Custom Controls in Silverlight 2

DE KRACHT VAN HET PARTS & STATES MODEL

Als je een custom control in Silverlight wilt ontwikkelen, is het verstandig een strikte scheiding aan te brengen tussen de logica en de look & feel van een control. In dit artikel beschrijft de auteur in detail hoe je deze scheiding kunt implementeren met het Parts & States Model in Silverlight 2.

De control is de elementaire bouwsteen van de User Interface in Silverlight. Een control is herbruikbaar, heeft visuele eigenschappen en is ontwikkeld voor een specifiek doel. Een button bijvoorbeeld heeft als doel om via een muisklik (of toetsaanslag) een actie te starten. De uitgebreide control library van Silverlight bestaat uit eenvoudige controls als de Button, CheckBox, ListBox en bevat daarnaast complexere controls als de DataGrid en Calendar. De controls van de Silverlight control library worden ook wel common controls genoemd.

Wanneer er behoefte is aan een control die niet voorkomt in de control library kun je besluiten zelf een control te ontwikkelen, de custom control. Dit gebeurt door gebruik te maken van een bestaande control of van de Control base class. Alle controls in Silverlight zijn direct of indirect afgeleid van deze Control base class (type System.Windows.Controls.Control).

Als je een custom control in Silverlight wilt ontwikkelen, is het verstandig een strikte scheiding aan te brengen tussen de logica en de look & feel van een control. Dit stelt een designer namelijk in staat de visuele aspecten van de control te wijzigen zonder de logica aan te hoeven passen. Andersom kan een control-ontwikkelaar de logica aanpassen zonder dat de designer een nieuwe look & feel moet maken.

Naast de control kent Silverlight de zogenaamde UserControl. Bij het ontwikkelen van een User Interface worden verschillende controls samengevoegd, waarbij de onderlinge samenhang van deze controls vast komt te liggen in code. Deze controls en logica worden typisch ondergebracht in de UserControl. Het gebruik van UserControls zorgt ervoor dat je de functionaliteit van een applicatie laat opsplitsen in beheersbare en herbruikbare onderdelen.

Styles

Het scheiden van logica en design is geen onbekend verschijnsel. Neem bijvoorbeeld de Cascading Style Sheet met de definitie van de look & feel van een HTML-pagina. In de HTML-pagina wordt bepaald wat er te zien is en in de CSS ligt vast hoe de user interface er uiteindelijk uit komt te zien. Een vergelijkbaar concept is geïmplementeerd in Silverlight door middel van Styles. Deze worden gedefinieerd in de applicatieresource App.xaml en zijn voorzien van een unieke sleutel. De style wordt vervolgens expliciet toegekend zoals weergegeven in codevoorbeeld 1. De blauwe en rode button style, vastgelegd in de applicatieresource, worden toegekend door middel van de Style-property op de buttons.

Silverlight gaat aanzienlijk verder. Een designer heeft namelijk de mogelijkheid om de complete lay-out van een control te vervangen. Wanneer we de vergelijking met HTML en CSS doortrekken, betekent dit dat de designer

```
<Application.Resources>
    <Style TargetType="Button" x:Key="Blauw">
        <Setter Property="Background" Value="Blue"/>
    </Style>
    <Style TargetType="Button" x:Key="Rood">
        <Setter Property="Background" Value="Red"/>
    </Style>
</Application.Resources>

<Button Content="Default" />
<Button Style="{StaticResource Blauw}" Content="Blauw" />
<Button Style="{StaticResource Rood}" Content="Rood" />
```

CODEVOORBEELD 1. STYLE-DECLARATIE EN -TOEKENNING

AFBEELDING 1. STYLED BUTTONS

in CSS de mogelijkheid zou hebben om bijvoorbeeld de standaard HTML-input-button te vervangen door een ronde image-button.

Om het vervangen van de lay-out van een custom control mogelijk te maken, zijn er wel afspraken nodig tussen de programmeur en de designer. Deze afspraken worden opgesteld in het zogenaamde control contract. Hierin staat exact waar een control uit bestaat en wat de mogelijke states van een control zijn. Een implementatie van een control contract is het Parts & States Model. Het is niet verplicht om dit model in Silverlight te gebruiken, maar zoals we zullen toelichten biedt het model dermate veel voordelen dat we met klem aanraden dit wel te doen.

