Michiel Lankamp en Willem Boeré zijn ontwikkelaars bij Winvision

Applicatieontwikkeling in SharePoint 2007

OVERVIEW VAN NIEUWE DEVELOPMENT EN DEPLOYMENT-METHODIEKEN

De manier van ontwikkelen met SharePoint is gewijzigd met de nieuwe versie, Windows SharePoint Services 3.0. SharePoint is nu op diverse manieren aan te passen; direct via de userinterface, via designtools als Office SharePoint Designer (vorige versie bekend als Office Frontpage) of via SharePoint Solutions. Dit artikel helpt de ontwikkelaar op weg met het ontwikkelen van SharePoint Solutions. De onderdelen hiervan worden 'high-level' beschreven. In de referenties vind je links naar sites waar je meer te weten kunt komen over SharePoint Solutions.

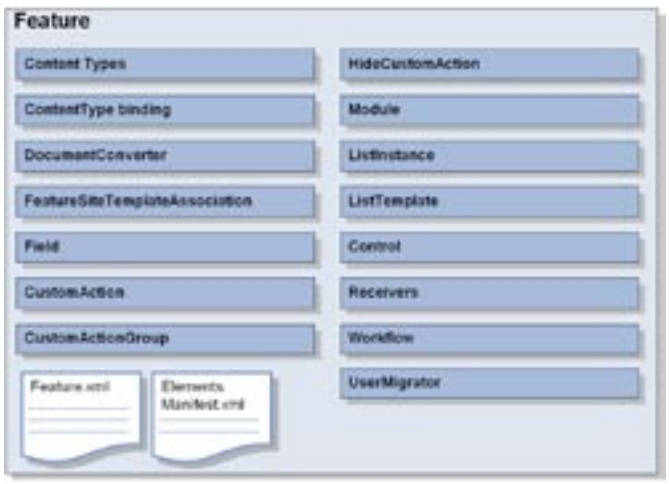
Wat is een SharePoint Solution? Dit is een uitbreiding van de bestaande functionaliteit van SharePoint. Je kunt hiermee bijvoorbeeld de userinterface, workflows, content-types en site templates aanpassen. Een SharePoint Solution is opgebouwd uit de volgende onderdelen:

- **Feature Framework**: kleine stukjes herbruikbare functionaliteit die je als legoblokken in SharePoint kunt schuiven.
- **Templates**: SharePoint-site-definities waarmee je een SharePoint-website definieert; wat komt er in en hoe.
- **Solution Framework**: een solution omvat alle templates en features en kan als een installatiepakket worden beschouwd van een oplossing binnen de SharePoint-omgeving.

Naast het ontwikkelen van features, templates en solutions zijn er nog andere onderdelen te ontwikkelen. Denk hierbij aan webparts, digitale formulieren en workflows.

Feature Framework

Bij het ontwikkelen van SharePoint-functionaliteit kun je als ontwikkelaar gebruikmaken van het zogenoemde Feature Framework. Dit framework biedt de ontwikkelaar de mogelijkheid kleine blokjes herbruikbare functionaliteit te ontwikkelen. Features kunnen los van templates en/of solutions geïnstalleerd worden. Wanneer een feature wordt geïnstalleerd, is de functionaliteit klaar om gebruikt te worden binnen de SharePoint-omgeving. Pas nadat de feature is geactiveerd, is de functionaliteit beschikbaar voor de gebruikers. Een feature wordt opgebouwd uit meer xml-files, de belangrijkste is feature.xml. Dit bestand bevat de naam en unieke ID van de feature. Daarnaast bevat het verwijzingen naar de andere xml-files, de elements manifest xml-files. Deze files beschrijven de aangeboden functionaliteiten. In afbeelding 1 wordt de opbouw van een feature weergegeven, inclusief alle acties die in het Feature Framework zijn gedefinieerd.



Afbeelding 1. De opbouw van een feature en een overzicht van de mogelijke acties.

