Wilco Bauwer en Herber de Ruijter zijn werkzaam bij Variomatic.

Webparts

TOOLS IN EEN TOOLBOX

Na het ontwikkelen van enkele ASP.NET webapplicaties, betrapten we onszelf erop dat we in verscheidene gevallen het wiel opnieuw aan het uitvinden waren. We hadden verschillende functionaliteiten al eerder ontwikkeld, die perfect zou kunnen worden hergebruikt in een nieuwe applicatie. Het nadeel met het standaard ASP.NET framework is dat een pagina-developer vaak bepaalde logica opnieuw moet implementeren voor een nieuwe applicatie.

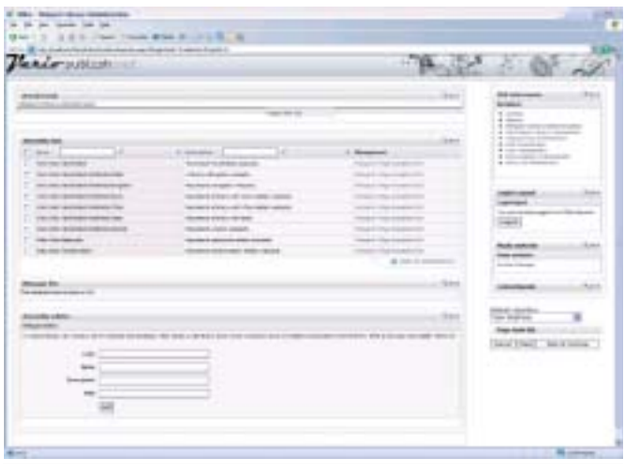
Sharepoint 2003 heeft een webpart-mechanisme dat de eindgebruiker de mogelijkheid geeft om pagina's op te zetten door webparts op de pagina te plaatsen. Webparts hoeven maar één keer te worden geïmplementeerd. Vervolgens kan de eindgebruiker deze webparts gebruiken in iedere Sharepoint 2003 applicatie. Het nadeel van Sharepoint is dat het niet flexibel genoeg was om ingezet te worden voor de applicaties die wij ontwikkelen. Eén van onze eisen was dat we zonder al te veel moeite iedere pagina eruit zouden kunnen laten zien zoals we wilden. Nog een eis was dat we een universeel systeem wilden hebben die het mogelijk maakt om pagina's toe te voegen, iedere pagina te onderhouden, en pagina's te verwijderen. Onze ervaring met Sharepoint is dat het een perfect systeem is, indien de aangeboden functionaliteit voor onder andere document-sharing centraal staat en een beperkte flexibiliteit aanvaardbaar is. Sharepoint is ons inziens minder aantrekkelijk om in te zetten voor het ontwikkelen van een webapplicatie, waar een compleet nieuwe functionaliteit en look & feel centraal staan.

Mogelijkheden van het webpart-mechanisme

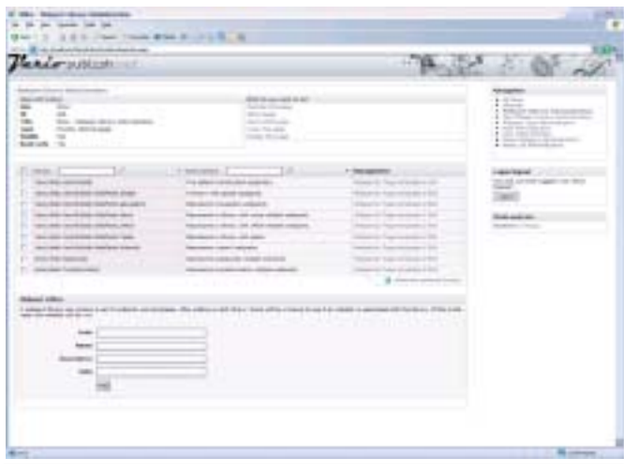
Het huidige ASP.NET framework biedt de developer de mogelijkheid een eigen toolbox van controls te ontwikkelen. Zowel de developer zelf als andere partijen kunnen deze toolbox uitbreiden. Bij het ontwikkelen van een nieuwe applicatie kan op deze manier veel tijd worden gewonnen door de tools in de toolbox te hergebruiken. De tools in de toolbox zijn relatief kleine elementen, die vaak in combinatie met andere elementen in een user control worden gebruikt. De verzameling user controls die dit oplevert, maakt vervolgens deel uit van de toolbox van de pagina-developer. Vervolgens kan een pagina-developer met deze toolbox de verschillende pagina's in elkaar zetten en er voor zorgen dat interactie plaatsvindt tussen de verschillende user controls. Het webpart-mechanisme zorgt ervoor dat de eindgebruiker de rol van een pagina-developer kan overnemen. De eindgebruiker kan de tools in de toolbox via de webinterface verslepen naar één van de voorgedefinieerde zones op de pagina; zie afbeeldingen 1 en 2. Met dezelfde interface kunnen tools op de pagina naar andere posities worden versleept, of verwijderd worden van de pagina. De look & feel van een pagina kan worden gewijzigd door de pagina met een andere pagina-template te associëren.

