Bob Swart is een onafhankelijk ontwikkelaar, auteur, trainer, webmaster en consultant

SharePoint webparts bouwen in Delphi for .NET

ZELF EXTRA FUNCTIONALITEIT ONTWIKKELEN

Na installatie van Microsoft Windows SharePoint Services 2003 kun je in Delphi 2005 webparts bouwen en integeren in de SharePoints-omgeving. In dit artikel laat ik zien waar je als Delphi-ontwikkelaar rekening mee moet houden en hoe je eenvoudig een aantal webparts kunt bouwen, onder meer aan de hand van een Provider en Consumer-combinatie.

Microsoft Windows SharePoint Services 2003 kan geïnstalleerd worden op Windows Server 2003 en biedt de mogelijkheid websites te bouwen waar gebruikers in teamverband informatie en documenten kunnen delen en uitwisselen. SharePoint Services 2003 kan gratis gedownload worden; je hebt verder alleen Windows Server 2003 nodig. Bovenop SharePoint Services 2003 kun je ook nog Microsoft Windows Sharepoint Portal Server 2003 aanschaffen en installeren; deze is niet gratis. Hierbij krijg je een aantal templates en hulpmiddelen voor het bouwen van gehele webportals. Zowel de SharePoint Services 2003 als Portal Server 2003 sites kunnen uitgebreid worden met in te pluggen bouwstenen - een soort ASP.NET custom controls - webparts genaamd. Hoe bouw je met Borland Delphi 2005 je eigen webparts en hoe deploy je die in een SharePoint Services 2003-omgeving of een Portal Server 2003-omgeving?

Zoals ik boven al beschreef heb je voor SharePoint Services 2003 een versie van Windows Server 2003 nodig, alsmede IIS 6.0 met ASP.NET enabled. Bij installatie wordt optioneel ook een versie van SQL Server 2000 voor je geïnstalleerd als je die nog niet hebt. De WMSDE – net als MSDE, maar dan zonder de beperking van vijf concurrent-connecties of een database van 2 GB die MSDE wel heeft. Omdat de Microsoft Windows SharePoint Services 2003 gratis is, maar de SharePoint Portal Server 2003 niet, zal ik voor de voorbeelden uitsluitend gebruik maken van de SharePoint Services 2003. Om de webparts te kunnen demonstreren heb ik een virtual server als demo sharepoint team website ingericht die staan op http://localhost:88.

Delphi 2005 webparts

In principe is elke versie van Delphi 2005 bruikbaar; ik heb het niet meer met Delphi 8 for .NET geprobeerd. Webparts kun je deployen in een .NET assembly. Met Delphi 2005 betekent dat dus een Package; via File | New – Other, en dan uit de Delphi for .NET projects kiezen voor een Package. Om de benodigde SharePoint-types, classes en methodes te kunnen gebruiken, moet de Microsoft.SharePoint assembly via “Add Reference” aan de “Requires”-lijst van het package-project toegevoegd worden. De Microsoft.SharePoint assembly bevat een aantal belangrijke namespaces die we aan de uses clause van onze SharePoint webpart-units moeten toevoegen, zoals Microsoft.SharePoint.WebPartPages, Microsoft.SharePoint.Utilities en Microsoft.SharePoint.WebPartPages.Communication.
Alle webparts die we bouwen moeten afgeleid zijn van de Microsoft.SharePoint.WebPartPages.WebPart-class, en enkele belangrijke methodes overrulen en implementeren. Allereerst is daar de RenderWebPart, die gebruikt wordt om de webpart te renderen, oftewel HTML te genereren met behulp van een HtmlTextWriter. Behalve het renderen van een webpart zelf kan een webpart ook child-controls bevatten. Om dit goed te kunnen moet de webpart wel bij de definitie aangeven de interface INamingContainer te implementeren. Dit is een zogenaamd “marker interface”, wat betekent dat er geen methodes geïmplementeerd hoeven te worden, maar dat de webpart nu gemarkeerd staat als zijnde een container voor subcontrols die alle wel een unieke naam moeten hebben. Zonder de INamingContainer-interface zou het potentieel fout kunnen gaan als er meer instanties van dezelfde webpart in dezelfde SharePoint-pagina (of workspace) gebruikt worden.

