12: Control Templates - Beyond Buttons

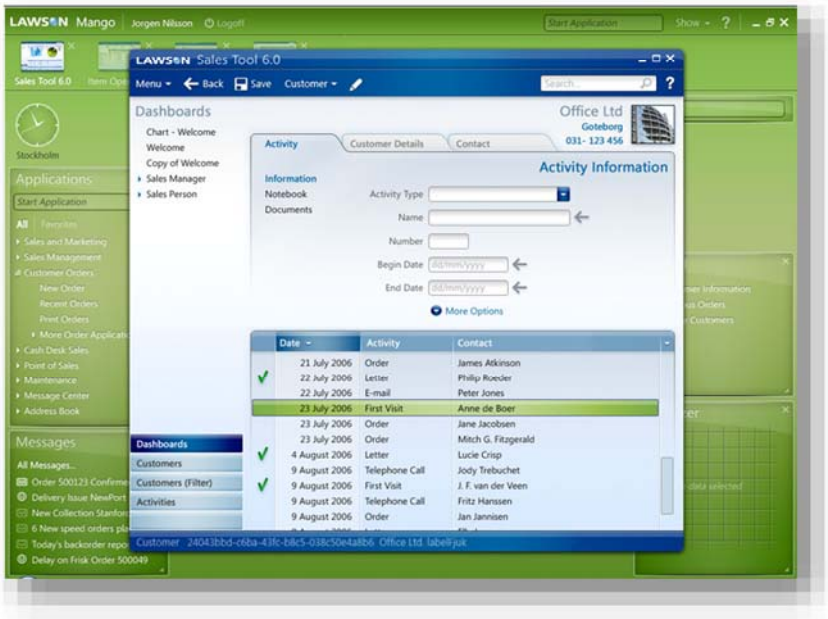
Control Templates: Beyond Buttons

DEVELOPMENTOR
DEVELOPING PEOPLE WHO DEVELOP SOFTWARE

12: Control Templates - Beyond Buttons

Going Beyond the simple Button

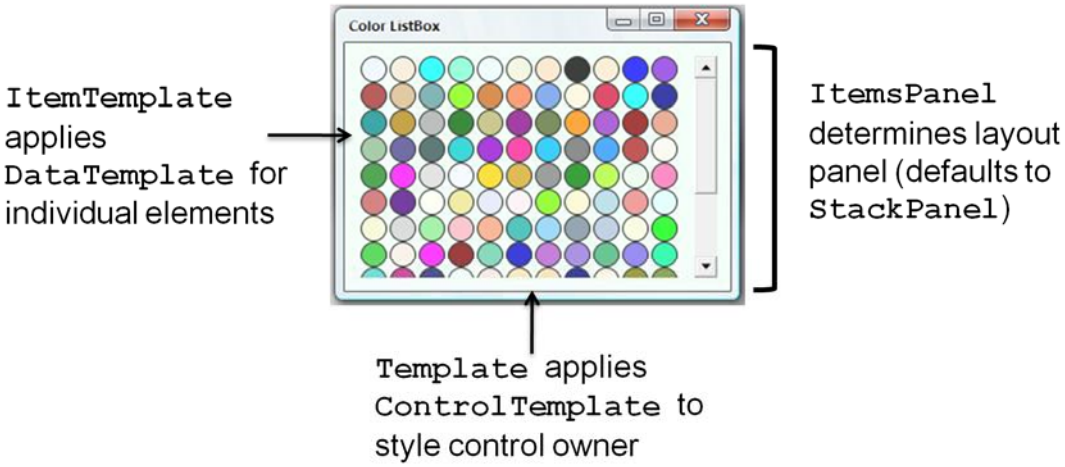
- **Every control has a Control Template which can be replaced**



DEVELOPMENTOR 2

Working with ItemsControls

- `ItemsControl` **is a visual container for elements**
  - base class for all controls which hold multiple children[1]
  - several templates allow complete visual customization



DEVELOPMENTOR 3

[1] This is a super important point! All of these templates are supported by TreeView, Menu, ListBox, ComboBox, TabControl, etc.