Bibliotheken met functionaliteit

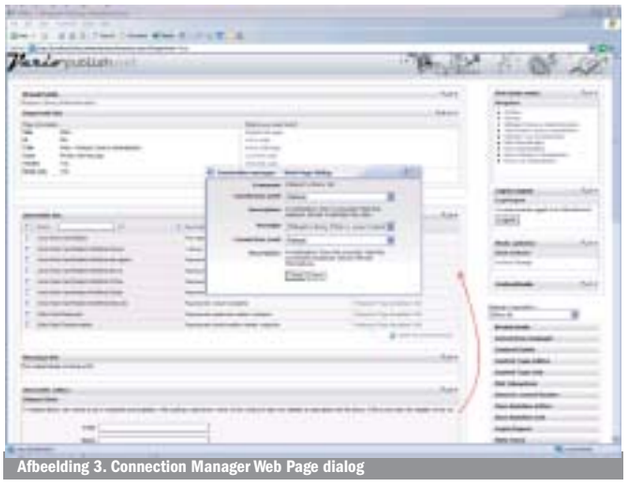
De tools in deze toolbox noemen we een webpart. Een webpart is een deel van een pagina dat verantwoordelijk is voor een bepaalde functionaliteit. De toolbox van de eindgebruiker kan worden uitgebreid door nieuwe webparts te ontwikkelen. Door meer applicaties te ontwikkelen met behulp van webparts, creëer je verschillende bibliotheken die elk voor een bepaalde functionaliteit kunnen zorgen. Dit maakt het mogelijk om bibliotheken die eerder zijn ontwikkeld te hergebruiken voor andere sites waar dezelfde functionaliteit gewenst is. Ten slotte kan een eindgebruiker webparts met elkaar laten communiceren. Iedere webpart kan verbindingspunten hebben. Verbindingspunten van webparts kunnen door de eindgebruiker aan elkaar worden gekoppeld om zo een verbinding te realiseren; zie afbeelding 3. Dit zorgt ervoor dat er nog steeds interactie kan plaatsvinden tussen webparts op een pagina. In combinatie met de mogelijkheid voor een eindgebruiker om dynamisch een webpart te configureren, zorgt het webpart-mechanisme ervoor dat de eindgebruiker op een eenvoudige manier pagina's



Afbeelding 1. Webpart Library Administration



Afbeelding 2. Webpart Library Administration



Afbeelding 3. Connection Manager Web Page dialog

kan beheren, zonder dat daar nog een pagina-developer voor nodig is. Het resultaat hiervan is dat tools in de toolbox van de eindgebruiker op zichzelf staan. De tools zijn 'loosely coupled' en zorgen ervoor dat tools eventueel vervangen kunnen worden door andere tools met een gelijke functionaliteit. Doordat de tools op zichzelf staan, kunnen bugs sneller worden gevonden en opgelost, zonder dat dit gevolgen heeft.

De uitgebreide life-cycle van een pagina

Het systeem dat we hebben ontwikkeld heeft één fysieke pagina. Deze pagina wordt gebruikt om alle pagina-requests af te handelen. Nadat een pagina is opgevraagd, zal deze pagina de opgevraagde URL vertalen naar een pagina-ID in een datastore. Dit gebeurt op het moment dat de pagina wordt geïnitialiseerd. In de standaard methode CreateChildControls wordt vervolgens een template-manager geïnstantieerd. Deze manager zal ervoor zorgen dat de juiste pagina-template wordt ingeladen. Indien de pagina niet in 'design mode' wordt opgevraagd, zal een WebPartMediator door de webparts gaan die op de pagina moeten worden geplaatst.
Als eerste zal de webpart via reflection - zie codevoorbeeld 1 - worden geïnstantieerd. Wanneer er een instantie van een webpart is, zal de WebPartMediator – indien dit in de configuratie van het systeem staat aangegeven – controleren of de webpart überhaupt mag worden ingeladen. Dit wordt gedaan door het WebPartPermission metadata-attribuut op te vragen die op de webpart is toegepast. Indien dit aangeeft dat de huidige gebruiker genoeg rechten heeft om de webpart in te laden, zal de dynamische configuratie van de webpart uit de datastore worden doorgegeven aan de webpart.