Parts & States

We geven uitleg over het Parts & States Model aan de hand van een TrafficLight custom control. TrafficLight is een eenvoudige control die een verkeerslicht representeert, bestaande uit een rode, oranje en groene lamp. Bij het klikken op één van de lampen springt het stoplicht op rood, op groen of gaat oranje knipperen.

Wanneer de control-ontwikkelaar de XAML van codevoorbeeld 2 hard codeert in de control, valt er weinig te designen. Als een designer besluit dat de lampen weergegeven moeten worden door middel van Rectangle shapes, kan de programmeur de code gaan aanpassen. Dit is geen wenselijke situatie, dus tijd om het Parts & States Model te introduceren.

Het Parts & States Model bestaat uit vier onderdelen: Parts, States, Tran-

feel free to innovate

We zijn er weer.
Opgeladen, bijgetankt en uitgerust.
Ruimte in ons hoofd voor nieuwe plannen en volop energie om ze uit te voeren.

Die ambitie is de drijfveer achter onze projecten en de motor van onze groei.
Je uitleven in je vak en in je passies.
Dat kan bij XCESS.

XCESS
expertise center

XCESS expertise center b.v.
Storkstraat 19 • 3833 LB LEUSDEN
Telefoon 033 - 433 51 51 • Fax 033 - 432 09 56
Web www.xcess.nl • E-mail info@xcess.nl

Wij hebben pas echte fans

Kom bij het winnende team met de prijswinnaars in software digital rights management.

HASP SRM
SOFTWARE RIGHTS MANAGEMENT

Robuuste softwarebescherming
Beveiligde & flexibele licentiering
Vertrouw voor uw softwarebeveiliging op
HASP SRM: de nummer 1 wereldwijd in softwarebescherming en licentieoplossingen*

- **Bescherm uw software tegen piraterij en illegaal gebruik**
- **Beveilig uw waardevol intellectueel eigendom tegen reverse-engineering & diefstal**
- **Creëer nieuwe businessmodellen met veilige & flexibele licentiering**
- **Integreer uw bescherming & licentiering volledig met uw software product-lifecycle**

Vraag een GRATIS HASP SRM Developer Kit aan op:
www.Aladdin.com/winner
*Frost & Sullivan N1AF-70, IDC #34452

Nu versie 3.50 met encryptie voor JAVA-applicaties

• BENELUX +31-30-688-0800
• UK • NORTH AMERICA • ISRAEL
• GERMANY • FRANCE • SPAIN
• ITALY • INDIA • CHINA
• JAPAN • PORTUGAL

Aladdin
SECURING THE GLOBAL VILLAGE

Aladdin.com/HASP

```
<StackPanel x:Name="Root" Background="#333333"
HorizontalAlignment="Center" VerticalAlignment="Center">
    <Ellipse x:Name="Red" Width="50" Height="50" Fill="Red" Mar-
gin="8,8,8,0" Cursor="Hand" />
    <Ellipse x:Name="Orange" Width="50" Height="50" Fill="Black"
Margin="8,8,8,0" Cursor="Hand" />
    <Ellipse x:Name="Green" Width="50" Height="50" Fill="Black"
Margin="8,8,8,8" Cursor="Hand" />
</StackPanel>
```