12: Control Templates - Beyond Buttons

Example: ItemsControl templates

```
<ListBox ItemsSource="{StaticResource brushCollection}"
         ItemTemplate="{StaticResource smallCircle}"
         ItemsPanel="{StaticResource colorPanel}"
         Template="{StaticResource colorTemplate}" />
```

template expanded for each item

template expanded to contain items[1]

template expanded to represent control appearance

```
<DataTemplate x:Key="smallCircle">
   <Ellipse Width="20" Height="20"
       Fill="{Binding}" Opacity=".75"
       Stroke="Black" StrokeThickness="1" />
</DataTemplate>
...
<ItemsPanelTemplate x:Key="colorPanel">
   <WrapPanel IsItemsHost="true" />
</ItemsPanelTemplate>
...
<ControlTemplate x:Key="colorTemplate">
  ...
</ControlTemplate>
```

DEVELOPMENTOR 4

[1] Note the usage of the IsItemsHost property on the panel – this tells the ItemsPanel which panel in the definition is the formal items host. It's not required in this case since there is only one and we are defining it in the template, but it's a good idea to mark your primary "item" panel.

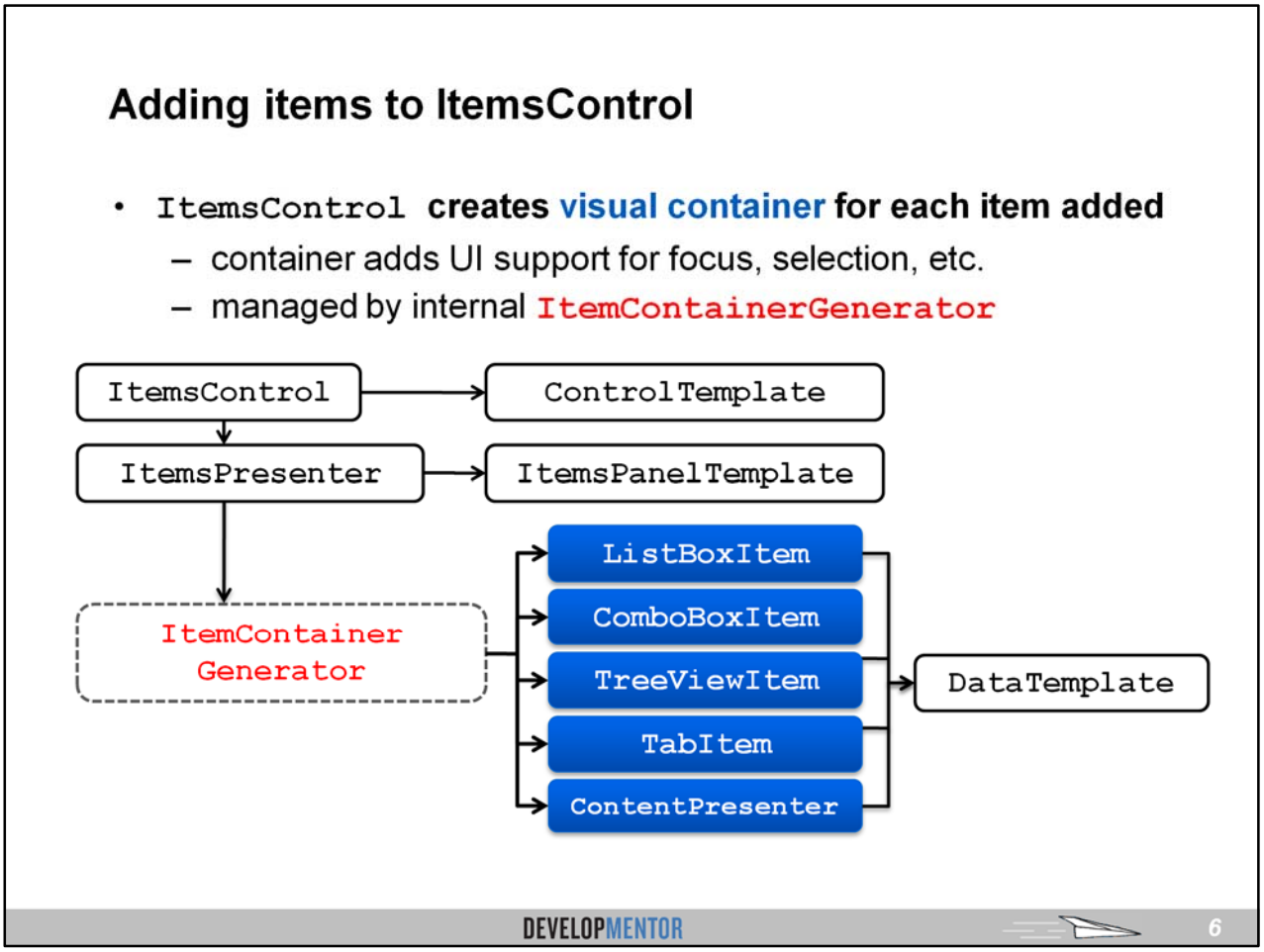
ItemsPresenter: where the content goes

- `ItemsPresenter` **used as placeholder for items in template**
  - expands to full tree through `ItemsPanelTemplate`

```
<ListBox.Template>
   <ControlTemplate TargetType="{x:Type ListBox}">
      <Border Background="{TemplateBinding Background}">
         <ScrollViewer HorizontalScrollBarVisibility="Auto">
            <ItemsPresenter Margin="5" />
         </ScrollViewer>
      </Border>
   </ControlTemplate>
</ListBox.Template>
```

