Dit jaar komt een nieuwe versie van Visual Studio 2005 uit. Deze nieuwe versie biedt behoorlijk wat nieuwe mogelijkheden voor ontwikkelaars. Zo ook het bouwen van websites. Het maken van de standaard webpagina-elementen en het consistent maken van lay-out kosten tot nu toe aardig wat moeite. De mogelijkheden van de style sheets beperken zich tot de standaard HTML-elementen. Style eigenschappen van ASP-controls komen ook in het stylesheet bestand terecht en aangezien ASP-controls vaak meerdere style-elementen gebruiken, is het consistent houden van opmaak lastig.

Visuele verbeteringen in Visual Studio 2005

Bouwen van websites sterk vereenvoudigd

In Visual Studio 2005 wordt het maken van een pagina-template door gebruik te maken van masterpages, skins en menucontrols veel gemakkelijker. Daarbij wordt er een nieuwe mogelijkheid geboden, die helpt bij het bouwen van website-portals. Het mechanisme is afgeleid van Microsoft Sharepoint Portal server. In Microsoft Sharepoint Portal Server krijgen gebruikers een persoonlijke webpagina, met allerlei informatie die voor deze persoon interessant is. De verscheidene informatieonderdelen krijgen op de webpagina een eigen venster. Dit venster wordt een webpart genoemd. In dit artikel worden de implementatie en mogelijkheden van masterpages, menu-elementen, skins en webparts uitgelegd.

MASTERPAGE De masterpage bepaalt de structuur van de pagina's. In de masterpage kunnen wederkerende elementen zoals menu's, logo's worden opgenomen zodat niet iedere pagina deze elementen opnieuw moet implementeren. Masterpages zorgen dat de pagina's dezelfde lay-out krijgen. De masterpage heeft een afwijkende naam. Je kunt het bestand een willekeurige naam geven met de extensie 'master'. Pagina's die van de masterpage gebruik willen maken moeten een verwijzing hebben naar de masterpage file. In het '@page'-element van de webpagina wordt het 'MasterPageFile'-attribuut opgenomen. Een '@page'-element dat een masterpage gebruikt ziet er zo uit:

```
<@page Language="C#" MasterPageFile="~/Site.
master" in het @page element
```

De master webpagina zelf begint met het 'master'-element:

```
<%@ Master Language="C#" %>
```

In de masterpage geef je aan welke delen van de pagina, specifieke paginacontent bevatten. Dit gebeurt met behulp van een contentplaceholder element. Deze content wordt op een andere pagina geïmplementeerd.

```
<asp:ContentPlaceHolder ID="headerPlaceHolder"
runat="server">
                    </asp:ContentPlaceHolder>
```

Dit is een voorbeeld van een masterpage. De inhoud heeft nu nog weinig om het lijf. Er is een contentplaceholder gedefinieerd voor de header van een pagina. Over het algemeen is het ook handig om een contentplaceholder op te nemen voor de body van de pagina. Vaste elementen zoals menu's kun je het beste in de masterpage zelf opnemen.

```
<%@ Master Language="C#" %>
<%@ Register Namespace="ServerWidgets"
TagPrefix="sw" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/
xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
```

```
</head>
<body>
  <form id="form1" runat="server">
   <table border="0" cellpadding="2" cellspacing="0" style="width: 100%; height: 100%">
    <tr height=30px>
     <td colspan =2></td>
      <td><asp:Image ID=logo Runat=Server ImageUrl="~/images/PageLogo.png"  /></td>
    </tr>
    <tr> <td width=100>menu</td> <td valign=top >
      <asp:ContentPlaceHolder ID="headerPlaceHolder" runat="server"></asp:ContentPlaceHolder>
    </td></tr>
    <tr><td colspan ="3">(C) Yourcompany </td></tr>
   </table>
  </form>
</body>
</html>
```