CODEVOORBEELD 2. TRAFFICLIGHT XAML

AFBEELDING 2. TRAFFICLIGHT UI

sitions en State Groups. De Parts van het Parts & States Model beschrijft de elementen waar een control uit bestaat. De control-logica heeft deze elementen nodig om zijn functionaliteit te implementeren. In het geval van de TrafficLight-control zijn er drie parts te definiëren, namelijk de drie Ellipse-shapes die de rode, oranje en groene lampen voorstellen. Klik je met de muis op één van de elementen, dan verandert de state van de TrafficLight-control.
De verschillende visual states van een control vormen het tweede onderdeel van het Parts & States Model. Voor de TrafficLight-control zijn de visual states Stop, Go en Off gedefiniëerd. Bij Stop brandt het rode licht, bij Go het groene en in de Off-state knippert het oranje licht.
Het Parts & States Model is vastgelegd in een zogenaamd Control Template. Het default Control Template, bestaande uit de Parts & States van de TrafficLight-control, is weergegeven in codevoorbeeld 3.
De lay-out van de TrafficLight bestaat nog steeds uit een StackPanel met drie Ellipse shapes. Aan het template zijn nu een aantal visual state-elementen toegevoegd. De visual states van een control worden beschreven met behulp van een Storyboard. Een Duration met waarde 0 geeft aan dat het Storyboard statisch is en dat de look & feel gelijk blijft zolang de control in deze visual state blijft. Het Storyboard van de Off-state heeft een Duration van één seconde en in deze seconde wordt de oranje Ellipse geanimeerd. Omdat het Storyboard AutoReverse op True heeft staan en zichzelf eindeloos herhaalt, bereik je hiermee het effect van een knipperende oranje lamp.
De standaard look & feel van de TrafficLight-control bestaat uit drie Ellipse shapes, alle gevuld met de kleur Black. In de verschillende visual states wordt vervolgens alleen beschreven welke elementen afwijken van de standaard look & feel. Zo beschrijft de Stop-state dat de rode Ellipse gevuld moet worden met de kleur Red. De Stop-state zegt niets over de andere Ellipse shapes. Dit betekent automatisch dat de overige Ellipse shapes de standaard look & feel krijgen en dus gevuld worden met de kleur Black. Dit komt doordat de states van de TrafficLight-control gegroepeerd zijn in een zogenaamde State Group, het derde onderdeel van het Parts & States Model. Een State Group wordt gebruikt om states die elkaar uitsluiten te groeperen. Wanneer de rode lamp brandt, kan het niet zo zijn dat de groene lamp ook brandt. De states Go en Stop sluiten elkaar uit.
Silverlight biedt de mogelijkheid om verscheidene State Groups te definiëren. In het geval van de TrafficLight-control zouden we een State Group kunnen introduceren voor de verschillende muis-states. De states Normal, Pressed en MouseOver worden in de standaard Silverlight-controls gebruikt om visuele feedback te geven over de status van de muis. Het gebruik van meer State Groups heeft tot gevolg dat een control tegelijkertijd verscheidene visual states kan hebben. Bijvoorbeeld bij TrafficLight zowel in de state Go als in de state MouseOver. Dit is mogelijk omdat beide states in verschillende State Groups zitten en elkaar daardoor niet uitsluiten.

Control implementatie

De functionaliteit van de TrafficLight-control wordt geïmplementeerd in een standaard Silverlight .NET class die afgeleid is van System.Windows.Controls.Control. Om gebruik te maken van het ControlTemplate zoals hiervoor beschreven, implementeren we de method OnApplyTemplate. Deze method wordt aangeroepen zodra een ControlTemplate door de Silverlight runtime is gekoppeld aan een control-instantie. In de OnApplyTemplate method worden de Parts van het control contract opgezocht met behulp van de GetTemplateChild method. Omdat de parts voorzien zijn van een uniek x:Name attribuut, heeft de programmeur de mogelijkheid references naar de verschillende Parts op te zetten. Met behulp van de reference van de verschillende parts kunnen vervolgens event handlers gekoppeld worden. De complete source van de TrafficLight-control is weergeven in codevoorbeeld 4.
De TrafficLight class is gedecoreerd met een aantal TemplatePart en TemplateVisualState attributen. Externe tools als Expression Blend gebruiken deze attributen om te lezen welke Parts & States verwacht worden door de control. Met deze attributen wordt het control contract declaratief opgesteld. De Silverlight runtime maakt zelf geen gebruik van deze design-time attributen. Bij het definiëren van de TemplatePart attributen wordt ook het verwachte type van de verschillende Parts aangegeven. Voor een flexibel contract is het zaak een zo generiek mogelijk type op te geven. Wanneer de TemplatePart van de TrafficLight-lampen voorzien zou zijn van het type Ellipse, is de designer verplicht altijd een Ellipse shape te gebruiken in zijn designs. In het voorbeeld is het type FrameworkElement gespecificeerd, nu is de designer vrij

```
<ControlTemplate TargetType="tl:TrafficLight">
    <StackPanel x:Name="Root" Background="#333333"
HorizontalAlignment="Center" VerticalAlignment="Center">
      <vsm:VisualStateManager.VisualStateGroups>
        <vsm:VisualStateGroup x:Name="TrafficLightStates">
          <vsm:VisualState x:Name="Off">
            <Storyboard Duration="0:0:1" AutoReverse="True"
            RepeatBehavior="Forever">
              <ColorAnimation
                Storyboard.TargetName="Orange"
                Storyboard.TargetProperty="(Fill).(Color)"
                From="Black" To="Orange" />
            </Storyboard>
          </vsm:VisualState>
          <vsm:VisualState x:Name="Go">
            <Storyboard Duration="0">
              <ColorAnimation
                Storyboard.TargetName="Green"
                Storyboard.TargetProperty="(Fill).(Color)"
                 To="Lime"/>
            </Storyboard>
          </vsm:VisualState>
          <vsm:VisualState x:Name="Stop">
            <Storyboard Duration="0">
              <ColorAnimation
                Storyboard.TargetName="Red"
                Storyboard.TargetProperty="(Fill).(Color)"
                To="Red"/>
            </Storyboard>
          </vsm:VisualState>
        </vsm:VisualStateGroup>
      </vsm:VisualStateManager.VisualStateGroups>

      <Ellipse x:Name="Red" Width="50" Height="50" Fill="Black"
      Margin="8,8,8,0" Cursor="Hand" />
      <Ellipse x:Name="Orange" Width="50" Height="50"
      Fill="Black" Margin="8,8,8,0" Cursor="Hand" />
      <Ellipse x:Name="Green" Width="50" Height="50" Fill="Black"
      Margin="8,8,8,8" Cursor="Hand" />
    </StackPanel>
</ControlTemplate>
```

