SharePoint SSP#: twee verdiende plusjes erbij

Je kunt nu zelf Service Applicaties bouwen

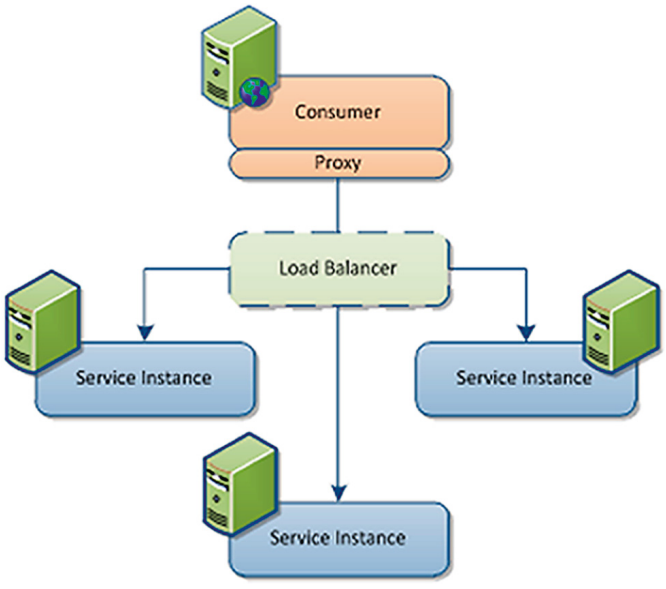
Wouter van Vugt

Zoals mogelijk elk vakgebied kent ook de software ontwikkeling zijn mythen en sagen. Wist je bijvoorbeeld dat naam 'C#' voortkomt uit het doel C++ te verbeteren? Het # teken is het resultaat van het toevoegen van nóg twee plussen. C# is dus eigenlijk C++++. Ook bij veel SharePoint 2010 onderdelen mag je een # teken noteren. Dit keer de focus op wat in 2007 nog Shared Services heette.

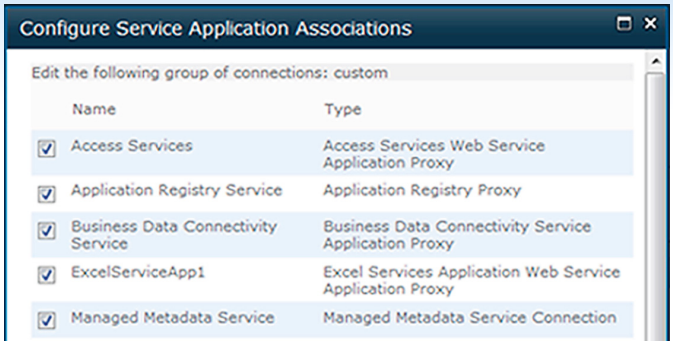
Shared Services is een concept uit SharePoint Server 2007. Het biedt een plek om gecentraliseerd, buiten de site collecties, een aantal gedeelde middle-tier services aan te bieden. Site collecties consumeren deze functionaliteit door bijvoorbeeld web parts, timer jobs en event receivers. Shared Services worden aangeboden via een Shared Service Provider. Het belangrijkste doel van dit provider-concept is partitionering van Shared Services. Elke provider bevat onafhankelijk alle Shared Services en heeft bijvoorbeeld een eigen set databases.Je koppelt een web applicatie aan een provider waarna de geboden services beschikbaar komen binnen die web app.
Hoewel het SSP model vernieuwend was voor SharePoint 2007 heeft het ook zo zijn nadelen. Zo is model niet erg flexibel. Web applicaties moeten álle services afnemen van één provider. Het model is geen onderdeel van de SharePoint core en zelf een nieuw service type toevoegen is helaas niet mogelijk. Als je eigen middle-tier functionaliteit wil delen of informatie wilt bewaren over meerdere site collecties dan bouw je al snel je eigen SSP-achtig mechanisme.