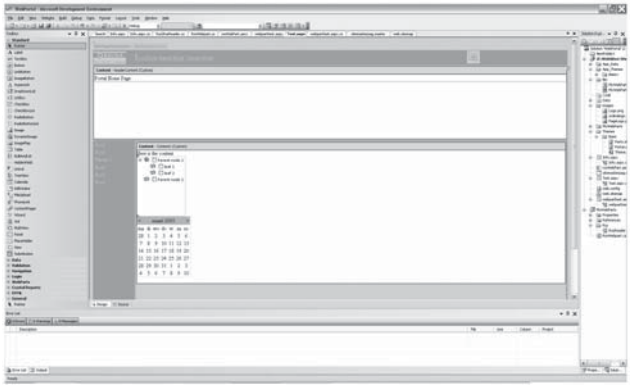
De webpagina die de masterpage implementeert kent geen HTML- en BODY-element. Alleen de contentplaceholder gebieden worden gedefinieerd. Hier wordt de invulling van een specifieke webpagina gegeven, die ontbreekt op de masterpage. In dit voorbeeld bevat de pagina de volgende contentplaceholders: een 'headerPlaceHolder' en een 'BodyPlaceHolder'. Alle contentplaceholders van de masterpage moeten op de gebruikmakende webpagina's ingevuld worden. De 'ID'-attributen van de placeholder dienen overeen te komen met die van de masterpage.

```
<%@ Page Language="C#" CompileWith="Test.aspx.cs" ClassName="Test_aspx"
  MasterPageFile="~/sitematerpag.master"
Title="Test Portal Home Page" %>

<asp:Content ID="headerContent"
runat="server"
  ContentPlaceHolderID="headerPlaceHolder">
  Portal Home Page</asp:Content>

<asp:Content ID="Content1" runat="server"
  ContentPlaceHolderID="bodyPlaceHolder">
  Here is the content</asp:Content>
```

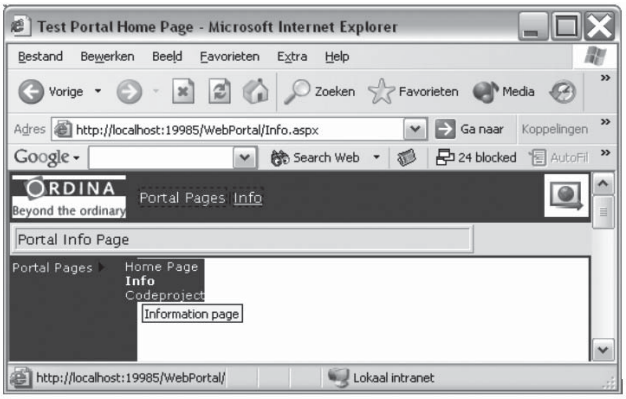
Bij het maken van een pagina die van een masterpage gebruik maakt, kun je in design mode de gehele pagina zien. Daarbij worden de delen van de masterpage in grey-style getoond en delen van de content in de normale stijl.

WEBSITE MENU Voor het maken van het menu biedt Visual Studio 2005 twee standaard controls.

- *Menu-control.* Dit is een scrollbaar menu en wordt meestal links op de pagina geplaatst. Maar kan ook bovenaan de pagina geplaatst worden. Het is mogelijk het menu een verticale of een horizontale lay-out te geven.
- *SitemapPath-control.* Dit is een menu, waarbij de hiërarchie van de site zichtbaar blijft voor de gebruiker en deze de keuze biedt om terug te keren naar een hoger onderdeel in de hiërarchie.



FIGUUR 1. Designmode van een contentpagina.



FIGUUR 2. Voorbeeld van een sitemap info control en een sitemap control.

