Dieper in Visual Basic .Net

7.1 Inleiding

Dit hoofdstuk is bedoeld om je wat handiger te maken in het programmeren in Visual Basic. Je leert geen nieuwe programmeervaardigheden, maar je leert wat event handlers zijn, hoe je een installatiepakket van je project maakt en hoe je werkt met het lezen en opslaan van bestanden.

In dit boek staat niet alles over Visual Basic. Daarom is het belangrijk dat je zelf op het internet op zoek kunt gaan naar de juiste informatie. Daarvoor krijg je in dit hoofdstuk enkele tips.

7

Index

event

event handler

Handles

MouseHover

MyBase.Load

7.2 Event handlers

7.2.1 Inleiding

Als je een knop op je formulier geplaatst hebt en je wilt code schrijven die moet worden uitgevoerd zodra de gebruiker op de knop klikt, dan dubbelklik je op de knop. Visual Basic genereert dan automatisch de volgende methode voor je:

```
Private Sub btnKnop_Click(ByVal sender As System.Object,
                          ByVal e As System.EventArgs)
                          Handles btnMijnKnop.Click

End Sub
```

De bovenstaande methode is de **event handler** voor het event btMijnKnop.Click. Dat wil zeggen dat deze methode het **event** (de gebeurtenis) Click van btnMijnKnop verwerkt. Je ziet dat aan het keyword **Handles**: daarachter staat de gebeurtenis die de methode verwerkt.

Je kunt ook methodes laten aanmaken bij andere events: in het rechter drop-downmenu (zie bovenstaande afbeelding) vind je een overzicht van alle events. Je kunt bijvoorbeeld het event **MouseHover** selecteren. Dat is de gebeurtenis dat de gebruiker met de muis over de knop beweegt.

7.2.2 Het event MyBase.Load

Een belangrijk event is **MyBase.Load**: het is het moment dat het formulier wordt geladen. De event handler hierbij wordt automatisch gemaakt als je dubbelklikt op het formulier:

```
Private Sub Formulier_Load(…) Handles MyBase.Load

End Sub
```

Je kunt in deze methode alles programmeren wat direct bij de start van je applicatie geregeld moet worden, zoals het geven van beginwaarden aan variabelen.

Index

parameter
sender

7.2.3 Zelf een event handler schrijven

Je bent niet afhankelijk van de methoden die Visual Basic automatisch voor je aanmaakt: je kunt ze ook zelf maken. Zo kun je de event handler btnKnop_Click die hierboven staat vervangen door een zelfgemaakte event handler:

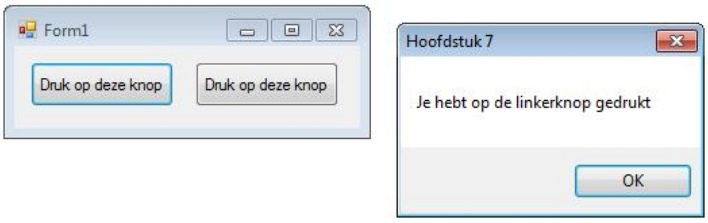
```
Private Sub ErIsOpMijnKnopGeklikt() Handles btnMijnKnop.Click

End Sub
```

Het is niet verplicht om de parameters `ByVal sender As System.Object` en `ByVal e As System.EventArgs` te gebruiken. Je kunt ook een event handler schrijven zonder parameters, zoals hierboven. Dat roept de vraag op waar de parameters *sender* en *e* eigenlijk voor dienen. Kort gezegd verwijst de **parameter sender** naar het object waar het event vandaan komt en bevat *e* meer informatie over het event zelf. We gaan iets dieper op dit onderwerp in.