Feature.xml

Elk feature wordt omschreven in feature.xml-file. Bij het omschrijven moet worden gedacht aan de metadata van de feature; de naam, de omschrijving en nog vele anderen. Eén meta-eigenschap is erg belangrijk, namelijk de scope-eigenschap. De scope-eigenschap vertelt SharePoint op het moment van implementatie voor welke SharePoint-onderdelen deze feature actief wordt. Er zijn vier mogelijke scopes:

- **Farm**: De featurefunctionaliteit wordt beschikbaar gemaakt voor de gehele farm, dat betekent alle SharePoint-websites die aangemaakt zijn.
- **Web Application**: De featurefunctionaliteit wordt beschikbaar gemaakt voor een enkele SharePoint-webapplicatie, een SharePoint-webapplicatie staat gelijk aan een website in Inter Information Services.
- **Site**: De featurefunctionaliteit wordt beschikbaar gemaakt voor een enkele sitecollectie binnen een webapplicatie. De functionaliteit is dan beschikbaar voor alle webs in de sitecollectie.
- **Web**: De featurefunctionaliteit wordt beschikbaar gemaakt voor een enkele SharePoint.

Zie codevoorbeeld 1 voor de mogelijke metadata-eigenschappen van de feature.

Naast de feature meta-eigenschappen, definieert de feature.xml-file nog een aantal andere zaken, namelijk:

- **Activation dependencies**: Stel voor dat je een feature bouwt, maar die is afhankelijk van reeds geïmplementeerde functionaliteit. De tag 'Activation dependencies' biedt de ontwikkelaar de mogelijkheid deze nieuwe feature afhankelijk

te maken van de reeds bestaande functionaliteit. Wanneer de feature wordt geactiveerd en de afhankelijk feature is niet actief in het systeem, dan geeft SharePoint een foutmelding, die meldt dat eerst de afhankelijke feature geactiveerd moet worden.

- **Elements manifests**: Elements manifests zijn de elements.xml-files. Het beschrijft waar en welke elements-files er aanwezig zijn voor deze feature. Ook beschrijft het de element-files. Wanneer een elements.xml-file definities (modules) bevat, oftewel pagina's, dan moeten deze extra worden gedefinieerd als element-file.
- **Properties**: Voor een feature kun je eigen properties maken, eigenlijk een soort verzameling van constanten. Het voordeel is dat je nu niet meer de web.config of resource-files hoeft te misbruiken.

Elements.xml

In de elements xml-file kun je door een aantal xml-tags simpelweg nieuwe functionaliteit aanbieden in SharePoint.Alle onderdelen die in afbeelding 1 worden getoond, kunnen gedefinieerd worden in de elements-file. Per feature is het mogelijk meer elements xml-files te maken. In codevoorbeeld 2 is een voorbeeld getoond van een elements xml-file waarin een contenttype en een custom action worden gedefinieerd. De software development kit van SharePoint (zie referenties onderaan dit artikel) geeft inzicht in hoe je de andere onderdelen van de elements kunt implementeren.
Kortom, als SharePoint-ontwikkelaar moet je veel xml schrijven, maar er is een oplossing. Naast het in xml definiëren van functionaliteit door middel van de elements xml-file, kun je er ook voor kiezen zogeheten feature event-receivers te gebruiken, waardoor je in C# of in VB.Net met behulp van de feature-implementatie je eigen code kunt schrijven.

```
<Feature xmlns="http://schemas.microsoft.com/sharepoint/"
  ActivateOnDefault="TRUE"
  AlwaysForceInstall="TRUE"
  AutoActivateInCentralAdmin="FALSE"
  Creator="Developer John Doe"
  DefaultResourceFile=""
  Description="My Feature Description"
  Hidden="FALSE"
  Id="_GENERATE_YOUR_GUID_"
  ImageUrl="MyCompany.gif"
  ImageUrlAltText="Feature created by My Company"
  ReceiverAssembly="MyCompany.SharePoint, Version=1.0.0.0,
      Culture=neutral, PublicKeyToken=0203e2b063d8c945"
  ReceiverClass="MyCompany.SharePoint.FeatureReceivers.
      MyFeatureReceiver"
  RequireResources="FALSE"
  Scope="Web"
  SolutionId="SOLUTION_GUID_GOES_HERE"
  Title="My Feature"
  Version="1.0.0.0">
  <ActivationDependencies>
    <!-- Activation dependency to my former feature "My former feature"-->
    <ActivationDependency FeatureId="A2DEB43C-D067-4669-A069-
        397D9693AE6D"/>
  </ActivationDependencies>
  <ElementManifests>
    <ElementManifest Location="Elements\ElementsManifest.xml"/>
    <ElementFile Location="Files\Mypage1.aspx"/>
    <ElementFile Location="Files\Mypage2.aspx"/>
  </ElementManifests>
  <Properties>
    <!-- Custom feature property to store constants-->
    <Property Key="MyFeatureKey" Value="1000"/>
  </Properties>
</Feature>
```