Service Applicaties

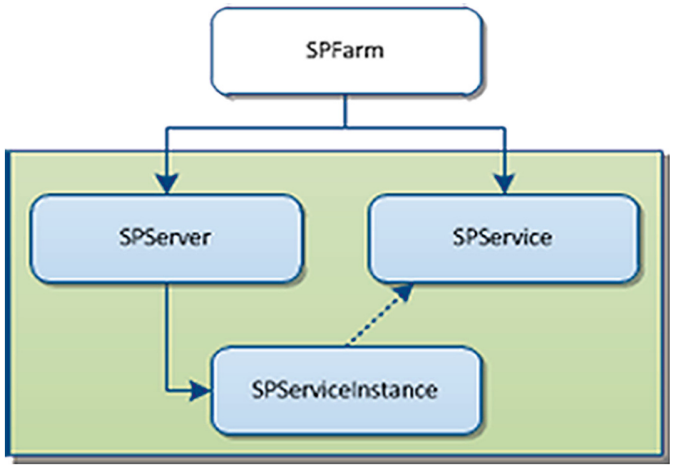
De komst van SharePoint 2010 brengt een heel scala aan interessante veranderingen. Een van de nieuwe Lego blokjes die je kunt inzetten om een klein, of misschien wel groot, paleis te bouwen is de Service Applicatie. Dit nieuwe model vervangt de SharePoint 2007 SSP en biedt voor SharePoint ongeëvenaarde programmeerbaarheid. Het nieuwe model biedt alle voordelen van Shared Services zoals gecentraliseerd management, maar dan wel zonder de opgeworpen restricties zoals de inflexibiliteit van SSP's. En.... tromgeroffel, je mag ze nu ook zelf maken!
Een service applicatie biedt middle-tier functionaliteit, dat wordt geconsumeerd door web applicaties. Deze functionaliteit wordt niet direct aangeroepen maar via een proxy. Figuur 1 toont dit proxy model. Via de proxy kunnen aanroepen naar de services worden verlegd naar een back-end server. Het mooie aan dit extra niveau van indirectie is de mogelijkheden die het biedt ten aanzien van de inrichting van de server farm. Als de Service Applicatie op de backend server stateless wordt opgebouwd dan maakt het natuurlijk niet uit welke server de code uitvoert. Stel bijvoorbeeld dat je de Word Automation Service gebruikt voor het converteren van documenten naar PDFs, en dat je een extra drukke tijd tegemoet ziet komen. Je kunt eenvoudig extra back-end servers bijschuiven. De ingebouwde load balancer pikt ze meteen op en je conversies zullen transparant op meerdere servers worden uitgevoerd. Weer in rustig vaarwater? Dan haal je de servers weer uit de farm en laat je ze weer andere dingen doen! De flexibiliteit uit zich ook op andere vlakken. Omdat de meeste service applicaties WCF gebruiken als communicatiemiddel tussen proxy en service is het eenvoudig om een service op een veilige manier cross-farm aan te bieden. Als je als grotere organisatie meerdere SharePoint installaties in de lucht houdt dan kan je toch gebruik maken van globale functionaliteit.
Een ander belangrijk voordeel is dat je in het nieuwe model zelf kunt bepalen welke services een web applicatie mag benaderen.



FIGUUR 1 SERVICE TOPOLOGIE



FIGUUR 2 CONSUMER CONFIGURATIE



FIGUUR3: FARMS, SERVICES EN INSTANCES

Geen alles of niets meer, maar een uiterst configurabel systeem dat door middel van standaard groeperingen toch eenvoudig is in gebruik. Figuur 2 toont het configuratiescherm voor een web applicatie waar gekozen kan worden welke services moeten worden geconsumeerd.

Services bouwen

Het ontwikkelen van een Service Applicatie is minder ingewikkeld dan je misschien denkt. Enkele basis componenten zijn vereist, maar hierna mag je zelf kiezen welke kamers in je kasteel gepimpt moeten worden. Laten we beginnen te kijken naar de elementen die je echt nodig hebt.

