**Michiel van Otegem**
is webdevelopment trainer bij ASPNL en voorzitter van de .NET gebruikersvereniging Stichting dotNED. Hij schrijft artikelen voor diverse bladen wereldwijd en is de auteur van Teach Yourself XSLT in 21 Days. Daarnaast spreekt hij ook regelmatig op conferenties in Europa en de VS. Hij is bereikbaar via michiel@aspnl.com, http://www.aspnl.com

# Gegevens en ASP.NET

## Effectief gebruik van de DataGrid-control en DataList-control

**In ASP.NET is het weergeven van gegevens uit een database of een andere gegevensbron betrekkelijk eenvoudig. Door gebruik te maken van de DataGrid-control of de DataList-control heb je slechts enkele regels code nodig om een mooi resultaat te krijgen. Hoewel het wijzigen van gegevens meer om het lijf heeft, is ook dat betrekkelijk makkelijk voor elkaar te krijgen.**

Gegevens weergeven gaat het snelste als je gebruik maakt van de DataGrid-control. Deze control geeft gegevens weer in een tabelweergave die je desgewenst kunt aanpassen. Wanneer je de DataGrid-control in Visual Studio .NET op de pagina sleept, krijg je een DataGrid-control te zien zonder enige opmaak. Je kunt hetzelfde resultaat bereiken door in de HTML <asp:DataGrid runat="server" id="DataGrid1"/> te zetten. De DataGrid-control is dan al volledig functioneel, en de opmaak kun je via het Properties-venster, of met de van daar uit te openen Property Builder wijzigen. Zo kun je de opmaak van een rij in de DataGrid aanpassen via ItemStyle, waarmee je onder andere de voor- en achtergrondkleur en het lettertype in kunt stellen. Wil je voor de even en oneven rijen een verschillende opmaak, dan kun je ook de AlternatingItemStyle wijzigen. Verder zijn er stijlen voor de kop, voettekst, paginering, de geselecteerde rij, en de rij die gewijzigd wordt. Je kunt ook een voorgedefinieerde opmaak voor de hele DataGrid-control gebruiken door in het Properties-venster te klikken op Auto Format. De gekozen opmaak kun je vervolgens verder verfijnen. In de HTML-weergave van de pagina zie je dat de opmaakwijzigingen opgenomen zijn in de HTML van de DataGrid.

Met de opmaak naar wens moet je nog wel aangeven welke gegevens moeten worden weergegeven. Dit doe je door de gegevensbron te koppelen aan de DataGrid, en dat kun je het beste doen in een aparte procedure. In codevoorbeeld 1 wordt code gekoppeld uit een SQL Server database (vergeet niet een referentie te maken naar de System.Data.SqlClient namespace met Imports [VB] of using [C#]), maar dit kan bijvoorbeeld ook een XML-bestand zijn. In Visual Studio .NET kun je ook via enkele wizards de gegevenstoegang verzorgen, maar je moet dan nog steeds een gedeelte via code doen, waardoor de voordelen daarvan niet opwegen tegen de eenvoud van codevoorbeeld 1. In de procedure maak je eerst een verbinding met een database aan de hand van connStr, die verwijst naar de Northwind-database. Eventueel kun je die in de Page_Load gebeurtenismethode laden uit de appSettings-sectie van web.config (zoals in de downloadbare voorbeeldcode). In de volgende stap moet je zorgen dat de juiste gegevens worden opgehaald zodat je een tabel in een DataSet-object kunt vullen. Je hoeft de database niet te openen en sluiten, want dat doet de .Fill-methode van de SqlDataAdapter automatisch. In de laatste stap koppel je de opgevraagde gegevens aan de DataGrid. Dat doe je door eerst de gegevensbron in te stellen en daarna de DataBind-methode aan te roepen. Die methode zorgt dat de gegevens in de DataGrid gezet worden. Dit gebeurt automatisch, dus je hoeft niet door de gegevens heen te bladeren. De procedure uit codevoorbeeld 1 *roep* je aan vanuit Page_Load, maar alleen als de gebruiker de pagina voor het eerst opvraagt. Als de pagina namelijk wordt hergeladen door een actie van de gebruiker moet de daarbij behorende gebeurtenismethode worden uitgevoerd voordat de DataGrid opnieuw wordt gevuld, terwijl Page_Load altijd eerst uitgevoerd wordt.