Los van de INamingContainer-interface zijn er voor het gebruik van child-controls twee routines van belang: eentje om te implementeren en eentje om aan te roepen. De routine die we moeten implementeren heet CreateChildControls. Het is de plek waar we de child-controls toevoegen aan de webpart. Codevoorbeeld 1 laat zien hoe ik een TextBox-, ListBox- en Button-control toevoeg aan de webpart. Uiteraard moeten de declaraties voor de FMyTextBox, FMyListBox en FMyButton aan de webpart-class zelf al zijn toegevoegd. Dat geldt ook voor de ButtonClick-methode, die aan de FMyButton.Click wordt toegekend. Op een vergelijkbare manier kun je events invullen van alle child-controls. En ook de AutoPostBack-property zal het gewenste effect hebben als je die op True zet, waardoor er automatisch een postback zal plaatsvinden wanneer de bezoeker de inhoud van de betreffende control wijzigt.

Naast de CreateChildControls heet de tweede belangrijke methode: EnsureChildControls. Deze methode hoeven we niet te implementeren, maar moeten we juist aanroepen in de code van de

```
procedure TDemoWebPart.CreateChildControls;
begin
  inherited;
  FMyTextBox := TextBox.Create;
  FMyTextBox.Text := 'Delphi 2005';
  Controls.Add(FMyTextBox);
  FMyListBox := ListBox.Create;
  Controls.Add(FMyListBox);
  FMyButton := Button.Create;
  FMyButton.Text := 'Click me!';
  Include(FMyButton.Click, ButtonClick);
  Controls.Add(FMyButton);
end;
```

Codevoorbeeld 1. Controls toevoegen aan de webpart

```
procedure TDemoWebPart.RenderWebPart(output: HtmlTextWriter);
begin
  EnsureChildControls;
  FMyTextBox.RenderControl(output);
  output.Write(' ');
  FMyButton.RenderControl(output);
  output.Write(' ');
  FMyListBox.RenderControl(output);
end;
```
Codevoorbeeld 2.

```
<?xml version="1.0"?>
<WebPart xmlns="http://schemas.microsoft.com/WebPart/v2">
   <Assembly>DelphiWebParts</Assembly>
   <TypeName>DrBob42.TDemoWebPart</TypeName>
   <Title>Delphi 2005 Web Part</Title>
   <Description>Demo Web Part geschreven met Delphi 2005</Description>
</WebPart>
```
Codevoorbeeld 3.

webpart vlak voor de momenten waarop we de child-controls daadwerkelijk gebruiken. Dit kunnen we bijvoorbeeld doen in de RenderWebPart-methode; te zien in codevoorbeeld 2, waarbinnen we nu voor ieder child-control de methode RenderControl moeten aanroepen. Rondom de child-controls kunnen we zelf de HTML genereren om de child-controls in "op te bergen".

De volledige broncode van de TDEmoWebPart is overgens te downloaden van de website van .NET Magazine of vanaf www.eBob42.com.

Deployment

Het deployen van een webpart bestaat uit twee stappen. Allereerst moeten we een .dwp-bestand maken met daarin de informatie voor de webpart, zoals de naam van de assembly, de naam van de class en extra (visuele) informatie zoals de titel en de omschrijving van de webpart. Deze laatste informatie is straks op het scherm terug te zien. Stel dat we de assembly de naam DelphiWebParts geven en dat de webpart van type TDemoWebPart in de unit DrBob42.WebPart.pas zit. Dan is de bijbehorende namespace DrBob42 (in Delphi 2005). Het uiteindelijke .dwp-bestand voor deze eerste demo-webpart is te zien in codevoorbeeld 3.

