Sandboxed Solutions: maatwerk in SharePoint

Juist SharePoint-beheerders profiteren hiervan

Robert Jaakke

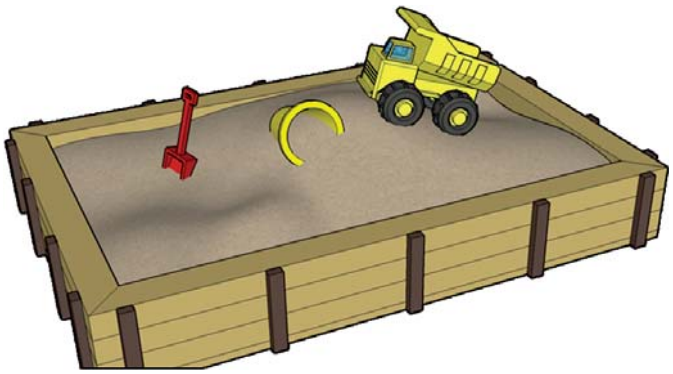
SharePoint Solutions is de magische term wanneer je spreekt over deployment van Features, Site definitions, Assemblies of andere maatwerkoplossingen in SharePoint. SharePoint 2010 doet hier nog een schepje bovenop door het mogelijk te maken om als site administrator een SharePoint Solution uit te rollen binnen een Site Collection.

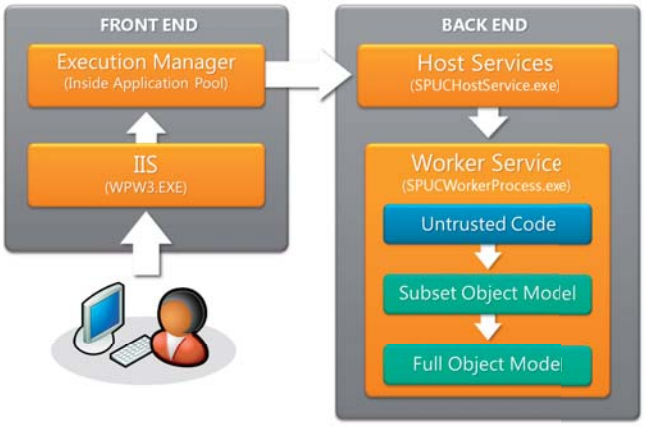
De gemiddelde SharePoint-beheerder is zich misschien een rolberoerte geschrokken bij het lezen van de inleiding maar juist deze groep mensen gaat van Sandboxed Solutions profiteren.

Door een Solution uit te rollen als een Sandboxed Solution zorg je er namelijk voor dat de Farm stabiel blijft ook als deze stinkende User Code bevat. Sandboxed Solutions worden beheerd op Farm-niveau en krijgen een quotum toegewezen voor het aantal resources dat ze kunnen gebruiken. Zijn de resources op dan krijgt de gebruiker netjes een melding dat hij de volgende dag terug kan komen. Webparts die meer dan dertig seconden nodig hebben om hun taken uit te voeren worden automatisch gestopt. Een andere belangrijke eigenschap is dat Sandboxed Solutions in een geïsoleerd proces draaien, dit is om te voorkomen dat Sandboxed Solutions de hele Farm neerhalen. Developers worden door de Sandbox dus gedwongen om goede code te schrijven en daarom is volgens Microsoft de Sandboxed Solution de preferred solution.

De User Code Service

Om te beginnen moet eerst de User Code Service gestart worden op elke server binnen de Farm die Sandboxed Solutions moet





gaan uitvoeren. Deze Service faciliteert het Sandbox Worker Process en kan gezien worden als een soort service host. De User-Code Service is een nieuwe service in SharePoint 2010 en is onderhuids eigenlijk een Service Application (net als Search, Excel Services, Managed Metadata Service).
Zoals misschien bekend zijn Service Application's de nieuwe SSP. De architectuur van de Service Applications zorgt ervoor dat deze volledig schaalbaar is. Deze eigenschap heeft de User Code Service dus ook. De User Code service kan bijvoorbeeld draaien op alle web front-end servers maar het is ook mogelijk om dedicated servers op te nemen in de Farm voor optimale performance en stabiliteit. Dit laatste kom je in de praktijk alleen tegen bij hele grote SharePoint-omgevingen of in Shared Hosting omgevingen zoals bijvoorbeeld SharePoint Online.