CODEVOORBEELD 3. TRAFFICLIGHT CONTROL TEMPLATE

om te kiezen voor elk type dat afgeleid is van FrameworkElement.
In de constructor van de TrafficLight-control wordt de DefaultStyleKey gezet. Deze property-toekenning zorgt er voor dat de TrafficLight-control wordt geïnitialiseerd met het default template. Default control templates worden opgenomen in een speciale XAML file, genaamd Generic.xaml. Deze file wordt geplaatst in het Silverlight project van de control. Tijdens het compileren wordt deze Generic.xaml als resource toegevoegd aan de assembly. In Generic.xaml worden de default styles opgeslagen, zoals de programmeur deze uitlevert. In feite is een default control template niets anders dan een control template zoals een designer deze aanlevert. Wanneer door middel van een Style Setter een ander Template wordt toegepast op een control, wordt het default template niet gebruikt.

VisualStateManager

Het wijzigen van de visual state van een control vindt plaats met behulp van de GoToState method van de class VisualStateManager (ook wel VSM genoemd). In code is er slechts één method-aanroep nodig, maar onder water gebeurt er een heleboel. De VisualStateManager berekent aan de hand van de visual state-instellingen in het ControlTemplate wat er met welke visuele elementen moet gebeuren. Wanneer de TrafficLight-control bijvoorbeeld van de Stop-state naar de Go-state gaat, moet de Ellipse Green gevuld worden met de kleur Green en krijgt de Ellipse Red de standaardkleur Black, zoals gedefinieerd in de standaard look & feel. De VisualStateManager berekent dit automatisch, creëert de verschillende animaties om de transitie van de begin-state naar de eind-state mogelijk te maken en voert deze vervolgens uit. De TrafficLight-control werkt nu redelijk naar behoren, maar toch kloppen er nog een aantal dingen niet. Zo moet bijvoorbeeld eerst de oranje lamp gaan branden voordat de rode lamp aan gaat wanneer de TrafficLight-control van de state "Go" naar de state "Stop" gaat. Dit gedrag wordt geïmplementeerd met behulp van zogenaamde Visual State Transitions, het vierde onderdeel van het Parts & States Model. Een transition beschrijft expliciet hoe een control van de ene state naar de andere state overgaat. Codevoorbeeld 5 geeft weer hoe de TrafficLight van state "Go" naar state "Stop" gaat.
De transition duurt twee seconden en gebruikt color animations om de lampen te animeren. De groene lamp verandert in zwart, terwijl de oranje lamp knippert om na twee seconden ook op zwart te gaan. De rode lamp ten slotte springt na anderhalve seconde op rood.
De To en From properties van een transition geven de begin- en eind-state van de transition aan. To en From zijn optioneel. Zo is het mogelijk om alleen een eind-state op te geven voor een transition. Welke visual transition uitgevoerd gaat worden, bepaalt uiteindelijk de VisualStateManager. Daarbij krijgt de meest specifieke transition een hogere prioriteit. Een transition waarbij zowel de To overeenkomt met de gewenste eind-state en From met de huidige state, heeft een hogere prioriteit dan de transition waarbij alleen de From overeenkomt met de huidige state.