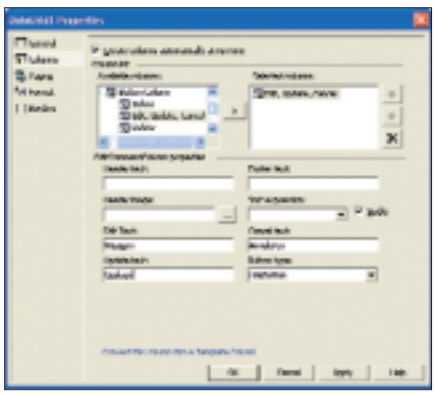
## Gegevens wijzigen

Normaal gezien worden de rijen in de DataGrid allemaal weergegeven zonder de mogelijkheid de gegevens te wijzigen. Je kunt een rij echter wijzigbaar maken, zodat de waardes in deze rij worden weergegeven in TextBox-controls. Een rij wijzigbaar maken doe je in de EditCommand-methode, waarin je aangeeft welke rij wijzigbaar moet worden. Door een EditCommandColumn in te voegen worden hiervoor de benodigde knoppen weergegeven bij elke rij. Je kunt een EditCommandColumn invoegen via de Columns-sectie in de Property Builder, zoals is te zien in afbeelding 1. Hierdoor wordt codevoorbeeld 2 toegevoegd aan de HTML van de DataGrid, hetgeen je ook handmatig in HTML-weergave kunt doen. Als je nu de pagina opvraagt bevat elke rij een knop 'Wijzigen'. Deze knop doet nog niets, want er is nog geen

```
Private Sub BindData()
    'Databaseverbinding en gegevensselectie verzorgen
    Dim sql As String = "SELECT * FROM Territories"
    Dim connection As New SqlConnection(connStr)
    Dim da As New SqlDataAdapter(sql, connection)

    'Gegevens uit database in DataSet laden
    Dim ds As New DataSet()
    da.Fill(ds, "Territories")

    'DataSet koppelen aan DataGrid control
    DataGrid1.DataSource = ds.Tables("Territories").DefaultView
    DataGrid1.DataBind()
End Sub
```

**Codevoorbeeld 1. Gegevens koppelen aan de DataGrid**

code die zorgt dat de gekozen rij wijzigbaar wordt. Hoe je die code het gemakkelijkst kunt toevoegen verschilt enigszins tussen de verschillende talen die je kunt gebruiken in Visual Studio .NET. In VB.NET ga je naar de codeweergave, waarna je via de dropdown-lists Class Name en Method Name een methode van de DataGrid kunt selecteren. In C# bevat het Properties-venster een aparte tab voor Events, te openen via het bliksemicoontje. Door op een Event te dubbelklikken, ga je automatisch naar de juiste plek in de codeweergave. De code die je moet invoegen voor de benodigde methodes van de DataGrid (EditCommand, CancelCommand, en UpdateCommand), zie je in codevoorbeeld 3.



**Afbeelding 1. Een EditCommandColumn toevoegen in de Property Builder**

Het EditCommand en CancelCommand bepalen alleen welke rij wijzigbaar is. In het eerste geval staat de index van de betreffende rij in de argumenten van de gebeurtenis. In het laatste geval zet je de index op -1, om geen enkele rij wijzigbaar te maken. Als de gebruiker op 'Opslaan' klikt bij een wijzigbare kolom, dan moet de gegevensbron bijgewerkt worden. Dit bestaat uit een aantal stappen. Eerst moet je de waardes uit de TextBox-controls halen. De code hiervoor is wat omslachtig, omdat de TextBox-controls dynamisch aangemaakt worden, en je daarom dus niet weet wat de namen zijn van de controls. Je moet de controls dus aan de hand van hun positie in de wijzigbare rij opvragen. Vervolgens bouw je een SqlCommand-object op voor het uitvoeren van een SQL-opdracht. Hierbij kun je parameters gebruiken om de waardes in te voegen, wat beter is dan de volledige opdracht op te bouwen als een string. Je kunt de gegevenstypes dan namelijk via de parameter bepalen en je hebt geen last met apostrofs en andere speciale karakters. Als laatste voer je de opdracht uit, waarbij je de databaseverbinding expliciet moet openen en sluiten. Om onnodige problemen te voorkomen, kun je de opdracht uit laten voeren binnen een Try...Catch-constructie om fouten af te vangen. Merk op dat in codevoorbeeld 3 de foutmelding via een Label-control aan de gebruiker getoond wordt. Het label moet je dus nog wel invoegen, waarbij het verstandig is om EnableViewState op False te zetten, zodat de melding niet blijft staan na nieuwe acties van de gebruiker.

```
<Columns>
    <asp:EditCommandColumn ButtonType="LinkButton"
        UpdateText="Opslaan" CancelText="Annuleren"
        EditText="Wijzigen">
    </asp:EditCommandColumn>
</Columns>
```