In figuur 2 zie je naast het logo de sitemappath-control. Onder de tekst 'portal info page' zie je de menu-control. Deze menu-control kan ook horizontaal door de orientation eigenschap op 'horizontal' te zetten. Het is ook gemakkelijk om een andere control als menu-control te gebruiken, zoals de treeview-control. Voor navigatie van de site kan voor al deze genoemde controls de sitemap als datasource gebruikt worden. Echter er is maar één sitemap datasource voor een site mogelijk. Hier volgt een voorbeeld van een sitemap datasource-bestand:

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode title="Hoofdmenu" description="" roles="*" >
```

```
        <siteMapNode url="~/Default.aspx"
title="Home Pagina"  description="" />
        <siteMapNode url="~/MyPage.aspx"
title="My Page"  description="" />
        <siteMapNode url="www.codeproject.
com" title ="Codeproject"
    description ="Famous .NET website with
lots of articles"></siteMapNode>
    </siteMapNode>
</siteMap>
<< einde kader met computercode >>
```

De sitemap-datasource is een XML-bestand met de lange extensie 'sitemap'. Het element sitemap bevat een verwijzing naar het sitemap namespace-schema. Binnen de sitemap kan slechts een sitemapnode worden opgenomen. Deze eerste node is de titel van het menu. Binnen een sitemapnode, zoals de root-sitemapnode, kunnen andere sitemapnodes worden opgenomen. Hierdoor ontstaat er een hiërarchisch menu. Een sitemapnode element bevat attributen, die het gedrag en de presentatie van een menu-item bepalen.

Attribuut	Omschrijving
Roles	Autorisatie rollen De provider in de web.config moet de attribuut securityTrimmingEnabled op true hebben staan om autorisatie toe te passen.
Keywords	Zoeksleutelwoorden
Url	De achterliggende URL
Title	Omschrijving van het menu-item
Description	De tooltiptekst van het menu-item

TABEL 1. Sitemap node-attributen

De sitemap wordt in het webconfiguratiebestand gedefinieerd. Hiermee wordt het sitemap-bestand automatisch gekoppeld aan de sitemap datasource-control.

```
<configuration>
 <system.web>
  <siteMap enabled ="true">
   <providers>
  <clear/>
  <add name="AspNetXmlSiteMapProvider"
         type="System.Web.XmlSiteMapProvider,
System.Web, Version=2.0.3600.0,
Culture=neutral,
         PublicKeyToken=b03f5f7f11d50a3a"
              siteMapFile="web.sitemap"
              securityTrimmingEnabled="false
"/>
   </providers>
  </siteMap>
 <system.web>
<configuration>
```

Op de pagina waar de sitemap gebruikt wordt dient de sitemap datasource-control worden geplaatst binnen het form-element. Dit maakt het mogelijk een control aan de sitemap-datasource te binden.

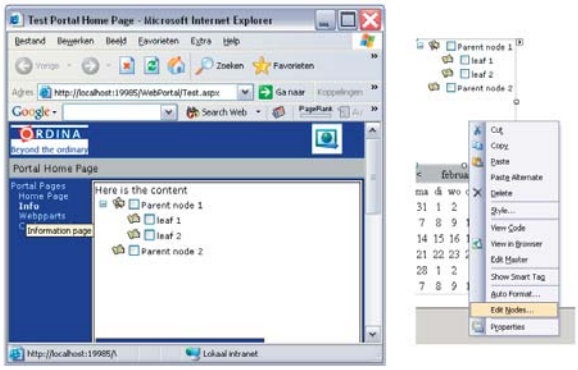
```
<asp:SiteMapDataSource ID="siteMapDataSource"
runat="server" />
```

De treeview-control kan aan de datasource gekoppeld worden door het datasource id-attribuut naar de sitemap datasource-control te laten verwijzen.

