpgradeAction.
In de handler voor het event kan je door middel van een switch statement verschillende codeblokken opnemen voor de acties die bij de verschillende CustomUpgradeAction Names horen.

Feature Upgrade Query Object Model

Via het objectmodel kan worden bekeken welke versie van features aanwezig is in een bepaalde scope. De QueryFeatures method is beschikbaar op objecten van het type SPWebService (Farm), SPWebApplication, SPContentDatabase en SPSite.
Het uitvoeren van een upgrade gebeurt niet automatisch!
Bij installatie van de solution worden bestanden en schema's vervangen maar worden de feature upgrade actions nog niet uitgevoerd. De versie van een feature verandert ook nog niet. Dit gebeurt alleen wanneer per scope de upgrade wordt uitgevoerd door Feature.Upgrade aan te roepen
Houd er rekening mee dat Feature.Upgrade (en dus FeatureUpgrading) alleen aangeroepen kan worden vanuit custom code of script. Er is geen kort powershell- of stsadm commando dat dit doet.
Het versienummer van een feature is niet zichtbaar via de web UI, waardoor het moeilijk is te zien welke versie van een feature op een site wordt gebruikt. Het volgende stukje code werkt (vanuit een commandline applicatie) alle versies van een feature op een Site collection scope bij.

```
SPSite site = new SPSite(args[0]);
foreach (SPFeature feature in site.Features)
{
if ((feature.DefinitionId == new Guid("641b0701-b490-48fb-a567-c42f-
becf294b"))&&
(feature.Version != feature.Definition.Version))
 {
  if (feature.Upgrade(false) != null)
 {
Console.WriteLine("Success ");
  }
 }
}
```

Door aan Feature.Upgrade de waarde false mee te geven, wordt de upgrade afgebroken als er tijdens de upgrade ergens een fout optreedt. In dat geval wordt als returnwaarde een collection met de opgetreden exceptions teruggegeven. Wanneer je true meegeeft aan de Update methode worden eventuele fouten genegeerd en wordt het versienummer gewoon opgehoogd.

Content types in features

Zoals eerder al genoemd is het upgraden van content types een verhaal apart.
Net als bij site definitions is de eerste versie van een content type dus ook de laatste. Ook hier is dat een reden om content types zoveel mogelijk in een aparte solution te deployen. Maar hoe pas je ze dan aan? In het verleden gebeurde dat via code en was er veel discussie over het maken van content types; vanuit code of vanuit CAML? De mogelijkheden vanuit CAML zijn in Visual Studio 2010 en SharePoint 2010 flink verbeterd. Zo is het nu mogelijk om bij een upgrade aan een content type dat vanuit CAML is gemaakt ook vanuit CAML kolommen toe te voegen. Veel andere wijzigingen, zoals het aanpassen van bestaande velden moeten nog steeds vanuit code.

AddContentTypeField element

Het AddContentTypeField element kan samen met het ApplyElementManifestselement gebruikt worden om tijdens feature upgrade een nieuw field te maken en deze aan een bestaand content type toe te voegen. In onderstaand voorbeeld wordt het NetMag.Summary field toegevoegd aan het Content.NetMag.ContentType. Het nieuwe veld wordt ook aan bestaande lijsten van het content type toegevoegd.

```
<?xml version="1.0" encoding="utf-8"?>
<Feature xmlns="http://schemas.microsoft.com/sharepoint/"
Title="Content.NetMag.ContentType.Feature" Id="afbe46f5-50b0-4c1c-
a542-8cd821607961" Scope="Site" Version="2.0.0.0">
<UpgradeActions>
<VersionRange BeginVersion="1.0.0.0" EndVersion="2.0.0.0">
<ApplyElementManifests>
<ElementManifest Location="NetMag.Summary\Elements.xml" />
</ApplyElementManifests>
</VersionRange>
<AddContentTypeField ContentTypeId="0x0101002c377367d6d642e5b215b45
d7c222da3" FieldId="{70B45039-E381-4CE8-889D-296B387F3A4A}"
PushDown="TRUE" />
</UpgradeActions>
<ElementManifests>
<ElementManifest Location="Content.NetMag.ContentType\Elements.
xml" />
<ElementManifest Location="NetMag.Summary\Elements.xml" />
</ElementManifests>
</Feature>
```

Het AddContentTypeField element heeft ook een PushDown attribuut. Hiermee wordt het veld ook toegevoegd op subsites.
Ook deze wijziging wordt niet automatisch actief wanneer je de feature opnieuw installeert. Je moet het nieuwe veld per scope upgraden (via de Feature.Upgrade methode) want misschien is de nieuwe kolom niet voor alle sites nodig of gewenst. Dit levert ook direct een aandachtspunt op. Een volgende upgrade naar v3 van het content type moet nog steeds upgrades vanaf v1 kunnen uitvoeren want niet alle sites hoeven immers de upgrade naar v2 al gehad te hebben. De feature xml dus niet 'opschonen' wanneer je aan V3 van de feature werkt.

Conclusie

Het upgraden van een SharePoint solution is iets waar je goed bij moet blijven nadenken. Dit begint al bij het opzetten van je Visual Studio solution, waarmee je je oplossing in meerdere WSP's opdeelt. SharePoint 2010 biedt veel nieuwe mogelijkheden voor upgrades, maar veel van de bestaande aandachtspunten blijven nog wel van kracht. Vooral het FeatureUpgrading event is erg interessant omdat je hierbij per scope vanuit code upgrade acties uit kan voeren. Wel jammer dat de upgrade zonder extra tools niet vanuit de UI kan worden uitgevoerd.
Upgraden blijft een spannende bezigheid waarover veel meningen, methoden en best practices bestaan. Ook met SharePoint 2010 zal dit ongetwijfeld weer veel voer voor discussies opleveren.

John Sanders, is SharePoint practice lead en software architect bij Logica.