Pieter Bos en Sven Vintges

De content van de website vanuit een database

DYNAMISCHE WEBSITES MET DE VIRTUAL PATH PROVIDER

Verhalen doen de ronde dat de Virtual Path Provider (kortweg VPP) speciaal voor het SharePoint-team van Microsoft is ontwikkeld. Toch blijkt uit alles dat dit niet een quick-and-dirty-oplossing is voor een probleem bij het Microsoft-ontwikkelteam, maar dat hij ook gebouwd is om binnen webapplicaties te kunnen gebruiken.

Stel: geen honderden losse bestanden meer op de harde schijf; de mogelijkheid om de mapstructuur van een website om te gooien zonder een langdurig migratietraject; de content van de website beschikbaar via één url in meerdere talen; de content beschikbaar maken door middel van meerdere url's zonder hiervoor complexe structuren op te bouwen; de hele website-content deployen door een database te plaatsen; of een zip-file naar de server te kopiëren. De Virtual Path Provider van ASP.NET 2.0 maakt dit allemaal mogelijk.

Binnen Internet Information Server (IIS) is het mogelijk om te werken met virtuele directories. Deze virtuele directories kunnen verwijzen naar een locatie buiten de website, zoals een andere directory of een url. Maar stel nu dat de geboden content uit een database moet worden gehaald. Dit kan met de Virtual Path Provider, omdat je daarmee in ASP.NET 2.0 pagina's van een andere locatie dan het file-systeem kunt halen, zoals een database, zip-file of ieder ander medium dat bedacht kan worden.
De Virtual Path Provider levert een aantal classes waarmee een virtueel bestandssysteem kan worden geïmplementeerd. In een virtueel bestandssysteem worden mappen en bestanden niet beheerd door het operating systeem, maar door een datastore zoals een (SQL Server) database of een zip-file. In ASP.NET 1.1 werd de System.IO-namespace door ASP.NET gebruikt om het bestandsysteem direct te benaderen. In ASP.NET 2.0 wordt de klasse MapPathBasedVirtualPathProvider daarvoor gebruikt. Deze klasse maakt op zijn beurt weer gebruik van de MapPathBasedVirtualPathFile die het bestandssysteem benadert via het object System.IO.FileStream. De MapPathBasedVirtualPathProvider, MapPathBasedVirtualDirectory en MapPathBasedVirtualFile zijn een implementatie van de abstracte klassen VirtualPathProvider, VirtualPathDirectory en VirtualPathFile. Het is in ASP.NET mogelijk een eigen VPP te bouwen op basis van deze drie abstracte klassen.

Ook in Windows SharePoint Services wordt een database gebruikt voor de implementatie van een virtueel bestandsysteem. In WSS V2.0 werd daarvoor nog een ISAPI-filter gebruikt. Het ISAPI-filter bepaalde of een pagina in WSS uit de database geladen moest worden, of van het bestandssysteem. Pagina's die vanuit de database worden geladen, worden unghosted pagina's genoemd en worden door de Safemode-parser van WSS gehaald. Pagina's die worden geladen van het bestandsysteem, worden ghosted pagina's genoemd en door de ASP.NET-compiler gehaald. In WSS V3.0 is het ISAPI-filter vervangen door een Virtual Path Provider, en de Safemode-parser is komen te vervallen. ASP.NET 2.0 heeft nu een CompilationMode-property waarmee kan worden aangegeven of een pagina runtime gecompileerd kan worden.

MSDN

Op Channel9 zijn vele interessante architectuurpodcasts te vinden van Ron Jacobs. Eén daarvan gaat over de MSDN-site en de Virtual Path Provider. Mark D'Urso van het MSDN-team legt uit hoe de nieuwe MSDN-website (MSDN2.microsoft.com) gebruikmaakt van één van de nieuwe features van ASP.NET, te weten: de Virtual Path Provider. Informatie over een bepaalde namespace of klasse is te vinden door de naam van de namespace of klasse gewoon in te typen in de url. Informatie over de System.IO.StreamWriter kan bijvoorbeeld gevonden worden door de volgende url in te typen: http://msdn2.microsoft.com/System.IO.StreamWriter. Afhankelijk van de Windows-instellingen wordt de gebruiker met behulp van URL-rewriting doorgestuurd naar de pagina die de beschrijving van de System.IO.StreamWriter in de voorkeurstaal weergeeft, en voor de laatste versie van het .NET framework.