panel container and child items will be placed **here** in the visual tree

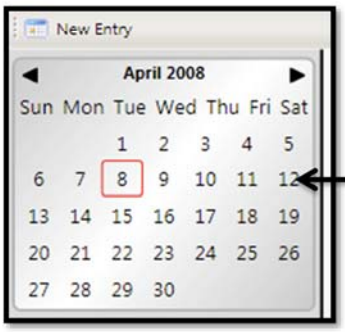
DEVELOPMENTOR 5



The actual creation of the container is done by the ItemsControl itself through ItemsControl.GetContainerForItemOverride(). ListBox, TreeView, TabControl and ComboBox provide specialized containers as shown above and everything else uses a ContentPresenter.

When the visual container gets in the way

- **Container is owned by the `ItemsControl` panel**
  - your `DataTemplate` is owned by the container



`ItemsControl` using `Grid` as panel

each day needs to be placed in proper column and row so we add attached properties onto our `DataTemplate`…

… but `Grid` does not directly own `TextBlock` so properties are ignored

```
<DataTemplate>
   <TextBlock Text="{Binding Day}"
       Grid.Column="{Binding ColumnIndex}"
       Grid.Row="{Binding RowIndex}" />
</DataTemplate>
```

DEVELOPMENTOR 7

Changing the visual container properties

- **Wrapper properties adjusted through** `ItemsContainerStyle`
  - applied to every generated wrapper
  - can supply triggers for selection and mouse over effects

```
<ListBox>
   <ListBox.ItemContainerStyle>
      <Style TargetType="ListBoxItem">
         <Setter Property="Grid.Column"
                 Value="{Binding ColumnIndex}" />
         <Setter Property="Grid.Row"
                 Value="{Binding RowIndex}" />
      </Style>
   </ListBox.ItemContainerStyle>
  ...
</ListBox
```

`DataContext` is inherited from owning control

DEVELOPMENTOR 8

An example of a trigger might be to provide alternate line colors for each row – this is actually very easy to do in WPF 3.5 SP1 due to a new property:

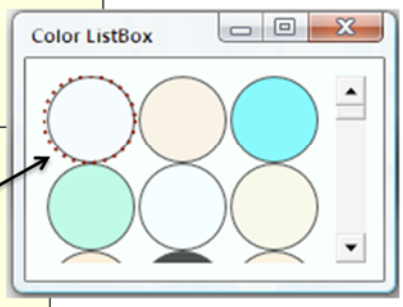
```
<ListBox AlternationCount="2">
  <ListBox.ItemContainerStyle>
    <Style TargetType="ListBoxItem">
      <Style.Triggers>
        <Trigger Property="ListBox.AlternationIndex" Value="0">
          <Setter Property="Background" Value="LightGray" />
        </Trigger>
        <Trigger Property="ListBox.AlternationIndex" Value="1">
          <Setter Property="Background" Value="White" />
        </Trigger>
      </Style.Triggers>
    </Style>
  </ListBox.ItemContainerStyle>
</ListBox>
```


Changing the focus style

- **Focus visual effect is determined by** `FocusVisualStyle` **template**
  - identified `Style` defines `ControlTemplate` to use for focus

```
<ListBox>
   <ListBox.ItemContainerStyle>
      <Style TargetType="ListBoxItem">
         <Setter Property="FocusVisualStyle"
              Value="{StaticResource fStyle" />
      </Style>
   </ListBox.ItemContainerStyle>
  ...
</ListBox
```

```
<Style x:Key="fStyle">
   <Setter Property="Control.Template">
      <Setter.Value>
         <ControlTemplate ... />
      </Setter.Value>
   </Setter>
</Style>
```



DEVELOPMENTOR 9

Other templates

- **More complex controls utilize multiple templates**
  - controls often composed of primitives which may be styled
- **Start with `Template` then apply styles to inner controls**
  - may also include multiple primary templates