Farms, Services en Service Instances

In het SharePoint 2007 object model zijn de SPFarm, SPServer en SPService klasse geintroduceerd. Deze klassen vormen de hierarchie die je ziet in Figuur 3. De SPFarm stelt de gehele SharePoint installatie voor die bestaat uit servers en services. Voor elke fysieke server uiteraard een bijbehorend SPServer object. De SPService werd gebruikt om een farm 'service' aan te bieden. Een beetje abstract, maar dat was ook de bedoeling. Je zou bijna kunnen zeggen dat de set aan SPService objecten eigenlijk hetgeen is wat SharePoint tot een product maakt. Bijna alle grote brokken functionaliteit zijn verwoord als service. Via het 'Services on Server' scherm in Central Admin kan je de infrastructuur die een service nodig heeft om zijn functionaliteit aan te bieden activeren op een server in de farm. Door dit activeren wordt bijvoorbeeld een Windows Service actief op die specifieke server, of misschien wordt er een ASMX endpoint beschikbaar gesteld waarachter een complexe applicatie schuilt. Dit concept, het hebben van een service die op een bepaalde server draait is ook benaderbaar via object model. Namelijk via de SPServiceInstance klasse. Deze klasse is de koppeling tussen een SPService en een SPServer. Niet iedere service heeft ook een service instance nodig, maar wel als je fysiek iets op een server wil activeren. De SPService klasse kan je gebruiken voor het maken van je eigen administratieve componenten. Stel bijvoorbeeld dat je SharePoint oplossing bestaat uit meerdere site collecties. Als je wilt vastleggen welke site collecties je oplossing omvat heb je een hoger gelegen niveau nodig voor het bewaren van deze informatie. Hiervoor is de SPService bijvoorbeeld geschikt. Zoals je kunt zien is de SPFarm de parent van de service. Het aan en uitzetten is een globale functie en is niet bekend met enige web applicatie om de service bij aan te bieden.

```
public class SimpleService
    : SPService
{
    public SimpleService()
    {
    }

    public SimpleService(SPFarm parentFarm)
        : base(String.Empty, parentFarm)
    {
    }

    public override void Provision()
    {
        ...
        Status = SPObjectStatus.Online;
        Update();
    }

    public override void Unprovision()
    {
        ...
        Status = SPObjectStatus.Offline;
        Update();
    }
}
```

CODE LISTING 1: EEN FARM-SERVICE

Zoals voor alle functionaliteit in SharePoint gebruik je een Feature om dit ding te installeren. Met enkele regels code kan je de service toevoegen aan de SharePoint farm. Dit toevoegen is een tweetraps raket. In het administratieve object model instantieer je eerst een object. Hierna roep je Update aan om dit object ook op te slaan in de database. Hieropvolgend gebruik je de Provision methode om de functionaliteit 'aan' te zetten. Uitzetten gaat via de Unprovision methode.

```
SimpleService service =
new NavigationService(farm);
service.Update();
service.Provision();
```

Ook de SPServiceInstance objecten maak je volgens hetzelfde principe. Overerven van SPServiceInstance, en in de Provision methode wat interessants doen met het server object dat je tot je beschikking hebt.

```
class SimpleServiceInstance
  : SPServiceInstance
{
  ...

public override void Provision()
  {
SPServer server = base.Server;
// do stuff on server
}
}
```

CODE LISTING 2 EEN SERVICE INSTANCE

Tijdens het activeren van je feature voeg je de relevante SPServiceInstance objecten toe aan de farm. Echter, dit keer hoef je niet hardcoded de Provision methode aan te roepen om de service te activeren op een server in de farm. Via het 'Services on Server' scherm zichtbaar in Figuur 4 kan een beheerder dit namelijk zelf bepalen. Door in dit scherm op de Start of Stop knop te klikken wordt ofwel Provision of Unprovision aangeroepen op de SPServiceInstance klasse.
Nu we begrip hebben voor het administratieve model uit SharePoint 2007 is het tijd om de toevoegingen in SharePoint2010 nader te onderzoeken. Alle structuren die we net hebben onderzocht zijn nog altijd relevant. Zo wordt bijvoorbeeld de SPServiceInstance gebruikt om te bepalen op welke server de Service Applicatie mag worden uitgevoerd.