**Codevoorbeeld 2. HTML voor een extra kolom in de DataGrid**

## Gegevens verwijderen

Voor het verwijderen van rijen heb je de primaire sleutel van de rij nodig. Verwijderen doe je echter als de rij niet wijzigbaar is, dus is er geen TextBox die deze waarde bevat. Je kunt dit verhelpen door het veld met de primaire sleutel toe te kennen aan de DataKeyField-eigenschap, in dit geval TerritoryID. Verder moet je een knop toevoegen waarmee je een rij kunt verwijderen. Dit doe je door een Button Column in te voegen (eventueel via de Property Builder) waarin de CommandName-eigenschap de waarde Delete krijgt. Hierdoor wordt de DeleteCommand-methode uitgevoerd als de gebruiker op de verwijderknop klikt. Die methode is vrijwel gelijk aan UpdateCommand-methode, behoudens de SQL-opdracht. Je moet hierbij de primaire sleutel, die je kunt opvragen met DataGrid1.DataKeys(e.Item.ItemIndex), gebruiken voor de TerritoryID-parameter. Let er op dat het toevoegen van een kolom mogelijk van invloed is op de positie van de TextBox-controls die nodig zijn voor het wijzigen van gegevens.

## Gegevens invoegen

Gegevens invoegen lijkt veel op het wijzigen van gegevens. Het is zelfs zo dat wanneer je gegevens toevoegt aan de gegevensbron je de UpdateCommand-methode gebruikt. De DataGrid kan namelijk geen onderscheid maken tussen die twee. Dat is waar gegevens invoegen lastig wordt, omdat je een aantal zaken moet regelen waardoor dit onderscheid er wel is. Gegevens invoegen doe je met een knop die geen onderdeel is van de DataGrid. Je kunt dus gewoon een knop toevoegen en

```
Private Sub DataGrid1_EditCommand(ByVal source As Object, _
  ByVal e As System.Web.UI.WebControls.DataGridCommandEventArgs) _
  Handles DataGrid1.EditCommand
    'Geselecteerde rij wijzigbaar maken
    DataGrid1.EditItemIndex = e.Item.ItemIndex
    BindData()
End Sub

Private Sub DataGrid1_CancelCommand(ByVal source As Object, _
  ByVal e As System.Web.UI.WebControls.DataGridCommandEventArgs) _
  Handles DataGrid1.CancelCommand
    'Wijzigen annuleren door geen rij wijzigbaar te maken
    DataGrid1.EditItemIndex = -1
    BindData()
End Sub

Private Sub DataGrid1_UpdateCommand(ByVal source As Object, _
  ByVal e As System.Web.UI.WebControls.DataGridCommandEventArgs) _
  Handles DataGrid1.UpdateCommand
    Dim TerritoryID As String
    Dim TerritoryDescription As String
    Dim RegionID As String

    'Nieuwe waardes ophalen uit rij
    TerritoryID = CType(e.Item.Cells(1).Controls(0), TextBox).Text
    TerritoryDescription = CType(e.Item.Cells(2).Controls(0), _
      TextBox).Text
    RegionID = CType(e.Item.Cells(3).Controls(0), TextBox).Text

    'Database opdrachten voorbereiden
    Dim connection As New SqlConnection(connStr)
    Dim updateCommand As SqlCommand = New SqlCommand()
    updateCommand.Connection = connection

    'SQL opdracht instellen
    Dim sql As New System.Text.StringBuilder()
    sql.Append("UPDATE Territories SET")
    sql.Append(" TerritoryDescription=@TerritoryDescription")
    sql.Append(", RegionID=@RegionID")
    sql.Append(" WHERE TerritoryID=@TerritoryID")
    updateCommand.CommandText = sql.ToString()

    'Parameters instellen
    updateCommand.Parameters.Add("@TerritoryID", _
      SqlDbType.NVarChar, 20).Value = TerritoryID
    updateCommand.Parameters.Add("@TerritoryDescription", _
      SqlDbType.NChar, 50).Value = TerritoryDescription
    updateCommand.Parameters.Add("@RegionID", _
      SqlDbType.Int).Value = RegionID

    'Opdracht uitvoeren
    Try
        connection.Open()
```