`GridView.ColumnHeaderContainerStyle` decides UI for header

`GridView.CellTemplate` provides UI for individual cells

Name	Occupation	Salary
Jim Wilson	Paris Hilton's Dog Walker	30000
Karen Jones	Oprah's Dietician	94350
Homer Simpson	Donut Specialist	10500
Jason Tardus	Michael Jackson's Cosmetic Surgeon	245234
Jason Whittington	Porta-Potty Cleaner	9500
Catbert	Evil HR Director	3235000
Theo Curry	Danny Bonaducci Stunt Double	10300
Sally Smith	Street Corner Chicken Sign	1000

primary template utilizes `ScrollViewer` which may be styled to change overall appearance such as how control is scrolled

DEVELOPMENTOR 10

Some rules to follow when templating controls

- **Many controls must tie behavior to visual elements**
  - so they name the elements to enable lookup in code behind
  - code logic and triggers looks for these named elements
- **Convention is to prefix names with `PART_`**
  - templates must supply properly named elements or control will not work properly
  - control should use `[TemplatePart]` to declare requirements
- **Requirements vary from control to control**
  - you will need to examine standard control template
- **Should generally always start with existing template**
  - ensures required elements are present with proper names
  - hide things you do not want vs. deleting them

DEVELOPMENTOR 11

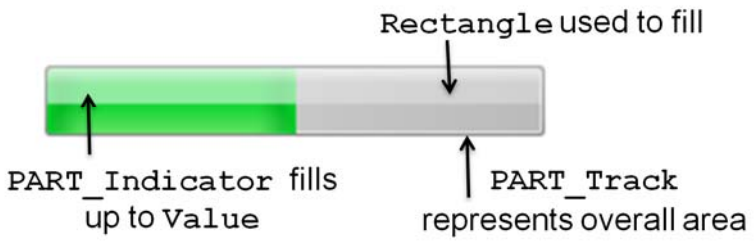
Controls sometimes have required elements

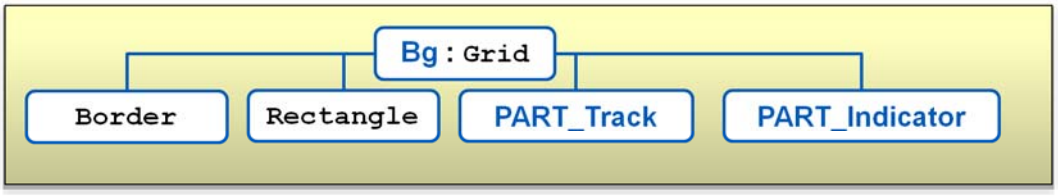
- `RangeBase` **controls (**`Progress Bar, Slider, ScrollBar`**)**
  - `PART_Track` defines the thumb and support for increase and decrease buttons
  - `PART_Indicator` defines the percentage of the current value
- `ComboBox`
  - `PART_Popup` defines the "menu" portion
  - `PART_EditableTextBox` provides synchronized selection as textbox changes
- `TextBox`**-based controls (including in** `ComboBox`**)**
  - `PART_ContentHost` implements editable portion of control