Indien de webpart op de pagina mag worden geplaatst, zal de configuratie van de webpart worden opgehaald en doorgegeven aan de webpart. Daarna zal bepaald worden of de webpart één of meer verbindingen heeft met andere webparts. Is dit het geval, dan zal de webpart tijdelijk worden geregistreerd als deelnemer van een verbinding. Een webpart kan worden geregistreerd als twee soorten deelnemers: een informatiegebruiker (consumer), of een informatieaanbieder (provider).

De volgende stap is het toevoegen van de webpart aan de juiste zone in de pagina-template. Dit is de zone waar de eindgebruiker de webpart heeft geplaatst. Wanneer alle webparts op de pagina zijn geplaatst, zal de pagina ervoor zorgen dat de WebPartMediator alle verbindingen tussen de webparts realiseert. Omdat alle deelnemers zijn geregistreerd tijdens het toevoegen van de webparts, kunnen nu de verbindingen voor iedere consumer-webpart één voor één gelegd worden. De verbinding zal gelegd worden tussen de aangegeven verbindingspunten, door een methode PartCommunicationConnect(IConnectionPoint,

```
public WebPart CreateWebPart(string typeName)
{
        // Haal het type van de webpart op aan de hand
        // van de volledige naam van het type.
        Type webPartType = Type.GetType(typeName);

        // Maak een instantie van het type.
        return Activator.CreateInstance(webPartType) as WebPart;
}
```

Codevoorbeeld 1.
Dynamisch een webpart instantiëren

NameValueCollection) van de consumer aan te roepen. Bij het aanroepen van deze methode wordt een referentie naar het provider-verbindingspunt doorgegeven aan het consumer-verbindingspunt. Optioneel zouden er verbindingsparameters kunnen worden meegegeven; dit is echter niet te configureren via de huidige webinterface. Nadat alle verbindingen zijn gelegd, zullen alle verbindingspunten worden geïnitialiseerd. Vanaf dit punt kunnen webparts communiceren met verbonden webparts.

DesignZone

Wanneer iemand met voldoende rechten de pagina in 'design mode' opvraagt, zal de WebPartMediator een DesignZone toevoegen aan iedere zone in de pagina-template. Ook zal een WebPartManager worden toegevoegd met daarin de toolbox van webparts die voor de gebruiker beschikbaar zijn. De WebPartManager zal een verwijzing naar iedere DesignZone krijgen, zodat deze ervoor kan zorgen dat alle wijzigingen op een pagina opgeslagen kunnen worden. De DesignZone zorgt er vervolgens voor dat alle webparts in de desbetreffende zone op een speciale manier worden geladen. Voor iedere webpart wordt een WebPartPanel geïnstantieerd. Een WebPartPanel is een webpart die versleept kan worden op een pagina. Iedere webpart is standaard geassocieerd met een 'designer'-webpart. Een webpart kan gebruik maken van een eigen 'designer' door een nieuw RuntimeDesigner metadata-attribuut toe te passen. Een designer krijgt altijd een verwijzing naar de webpart die het moet representeren. Dit maakt het mogelijk om eventueel de oorspronkelijke webpart weer te geven. Wanneer er een WebPartPanel voor de webpart gemaakt is, zal de geassocieerde 'designer' via reflection worden geïnstantieerd. Vervolgens zal deze 'designer' in het WebPartPanel worden geplaatst.