Codevoorbeeld 1. Feature.xml

Feature event-receivers

Wanneer een feature wordt geïnstalleerd, geactiveerd, gedeactiveerd of gedeïnstalleerd, gaat er een event af. Dit event kan worden opgevangen door zelf een assembly te schrijven die de SPFeatureReceiver overerft. Binnen deze assembly kan voor elk van deze vier events custom code worden geschreven. Codevoorbeeld 3 toont een voorbeeld van een feature receiver- class.
De feature receiver-class die wordt getoond in codevoorbeeld 3 laat de vier mogelijke override-methods van de feature receiver-class zien. Wanneer je als ontwikkelaar je class schrijft, is het erg belangrijk om aan de feature-eigenschap 'scope' te denken. In dit voorbeeld gaat het om een feature die wordt geactiveerd met de scope Web.Vanuit de feature properties-variabele, die meegeven wordt aan de methode, kunnen we de parent van de feature halen en casten naar een SPWebobject. In het geval van een andere scope kunnen we de parent casten naar een van de andere drie objecten, zie voorbeeld commentaar.

Templates

Templates binnen SharePoint zijn definities waarmee ontwikkelaars de vorm en opzet van een SharePoint-web definiëren. In de vorige versie van SharePoint had je als ontwikkelaar te maken met zogenoemde site-definitions. Een site-definition bestond uit een onet.xml die de definitie beschreef (vaak een enorm bestand), plus een berg bestanden. Deze bestanden bestonden uit aspx-pagina's, lijstdefinities voor documentbibliotheken en voor andere typen lijsten. Het maken en aanpassen van site-definities was mogelijk, maar omslachtig. In de huidige versie van SharePoint is het eenvoudiger opgelost. Het bestand onet.xml is blijven bestaan, maar is kleiner geworden. Dit is opgelost door gebruik te maken van een soort basistemplate, het Global-template. Alle templates in Sharepoint gebruiken deze template nu als basis, hierdoor is het nu veel eenvoudiger om zelf een nieuwe template te maken. De onderdelen van een site-template zijn weergegeven in afbeelding 2.

Configurations

De huidige site-template kan meer configuraties bevatten, elke configuratie is in feite een unieke definitie. Bijvoorbeeld , je maakt een definitie die een navigatiebalk bevat en een documentbliblioteek. Dan kun je als ontwikkelaar twee configuraties aanmaken, zeg configuratie 0 en 1. Configuratie 0 maakt alleen de navigatiebalk aan en configuratie 1 maakt de navigatiebalk en de documentbibliotheek aan. Wanneer we een SharePoint-web willen aanmaken met deze definities en configuratie, hanteert

```
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <ContentType ID="0x01B1" Name="MyNewContentType"
               Group="My Company Content Types" Version="0">
    <FieldRefs>
      <FieldRef ID="_MY_FIELD_GUID_COMES_HERE_" Required="TRUE"/>
    </FieldRefs>
  </ContentType>
  <CustomAction Id="SiteActionsToolbarSiteInformation"
                GroupId="SiteActions"
                Location="Microsoft.SharePoint.StandardMenu"
                Sequence="101"
                Title="Site information"
                Description="Get overview of information of this site"
                Rights="ManageLists">
               <UrlAction Url="\_layouts\siteInformation.aspx"/>
  </CustomAction>
</Elements>
```

Codevoorbeeld 2. Elements manifest xml-file

SharePoint de templatenaam + # + configuratienummer. Bijvoorbeeld 'mijntemplate#0' en 'mijntemplate#1'.

Features versus site-templates

De functie van een site-template is simpel, een nieuwe site aanmaken volgens een bepaalde vastgestelde configuratie. De functie van een feature is het aanbieden van functionaliteit, zoals eerder beschreven. Er zijn onderdelen die je zowel in een site-template als in een feature kunt definiëren, bijvoorbeeld list-templates. Daarbij kan de vraag rijzen: 'Moeten we een feature bouwen voor die listtemplate of kunnen we deze in de site-template definiëren?' Het antwoord op die vraag is, als je de functionaliteit van de listtemplate op meer en verschillende type sites wil deployen, dat je dan een feature te gebruiken. De feature kan dan worden hergebruikt in meer site-templates.
Zoals beschreven kun je in een site-template verwijzigingen maken naar features. Het komt voor dat de site-template al is gedefinieerd en deze niet aangepast kan worden. Denk hierbij aan standaard SharePoint-templates. Als je dan toch als ontwikkelaar wilt dat jouw feature wordt meegenomen in deze templates, dan kun je dit doen door middel van de Feature Site Template-association tags. Dit zorgt er voor dat wanneer een web wordt aangemaakt met een standaard template, jouw feature tevens wordt geactiveerd binnen het web.