*(Zie vervolg code op volgende pagina)*

**Codevoorbeeld 3. Code voor het wijzigen van gegevens in een rij**

daar een Click-event aan koppelen. Als de gebruiker op die knop klikt, moet je een lege rij toevoegen aan de DataGrid en deze wijzigbaar maken. Dit doe je door eerst de gegevens in een DataSet te lezen, zoals in de BindData-procedure, en daarna een rij toe te voegen aan de tabel in de DataSet, voordat je die koppelt aan de DataGrid. Ook moet de EditItemIndex naar de laatste rij in de DataGrid wijzen. Codevoorbeeld 4 laat het fragment zien van het Click-event dat verantwoordelijk is voor deze stappen.

Codevoorbeeld 4 geeft AddingNew de waarde True. Hiermee geef je aan dat een nieuwe rij wordt ingevoegd. Door de waarde van AddinNew in het UpdateCommand te controllen, kun je de SQL-opdracht aanpassen aan wijzigen of invoegen. AddingNew is een eigenschap waarvan je de waarde op moet slaan in de ViewState, zoals je kunt zien in codevoorbeeld 5. De ViewState zorgt dat de staat van de pagina bewaard blijft tussen acties van de gebruiker. Door AddingNew in de ViewState op te slaan, blijft die waarde beschikbaar als de gebruiker de gegevens gaat opslaan.

## Kolommen aanpassen

Totnogtoe worden gegevens op een standaardmanier weergegeven, omdat de daarvoor verantwoordelijke kolommen automatisch gegenereerd worden. Dit betekent onder andere dat de volgorde en de kolomkoppen meestal niet aan de wensen voldoen. Als je de eigenschap AutoGenerateColumns op False zet, worden de kolommen niet automatisch gegenereerd, en moet je ze net als de EditCommandColumn toevoegen. Dit kan uiteraard weer via de Property Builder, waarbij je keuze hebt uit verschillende soorten kolommen. De simpelste kolom is een Bound Column, die min of meer gelijk is aan de kolommen die automatisch gemaakt worden. Voor iedere kolom moet je aangeven welk veld je wilt weergeven. Je kunt verder onder andere de tekst van de kolomkop instellen, een kolom niet wijzigbaar maken, en het formaat aanpassen waarin waardes weergegeven worden. Als je bijvoorbeeld een geldbedrag wilt weergeven, geef je bij Data formatting expression de

volgende waarde: {0:c}. Elk getal wordt dan met twee cijfers achter de komma weergegeven en met het geldsymbool van het huidige land. Wil je de weergave nog verder af laten wijken van de standaardkolom, dan kun je ook gebruik maken van een Template Column. Hiermee kun je een HTML-sjabloon maken waarin de waarde uit de kolom wordt weergegeven, zodat je de opmaak precies kunt bepalen. Als je een Template Column gebruikt, kun je het beste eerst een Bound Column aanmaken en dan in de Property Builder aangeven dat je er een Template Column van wilt maken. Je krijgt dan de basisopmaak van de Bound Column cadeau, die je verder kunt aanpassen. Je kunt de sjablonen van de kolom wijzigen door met de rechter muisknop op de DataGrid-control te klikken, en dan te kiezen voor Edit Template, zoals je kunt zien in afbeelding 2. Je kunt nu net als in een pagina de controls in de sjablonen slepen en deze aanpassen. Ook kun je de sjablonen wijzigen in HTML-weergave.