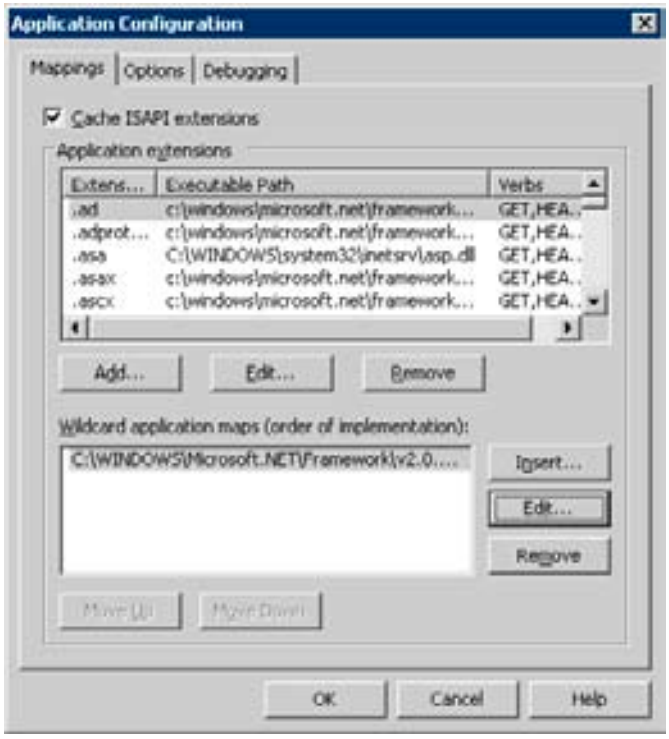
Zelf een implementatie maken

In deze paragraaf wordt dieper ingegaan op de VPP-klassen, en hoe deze gebruikt kunnen worden om jouw eigen implementatie te maken. Allereerst zal aan de ASP.NET-runtime bekend moeten worden gemaakt dat er een custom virtual path provider is geïmplementeerd voor de website. Dit kan door de virtual provider te registreren in een van de life-cycle start-up events in de global.asax-file (codevoorbeeld 1) of in de methode AppInitialize.
De ASP.NET-runtime zal nu bij iedere request aan de VPP vragen of het een virtual file is of niet. Als dit niet het geval is, zal er op de reguliere manier met de file worden omgegaan (hosting vanuit de standaard ASP.NET-runtime). Hier zit nog wel een

```
public class Global : System.Web.HttpApplication
{
    protected void Application_Start(object sender, EventArgs e)
    {
      HostingEnvironment.RegisterVirtualPathProvider(
        new DemoVirtualProvider());
    }

    protected void Application_End(object sender, EventArgs e)
    {
    }
}
```

Codevoorbeeld 1. Het registreren van de virtual provider



Afbeelding 1. Wildcard application mappings

detail in voor IIS 6 en hoger. Vanaf versie 6 van de Internet Information Services (Windows 2003) is de hosting van statische bestanden gewijzigd. Zo zal de hosting er voor zorgen dat static content niet via de asp.net-runtime wordt geleverd. Indien er gekozen is om bijvoorbeeld txt-bestanden of doc-bestanden in het virtual file-system te laden, zal moeten worden gezorgd dat de bestanden wel door de ASP.NET-runtime worden gehost. In IIS 6 wordt dat als volgt gerealiseerd:

- Open de IIS-manager en klik met de rechtermuisknop op de website.
- Vraag de properties op en ga naar het tabblad 'Document'.
- Kies vervolgens de knop 'Configuration' en dan het tabblad 'Mappings'. Hier zal een 'Wildcard application' aan moeten worden toegevoegd.
- Voeg met de 'Insert...'-knop de fileC:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll toe. Vink 'Verify that file exists' uit; zie afbeelding 1.

Meer informatie over 'Wildcard application mappings' is te vinden op de website van Microsoft: http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/ad520e25-877c-4764-bfe5-a9d5a9a5d3bb.mspx?mfr=true. Voor de virtual path provider is een aantal klassen aanwezig die kunnen worden geïmplementeerd. Het betreft de VirtualPathProvider, VirtualDirectory en VirtualFile. De VirtualPathProvider is de class die, zoals eerder genoemd, geregistreerd zal worden in de ASP.NET-runtime. De classes dienen in de App_Code of een assembly te staan. Allereerst wordt in de VirtualPathProvider de methode IsFolderVirtual geïmplementeerd; zie codevoorbeeld 2. Deze methode zorgt voor de initiële communicatie met de ASP.NET-runtime om te bepalen of het bestand in het pad van het virtuele bestandsysteem voorkomt. Als dit het geval is, zal de ASP.NET-runtime de afhandeling van de content via de VPP laten lopen. Is dit niet het geval, dan zal de reguliere manier van afhandeling plaatsvinden. Dit kan ASP.NET zijn voor de aspx-files, maar het zou ook een andere manier van afhandeling kunnen zijn.
Nu de ASP.NET-runtime weet dat het path op zijn minst virtual is, wordt bepaald of de gevraagde content ook virtual is. Dit gebeurt

```
private bool IsFolderVirtual(string virtualPath)
{
   string checkPath = VirtualPathUtility.GetDirectory(virtualPath);
   return utility. IsFolderVirtual(checkPath);
}

//In VirtualPathUtility class
/* Controleert of de folder virtueel is, met andere woorden in de
* database path aanwezig is dat voldoet aan de opgegeven parameter
*/
public bool IsFolderVirtual(string virPath)
{
   bool pathIsVirtual;
   string sQuery = String.Format("select count(0) from Folders where
path = \"{0}\" or path & \"/\" = \"{0}\"", virPath);
   pathIsVirtual = ((Int32)this.ExecScalar(sQuery) != 0);
   return pathIsVirtual;
}
```