```
<asp:TreeView runat="server" ID="siteMapTree"
                 Width="125px"
                 DataSourceID="siteMapDataSo
urce"
                 SkinID="SiteMap" />
```

De opmaak van een treeview kan makkelijk aangepast worden. De imageset kan in diverse smaken worden gezet, zodat de lay-out als een folderbrowser of een outlook bar lijkt. De imageset zorgt voor de images voor de uitklapnode, de uitgekapte node, de lijnen en de leave-nodes. Standaard staat de imageset op custom. Via de custom-eigenschap kunnen de plaatjes gezet worden op de eigenschappen: 'CollapseImageUrl', 'ExpandImageUrl', 'LineImagesFolderUrl' en 'NoExpandImageUrl'. De showlines eigenschap kan gezet worden om de treeview-lijnen wel of niet te tonen. De treeview kent ook een eigenschap 'ShowCheckBoxes'. Deze checkbox-controls kunnen op de root nodes, de parent-nodes, de leave-nodes of alle nodes gezet worden.

Op de pagina is de menu-control of treeview-control goed te gebruiken samen met de sitemap path-control. Hierbij geeft de sitemap path-control de gekozen keuzes van het menu aan. Figuur 3 toont opnieuw de menu-control. Hier is de eigenschap 'StaticDisplayLevels' op twee gezet, zodat niet alleen de titel van het menu zichtbaar wordt. De treeview in de figuur is gedefinieerd met checkboxes en de imageset eigenschap is ingesteld op 'Inbox'. De elementen van de getoonde treeview zijn via het pop-up menu van de webpagina-designer aangemaakt.



FIGUUR 3. Treeview-control (l) en het bewerken van de treeview nodes met behulp van het contextmenu.

SKINS De visuele ASP-controls, zoals de menu-control of de calendar-control hebben vele opmaakeigenschappen. Deze eigenschappen zijn via de eigenschapeditor of via de auto-opmaakfunctie van de control-designer gemakkelijk te wijzigen. Als de opmaakeigenschappen voor controls consistent moeten zijn, is dit geen handige manier. Een Cascading Style Sheet kan voor de opmaak van HTML-elementen een oplossing hierin bieden. Voor ASP-controls voorzien skins in deze behoefte.

Een skin staat in een 'Themes'-subfolder van de root van het project. In deze folder kunnen de diverse skinbestanden geplaatst worden. Dit geldt ook de standaard-stylesheet. De extensie van een skinbestand is 'skin'. Er zijn twee type skins: de standaardskin en de specifieke skin. De visuele ASP-controls hebben een 'skin-id'-eigenschap. Deze kunnen gezet worden op het specifieke 'skin-id', zoals gedeclareerd is in het skinbestand. In het skin-bestand kunnen alle opmaakeigenschappen van de control gedefinieerd worden. Behalve een specifieke skin is er een basisskin. Bij ASP-controls die volgens een basisskin moeten worden opgemaakt wordt het skin-id leeggelaten. Dit geldt ook voor de definitie van de skin. Het voordeel van een specifieke skin is, dat je van de basisskin kunt afwijken en niet altijd een verplichte skin aan een control hoeft te geven. Skins kunnen voor ASP-controls ook uitgezet worden de eigenschap 'EnableTheming' van de control op false te zetten. Het uitzetten van de theme eigenschap kan ook op de pagina worden toegepast.

Dit is een voorbeeld van een skin-bestand, waar de basisopmaak van de calendar-control en een specifieke control:

```
<asp:Calendar BackColor="#FFFFCC"
BorderColor="#FFCC66" BorderWidth="1px" DayN
ameFormat="FirstLetter" Font-Names="Verdana"
Font-Size="8pt" ForeColor="#663399"
runat="server" ShowGridLines="True"
Width="220px">

  <TodayDayStyle BackColor="#FFCC66"
ForeColor="#FFFFFF" />
  <NextPrevStyle Font-Size="9pt"
ForeColor="#FFFFCC" />
  <DayHeaderStyle BackColor="#011C8D"
ForeColor="#FFFFFF" Font-Bold="True"
Height="1px" BorderStyle="None" />
  <TitleStyle BackColor="#011C8D"
ForeColor="#FFFFFF" Font-Bold="True" Font-
Size="9pt"  />
</asp:Calendar>

<asp:Calendar Runat="server" skinid="SkinSpec
ifiekeCalendar">
  <SelectedDayStyle BackColor="#"
ForeColor="FFFFFF" Font-Bold="True"></
SelectedDayStyle>
  <SelectorStyle BackColor="#FFCA65"></Select
orStyle>
  <DayHeaderStyle Font-Bold="True"
Height="1px" backColor="#FFCC66"> </
DayHeaderStyle>
  <TitleStyle ForeColor="#FFFFCC" Font-
Size="9pt" Font-Bold="True"
            BackColor="#990000"></TitleStyle>
</asp:Calendar>
```

FIGUUR 4. ASP calendar control opgemaakt met een skin-bestand

Binnen een themes-folder kunnen ook meerdere thema's worden gemaakt met eigen skins. Deze thema's kunnen dan in een eigen subfolder van de themes-folder geplaatst worden. Een pagina kan een specifiek thema gebruiken, door een theme-attribuut in het page-element van de pagina op te nemen.