De GotoState method bevat een parameter die aangeeft of er wel of niet gebruik wordt gemaakt van transitions. Gebeurt dit niet, dan springt de control direct naar de opgegeven eind-state. Het niet gebruiken van transitions vindt met name plaats bij het initialiseren van een control in de OnAppyTemplate. GotoState retourneert een boolean die aangeeft of de actie succesvol was. Wanneer de waarde false is, kan dit betekenen dat de state niet bestaat of dat de control zich al in de gewenste state bevindt.

```
[TemplatePart(Name = "Red", Type = typeof(FrameworkElement))]
[TemplatePart(Name = "Orange", Type = typeof(FrameworkElement))]
[TemplatePart(Name = "Green", Type = typeof(FrameworkElement))]
[TemplateVisualState(Name = "Go")]
[TemplateVisualState(Name = "Stop")]
[TemplateVisualState(Name = "Off")]
public class TrafficLight : Control {
    public TrafficLight() {
      this.DefaultStyleKey = typeof(TrafficLight);
    }
    public override void OnApplyTemplate() {
      base.OnApplyTemplate();

      FrameworkElement red = GetTemplateChild("Red") as Framewor-
      kElement;
      FrameworkElement orange = GetTemplateChild("Orange") as
      FrameworkElement;
      FrameworkElement green = GetTemplateChild("Green") as
      FrameworkElement;

      red.MouseLeftButtonUp += red_MouseLeftButtonUp;
      orange.MouseLeftButtonUp += orange_MouseLeftButtonUp;
      green.MouseLeftButtonUp += green_MouseLeftButtonUp;

      // set initial state
      VisualStateManager.GoToState(this, "Off", false);
    }
    private void red_MouseLeftButtonUp(object sender, MouseButto-
    nEventArgs e) {
      VisualStateManager.GoToState(this, "Stop", true);
    }
    private void orange_MouseLeftButtonUp(object sender, MouseBut-
    tonEventArgs e) {
      VisualStateManager.GoToState(this, "Off", true);
    }
    private void green_MouseLeftButtonUp(object sender, MouseBut-
    tonEventArgs e) {
      VisualStateManager.GoToState(this, "Go", true);
    }
}
```

CODEVOORBEELD 4. TRAFFICLIGHT SOURCE

Custom designs

Wat is er nu eigenlijk bereikt? Een implementatie van de TrafficLight-control met een minimale hoeveelheid code die zwaar leunt op de VisualStateManager om de control in de verschillende visual states te brengen. Maar het allerbelangrijkste is wel dat het complete default control contract, opgeslagen in het ControlTemplate, nu volledig vervangen kan worden door de designer. Het enige dat de programmeur nodig heeft zijn de verschillende parts en states van de control. De designer bepaalt de manier waarop de control overgaat van de ene naar de andere state. Deze kan nu een compleet nieuwe look & feel samenstellen en deze toepassen op de control. Afbeelding 3 laat verschillende designs zien van de TrafficLight-control. Het design van het voetgangersverkeerslicht bestaat uit slechts een rode en een groene lamp en heeft een compleet ander visual transition-model dan de default TrafficLight. Dankzij het Parts & States Model zijn er geen wijzigingen nodig in de control om de nieuwe designs mogelijk te maken.
In het control contract van de TrafficLight-control is er voor gekozen veel verantwoordelijkheid bij de designer neer te leggen. De business logica van de TrafficLight-control is minimaal en de designer kan zelf bepalen hoe en welke lampen op welke momenten gaan branden. Dat maakt het contract zeer flexibel en biedt de designer veel vrijheid. In de praktijk zullen echter met het toenemen van de complexiteit van de control logica meer restricties in het control contract worden vastgelegd.