Codevoorbeeld 2. De implementatie van de methode IsFolderVirtual

```
/*
 * FileExists controleert of het bestand werkelijk
 * aanwezig is op het virtuele filesysteem.
 * Hiervoor wordt de hulpklasse utility gebruikt.
 */
public override bool FileExists(string virtualPath)
{
   if (IsFolderVirtual(virtualPath))
   {
      if (utility.CheckIfFileExists(virtualPath))
         return true;
      else
         return Previous.FileExists(virtualPath);
   }
   else
   {
      return Previous.FileExists(virtualPath);
   }
}
```

Codevoorbeeld 3. De overload van FileExists

```
/*
 * DirectoryExists bepaalt of een specifieke
 * directory bestaat op het virtuele filesystem m.b.v
 * GetDirectory.
 */
public override bool DirectoryExists(string virtualDir)
{
   if (IsFolderVirtual(virtualDir))
   {
      DemoVirtualDirectory dir = (DemoVirtualDirectory)
GetDirectory(virtualDir);
      return true;
   }
   else
      return Previous.DirectoryExists(virtualDir);
}

/*
 * De GetDirectory functie retourneert een
 * VirtualDirectory aan de run-time.
 */
public override VirtualDirectory GetDirectory(string virtualDir)
{
   if (utility.IsFolderVirtual(virtualDir))
      return new DemoVirtualDirectory(virtualDir, this);
   else
      return Previous.GetDirectory(virtualDir);
}
```

Codevoorbeeld 4. Voor DirectoryExists geldt hetzelfde implementatieprincipe

```
using System;
using System.Collections;
using System.Web.Hosting;

namespace AtosOrigin.VPPDemo.Presentation.Hosting
{
  public class DemoVirtualDirectory : VirtualDirectory
  {
    DemoVirtualProvider spp;

    private ArrayList children = new ArrayList();
    private ArrayList directories = new ArrayList();
    private ArrayList files = new ArrayList();

    public DemoVirtualDirectory(string virtualDir,
 DemoVirtualProvider provider) : base(virtualDir)
{ spp = provider; }

    public override IEnumerable Children
    {
      get
      {
        return children;
      }
    }

    public override IEnumerable Directories
    {
      get
      {
        return directories;
      }
    }

    public override IEnumerable Files
    {
      get
      {
        return files;
      }
    }
  }
}
```