```
<%@ page theme="MyTheme" EnableTheming=true
%>
```

Voor de gehele website wordt een specifiek thema in de config-file opgenomen. Echter de waarde 'Basic' is de uitzondering op de regel, omdat het hier een basisthema betreft en bestaat niet als een theme-folder.

```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.micro-
soft.com/.NetConfiguration/v2.0">
  <system.web>
     <authentication mode="Forms"/>
     <pages theme="Basic"/>
         </system.web>
</configuration
```

WEBPARTS Microsoft heeft de webparts van Sharepoint Portal Server in Visual Studio 2005 overgenomen. Sharepoint Portal is bedoeld om toegang tot informatie en communicatie van informatie gemakkelijk te maken. Webparts zijn een makkelijk hulpmiddel om veel informatie op één webpagina tonen. Ook worden webparts gebruikt om de informatie persoonlijk te maken. Gebruikers kunnen zelf bepalen welke, hoe en waar de webparts op de pagina komen te staan.

Webparts zijn door de eindgebruiker te configureren. Webparts kunnen bijvoorbeeld verplaatst worden en eigenschappen kunnen veranderd worden. Ook kunnen webparts op de pagina geplaatst worden en ze kunnen er ook weer van afgehaald worden. Hierbij wordt gebruik gemaakt van een webpart-catalog. Omdat webparts door de eindgebruiker aangepast kunnen worden, dient er ook een personalisatiemogelijkheid aanwezig te zijn. Deze mogelijkheid wordt gedefinieerd in het 'web.config'-bestand. In de personalisatiesectie wordt aangegeven welke rollen recht hebben en onder welke verbs. Een verbs is een werkwoord. Deze verbs kunnen aan de diverse webpartcontrol eigenschappen verbonden worden.

```
<webParts>
      <personalization>
        <authorization>
          <allow roles="Admin"
verbs="enterSharedScope" />
        </authorization>
      </personalization>
 </webParts>
```

Er zijn diverse standaardcontrols beschikbaar voor webparts. De webpartmanager is de basiscontrol. De webpartmanager is een niet-visuele control die de status van webparts bewaart en zorgt dat er communicatie tussen de webparts mogelijk is. Binnen het gebied van de pagina worden de webpart-zones geplaatst. Dit zijn de eigenlijke gebieden voor de webparts. Ze lijken op windows binnen een webpagina. De webparts-content kan gemaakt worden door web-user control te maken en deze control in de webpartzone te plaatsen. Dit is een overzicht van de belangrijkste webpart onderdelen:

- *Webpartmanager.* Het Webpartmanager-object is de basiscontrol op een webpart-pagina. Het is een onzichtbare control, verantwoordelijk voor het beheer, gedrag en functioneren van webparts. De webpartmanager beheert de instellingen van de webparts en maakt het mogelijk dat webparts van een pagina gehaald en op een pagina geplaatst kunnen worden.
- *WebPart.* Dit is een visuele servercontrol van het type webpart. Deze kan op verschillende manieren worden gedefinieerd.
- *WebPartZone.* Dit is het gebied waarin webparts geplaatst worden. Een pagina kan meerdere webpartzones bevatten. Webparts kunnen tussen de verschillende webzones verplaatst worden.
- *CatalogZone.* In de CatalogZone wordt de lijst van mogelijke webparts getoond. Gebruikers kunnen webparts uit de CatalogZone op de pagina plaatsen.
- *WebPartPageMenu.* Het WebPartPageMenu zorgt dat de gebruiker de instellingen van de pagina kan wijzigen. De gebruiker kan van browse mode naar edit mode switchen en daarbij de webparts gaan configureren.
- *EditorZone.* Dit is het gebied waarin de webpart-eigenschappen ingesteld worden. De eigenschappen van de webpart, dat in editmode is gezet, wordt in deze zone getoond.
- *ConnectionsZone.* De connection-controls zorgen voor de verbindingen tussen de webparts. Hierdoor kunnen webparts invloed op elkaar uitoefenen.

Het volgende voorbeeld toont de webpartzones en de webpartmanager.