De User Code Service bestaat uit drie processen:

- User Code Service (SPUCHostService.exe)
- Sandbox Worker Process (SPUCWorkerProcess.exe)
- Sandbox Worker Process Proxy (SPUCWorkerProcessProxy.exe)

De subset van Microsoft.SharePoint

Het Sandbox Worker Process is verantwoordelijk voor het uitvoeren van de code in een Sandboxed Solution en zorgt er ook voor dat alle code netjes binnen de grenzen blijft opereren. De grenzen binnen het SPUCWorkerProcess.exe worden bepaald door de beschikbare classes. Binnen het SPUCWorkerProcess.exe heeft de developer de beschikking over een subset van de Microsoft.SharePoint namespace. In namspaces zoals Microsoft.SharePoint.Administration heeft de code van een Sandboxed Solution natuurlijk niks te zoeken. Het doel van de Sandbox is immers stabiliteit en de code van een Sandboxed Solution mag dan niet zelf zijn quota aanpassen om toch net even die 'dure' operatie uit te kunnen voeren. Met deze subset van Microsoft.SharePoint zullen de meeste developers goed uit de voeten kunnen bij het ontwikkelen van hun maatwerk.

Hier volgt een opsomming van de beschikbare classes:

- Microsoft.SharePoint behalve
  - SPSite constructor
  - SPSecurity object
  - SPWorkItem and SPWorkItemCollection objects
  - SPAlertCollection.Add method
  - SPAlertTemplateCollection.Add method
  - SPUserSolution and SPUserSolutionCollection objects
  - SPTransformUtilities
- Microsoft.SharePoint.Navigation
- Microsoft.SharePoint.Utilities behalve
  - SPUtility.SendEmail method
  - SPUtility.GetNTFullNameandEmailFromLogin method
- Microsoft.SharePoint.Workflow
- Microsoft.SharePoint.WebPartPages behalve
  - SPWebPartManager object
  - SPWebPartConnection object
  - WebPartZone object
  - WebPartPage object
  - ToolPane object
  - ToolPart object

Ben je geïnteresseerd in de complete lijst van de beschikbare classes? Raadpleeg dan de SDK.

Hoe werkt het in Visual Studio 2010?

Bij het aanmaken van een nieuwe SharePoint Solution in Visual Studio 2010 kun je de vraag krijgen wat het Trust Level moet zijn. Hierbij is er keuze uit een Sandboxed Solution of een Farm Solution. Het enige zichtbare verschil tussen deze twee solutions is dat de eigenschap Sandboxed Solution op True staat in de eigenschappen van de Visual Studio Solution en de regel [assembly: AllowPartiallyTrustedCallers()] in het bestand AssemblyInfo.cs van elk project. Voor het debuggen van een Sandboxed Solution moet de debugger attached worden aan het SPUCWorkerProcess.exe. In Visual Studio 2010 is dit met de nieuwe SharePoint tools zo simpel als op F5 drukken en wachten tot er een breakpoint geraakt wordt. Visual Studio zal eerst een eventuele aanwezige versie van solution retracten, de solution opnieuw deployen en daarna automatisch attachen aan SPUCWorkerProcess.exe.

Deployment

De uitrol van een Sandboxed Solution vindt plaats in de Solution Gallery op Site Collection niveau. De Solution Gallery is een spe-

ciale SharePoint-lijst en is te vinden in de RootWeb van de Site Collection. Hier vind het beheer plaats en kunnen solutions geüpload, geactiveerd en gedeactiveerd worden. Ook staat hier een overzicht van de door de Solutions verbruikte resources en het gemiddelde verbruik van resources van de afgelopen veertien dagen. Na het activeren zullen eventuele Site Scoped Features van de Solution automatisch geactiveerd worden, Web Scoped Features moeten handmatig geactiveerd worden. De Solution Gallery is te vinden onder Site Actions->Site Settings->Galleries->Solutions.