Tips

Parts en Visual States in het Parts & States Model kunnen verplicht of optioneel zijn. Door middel van het raisen van exceptions in OnApplyTemplate kan de programmeur het gebruik van bepaalde parts en states afdwingen. Daarnaast dient de control-implementatie flexibel genoeg te zijn om correct te blijven werken wanneer optionele onderdelen niet aanwezig blijken te zijn in een design.
Denk goed na of het nodig is een custom control zelf te bouwen. Zoals aangegeven biedt Silverlight uitgebreide mogelijkheden om de visuele representatie van bestaande controls te wijzigen en zelfs te vervangen. Vaak biedt het re-designen van bestaande controls uitkomst en hoef je geen eigen custom control te ontwikkelen.

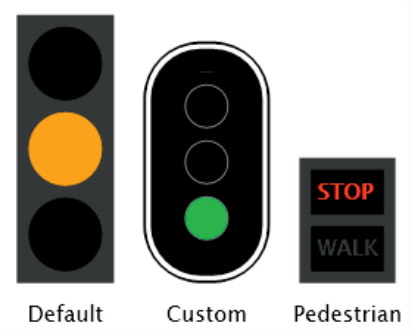
```
<vsm:VisualTransition Duration="0:0:2" From="Go" To="Stop">
    <Storyboard>
      <ColorAnimationUsingKeyFrames
        Storyboard.TargetName="Orange"
        Storyboard.TargetProperty="(Fill).(Color)"
        Duration="0:0:2">
        <LinearColorKeyFrame KeyTime="0:0:0.5" Value="Orange"/>
        <LinearColorKeyFrame KeyTime="0:0:1.5" Value="Orange"/>
        <LinearColorKeyFrame KeyTime="0:0:2" Value="Black"/>
      </ColorAnimationUsingKeyFrames>
      <ColorAnimation
        Storyboard.TargetName="Green"
        Storyboard.TargetProperty="(Fill).(Color)"
        Duration="0:0:0.5" To="Black"/>
      <ColorAnimation
        Storyboard.TargetName="Red"
        Storyboard.TargetProperty="(Fill).(Color)"
        BeginTime="0:0:1.5" Duration="0:0:0.5" To="Red"/>
    </Storyboard>
</vsm:VisualTransition>
```

CODEVOORBEELD 5. VISUAL TRANSITION

Met de release van Silverlight 2 Beta 2 begin mei 2008 is het Parts & States Model geïntroduceerd. Er is nog weinig ervaring met het gebruik van het model in 'real-life' applicaties. Microsoft is daarom zeer geïnteresseerd in de ervaringen van zowel ontwikkelaars als designers. Opmerkingen en suggesties zijn welkom.

Samenvatting

De VisualStateManager vervult een centrale rol in het Parts & States Model en neemt de control-ontwikkelaar veel werk uit handen door zelf te berekenen welke transitions nodig zijn om van een begin-state in een bepaalde eind-state te geraken. Het Parts & States Model vereenvoudigt de samenwerking tussen de ontwikkelaar en de designer. Door middel van het opstellen van



AFBEELDING 3. CUSTOM DESIGNED TRAFFICLIGHTS

een expliciet control contract kunnen beide partijen onafhankelijk van elkaar wijzigingen aanbrengen in een control. Alhoewel het Parts & States Model niet verplicht is, raden wij het ten sterkste aan bij het ontwikkelen van custom controls. Met de ondersteuning van dit model in alle standaard Silverlight controls en in designer-tools als Expression Blend zet Microsoft volledig in op het Parts & States Model. .net

Referenties

The Official Microsoft Silverlight Site: http://silverlight.net

Blog Karen Corby (Microsoft Product Manager): http://scorbs.com

Blog Koen Zwikstra: http://firstfloorsoftware.com

Koen Zwikstra (kozw@firstfloorsoftware.com) is een Silverlight-ontwikkelaar van het eerste uur. Sinds de eerste publieke releases van Silverlight heeft hij verschillende applicaties en tools ontwikkeld en mede door zijn actieve participatie in de Silverlight-community heeft hij veel kennis en ervaring opgedaan met Silverlight.