```
<%@ Page Language="C#"
CompileWith="webparttest.aspx.cs"
ClassName="Webparttest_aspx"
  MasterPageFile="~/sitemasterpag.master"
Title="Webparts" %>

<asp:Content ID="headerContent"
runat="server"
  ContentPlaceHolderID="headerPlaceHolder">
  <table border=0 width=100% height=20px><tr>
<td>Portal webparts Page</td></tr></table>
</asp:Content>

<asp:Content ID="Content1" runat="server" Con
tentPlaceHolderID="bodyPlaceHolder">
<asp:WebPartManager ID="WebPartManager1"
Runat="server">
<Personalization Enabled="true" />
</asp:WebPartManager>
    <table border =1 cellpadding =2 width
=100%>
    <tr><td>
         <asp:WebPartZone ID="WebPartZone3"
```

```
runat="server"><ZoneTemplate></ZoneTemplate>
        </asp:WebPartZone>
      </td><td>
        <asp:WebPartZone ID="WebPartZone2"
runat="server"><ZoneTemplate></ZoneTemplate>
      </asp:WebPartZone>
      </td></tr>
      <tr><td colspan=2>
       <asp:WebPartZone ID="WebPartZone1" Run
at="server"><ZoneTemplate></ZoneTemplate>
          </asp:WebPartZone>
       </td></tr>
  </table>
</asp:Content>
```

Wanneer de webparts door de gebruikers aangepast moeten kunnen worden, zal de personalisatie in de webpartmanager-control aangezet moeten worden.

```
<asp:WebPartManager ID="webPartManager"
runat="server">
        <Personalization Enabled="true" />
      </asp:WebPartManager>
```

VERANDERINGEN VAN SELECTIE De flexibiliteit van webparts wordt zichtbaar door de webpart Pagemenu-control te plaatsen. Het menu heeft een lijst van werkwoorden. Deze verbs kunnen aan een control gekoppeld worden. De webpart Pagemenu-control zorgt ervoor dat de eigenschappen gezet van webparts gezet kunnen worden, elementen verplaatst kunnen worden, webparts verbonden kunnen worden en webparts toegevoegd of verwijderd worden. Webparts kunnen eventueel hierbij in de catalog-zone geplaatst worden of er juist uit gehaald worden. Hier volgt een voorbeeld van een WebPartPageMenu:

```
<asp:WebPartPageMenu ID="WebPartPageMenu1"
             Runat="server"
             Text="Change the Layout"
             Mode="Menu"
             HoverStyle-BorderWidth="1"
             MenuStyle-BorderWidth="1"
             BrowseModeVerb-Text="Browse"
             DesignModeVerb-Text="Design"
             EditModeVerb-Text="Edit"
             CatalogModeVerb-Text="Catalog">

</asp:WebPartPageMenu>
```

Het WebPartPageMenu control heeft vier werkwoorden

- BrowseMode- de user komt weer terug in de weergave modus.
- DesignMode- de gebruiker kan webparts op het scherm verplaatsen.
- EditMode- de gebruiker kan weergave en de titel van webparts veranderen.
- CatalogMode- de gebruiker kan webparts verwijderen van de pagina en kan webparts toevoegen aan webpart-zones.

De mode kan in code worden gezet op de Displaymode eigenschap van de webpartmanager.