Communicatie tussen webparts

Een webpart kan verbindingspunten registreren door de methode CreateConnectionPoints() te overerven. Deze methode kan op een soortgelijke manier als CreateChildControls() worden geïmplementeerd; zie codevoorbeeld 2. Het systeem kan deze verbindingspunten opvragen door gebruik te maken van de eigenschap ConnectionPoints. In de PartCommunicationConnect-methode kan een consumer verbindingspunt-eventhandlers toevoegen voor gebeurtenissen die kunnen plaatsvinden bij een provider-verbindingspunt. In de eventhandler zal de gebeurtenis worden afgehandeld door een nieuwe gebeurtenis in het consumer-verbindingspunt te laten plaatsvinden. De webpart kan deze gebeurtenis weer afhandelen. Bij het afhandelen van een dergelijke gebeurtenis, heeft de webpart de beschikking over de informatie die is aangeboden. Dit zou bijvoorbeeld kunnen gaan om zoekcriteria die zijn aangeboden door een webpart met een invoerveld en een knop. Na het drukken op de knop, zou de webpart de ingevoerde tekst kunnen doorgeven aan alle verbonden webparts. Een verbonden webpart zou een lijst met namen van medewerkers kunnen zijn. De webpart met de lijst van namen kan vervolgens de lijst van namen filteren wanneer deze de zoekcriteria ontvangt.

```
protected override void CreateConnectionPoints()
{
   // Maak een verbindingspunt die een cell
   // van een provider verbindingspunt kan gebruiken.
   CellConsumerConnectionPoint searchCriteriaConsume =
   new CellConsumerConnectionPoint("{guid}",
   new CommunicationInformation("Search criteria",
   "Zoek argumenten voor het filteren van de lijst.",
   ConnectionPoint.OneConnection),
   new CellReadyEventArgs(search_Ready));
   this.ConnectionPoints.Add(searchCriteriaConsumer);

   // Maak een verbindingspunt die een notificatie
   // verstuurd naar verbonden webparts van het aantal
   // medewerkers dat in de huidige lijst getoond wordt.
   this.employeeCountProvider = new CellProviderConnectionPoint(
   "{guid}",
   new CommunicationInformation("Employee Count", "# of employees",
   ConnectionCount.UnlimitedConnections));
   this.ConnectionPoints.Add(this.employeeCountProvider);
}

private void search_Ready(object sender, RowReadyEventArgs e)
{
   if (e.Field.Name == "SearchCriteria")
   {
   // Pas een filter toe.
   this.Filter = e.Field.CurrentValue.ToString();
   // Ververs de lijst.
   this.Bind();
   }
}

private void Bind()
{
   // Haal de medewerkers op aan de hand van een filter
   // en toon deze in een ASP.NET control.
   EmployeeCollection employees = this.FindEmployees(this.Filter);
   this.employeeGrid.DataSource = employees;
   this.employeeGrid.DataBind();

   // Stuur een notificatie naar alle verbonden webparts met daarin
   // de hoeveelheid employees die in de lijst getoond worden.
   this.employeeCountProvider.NotifyConsumers(
   new CellReadyEventArgs(new CellItem("EmployeeCount", employees.Count));
}
```

Codevoorbeeld 2.
Verbindingspunten definiëren

Standaard biedt het systeem vier verbindingspunten aan die een webpart kan gebruiken: een CellConsumerConnectionPoint, RowConsumerConnectionPoint, CellProviderConnectionPoint en RowProviderConnectionPoint. Wanneer een webpart een verbindingspunt gebruikt, moet deze een ID, naam, omschrijving en het maximale aantal verbonden verbindingspunten opgeven. De ID moet uniek zijn voor de verbindingspunten binnen een webpart, zodat het systeem een verbindingspunt kan identificeren. De naam en omschrijving beschrijven een verbindingspunt. De eindgebruiker zal dit ook terugzien in de webinterface. Het communicatie-framework in het systeem kan worden uitgebreid met nieuwe verbindingspunten. Dit kan worden gedaan door de IConnectionPoint-interface te implementeren. In Sharepoint 2003 zit een soortgelijk communicatie-framework. Het verschil is echter dat de webparts in Sharepoint 2003 zelf bepaalde interfaces moeten implementeren. Dit resulteert vaak in repeterende code voor verschillende webparts. Dit zou kunnen worden voorkomen door basisklassen te schrijven. Er zijn echter tal van mogelijke combinaties van communicatie-interfaces die kunnen worden geïmplementeerd, waardoor dit geen aantrekkelijke optie is. In ons systeem wordt dit geminimaliseerd door gebruik te maken van verbindingspunten, die door webparts kunnen worden hergebruikt. Een bijkomend voordeel is dat user controls ook gebruik kunnen maken van de verbindingspunten van een webpart. De communicatielogica kan op deze manier door zowel de webpart als de bijbehorende user control worden geregeld.