Solution Framework

Het uitrollen van een applicatie in SharePoint 2003 was voor ontwikkelaars een hele klus. Gelukkig heeft Microsoft in de nieuwe versie van SharePoint rekening gehouden met de ontwikkelaars. In Windows SharePoint Services 3.0 heeft de ontwikkelaar de beschikking over het Solutions Framework. Het Solutions Framework biedt een manier om alle uitbreidingen in SharePoint in een nieuw bestand te bundelen, een *solution-file*. Een solution-file is een cabinetbestand (.cab) met een .wsp-extensie. Een solution, bestaande uit één of meer features, site-templates en assemblies, kan verscheidene keren worden gebruikt en op individuele sites worden uitgerold. Met behulp van het Solution Framework kunnen de volgende zaken worden geïnstalleerd (zie afbeelding 3):

- Site-templates
- Features inclusief bijbehorende bestanden
- Webpart-files (*.webpart & *.dwp)
- Template-files en root-files, bijvoorbeeld:
- _layout-files
- Resources (*.resx)
- Resources-files (bijvoorbeeld *.doc, *.xls)
- Assemblies, inclusief:
- Safecontrols aanpassingen in web.config
- Resources
- Code access security-policies

```
public class MyFeatureReceiver : SPFeatureReceiver
{
   public override void FeatureActivated(SPFeatureReceiverProperties
properties)
   {
      // Method is called on activation

      SPWeb currentWebSite = (SPWeb) properties.Feature.Parent;
      // SPSite currentSiteCollection = (SPSite) properties.Feature.Parent;
      // SPWebApplication currentWebApp = (SPWebApplication) properties.
                                          //Feature.Parent;
      // SPFarm currentFarm = (SPFarm) properties.Feature.Parent;
   }

   public override void FeatureDeactivating(SPFeatureReceiverProperties
                        properties)
   {
      // Method is called on de-activation
   }

   public override void FeatureInstalled(SPFeatureReceiverProperties
                        properties)
   {
      // Method is called on installation
   }

   public override void FeatureUninstalling(SPFeatureReceiverProperties
                        properties)
   {
      // Method is called on uninstalling
   }
}
```

Codevoorbeeld 3. De feature receiver-class

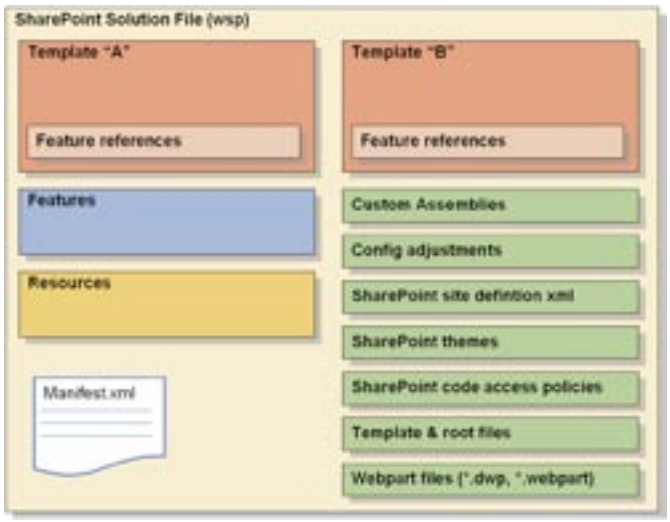
Voordelen van het gebruik van het Solutions Framework:

- Geïntegreerde deployment: Solutions stellen developers en administrators in staat gemakkelijk bestanden te installeren op de front-end server van een serverfarm. Hierbij is het framework verantwoordelijk voor de uniformiteit van alle servers in de SharePoint-omgeving.
- Meertaligheid: Solutions kunnen gebruikt worden om een applicatie in meer talen beschikbaar te stellen.