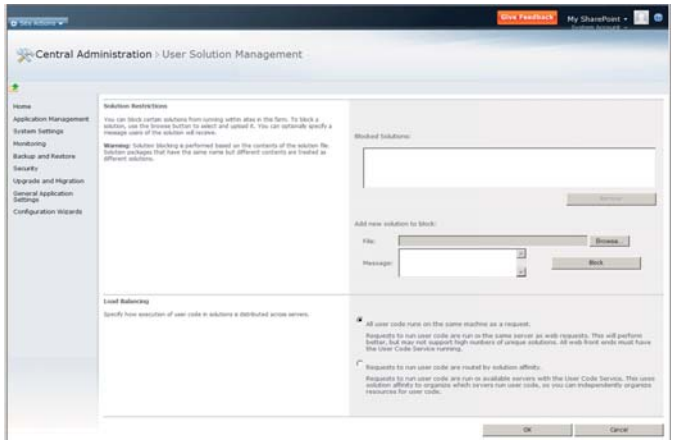
Wat wel en wat niet!

Omdat Sandboxed Solutions niet op Farm-niveau uitgerold worden, maar op Site Collection Niveau kan niet alles opgenomen worden in de solution. Er kan bijvoorbeeld niets uitgerold worden in de 14 hive en dit betekent dus, geen Application Pages, geen Site Definitions, geen Resource Files, etc. Op zich niet gek, want in een Shared Hosting omgeving mag de klant niet zomaar bestanden wegschrijven in de 14 hive.

Hier volgt een lijstje van de oplossingen die in een SandBoxed Solution opgenomen kunnen worden:

- Content Types, Site Columns
- Custom Actions
- Declarative Workflows
- Event Receivers, Feature Receivers
- InfoPath Forms Services (not admin-appr)
- JavaScript, AJAX, jQuery, Silverlight
- List Definitions
- Non-visual web parts
- Site Pages

Helaas is de Visual Webpart niet beschikbaar voor Sandboxed Solutions, dit heeft te maken met de manier waarop Visual Webparts geladen worden.



Het beheer van Sandboxed Solutions

Het beheer van de User Code Service en de Sandbox vindt plaats in Central Administration. Hier kan de beheerder quota instellen als blijkt dat de standaardwaarden niet voldoende zijn. Solutions die keer op keer problemen veroorzaken kunnen geblokkeerd worden en de Load Balancing kan ingeregeld worden.

Resources quota's worden gemeten aan de hand van veertien verschillende waarden. Zo is bijvoorbeeld één AbnormalProcessTermination een punt, tien Citical Exceptions een punt en vijf minuten CPU Time een punt. Het mooie is dat deze waarden naar eigen behoeften aangepast kunnen worden. Denk hier wel eerst goed over na, want deze waarden zijn niet willekeurig gekozen door de mensen van Microsoft. Het is dus beter om eerst de quota-instellingen voor Sandboxed Solutions aan te passen.

De beheerder kan ook de Load Balancing instellen voor de User Code Service. Voor Load Balancing kan er gekozen worden uit twee opties: Local Mode en Remote Mode. Bij Local Mode wordt alle User Code uitgevoerd op de server waar het request op binnen is gekomen. Dit zal qua performance iets beter werken, maar kan problemen opleveren als er veel Solutions zijn. Bij Local Mode moet de User Code Service op alle web front-end servers gestart zijn. Bij de optie Remote Mode zal de User Code uit gevoerd worden op een beschikbare server waar de User Code Service op gestart is.

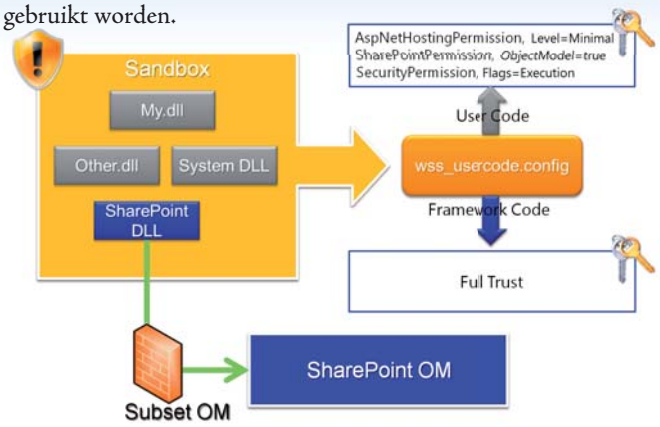
Code Access Security en de fully-trusted Proxy

Nog even terug naar het Sandbox Worker Process. Dit proces heeft namelijk ook zijn eigen Code Access Security (CAS) Policy. De CAS policy heeft een API blocklist en hiermee kan de beheerder de beschikbare API's nog verder beperken. Let op, de standaard Subset van Microsoft.SharePoint namespace kan niet verder beperkt worden. De CAS policy wordt bijgehouden in de file '%Program Files%\Common Files\Microsoft Shared\web server extensions\14\CONFIG\wss_usercode.config'.
In deze CAS Policy worden standaard de volgende drie permissies verleend:

