**[Difference between Control Template and DataTemplate in WPF](http://stackoverflow.com/questions/1340108/difference-between-control-template-and-datatemplate-in-wpf)**

Typically a control is rendered for its own sake, and doesn't reflect underlying data. For example, a Button wouldn't be bound to a business object - it's there purely so it can be clicked on. A ContentControl or ListBox, however, generally appear so that they can present data for the user.

***A DataTemplate, therefore, is used to provide visual structure for underlying data, while a ControlTemplate has nothing to do with underlying data and simply provides visual layout for the control itself***.

A ControlTemplate will generally only contain TemplateBinding expressions, binding back to the properties on the control itself, while a DataTemplate will contain standard Binding expressions, binding to the properties of its DataContext (the business/domain object or view model).

They're mostly the same, getting most of their behavior from the base FrameworkTemplate class.  The primary difference is that ControlTemplate has a TargetType property and DataTemplate has a DataType property.

Another difference is that the DataTemplateSelector can be used to choose a DataTemplate at runtime.  So, for example, the ColumnHeaderTemplate that you mentioned has a companion ColumnHeaderTemplateSelector property.

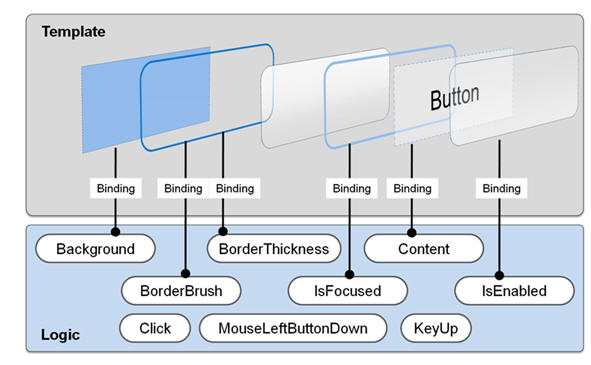
# Control Templates

## Introduction

Controls in WPF are separated into **logic**, that defines the states, events and properties and **template**, that defines the visual appearance of the control. The wireup between the logic and the template is done by DataBinding.

Each control has a default template. This gives the control a basic appearance. The default template is typically shipped together with the control and available for all common windows themes. It is by convention wrapped into a style, that is identified by value of the DefaultStyleKey property that every control has.

The template is defined by a dependency property called Template. By setting this property to another instance of a control template, you can completely replace the appearance (visual tree) of a control.

The control template is often included in a style that contains other property settings. The following code sample shows a simple control template for a button with an ellipse shape.

<Style x:Key="DialogButtonStyle" TargetType="Button">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Grid>

<Ellipse Fill="{TemplateBinding Background}"

Stroke="{TemplateBinding BorderBrush}"/>

<ContentPresenter HorizontalAlignment="Center"

VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

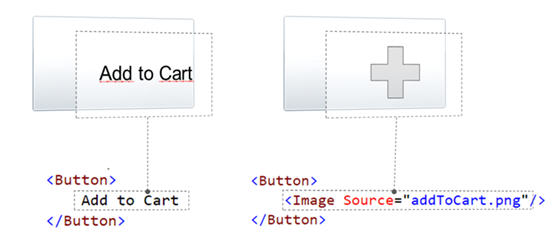
<Button Style="{StaticResource DialogButtonStyle}" />



*A Button without and with a custom control template*

## ContentPresenter

When you create a custom control template and you want to define a placeholder that renders the content, you can use theContentPresenter. By default it adds the content of the Content property to the visual tree of the template. To display the content of another property you can set the ContentSource to the name of the property you like.



## Triggers

### {RelativeSource TemplatedParent} not working in DataTriggers of a ControlTemplate

If you want to bind to a property of a property on your control like Data.IsLoaded you cannot use a normal Trigger, since it does not support this notation, you have to use a DataTrigger.

But when you are using a DataTrigger, with {RelativeSource TemplatedParent} it will not work. The reason is, that**TemplatedParent can only be used within the ControlTemplate**. It is not working in the Trigger section. You have to use the {RelativeSource Self} instead.

## What if a Binding working or a Setter is not applied when using a control template

There is something you need to know when setting a value of an element within a control template: The value does have a lower precendence as the local value! So if you are setting the local value in the constructor of the contained element, you cannot override it within the controltemplate. But if you use the element directly in your view, it will work. So be aware of this behavior!.

# ControlTemplate in WPF

The ControlTemplate contains the tree of elements that define the desired look. After you define a ControlTemplate you can attach it to any Control or Page by setting it's TemplateProperty.

In this example I am also showing you how to use Triggers with ControlTemplate. I am using Triggers on ButtonClick and MouseOver.