7.2.4 De parameter *sender*

Op het bovenstaande formulier staan twee knoppen, btnKnopLinks en btnKnopRechts.
We maken een methode die voor beide knoppen het event Click afhandelt:




```
Private Sub OpDeKnopGeklikt(ByVal sender As System.Object,
                            ByVal e As System.EventArgs)
                            Handles btnKnopLinks.Click, btnKnopRechts.Click
End Sub
```

We kunnen nu met een if-statement controleren op welke van de twee knoppen is geklikt:

```
Private Sub OpDeKnopGeklikt(ByVal sender As System.Object,
                            ByVal e As System.EventArgs)
                            Handles btnKnopLinks.Click, btnKnopRechts.Click

    If sender.Equals(btnKnopLinks) Then
        MsgBox("Je hebt op de linkerknop gedrukt")
    Else
        MsgBox("Je hebt op de rechterknop gedrukt")
```

Index

Equals()

MenuStrip

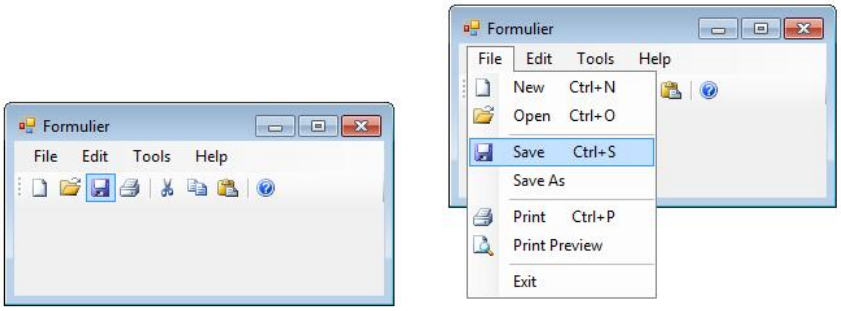
ToolStrip

```
    End If
End Sub
```

De methode `sender.Equals()` controleert of sender gelijk is aan het object dat je als parameter opgeeft, in dit geval btnKnopLinks.

Het is in het voorbeeld hierboven niet zo nuttig om één methode te schrijven voor twee Click events. Het is wel nuttig in bijvoorbeeld het volgende voorbeeld.

We plaatsen een **MenuStrip** en een **ToolStrip** op het formulier. Als je op de MenuStrip rechts klikt, kun je de optie 'Insert Standard Items' kiezen. Dat kan ook bij de ToolStrip. Je krijgt dan het volgende formulier:



De optie 'Opslaan' krijg je via de MenuStrip en via de ToolStrip. Daarbij horen de volgende event handlers:

```
Public Class Formulier

    Private Sub SaveToolStripButton_Click(…) Handles SaveToolStripButton.Click

    End Sub

    Private Sub SaveToolStripMenuItem_Click(…) Handles
                                                SaveToolStripMenuItem.Click

    End Sub

End Class
```

Het is veel handiger om beide event handlers samen te voegen tot één:

```
Public Class Formulier

    Private Sub Opslaan_Click() Handles SaveToolStripButton.Click,
                                        SaveToolStripMenuItem.Click
       'Code voor het opslaan
    End Sub

End Class
```

Index

parameter e

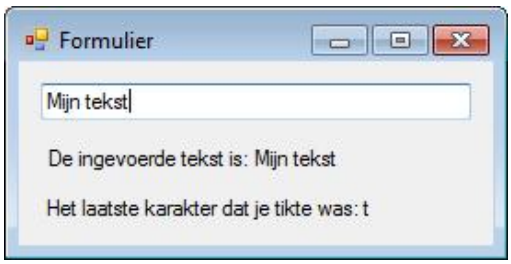
TextChanged

KeyPress

MouseClick

7.2.5 De parameter *e*

De **parameter e** vertelt je iets meer over het event zelf. Welke informatie je krijgt, hangt af van het event. We bekijken het onderstaande voorbeeld van een formulier met een tekstvak.




```
Public Class Formulier

    Private Sub tbMijnTextBox_TextChanged(ByVal sender As System.Object,
                                          ByVal e As System.EventArgs)
                                          Handles tbMijnTextBox.TextChanged

        lblTekst.Text = "De ingevoerde tekst is: " & tbMijnTextBox.Text
    End Sub

    Private Sub tbMijnTextBox_KeyPress(ByVal sender As Object,
            ByVal e As System.Windows.Forms.KeyPressEventArgs)
            Handles tbMijnTextBox.KeyPress

        lblLaatsteKarakter.Text = "Het laatste karakter dat je tikte was: "
                                & e.KeyChar
    End Sub

End Class
```

De eerste methode verwerkt het event tbMijnTextBox.**TextChanged**. Dit event vindt plaats zodra de tekst in het tekstvak wijzigt. De parameter *e* geeft in dit geval weinig nuttige extra informatie.