De ontwikkeling van een solution

Als de ontwikkelaar klaar is met het maken van de features, site-templates, webservices, enzovoort, kan de solution worden gemaakt. Dit begint bij het maken van een manifest.xml, wat in de root van de solution-file komt te staan. Dit bestand definieert de lijst met features, site-templates, resourcesfiles en assemblies, die door het Solution Framework moeten worden verwerkt. In de Windows SharePoint Services SDK is het schema voor manifest.xml opgenomen. In codevoorbeeld 4 staat een voorbeeld van een manifest.xml. In dit bestand staat de unieke ID van de solution; deze GUID kan gegenereerd worden door een Visual Studio-command-prompt te openen en 'uuidgen –c' uit te voeren. Een solution-file is eigenlijk een cabinetbestand (.cab). De makecab.

Afbeelding 2. De onderdelen van een site-template.



Afbeelding 3. De inhoud van een SharePoint Solution (.wsp)

exe-tool gebruikt een ddf-bestand om de structuur van de cabinet-file op te bouwen. De opmaak van een ddf-bestand heeft in essentie dezelfde opmaak als een inf-bestand, namelijk een standaard header en daarna op elke regel een verwijzing naar een bestand op disk en de locatie in het .cab- bestand. In codevoorbeeld 5 is een voorbeeld opgenomen van een ddf-bestand. De solution-file wordt gemaakt met behulp van makecab.exe (onderdeel van de Microsoft Cabinet SDK), zie commando 1. De inhoud van de solution-file is te bekijken door het bestand te hernoemen naar een cab-bestand en het direct vanuit de Windows Verkenner te openen.

```
<Solution SolutionId="_GENERATE_YOUR_GUID_" xmlns="http://schemas.
microsoft.com/sharepoint/">
  <FeatureManifests>
    <FeatureManifest Location="MyFeature\feature.xml"/>
  </FeatureManifests>
  <TemplateFiles>
    <TemplateFile Location="ControlTemplates\MyFeatureForm.ascx"/>
  </TemplateFiles>
  <RootFiles>
    <!-- These files go into the 12\ directory and can be used for
 Web services and global resources -->
    <RootFile Location="ISAPI\MyWebService.asmx" />
  </RootFiles>
  <Assemblies>
    <Assembly DeploymentTarget="GlobalAssemblyCache" Location="
 myAssembly.dll"/>
  </Assemblies>
</Solution>
```

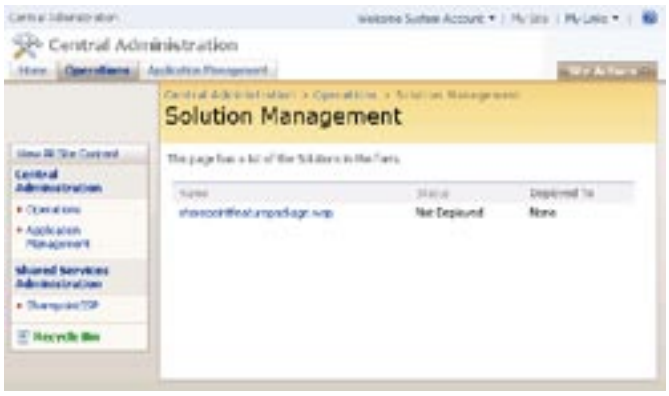
Codevoorbeeld 4. Manifest.xml

```
OPTION EXPLICIT     ; Generate errors
.Set CabinetNameTemplate=MySolutionFile.wsp
.set DiskDirectoryTemplate=CDROM ; All cabinets go in a single directory
.Set CompressionType=MSZIP;** All files are compressed in cabinet files
.Set UniqueFiles="ON"
.Set Cabinet=on
.Set DiskDirectory1=Package
;*****************************************
Manifest.xml

.Set DestinationDir=Assemblies
bin\MyAssembly.dll

.Set DestinationDir=Features\MyFeature
MyFeature\feature.xml
...
```

Codevoorbeeld 5. Mysolution.ddf



Afbeelding 4. Solution Management in Central Admininstration

```
makecab.exe /F MySolution.ddf /D CabinetNameTemplate=MySolution.WSP /D
 DiskDirectory1=c:\MyOutputDir
```