Het handige van de sjablonen is dat je andere controls kunt invoegen voor de weergave van waardes. Dit is vooral interessant als je Booleaanse waardes wilt wijzigen via een CheckBox, of een DropDownlist wilt weergeven met mogelijke waardes, misschien zelfs wel uit een andere tabel in de database. Codevoorbeeld 6 laat de HTML zien waarmee je dit kunt doen.

Codevoorbeeld 6 maakt gebruik van expressies tussen <%# en %>. Dit zijn zogenaamde databind-expressies, die je gebruikt om waardes in te voegen in een sjabloon of te koppelen aan een control. Je bind de waarde uit een veld in de rij die je wilt weergeven met de expressie DataBinder.Eval(Container, “DataItem.RegionID”). Je kunt echter verder gaan, en zoals met de RegionData-functie in codevoorbeeld 6 en 7 een hele tabel met gegevens koppelen aan een control zoals de DropDownList. Dit moet ook wel,

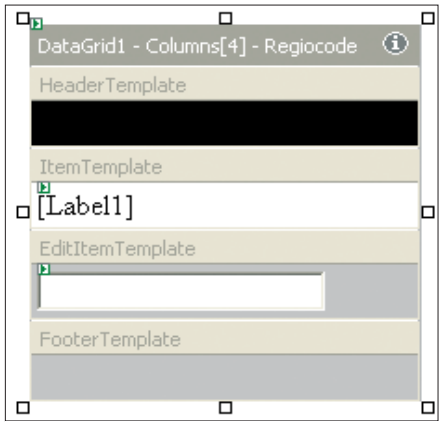
```
(vervolg code vorige pagina)
        updateCommand.ExecuteNonQuery()
    Catch ex As Exception
        Message.Text = ex.ToString()
    Finally
        connection.Close()
    End Try

    ‘Rij weer gewoon weergeven
    DataGrid1.EditItemIndex = -1
    BindData()
End Sub
```

```
‘Lege rij toevoegen
Dim rowValues As Object() = {“”, “”, 0}
ds.Tables(0).Rows.Add(rowValues)

‘Toegvoegde rij wijzigbaar maken
DataGrid1.EditItemIndex = ds.Tables(0).Rows.Count - 1

‘Geef aan dat de nieuwe rij toegevoegd moet worden
AddingNew = True
```

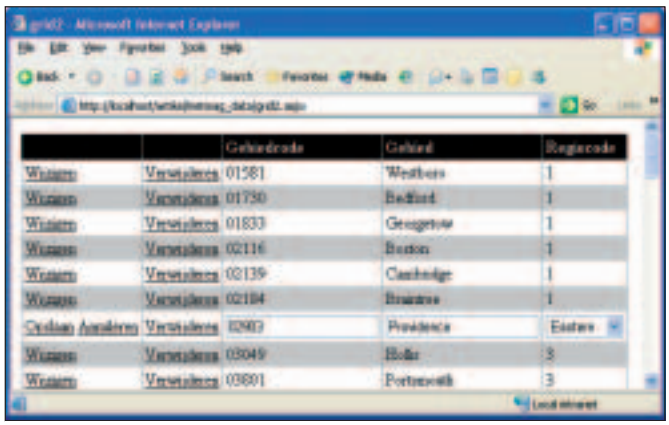
**Codevoorbeeld 4. Een lege rij toevoegen aan de DataGrid**

```
Property AddingNew() As Boolean
    Get
        ‘Haal waarde AddingNew uit ViewState
        Dim o As Object = ViewState(“AddingNew”)

        ‘Retourneer waarde (of False als deze er niet is)
        If o Is Nothing Then
            Return False
        End If
        Return CBool(o)
    End Get
    Set(ByVal Value As Boolean)
        ‘Waarde AddingNew opslaan in ViewState
        ViewState(“AddingNew”) = Value
    End Set
End Property
```

**Codevoorbeeld 5. De AddingNew-eigenschap werkt met de ViewState**




**Afbeelding 2. Sjablonen wijzigen**



**Afbeelding 3. De resulterende DataGrid in de browser met een wijzigbare rij**

```
<asp:TemplateColumn HeaderText="Regiocode">
    <ItemTemplate>
      <asp:Label id=Label1 runat="server"
       Text='<%# DataBinder.Eval(Container, _
            "DataItem.RegionID") %>'>
      </asp:Label>
    </ItemTemplate>
    <EditItemTemplate>
      <asp:DropDownList id="Region" runat="server"
        DataSource='<%# RegionData()%>'
        DataValueField="RegionID"
        DataTextField="RegionDescription"
        SelectedIndex='<%# RegionIndex(DataBinder.Eval(Container, _
                        "DataItem.RegionID")) %>'>
     </asp:DropDownList>
  </EditItemTemplate>
</asp:TemplateColumn>
```