DEVELOPMENTOR 12

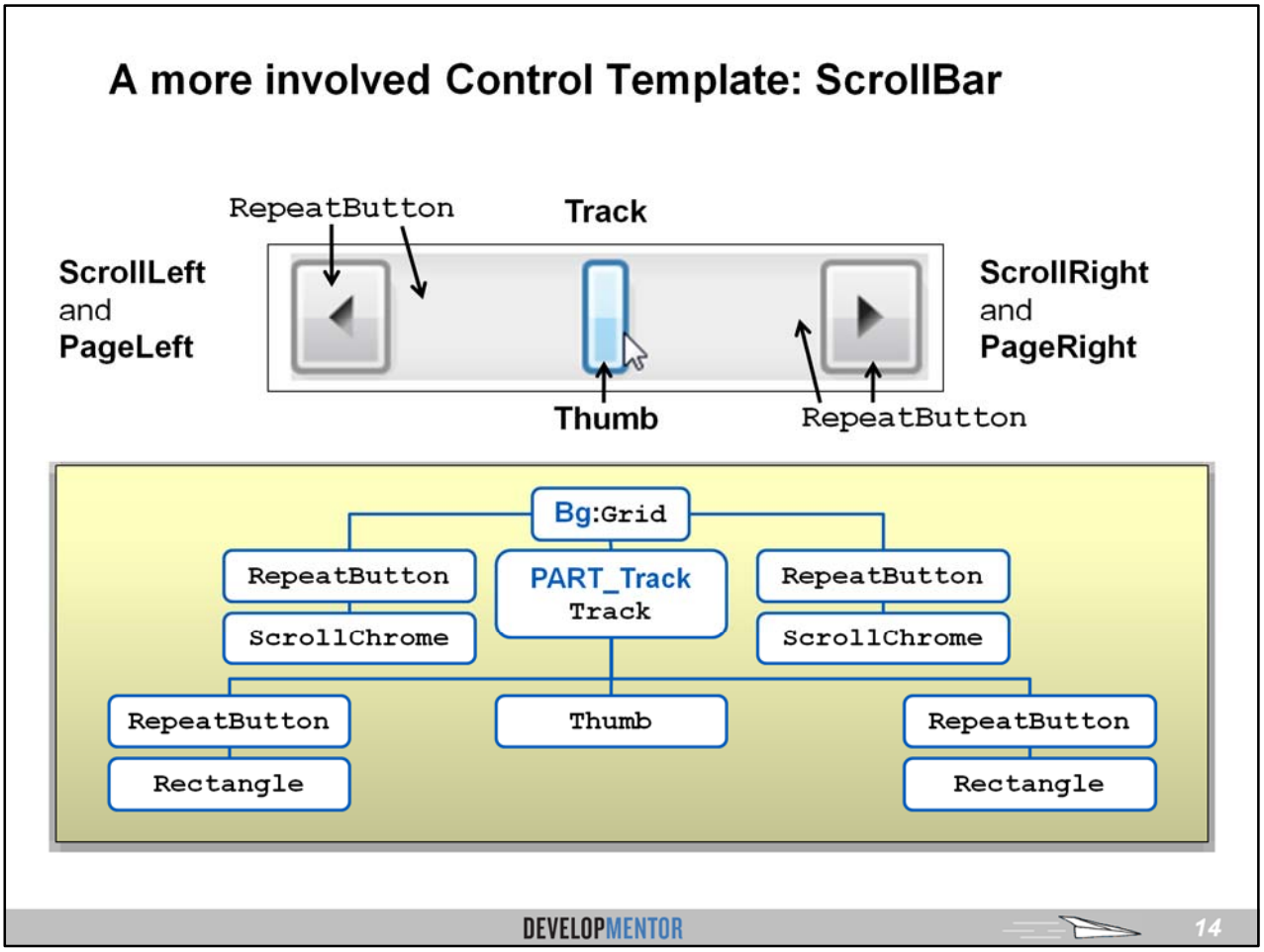
Every control is different – for example, the TabControl uses the name PART_SelectedContentHost for the content presenter used to display the active tab item. When templating a control, use Blend (which will give you a nice editable version of the template) or go look at the existing control template available from MSDN.

A Simple Control Template: ProgressBar





12: Control Templates - Beyond Buttons



Supporting commands in the templates

- **Template controls often use commands to drive behavior**
  - avoids requiring event handlers on template parts
  - code behind implements command handler vs. event handler

```
partial class ScrollBar
{
 public static readonly RoutedCommand LineUpCommand;
 public static readonly RoutedCommand LineDownCommand;
 public static readonly RoutedCommand LineLeftCommand;
 public static readonly RoutedCommand LineRightCommand;
 public static readonly RoutedCommand PageUpCommand;
 public static readonly RoutedCommand PageDownCommand;
 public static readonly RoutedCommand ScrollToHomeCommand;
 public static readonly RoutedCommand ScrollToEndCommand;
   ...
}
```

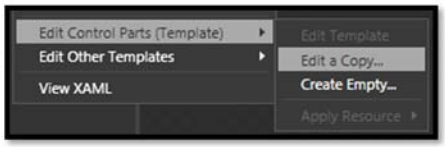
DEVELOPMENTOR 15

There are quite a few more defined on the ScrollBar and these commands are used for anything that has scrollbars on it – e.g. ScrollViewer, ListBox, etc.

Providing behavior using named elements

- **Some of the control behavior is provided in code behind**
  - ex: text entry in `TextBox`
  - wired up by locating template part by name
- **Some of the control behavior is provided through triggers**
  - ex: `RangeBase.Orientation` property
  - supplied as part of `ControlTemplate`
- **Only way to know is to look at existing template**
  - always start by examining existing `ControlTemplate`

Blend is ideal for this task ..
right click on the control you want
to template to get context menu



DEVELOPMENTOR 16

Summary

- **Every control in WPF can be templated**
  - many have multiple templates to edit
- **Use Blend to examine and alter existing templates**
  - easiest approach
- **Full control template source is included with SDK and Blend**
  - part of system themes
- **Also look at other sample and commercial themes**
  - "simple" theme in Blend
  - http://wpfthemes.codeplex.com/
  - http://www.nukeation.com/reuxables.aspx
  - http://www.xamltemplates.net/
  - also possible to port themes from Silverlight (e.g. JetPack)

DEVELOPMENTOR 17