Het .dwp-bestand kunnen we samen met de gecompileerde .NET assembly in de bin-directory van de SharePoint virtual server directory zetten. Dit is echter nog niet voldoende, want we moeten nu ook nog SharePoint vertellen dat de types in de assembly veilig zijn om te gebruiken. Hiervoor moeten we de web.config wijzigen die in de SharePoint virtual server root staat. In deze web.config moeten we een nieuwe SafeControl-regel toevoegen, die er als volgt uitziet.

```
<SafeControl Assembly="DelphiWebParts"
 Namespace="DrBob42" TypeName="*" Safe="True" />
```

Een * bij TypeName geeft aan dat alle part-types in de aangegeven assembly en namespace veilig zijn om te gebruiken. Je kunt natuurlijk ook alleen maar de types toevoegen die je echt wilt (laten) gebruiken.

Webparts toevoegen

Het daadwerkelijk toevoegen van een webpart aan een SharePoint Workspace gaat met behulp van de "Modify This Workspace"-optie. Het zal duidelijk zijn dat ik de Engelstalige versie van SharePoint Services 2003 gebruik. Klik hierna op Create List en kies voor Import. Je kunt dan via een browse-dialoog het webpart-bestand .dwp-definitie laden. Dit kun je vervolgens ergens op de workspace toevoegen zoals in afbeelding 1 te zien is.
Zodra je het "Add Web Parts"-deel weer sluit, krijg je de nieuwe workspace te zien, inclusief het nieuwe Delphi 2005 webpart. Je kunt de assembly waar de webpart(s) in zit uiteraard ook signen met een strong name en dan deployen in de Global Assembly Cache, net als iedere andere .NET assembly.

Events

Als ontwikkelaar is het fijn om te weten dat er verschillende events zijn waar een webpart op reageert. In de juiste volgorde is de lijst van de belangrijkste events als volgt:

1. Create
2. OnInit
3. OnLoad
4. CreateChildControls
5. OnPreRender
6. RenderWebPart

Een verschil met ASP.NET controls is dat de IsPostBack al True is als je voor de eerste keer in een pagina komt tijdens runtime. Maar de ViewState bevat op dat moment nog geen "inhoud" van de controls. Het is dus niet langer veilig om (alleen) op IsPostBack te controleren als je bijvoorbeeld een DropDownList wilt vullen met items. Iets om wellicht rekening mee te houden in praktijk.

Trace

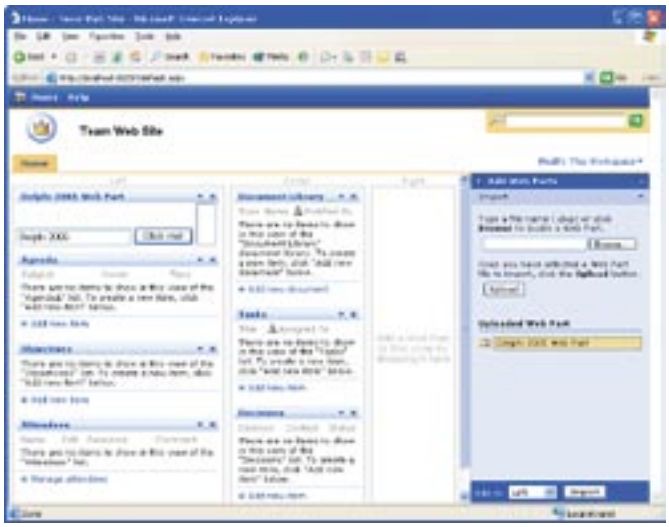
Het is ook mogelijk om Trace te gebruiken via Context.Trace. Op die manier kun je Write en Warn-berichten naar de Trace.axd log schrijven. Je moet hiervoor wel de web.config aanpassen en vlak voor de </system.web> tag het volgende toevoegen:

```
    <trace enabled="true" pageOutput="false" localOnly="true" />
```

Door localOnly op true te zetten zorg je ervoor dat je alleen berichten in Trace.axd zult zien als je direct bent ingelogd op de Windows Server 2003-machine. Andere bezoekers van de site zullen dit niet kunnen.