- SharePointPermission.ObjectModel
- SecurityPermission.Execution
- AspNetHostingPermission = Minimal

Op het eerste gezicht lijkt dit redelijk kansloos, want met zo'n permissie set kom je niet veel verder dan een 'Hello World' webpart. Neem bijvoorbeeld een webpart die weer informatie van het internet op wil vragen. Klinkt redelijk onschuldig, maar zonder WebPermission gaat het niet lukken. Gelukkig snappen de mensen van Microsoft dat ook en is er een oplossing voor dit probleem bedacht. Het is namelijk mogelijk om een fully-trusted Proxy Class te schrijven om toegang te krijgen tot resources die

buiten de policy vallen. Een Proxy Class moet op Farm-niveau uitgerold worden en kan vervolgens door iedereen binnen de farm gebruikt worden.

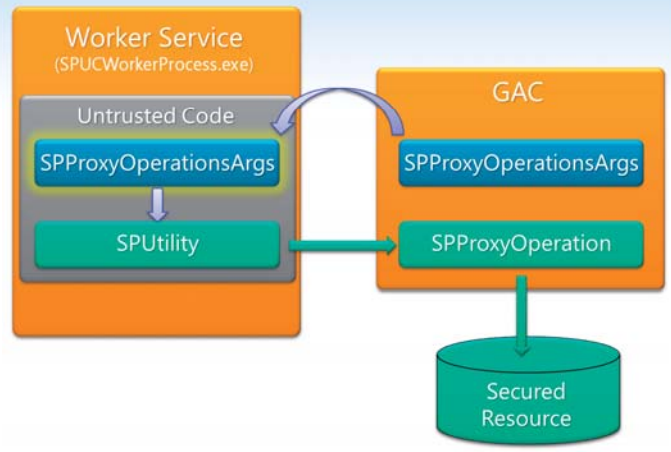


De code van een fully-trusted proxy wordt opgebouwd uit twee classes. De eerste class erft van de SPProxyOperationArgs class en de tweede class erft van de SPProxyOperation class. Deze twee SharePoint classses zitten in de Microsoft.SharePoint.UserCode namespace. De class die erft van SPProxyOperationArgs is eigenlijk niks meer dan een class om argumenten te definiëren die doorgegeven moeten worden aan de proxy. Hier volgt een voorbeeld van een SPProxyOperationArgs class.

```
[assembly: AllowPartiallyTrustedCallers()]
namespace FullTrustWeather.ProxyArgs
{
[Serializable]
public class GetWeatherReportProxyArgs : SPProxyOperationArgs
{
  public DateTime ReportDate { get; set; }
public  GetWeatherReportProxyArgs(DateTime ReportDate)
{
  this. ReportDate =  ReportDate;
}
}
}
```

Deze class moet Serializable zijn en AllowPartiallyTrustedCallers moet aangezet zijn, omdat de class tussen Trust Domains gebruikt gaat worden. Verder heeft de class een constructor method en een property. Niet heel erg spannend dus. De class die erft van SPProxyOperation is de ontvanger van de SPProxyOperationArgs.

```
public class GetWeatherProxyOps : SPProxyOperations
{
public override object Execute(SPProxyOperationArgs args)
  {
    if (args != null)
    {
      GetWeatherReportProxyArgs proxyArgs = args as  GetWeather-
      ReportProxyArgs;
      DateTime  reportDate =  proxyArgs. ReportDate;
      List<string> results =  new List<string>;
      // Code here to get weather…..
      return  results;
    }
return null;
}
}
```



In deze class zit een override van de Execute method die als parameter de SPProxyOperationArgs heeft. Voor de zekerheid wordt er gecontroleerd op null waarden en dan wordt de args parameter geparsed naar de GetWeatherReportProxyArgs. Hierna kunnen alle public properties van de GetWeatherReportProxyArgs opgevraagd worden en gebruikt worden voor het ophalen van de data. De execute method heeft een object als returnwaarde. Dit is alles wat er nodig is voor het maken van een fully-trusted Proxy.