Service Applicaties en Proxies

Wat ontbreekt aan het oude object model met de SPService en SPServiceInstance is een door het systeem afgedwongen representatie van een service die door web applicaties kan worden afgenomen. In SharePoint Foundation is een belangrijke nieuwe bouwsteen toegevoegd. De nieuwe SPServiceApplication klasse is de vervolmaking van het nieuwe service model.
In SharePoint 2010 is ervoor gekozen de nieuwe Service Applicatie functionaliteit te ontsluiten via de SPService klasse. Immers, de SPService representeert jouw service, maar dan farm breed. De oude SPServiceInstance representeerde de toestand van jouw service gepartitioneerd per server in de farm. Net als een Shared Service Provider representeerde SPServiceApplication jouw service, maar dan meer logisch gepartitioneerd. Ook wordt de oude SPServiceInstance nog altijd gebruikt. De Service Applicatie is met name configuratie. Welke database wordt gebruikt, welke settings, security, etcetera. De SPServiceInstance bepaalt echter op welke server de code wordt uitgevoerd die gebruik maakt van de configuratie in de Service App. De collectie van SPServiceInstance objecten wordt dan gebruikt door de load balancer om de binnenkomende verzoeken te verdelen over de beschikbare server capaciteit.
Een deel van de functionaliteit die je vroeger in de SPService zou onderbrengen verplaatst zich naar de SPServiceApplication. Zo kan je verwachten dat een service applicatie een eigen database heeft. Deze database wordt bewaard als veld binnen de SPServiceApplication klasse. Een belangrijk bijkomend voordeel van het nieuwe programmeermodel is dat databases automatisch worden aangeboden aan de backup technologie van SharePoint.

FIGUUR 4 SERVICES ON SERVER

```
class SimpleApplication
  : SPServiceApplication
{
  [Persisted]
SPDatabase _database;

  ...

public override void Provision()
```

```
  {
    _database.Provision();
    Status = SPObjectStatus.Online;
  }

public override void Unprovision(bool deleteData)
  {
if (deleteData)
    {
      _database.Unprovision();
    }
    Status = SPObjectStatus.Offline;
}
}
```

CODE LISTING 3 SERVICE APPLICATIE BASIS CODE

Nadat je de service hebt ontwikkeld is het tijd om aan de proxy te beginnen. Hierbij ook een ruime keuze aan wat je nu gaat bouwen. Ten eerste, een proxy hoeft niet. Je kunt een applicatie hebben zonder proxy, waarbij de applicatie back-end taken uitvoert niet zichtbaar voor front-end machines. Je kunt een proxy hebben zonder applicatie, als de applicatie bijvoorbeeld geimplementeerd is in een ander systeem. Neem een CRM-proxy of de out-of-the-box beschikbare Notes connector. Het niet hebben van een proxy wordt echter niet geadviseerd. Om een proxy te bouwen moet je overerven van de SPServiceApplicationProxy klasse. Uiteraard moet je proxy weten met welke Service Applicatie moet worden gepraat. Deze informatie wordt meegegeven tijdens het maken van de proxy.