Provider - Consumer

Behalve dat webparts child-controls kunnen bevatten, kunnen we ook speciale webparts bouwen die verbonden kunnen worden met andere webparts. Dit doen we in een zogenaamde provider – consumer-opzet, waarbij de provider-webpart een datacell aanbiedt aan een consumer webpart. Dit kan door gebruik te maken van twee voorgedefinieerde interfaces: ICellConsumer en ICellProvider. Deze keer bevatten ze wel een aantal methodes die we moeten implementeren. Dat lijkt eenvoudiger dan gezegd, want een aantal methodes is eigenlijk access-methodes voor events; CellProviderInit en CellReady aan de Providerkant en CellConsumerInit aan de Consumerkant. De benodigde broncode om de events in de



Afbeelding 1. Delphi 2005 webpart tijdens design time

```
strict private
  FCellProviderInit: CellProviderInitEventHandler;
  FCellReady: CellReadyEventHandler;
public
  procedure set_CellProviderInit(const Value:
   CellProviderInitEventHandler);
  procedure set_CellReady(const Value: CellReadyEventHandler);
published
  property CellProviderInit: CellProviderInitEventHandler
    read FCellProviderInit write set_CellProviderInit;
  property CellReady: CellReadyEventHandler
    read FCellReady write set_CellReady;
end;
```

Codevoorbeeld 4.

```
strict private
  FCellConsumerInit: CellConsumerInitEventHandler;
public
  procedure set_CellConsumerInit(value: CellConsumerInitEventHandler);
  property CellConsumerInit: CellConsumerInitEventHandler
    read FCellConsumerInit write set_CellConsumerInit;
end;
```

Codevoorbeeld 5.

```
public
  procedure PartCommunicationInit; override;
  procedure PartCommunicationMain; override;
  procedure PartCommunicationConnect(interfaceName: string;
    connectedPart: WebPart; connectedInterfaceName: string;
    runAt: ConnectionRunAt); override;
public
  procedure EnsureInterfaces; override;
  function CanRunAt: ConnectionRunAt; override;
```

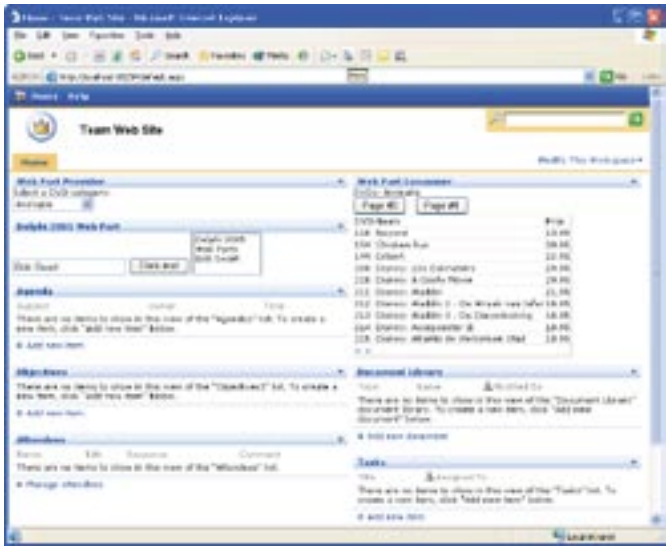
Codevoorbeeld 6.

ICellProvider-interface correct te implementeren is te zien in codevoorbeeld 4.

De broncode die nodig is om de events in de ICellConsumer-interface correct te implementeren is te zien in codevoorbeeld 5.

Daarnaast bevatten zowel de ICellProvider- en ICellConsumer-interface methodes om de communicatie en het doorgeven van data te verzorgen; zie codevoorbeeld 6.

Voor de daadwerkelijke implementatie van deze methodes verwijs ik naar de broncode bij dit artikel. Hierin heb ik als voorbeeld een kleine databasetoepassing gebouwd – of eigenlijk twee webparts die gebruik maken van een database die via een webservice toegankelijk wordt gemaakt. In beide gevallen zal dit niet zonder meer werken vanwege onvoldoende trust level.