Dat is anders in het geval van de tweede methode. Deze verwerkt het event tbMijnTextBox.**KeyPress**. Dit event vindt plaats op het moment dat er een toets is ingedrukt door de gebruiker. De parameter *e* heeft in dit geval de eigenschap KeyChar die je vertelt welke toets er precies is ingedrukt.

Bij het event **MouseClick** geeft *e* weer andere informatie: wat de coördinaten van de muis zijn en of de gebruiker de linker- of rechtermuisknop heeft ingedrukt.

Er zijn zeer veel verschillende events. Om erachter te komen welke informatie de parameter *e* je bij elk event geeft, kun je het beste op het internet zoeken of ontdekken door te kijken welke eigenschappen *e* heeft door e. te tikken.

Index

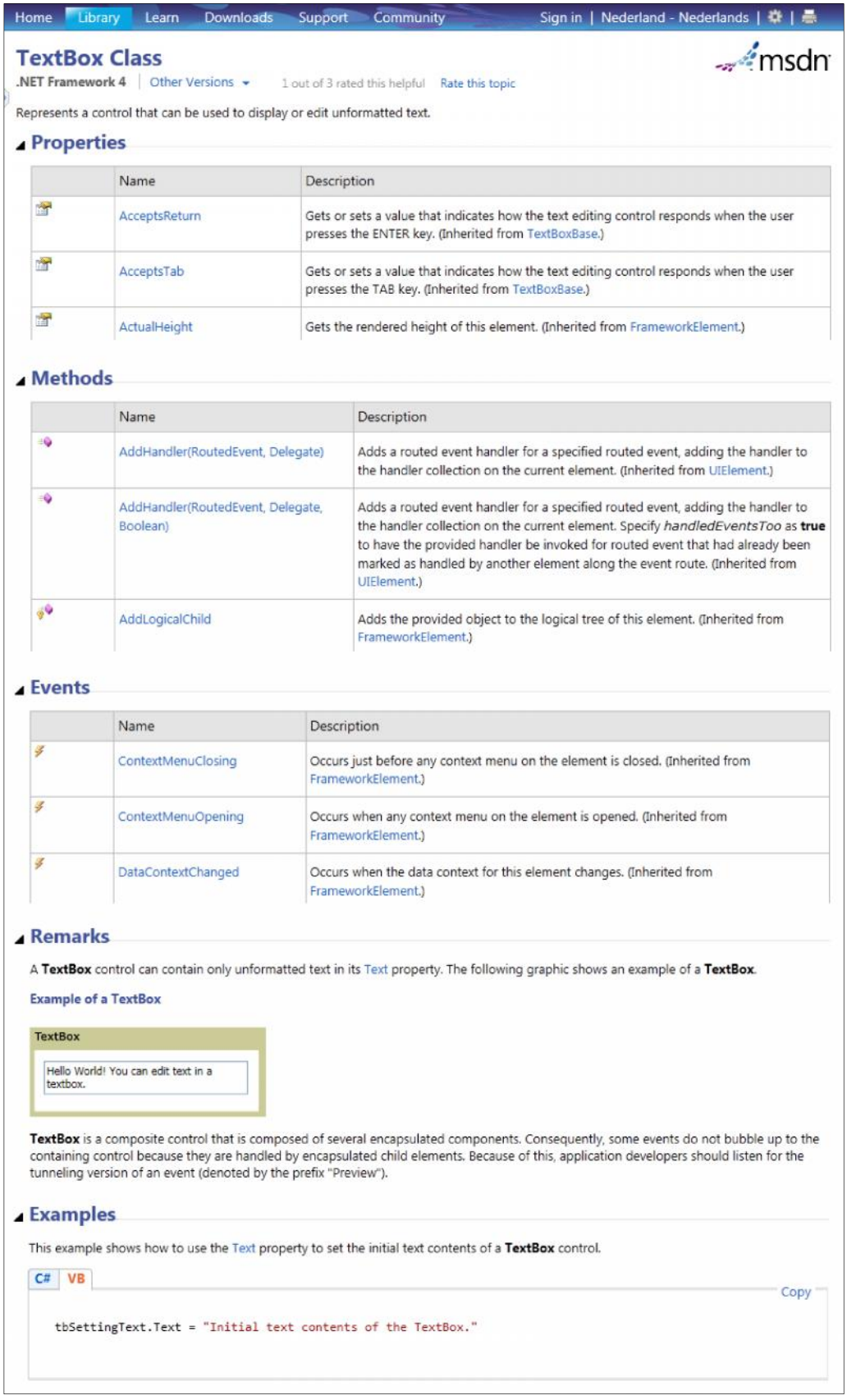
MSDN

7.3 Informatie op internet en via het MSDN

Je kunt op het internet heel veel informatie over Visual Basic vinden. Niet alleen uitleg maar ook veel codevoorbeelden. Een lastig punt daarbij is dat er veel verschillende oudere versies van Visual Basic zijn. De uitleg en de codevoorbeelden bij oudere versies kan niet meer geldig zijn. Als je via Google zoekt naar informatie kun je daarom het beste de zoekwoorden 'Visual Basic 2010' toevoegen.

Microsoft heeft een website waarop alle informatie over Visual Basic 2010 te vinden is: het Microsoft Developer Network (**MSDN**), te bereiken via http://msdn.microsoft.com. Een van de grote voordelen van MSDN is dat je van alle controls uit de Toolbox een overzicht kunt vinden van de eigenschappen, methoden en events van die control. Op de volgende bladzijde zie je een voorbeeld van informatie op het MSDN over de TextBox.

Helaas is de meeste informatie over Visual Basic 2010 alleen in het Engels op het internet beschikbaar. Maar er zijn veel codevoorbeelden en (video)tutorials die je goed op weg kunnen helpen bij jouw vraag. Daarnaast zijn er vele forums op het internet waar heel veel informatie te vinden is: vaak is je vraag al eens door een ander op zo'n forum aan de orde gesteld en vind je er uitstekende oplossingen.



Home | Library | Learn | Downloads | Support | Community | Sign in | Nederland - Nederlands

msdn

TextBox Class

.NET Framework 4 | Other Versions | 1 out of 3 rated this helpful | Rate this topic

Represents a control that can be used to display or edit unformatted text.