Codevoorbeeld 5. De implementatie van de Virtual Directory

door middel van de overload van FileExists; zie codevoorbeeld 3. Wanneer de huidige (this) Virtual Path Provider niet kan bepalen dat de content virtual is (IsFolderVirtual(virtualPath geeft false als return-waarde), zal de vorige VPP in de keten worden geraadpleegd. Dit gebeurt door middel van de klasse-member Previous. Op deze manier kan een keten van Virtual Path Providers geïmplementeerd worden; meerdere virtual path providers in één ASP.NET-runtime. Voor DirectoryExists geldt hetzelfde implementatieprincipe; zie codevoorbeeld 4.
Nadat bepaald is dat de content virtual is, zal de content gehost moeten worden door middel van de methode GetFile en/of GetDirectory die respectievelijk een VirtualFile-object of een VirtualDirectory-object teruggeven. Voor de implementatie van de FileExists, GetFile, DirectoryExists en GetDirectory maken we twee specifieke implementaties die VirtualDirectory en VirtualFile implementeren. De implementatie van de Virtual Directory ziet er als volgt uit; zie codevoorbeeld 5. De implementatie van de VirtualFile is redelijk eenvoudig. Hiervoor extenden we enkel de VirtualFile-klasse uit het .NET Framework; zie codevoorbeeld 6. Een van de belangrijkste methoden van deze class is de Open-methode, die een stream teruggeeft aan de runtime met daarin de content van de file. Dit kan dus een memorystream, text-stream of een eigen implementatie van een stream zijn. De ASP.NET-runtime zorgt er voor dat een specifieke aspx-pagina (of andere content die door de runtime wordt gecompileerd) wordt gecached. Op deze manier is het niet nodig dat de runtime telkens de pagina's opnieuw compileert. Voor het virtuele path geldt ook dat dit niet nodig is, uit het oogpunt van performance. Daarom zijn er nog twee zeer belangrijke, maar moeilijk te begrijpen methoden, GetCacheDependency en GetFileHash. In de demo-applicatie is aan deze methoden geen specifieke invulling gegeven, maar is de implementatie van deze methoden overgelaten aan de base-class.
Om de werking van een VPP echt goed te kunnen begrijpen, is achtergrondkennis van de implementatie van ASP.Net en specifiek de MapBasedVirtualProvider vereist. Deze VPP is dé standaard virtual path provider van Microsoft. Deze provider zorgt er voor dat de virtual paden worden gemapped naar bestanden op het fysieke filesysteem van de server. Voor de implementatie van deze provider zijn ook implementaties gemaakt van de VirtualFile en VirtualDirectory. Deze twee implementaties van de abstracte klassen gebruiken de namespace System.IO om te bepalen of de bestanden op de harde schijf aanwezig zijn. Dit gebeurt onder andere met behulp van System.IO.Directory.Exists. Belangrijk om te weten is dat er een aantal bestanden niet opgeslagen kan worden in een virtual filesysteem.

Het gaat om de volgende bestanden:
- de global.asax
- de web.config
- de directories bin, app_data, app_code, app_globalresources, app_localresources
- site-mapbestanden van de XmlSiteMapProvider

Het feit dat deze bestanden niet kunnen worden opgenomen in een virtual file-systeem heeft alles te maken met de aard en de rol van deze bestanden in de runtime. Zo is de global.asax de plek waarin de runtime geconfigureerd kan worden met behulp van de life-cycle-events. Doordat de virtual path provider geregistreerd moet worden in de runtime in een van de start-up-events

```
using System;
using System.Web;
using System.Web.Hosting;
using System.IO;

namespace AtosOrigin.VPPDemo.Presentation.Hosting
{
  public class DemoVirtualFile : VirtualFile
  {
    private DemoVirtualProvider spp;
    private string virPath;

    public DemoVirtualFile(string virtualPath,
DemoVirtualProvider provider) : base(virtualPath)
    {
      this.spp = provider;
      this.virPath = virtualPath;
    }

    public override Stream Open()
    {
      //Set the Response.Contenttype i.e. "image/jpeg"
      HttpContext.Current.Response.ContentType =
spp.GetFileContentType(virPath);
      //Stream the requested content.
      return spp.GetFileContents(virPath);
    }
  }
}
```

Codevoorbeeld 6. Het extenden van de VirtualFile-klasse uit het .NET Framework

(die zich in de global.asax) bevinden, zou zich hier een kip-en-ei-probleem gaan voordoen. De web.config is de configuratie van de website voor ASP.NET. Om deze reden kunnen we deze niet virtueel maken. De app_folders en bin directory zijn voorgecompileerde speciale folders die hier ook buiten vallen. De XmlSiteMapProvider wordt ook gebruikt voordat de virtual path provider in de lucht is.

Vele mogelijkheden

Samenvattend is de virtual path provider een mooie oplossing om problemen met grote hoeveelheden bestanden en customizable pages op te lossen door deze in de database op te slaan. Ook biedt het een eenvoudige toepassing voor gezipte websites. Deze zip-files zouden bijvoorbeeld een wachtwoord kunnen bevatten, waardoor de beheerders van de machine niet zomaar in de pagina's kunnen kijken. Het biedt dus ook mogelijkheden op het gebied van afscherming van de content van websites. Doordat de pagina's ook nog eens compileerbaar zijn, kunnen we hier erg veel mee.

Pieter Bos is Lead Architect en **Sven Vintges** is Architect bij Atos Origin Nederland B.V. Beiden hebben jarenlange ervaring met het ontwikkelen en onderhouden van .NET-applicaties. Pieter Bos is te bereiken via Pieter.Bos@atosorigin.com en Sven Vintges is te bereiken via Sven.Vintges@atosorigin.com.

Referenties

Virtualizing Access to Content: Serving Your Web Site from a ZIP File: http://msdn2.microsoft.com/en-us/library/aa479502.aspx

ASP.NET Virtual path provider: http://channel9.msdn.com/ShowPost.aspx?PostID=170643

http://support.microsoft.com/kb/910441