```
WebPartManager1.DisplayMode = WebPartManager.
DesignDisplayMode;
```

BOUWEN VAN EEN WEBPART Een webpart kan via diverse mogelijkheden worden gemaakt. Dit kan door gebruik te maken van de zone template-control. Dit is het gemakkelijkste. Er hoeven alleen maar controls op de pagina gesleept worden naar de zone template. Een tweede mogelijkheid is een control te maken in ASPX. Deze control kan eenvoudig als webpart-control worden gezien. Een derde mogelijkheid is om een webpart te maken in code door de klasse van de webpart-classe af te leiden. Een vierde mogelijkheid is om van een willekeurige server-control een generieke webpart-control te maken. Er zijn daarnaast nog meerdere mogelijkheden die hier niet worden behandeld om een webpart te maken.

1. Webpart, gemaakt door implementatie van een content-webpart:

```
<asp:WebPartZone id="HelloPart"
runat="Server">
  <ZoneTemplate>
    A simple webpart
  </ZoneTemplate>
</asp: WebPartZone >
```

2. Webpart, gemaakt door definitie van een custom webcontrol:

```
<%@ Control EnablePersonalization="True" %>
<script runat="server">
  [Personalizable, WebBrowsable ]
  Public string Tekst { get { return
"Label";}}
</script>
  <asp:Label runat="server" ID="myLabel" />
A simple webpart
```

3. Webpart als control gemaakt door een klasse af te leiden van de webpart-klasse:

```
using System.Web.UI;
using System.Web.UI.WebControls;
public class cODEWebPart : WebPart {
```

Website	Omschrijving + URL
Microsoft ASP 2.0	Microsoft tutorial site for ASP 2.0 (Beta)
	http://beta.asp.net
Developer source	Designing with Webparts in Visual Web Developer
	http://www.devsource.com/article2/0,1759,1751857,00.asp
CarlosAg.Net	ASP 2.0 Webparts Connection tutorial
	http://www.carlosag.net/Articles/WebParts/connectionsTutorial.aspx#scenario
Nikhilk.Net	Webparts and crosspage connections
	http://www.nikhilk.net/CrossPageConnections.aspx
MSDN	Masterpages
	http://msdn.microsoft.com/library/default.asp?url=/msdnmag/issues/04/06/ASPNET20MasterPages/toc.asp
	ASPNET meets webparts framework
	http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnaspp/html/aspnet-meetwebpartfrmwrk.asp

```
  protected override void RenderContents(Html
TextWriter writer) {
    writer.Write( "This is a code webpart!"
);
  }
}
```

4. Generieke webpart gemaakt door aanroep van de CreateWebPart methode met de user-control.als parameter:

```
GenericWebPart webPart = WebPartManager1.Crea
teWebPart(customUserControl);
```

Het toevoegen van een webpart aan een pagina in code is niet moeilijk. De webpart moet via de webpartmanager aan een webpartzone gekoppeld worden:

```
WebPartCalendar webPartCalendar = new
WebPartCalendar("WebPart Calendar",today);
WebPartManager1.AddWebPart(wc, WebPartZone1,
0);
```

De webpartmanager zorgt er ook voor of een webpart gewijzigd mag worden. Het verplaatsen tussen de webpartzones van een webpart kan uitgezet worden:

```
WebPartManager1.WebParts["wcCalendar"].
AllowZoneChange = false;
```

TENSLOTTE Bouwen van websites met Visual Studio 2005 lijkt gemakkelijker te worden. Masterpages zorgen voor een consistentere lay-out, skins zorgen voor een consistente opmaak van controls en de menu-control is handig voor het navigeren van de site. Daarbij geeft de sitemappath-control de plek in de site aan waar je je op dat moment bevindt. Maar de beste vooruitgang wordt geboekt door de mogelijkheid webparts op de website te gebruiken. Weliswaar kon dit al met Sharepoint Portal Server, maar nu kan het ook zonder. De hier besproken mogelijkheden zijn slechts een deel van het geheel. Het ziet er allemaal veelbelovend uit. Eén ding wordt lastiger: er zijn steeds meer zaken waar je rekening mee moet houden om een website te bouwen. Het werkend krijgen van een website wordt daardoor lastiger.

Dieder Timmerman is werkzaam bij Ordina