De fully-trusted Proxy registreren

Wat er nu nog overblijft, is het registreren van de fully-trusted Proxy, dit kan met PowerShell maar het kan ook via het object model.

```
class ConsoleApp
{
  static void Main(string[] args)
  {
    SPUserCodeService userCodeService = SPUserCodeService.local;
    if(userCodeService  != null)
    {
      SPProxyOperationType getWeatherOperation = new SPProxy-
      OperationType("FullTrustWeather.ProxyOps, version=1.0.0.0,
      Culture=neutral, PublicKeyToken=b00a5c579934e053", "FullTrust
      Weather.ProxyOps.GetWeatherProxyOps");
      userCodeService.ProxyOperationTypes.Add(getWeatherOperation);
      userCodeService.Update();
      Console.WriteLine("Success");
    }
    else
    {
      Console.WriteLine("Failed");
    }
  }
}
```

In dit voorbeeld wordt eerst even gecontroleerd of de User Code Service wel draait op de machine waar de ConsoleApp op gedraaid wordt, daarna wordt de fully-trusted Proxy geregistreerd bij de lokale User Code Service. Als de fully-trusted Proxy geregistreerd is, kan deze als volgt gebruikt worden in bijvoorbeeld een webpart:

```
GetWeatherReportProxyArgs proxyArgs = new GetWeatherReportProxy-
Args(DateTime.Now);
List<string> weatherList = SPUtility.ExecuteRegisteredProxy-
Operation("FullTrustWeather.ProxyOps, version=1.0.0.0,
Culture=neutral, PublicKeyToken=b00a5c579934e053", "FullTrust-
Weather.ProxyOps.GetWeatherProxyOps", proxyArgs ) as List<string>;
```

Solution Validation

Een laatste belangrijke mogelijkheid is dat er naast de mogelijkheid om solutions te blokkeren ook de mogelijkheid is om solutions te valideren voordat ze toegelaten worden. In de namespace Microsoft.SharePoint.UserCode vinden we de class SPSolutionValidator. Deze class biedt de mogelijkheid voor validatie.

```
[GuidAttribute("34805697-1FC4-4b66-AF09-MAVENTIOND97")]
public class PublisherValidator : SPSolutionValidator
{
    [Persisted]
    List<string> _allowedPublishers;

    public override void ValidateSolution(SPSolutionValidation-
Properties properties){
    }
    public override void ValidateAssembly(SPSolutionValidation-
Properties properties,

SPSolutionFile assembly){
    }
}
```

In een class die erft van de Solution Validator kunnen twee methods met een override geïmplementeerd worden. De ValidateSolution method geeft de mogelijkheid om waarden uit de Manifest.xml te controleren en met de ValidateAssembly kunnen de aanwezige assembly files gecontroleerd worden. De mogelijkheden voor validatie zijn hier dus vrijwel oneindig. Je zou bijvoorbeeld in de ValidateAssembly method de SharePoint Dispose Checker Tool kunnen aanroepen om te controleren of voor bepaalde SharePoint objecten netjes de Dispose method aangeroepen wordt.

Wanneer wel en wanneer niet

Met een Sandbox Solution heb je niet dezelfde feature set als met een Solution die op Farm-niveau uitgerold wordt. Moet er iets weggeschreven worden in de 14 hive zoals een Site Definition dan is een Sandboxed Solution geen optie. Komt er custom code aan te pas dan is de afweging: ga je voor optimale performance of ga je voor stabiliteit en veiligheid?

De Sandbox zorgt voor stabiliteit en veiligheid, maar er komt wel een kleine performance penalty bij. Performance wordt pas echt een zwaar punt in situaties waar er sprake is van grote aantallen gebruikers die bijvoorbeeld allemaal dezelfde homepage hebben met daarop een webpart. Het zou niet goed zijn als er meer Application Servers met de User Code Service in de Farm zitten als web front-ends. De CAS Policy kan ook een show stopper zijn bij de keuze tussen Sandbox of Farm Solution.
Het schrijven van een fully-trusted Proxy is zeker aan te bevelen, want alles wat in de Sandbox draait kan de stabiliteit van de Farm niet beïnvloeden. Zit er geen custom code in je Solution, maar bijvoorbeeld alleen Features, Site Actions, Content Types en Columns dan kun je in elke SharePoint Farm prima uit de voeten met de Sandbox, ongeacht het aantal gebruikers.

Robert Jaakke, werkt bij Mavention en is Microsoft Certified Master voor SharePoint 2007. Hij is bereikbaar via email robert.jaakke@mavention.nl.