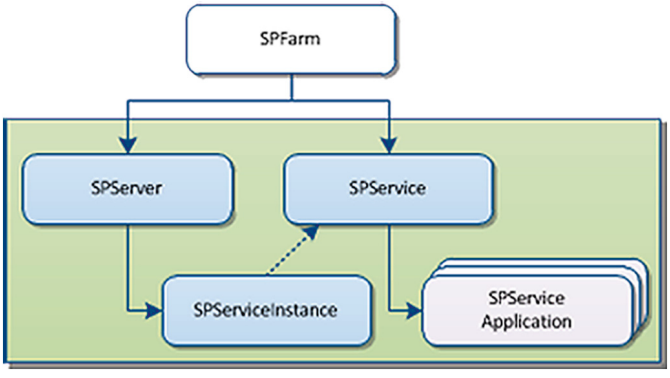
```
[GuidAttribute("0c1a4b04-90d5-4cef-be3a-f4591b1ac873")]
class SimpleApplicationProxy
  : SPServiceApplicationProxy
{
  [Persisted]
Guid _serviceApplicationID;

public SimpleApplicationProxy()
  {
  }

public SimpleApplicationProxy(string name,
NavigationApplication application,
NavigationServiceProxyserviceProxy)
    : base(name, serviceProxy)
  {
    _serviceApplicationID = application.Id;
  }

public void DoStuff()
  {
SimpleApplication application = Farm.GetObject(
      _serviceApplicationID) asSimpleApplication;
application.DoStuff();
  }
}
```

CODE LISTING 4 SERVICE APPLICATIE PROXY



FIGUUR 5 SERVICE APPLICATIES IN HET OBJECT MODEL

Het belangrijkste stukje code is uiteraard het aanroepen van de proxy. Dat doe je met een speciale klasse: SPServiceContext

```
SPServiceContext context = SPServiceContext.Current;
SimpleApplicationProxy proxy =
context.GetDefaultProxy(
typeof(SimpleApplicationProxy))
asSimpleApplicationProxy;
proxy.DoStuff();
```

WCF

De code voorbeelden die ik tot nu toe heb gebruikt zijn alleen geschikt voor basis service applicaties. Applicaties die met name configuratie bevatten en geen zware componenten in termen van CPU en memory gebruik. Als je de kracht van WCF wilt gaan gebruiken om wél je zware applicatie op een backend server te plaatsen moet je wat meer werk doen. Denk hierbij aan het deployen van WCF endpoints en integratie met Claims security. Met behulp van extra basis klassen zoals de SPIIsWebServiceApplication kun je vrij snel beschikken over de extra capaciteit in de farm. Het meest spannende stukje code is het aanroepen van de WCF server. Omdat je offloading wilt krijg je te maken met de load balancer en het aanroepen van WCF. Het leuke is dat je dit alles kunt inpakken in een generieke methode die dit alles afschermt en dat ik er nog een lekker complex codevoorbeeld bij kan stoppen.

```
TResult ExecuteOnChannel<TResult, TInterface>(
Func<TInterface, TResult> method)
{
Configuration clientConfig =
OpenClientConfiguration(channelConfigPath);
ChannelFactory<TInterface> factory =
new ConfigurationChannelFactory<TInterface>(
"http", clientConfig, null);
factory.ConfigureCredentials(SPServiceAuthenticationMode.Claims);

SPServiceLoadBalancerContext context = _loadBalancer.Begin-
Operation();
try
  {
TInterface application =
factory.CreateChannelActingAsLoggedOnUser<TInterface>(
 new EndpointAddress(context.EndpointAddress));
return method(application);
}
finally
  {
    context.EndOperation();
  }
}
```

CODE LISTING 5 AANROEPEN VAN EEN WCF APPLICATIE

In Code listing 5 zie je de generieke aanroep naar een service. De service interface is een type parameter, de return value is een type parameter. Verder biedt je als delegate de methode op de service interface aan die moet worden aangeroepen. Het mooie is dat je de aanroep naar de WCF server, inclusief load balancing, kan doen met een simpele call als ExecuteOnChannel<string>(a => a.DoStuff()). Op deze momenten vind ik C++++ toch een erg mooie, expressieve taal.

Wouter van Vugt, is MVP en expert op het gebied van SharePoint en Office development. Hij is medeauteur van het Microsoft SharePoint 2010 early-adopters programma en heeft sinds de eerste alpha versies ervaring opgedaan met de nieuwste versie van Microsoft SharePoint Server en Microsoft Office. Wouter is te bereiken op wouter@code-counsel.net.