Webparts en user controls

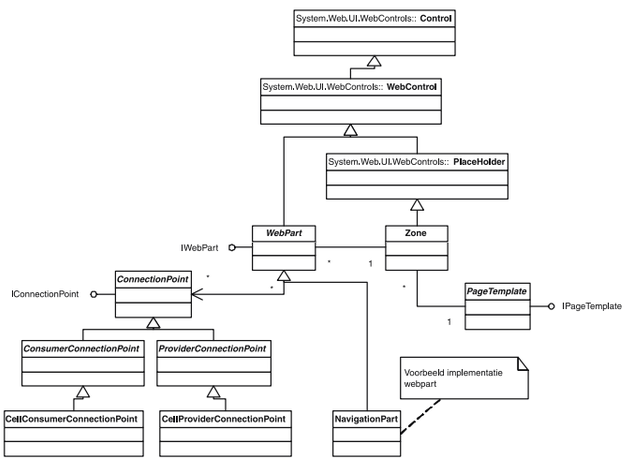
User controls in ASP.NET worden voornamelijk gebruikt voor specifieke functionaliteiten binnen één webapplicatie. Omdat een uitgebreide uitleg van user controls buiten de scope van dit artikel valt, zullen we enkel de voordelen van user controls opnoemen:

- design-time support binnen onder andere VS.NET
- opmaak gescheiden van code
- opmaak op declaratieve manier definiëren.

Deze voordelen hebben een hogere productiviteit tot gevolg. Het is daarom belangrijk dat webparts op dezelfde manier kunnen worden ontwikkeld. Het ASP.NET-model biedt de mogelijkheid om user controls dynamisch in te laden. Dit maakt het weer mogelijk om user controls te ontwikkelen en deze te gebruiken binnen een webpart. Dit betekent dat een webpart een wrapper is tussen het systeem en de user control. De toegevoegde waarde is dat een user control gebruik kan maken van de functionaliteiten die het systeem aanbiedt, waaronder het communicatiemechanisme. Doordat een user control pas werkelijk gecompileerd wordt als een pagina voor de eerste keer wordt opgevraagd, zal het type van de user control niet eerder bekend zijn. Dit betekent dat een webpart – die van te voren gecompileerd wordt – niet het werkelijke type van een user control weet. Een webpart kan dus geen methodes aanroepen op de user control, die niet onderdeel zijn van het type UserControl. Het is daarom noodzakelijk de user control een interface te laten implementeren. Dit maakt het mogelijk om aan de hand van de interface de methodes en eigenschappen van een user control te gebruiken. In een huidige implementatie hebben we een interface ontwikkeld met een methode Initialize(WebPart). Iedere webpart zorgt er dan voor dat op iedere user control deze methode wordt aangeroepen met een verwijzing naar zichzelf. De user control kan vervolgens de referentie casten naar het werkelijke type van de webpart. Er kan dan gebruik gemaakt worden van alle methodes en eigenschapen die het type bevat.

De toekomst

In de volgende versie van ASP.NET zal ook een webpart- en communicatiemechanisme komen. Dat webpartmechanisme zal enkele



Afbeelding 4. Objectstructuur van generieke webpart en specifieke implementatie daarvan

toevoegingen hebben ten opzichte van ons systeem. Hierbij valt te denken aan personalisatie en de mogelijkheid om verschillende soorten zones te definiëren. De implementatie van het communicatiemechanisme in ASP.NET 2.0 is – zoals het er nu naar uit ziet – anders dan de implementatie binnen Sharepoint of ons systeem. Er zal gebruik worden gemaakt van metadata-attributen, die in webparts kunnen worden toegepast. Voor informatie die door een provider moet worden aangeboden, zal een interface moeten worden gemaakt en geïmplementeerd. Een consumer kan vervolgens een metadata-attribuut toepassen op een zogenaamde callback-methode. Deze methode verwacht een object dat de gemaakte interface implementeert, en zal worden aangeroepen zodra een provider de informatie aanbiedt. Het voordeel hiervan is dat je ten opzichte van Sharepoint 2003 minder code nodig hebt om een verbinding te realiseren. De focus bij ons systeem zal komen te liggen op interoperabiliteit met Sharepoint en – zodra dit mogelijk is – met ASP.NET 2.0. Op deze manier kunnen we zo snel mogelijk het beste van ASP.NET combineren met ons systeem.

Nuttige URL's

http://www.microsoft.com/sharepoint/
http://msdn.microsoft.com/sharepoint/
http://msdn.microsoft.com/library/default.asp?url=/nhp/default.asp?contentid=28001891
http://www.devx.com/dotnet/Article/17518

Wilco Bauwer en Herber de Ruijter zijn respectievelijk als Lead .Net Developer en Creative Technologist werkzaam bij Variomatic (www.variomatic.biz) te Amsterdam. Voor meer informatie over Variomatic's SmartStick en VarioPublish framework stuur een e-mail naar: herber@variomatic.biz