Afbeelding 2. Delphi 2005 webpart tijdens runtime

Trust Levels

Omdat webparts een beperkte bewegingsvrijheid hebben, is er standaard geen recht om met databases te werken, file I/O te doen en om met het internet te communiceren. Als je dat wel wilt (laten) doen door de webparts, zul je de security-instellingen moeten aanpassen. Dit kan in de web.config, waar het default trust level op WSS_Minimal zal staan. Voor bijvoorbeeld SQL Server-toegang of File I/O-mogelijkheden moeten we dit trust level op WSS_Medium zetten. Dit gaat als volgt:

```
<trust level="WSS_Medium" originUrl="" />
```

Wie ook nog met het internet wil communiceren, bijvoorbeeld om een webservice aan te roepen (zoals ik doe in mijn voorbeeld), zal het trust level zelfs op Full moeten zetten. Tenzij de URL van de webservice van dezelfde host komt. In de praktijk is het echter aan te raden om de betreffende assembly te signen met een strong name, te deployen in de Global Assembly Cache, en aldaar een hoger trust level toe te kennen. Als derde alternatief bestaat de mogelijkheid een custom policy-bestand voor de betreffende assembly te maken, waarin ook het specifieke trust level gedefinieerd kan worden.

In afbeelding 2 is te zien dat de consumer webpart (aan de rechter bovenkant van het scherm) gevoed wordt door de provider webpart (linksboven). Hier kan ik de category van mijn DVD-verzameling bepalen. De gekozen category wordt doorgegeven van de provider aan de consumer, die vervolgens een nieuwe selectie uit een database (of in dit geval een webservice) kan halen. De kracht van webpart providers en consumers zit hem in de mogelijkheid dat een provider ook aan andere consumers gekoppeld kan worden; bijvoorbeeld een kalender-provider met een gekozen dag of periode, die als input gebruikt kan worden voor consumer webparts die dan gebruik kunnen maken van de geselecteerde dag of periode.

RAD

Microsoft Windows SharePoint Services 2003 is een krachtig hulpmiddel om teamwebsites te bouwen en is uitbreidbaar met inklikbare onderdelen die webparts heten. Ook met Delphi 2005 kunnen we eigen webparts bouwen en deployen, waarbij ik voorbeelden met child-controls, events, tracing en het provider-consumermodel heb laten zien. Het enige dat in Delphi 2005 ontbreekt is een manier om de child-controls op een visuele manier te ontwerpen: het plaatsen van de individuele child-controls vindt plaats op een non-visuele manier in de RenderWebPart, waarin de child-controls stuk voor stuk op hun plek gezet moeten worden. Het zou fijn zijn als er een soort frame of panel-designer was, vergelijkbaar met user controls, die we kunnen gebruiken op een RAD-manier bij het bouwen van onze eigen webparts.

Bob Swart Bob@eBob42.com - www.eBob42.com - is een onafhankelijk ontwikkelaar, auteur, trainer, webmaster en consultant werkzaam als de eenmanszaak Bob Swart Training & Consultancy (eBob42) in Helmond. Bob is een internationale spreker op Borland en Delphi Conferenties sinds 1993 en is groot fan van Delphi en .NET – in het bijzonder ASP.NET. Bob schrijft columns en artikelen voor onder andere The Delphi Magazine, SDN Magazine en hij is de auteur van Borland's officiële Delphi 8 for .NET en ASP.NET Essentials trainingsmateriaal.

Nuttige internetadressen

Overzicht Windows Sharepoint Services: http://www.microsoft.com/netherlands/windowsserver2003/windowssharepointservices.aspx
SharePoint Products and Technologies: http://msdn.microsoft.com/sharepoint
SharePoint Products & Technologies Resource Kit (via Amazon.co.uk): http://www.amazon.co.uk/exec/obidos/ASIN/073561881X/drbobsdelphclini