**Codevoorbeeld 6. HTML voor de Template Column**

```
Function RegionData() As DataTable
    If dtRegion Is Nothing Then
        'Databaseverbinding en gegevensselectie verzorgen
        Dim sql As String = "SELECT * FROM Region"
        Dim connection As New SqlConnection(connStr)
        Dim da As New SqlDataAdapter(sql, connection)

        'Gegevens uit database in DataTable laden
        dtRegion = New DataTable()
        da.Fill(dtRegion)
    End If

    Return dtRegion
End Function

Function RegionIndex(ByVal ID As Integer) As Integer
    Dim i As Integer

    For i = 0 To dtRegion.Rows.Count - 1
        If dtRegion.Rows(i).Item("RegionID") = ID Then
            Return i
        End If
    Next
End Function
```

**Codevoorbeeld 7. Functies voor het instellen van de DropDownList in codevoorbeeld 6**

omdat de DropDownList als onderdeel van de DataGrid wordt aangemaakt, en je dus niet zomaar de gegevens kunt koppelen zoals je dat met BindData doet. Natuurlijk wil je wel dat de DropDownList de juiste waarde laat zien, dus moet je wel de juiste index instellen. Hiervoor koppel je de functie RegionIndex aan de SelectedIndex-eigenschap. Met ook de kolommen helemaal naar wens ziet de resulterende DataGrid eruit zoals in afbeelding 3.

## DataList in plaats van DataGrid

De DataList-control werkt volledig op basis van sjablonen. Er is dus geen standaard tabelweergave, je zult de gehele opmaak zelf moeten verzorgen. Dat is moeilijker, maar geeft ook meer vrijheid. De sjablonen werken helemaal hetzelfde als de Template Column-elementen uit de DataGrid, maar gelden voor de hele rij. Welke waardes je dus invoegt hangt helemaal af van de databind-expressies die je gebruikt. Voor bijvoorbeeld een wijzigknop kun je een gewone knop (Button, LinkButton, of ImageButton) invoegen. Van die button moet je wel de CommandName-eigenschap de naam geven van wat de knop moet doen, dus Edit voor wijzigen, Cancel voor annuleren, Update voor opslaan, en Delete voor verwijderen.

## Veelzijdige en flexibele controls

De DataGrid-control en DataList-control zijn zeer veelzijdig en door de juiste methodes te gebruiken ook zeer flexibel. De kunst is te weten welke methode je waarvoor moet gebruiken. Naast de gedemonstreerde mogelijkheden zijn er nog meer. Zo is het met de DataGrid uitermate eenvoudig om, zonder de wijzigingscapaciteiten aan te tasten, paginering en sortering te implementeren. Al met al kun je snel op weg als je met gegevens wilt werken, maar wordt het moeilijker naarmate je meer wilt.

**Nuttige internetadressen**

- Tutorial ASP.NET inclusief DataGrid.DataList - http://www.microsoft.com/netherlands/msdn/asp-net/menu.asp
- DataGrid vs. DataList - http://msdn.microsoft.com/msdnmag/issues/01/12/asp/asp0112.asp
- Forum voor vragen over DataGrid/DataList - http://www.aspnl.com/aspnl/nl/forums/ShowForum.aspx?ForumID=22

**.NET Gebruikersgroep**

Wil je meer weten over het .NET Framework, en kennis maken met andere .NET ontwikkelaars? Kom dan eens naar de gratis bijeenkomsten van Stichting dotNED, de Nederlandse .NET gebruikersgroep. Op deze maandelijkse bijeenkomsten wordt telkens een presentatie gegeven die je meer leert over .NET en aanverwante technologie. Kijk voor meer informatie over Stichting dotNED en de bijeenkomsten op http://www.dotned.nl.

**Voorbeelden downloaden**

Op de website van Microsoft Nederland zijn in het kader van dit artikel extra voorbeelden te downloaden van zowel VB.NET als C#