Commando 1. Genereren van de solution-file

Solution deployment

De installatie en deïnstallatie van een solution-file is alleen mogelijk met behulp van stsadm.exe, zie commando 2. Vanuit de central administration-website is het mogelijk een solution uit te rollen naar een SharePoint-site. Ook is daar een overzicht te vinden van alle beschikbare solutions (Op tabblad 'Operations' optie Manage Solutions, zie afbeelding 4 en 5). Naast de central administration-website biedt stsadm.exe ook de mogelijkheid een solution uit te rollen naar een site, zie commando 3.

```
stsadm -o addsolution -filename MySolutionFile.wsp
stsadm -o deletesolution -name MySolutionFile.wsp –override
```

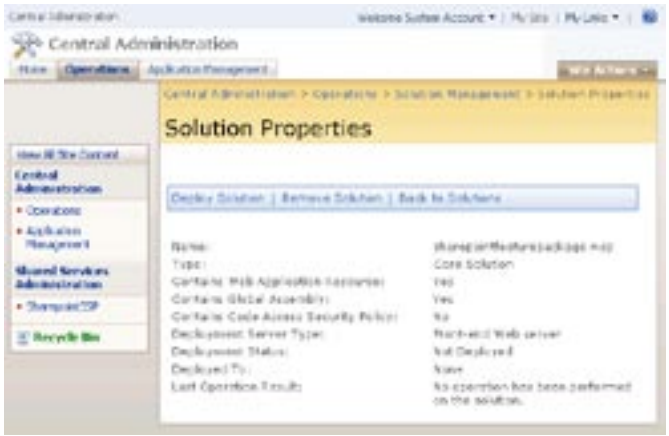
Commando 2. Installatie en deïnstallatie van een solution-file

```
stsadm -o deploysolution -name MySolutionFile.wsp -immediate –allcon-
tenturls
-allowGacDeployment –allowCasPolicies
stsadm -o retractsolution -name MySolutionFile.wsp -immediate –allcon-
tenturls
```

Commando 3. Deployment-commando's

Handige hulpmiddelen

Het genereren van een solution-file hoeft geen handmatige actie te zijn, Microsoft heeft de ontwikkelaar reeds met 'Visual Studio 2005 extensions for Windows SharePoint Services 3.0' voorzien van een aantal hulpmiddelen om snel solutions mee te maken. Op Codeplex (www.codeplex.com) is een aantal projecten aanwezig, dat de ontwikkelaar kan gebruiken bij het maken van een SharePoint Solution. Naast deze hulpmiddelen kan een aangepaste msbuild-file in Visual Studio een oplossing zijn voor het genereren van het solution-file.



Afbeelding 5. Solution Properties en Actions

App_GlobalResources

In manifest.xml is geen mogelijkheid aanwezig om resourcefiles op te nemen die worden geïnstalleerd in de folder app_globalresources van de webapplicatie. De resources in de app_globalresources worden gebruikt in site-templates en in custom aspx-bestanden .Via een feature in combinatie met een feature-receiver-assembly is het toch mogelijk resourcefiles te installeren in de folder app_globalresources. Lees voor meer informatie de blog van Mikhail Dikov (zie referenties).

Samenvatting

De manier van ontwikkelen in SharePoint is gewijzigd en sluit nu beter aan bij de wensen van ontwikkelaars. SharePoint Solutions biedt een development-methodiek aan om de functionaliteit van SharePoint uit te breiden met features en templates. Hiernaast biedt SharePoint Solutions een deployment-methodiek aan, het Solutions Framework. Dit geeft de ontwikkelaar de mogelijkheid op een gestructureerde wijze software te installeren binnen diverse SharePoint-omgevingen.

Michiel Lankamp (michiel.lankamp@winvision.nl) en **Willem Boeré** (willem.boere@winvision.nl) zijn ontwikkelaars bij Winvision (www.winvision.nl), een Microsoft Gold Certified Partner in Nieuwegein. Zij hebben een gecombineerde SharePoint-ervaring van zo'n tien jaar en doen dagelijks ervaring op in ontwikkelingstrajecten met SharePoint 2007.

Referenties

MSDN, WSS TechCenter: www.microsoft.com/technet/windowsserver/sharepoint

MSDN, Working with Features: http://msdn2.microsoft.com/en-us/library/ms460318.aspx

MSDN, Solutions Overview: http://msdn2.microsoft.com/en-us/library/aa543214.aspx

Microsoft Cabinet SDK: http://support.microsoft.com/default.aspx/kb/310618

Mikhail Dikov, Deploying resource files to the App_GlobalResources: http://dikov.blogspot.com/2007/03/sharepoint-resources-types-use-and_2163.html