Properties

Name	Description
AcceptsReturn	Gets or sets a value that indicates how the text editing control responds when the user presses the ENTER key. (Inherited from TextBoxBase.)
AcceptsTab	Gets or sets a value that indicates how the text editing control responds when the user presses the TAB key. (Inherited from TextBoxBase.)
ActualHeight	Gets the rendered height of this element. (Inherited from FrameworkElement.)

Methods

Name	Description
AddHandler(RoutedEvent, Delegate)	Adds a routed event handler for a specified routed event, adding the handler to the handler collection on the current element. (Inherited from UIElement.)
AddHandler(RoutedEvent, Delegate, Boolean)	Adds a routed event handler for a specified routed event, adding the handler to the handler collection on the current element. Specify *handledEventsToo* as **true** to have the provided handler be invoked for routed event that had already been marked as handled by another element along the event route. (Inherited from UIElement.)
AddLogicalChild	Adds the provided object to the logical tree of this element. (Inherited from FrameworkElement.)

Events

Name	Description
ContextMenuClosing	Occurs just before any context menu on the element is closed. (Inherited from FrameworkElement.)
ContextMenuOpening	Occurs when any context menu on the element is opened. (Inherited from FrameworkElement.)
DataContextChanged	Occurs when the data context for this element changes. (Inherited from FrameworkElement.)

Remarks

A **TextBox** control can contain only unformatted text in its Text property. The following graphic shows an example of a **TextBox**.

Example of a TextBox

TextBox

Hello World! You can edit text in a textbox.

TextBox is a composite control that is composed of several encapsulated components. Consequently, some events do not bubble up to the containing control because they are handled by encapsulated child elements. Because of this, application developers should listen for the tunneling version of an event (denoted by the prefix "Preview").

Examples

This example shows how to use the Text property to set the initial text contents of a **TextBox** control.

C# VB Copy

```
tbSettingText.Text = "Initial text contents of the TextBox."
```