<Window x:Class="WpfApplication4.Window1"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    Title="Window1" Height="300" Width="300">

    <Grid>

        <Grid.Resources>

            <ControlTemplate x:Key="buttonTemplate">

                <Grid>

                    <Ellipse Width="160" Height="160" x:Name="outerCircle">

                        <Ellipse.Fill>

                            <LinearGradientBrush StartPoint="0,0" EndPoint="0,1">

                                <GradientStop Offset="0" Color="Green"></GradientStop>

                                <GradientStop Offset="1" Color="Purple"></GradientStop>

                            </LinearGradientBrush>

                        </Ellipse.Fill>

                    </Ellipse>

                    <Ellipse Width="120" Height="120">

                        <Ellipse.Fill>

                            <LinearGradientBrush StartPoint="0,0" EndPoint="0,1">

                                <GradientStop Offset="0" Color="Gray"></GradientStop>

                                <GradientStop Offset="1" Color="Blue"></GradientStop>

                            </LinearGradientBrush>

                        </Ellipse.Fill>

                    </Ellipse>

                </Grid>

                <ControlTemplate.Triggers>

                    <Trigger Property="Button.IsMouseOver" Value="True">

                        <Setter TargetName="outerCircle" Property="Fill" Value="Black"></Setter>

                    </Trigger>

                    <Trigger Property="Button.IsPressed" Value="True">

                        <Setter Property="RenderTransform">

                            <Setter.Value>

                                <ScaleTransform ScaleX=".8" ScaleY=".8"></ScaleTransform>

                            </Setter.Value>

                        </Setter>

                        <Setter Property="RenderTransformOrigin" Value=".6,.6"></Setter>

                    </Trigger>

                </ControlTemplate.Triggers>

            </ControlTemplate>

        </Grid.Resources>

        <Button Template="{StaticResource buttonTemplate}">Click Me</Button>

    </Grid>

</Window>

Result looks like this :

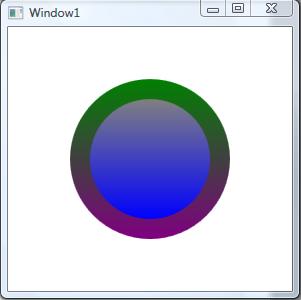


Figure 1.

After MouseOver:

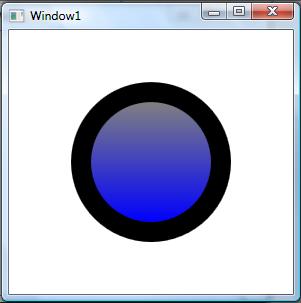
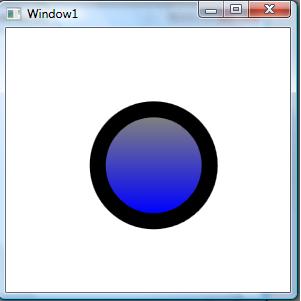


Figure 2.

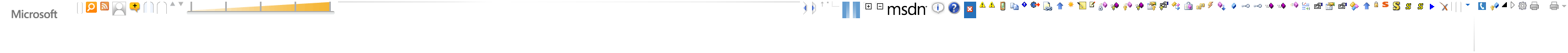
After KeyPressed:



# Data Templating Overview

**.NET Framework 4.5**

[Other Versions](javascript:;)



5 out of 7 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/ms742521.aspx#feedback)

The WPF data templating model provides you with great flexibility to define the presentation of your data. WPF controls have built-in functionality to support the customization of data presentation. This topic first demonstrates how to define a [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) and then introduces other data templating features, such as the selection of templates based on custom logic and the support for the display of hierarchical data.

This topic contains the following sections.

* [Prerequisites](http://msdn.microsoft.com/en-us/library/ms742521.aspx#Prerequisites)
* [Data Templating Basics](http://msdn.microsoft.com/en-us/library/ms742521.aspx#DataTemplating_Basic)
* [Adding More to the DataTemplate](http://msdn.microsoft.com/en-us/library/ms742521.aspx#adding_more_to_datatemplate)
* [Choosing a DataTemplate Based on Properties of the Data Object](http://msdn.microsoft.com/en-us/library/ms742521.aspx#Styling_StyleSelection)
* [Styling and Templating an ItemsControl](http://msdn.microsoft.com/en-us/library/ms742521.aspx#DataTemplating_ItemsControl)
* [Support for Hierarchical Data](http://msdn.microsoft.com/en-us/library/ms742521.aspx#DataTemplating_HeirarchicalDataTemplate)
* [Related Topics](http://msdn.microsoft.com/en-us/library/ms742521.aspx#seeAlsoToggle)

[**Prerequisites**](javascript:void(0))

This topic focuses on data templating features and is not an introduction of data binding concepts. For information about basic data binding concepts, see the [Data Binding Overview](http://msdn.microsoft.com/en-us/library/ms752347.aspx).

[DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) is about the presentation of data and is one of the many features provided by the WPF styling and templating model. For an introduction of the WPF styling and templating model, such as how to use a [Style](http://msdn.microsoft.com/en-us/library/system.windows.style.aspx) to set properties on controls, see the [Styling and Templating](http://msdn.microsoft.com/en-us/library/ms745683.aspx) topic.

In addition, it is important to understand **Resources**, which are essentially what enable objects such as [Style](http://msdn.microsoft.com/en-us/library/system.windows.style.aspx) and [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) to be reusable. For more information on resources, see [XAML Resources](http://msdn.microsoft.com/en-us/library/ms750613.aspx).

[**Data Templating Basics**](javascript:void(0))

This section contains the following subsections.

* [Without a DataTemplate](http://msdn.microsoft.com/en-us/library/ms742521.aspx#without_a_datatemplate)
* [Defining a Simple DataTemplate](http://msdn.microsoft.com/en-us/library/ms742521.aspx#defining_simple_datatemplate)
* [Creating the DataTemplate as a Resource](http://msdn.microsoft.com/en-us/library/ms742521.aspx#defining_datatemplate_as_a_resource)
* [The DataType Property](http://msdn.microsoft.com/en-us/library/ms742521.aspx#Styling_DataType)

To demonstrate why [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) is important, let's walk through a data binding example. In this example, we have a [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) that is bound to a list of Task objects. Each Task object has a TaskName (string), a Description (string), a Priority (int), and a property of type TaskType, which is an Enum with values Home and Work.

**XAML**

<Window x:Class="SDKSample.Window1"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:SDKSample"

Title="Introduction to Data Templating Sample">

<Window.Resources>

<local:Tasks x:Key="myTodoList"/>

...

</Window.Resources>

<StackPanel>

<TextBlock Name="blah" FontSize="20" Text="My Task List:"/>

<ListBox Width="400" Margin="10"

ItemsSource="{Binding Source={StaticResource myTodoList}}"/>

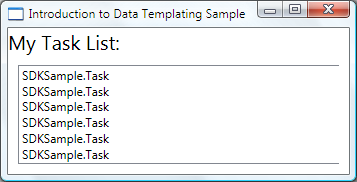
...

</StackPanel>

</Window>

[**Without a DataTemplate**](javascript:void(0))

Without a [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx), our [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) currently looks like this:



What's happening is that without any specific instructions, the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) by default calls **ToString** when trying to display the objects in the collection. Therefore, if the Task object overrides the **ToString** method, then the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) displays the string representation of each source object in the underlying collection.

For example, if the Task class overrides the **ToString** method this way, where name is the field for the TaskName property:

**C#**

[**VB**](javascript:%20CodeSnippet_SetLanguage('CodeSnippetContainerLang',%20'Programming',%20'Visual%20Basic');)

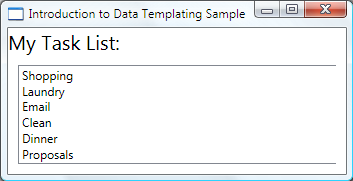
public override string ToString()

{

return name.ToString();

}

Then the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) looks like the following:



However, that is limiting and inflexible. Also, if you are binding to XML data, you wouldn't be able to override **ToString**.

[**Defining a Simple DataTemplate**](javascript:void(0))

The solution is to define a [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx). One way to do that is to set the [ItemTemplate](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.itemtemplate.aspx) property of the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) to a[DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx). What you specify in your [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) becomes the visual structure of your data object. The following[DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) is fairly simple. We are giving instructions that each item appears as three [TextBlock](http://msdn.microsoft.com/en-us/library/system.windows.controls.textblock.aspx) elements within a [StackPanel](http://msdn.microsoft.com/en-us/library/system.windows.controls.stackpanel.aspx). Each [TextBlock](http://msdn.microsoft.com/en-us/library/system.windows.controls.textblock.aspx) element is bound to a property of the Task class.

**XAML**

<ListBox Width="400" Margin="10"

ItemsSource="{Binding Source={StaticResource myTodoList}}">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel>

<TextBlock Text="{Binding Path=TaskName}" />

<TextBlock Text="{Binding Path=Description}"/>

<TextBlock Text="{Binding Path=Priority}"/>

</StackPanel>

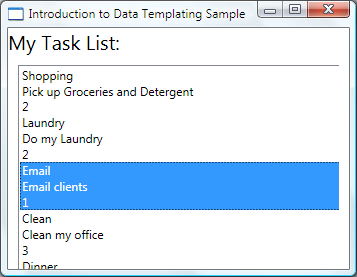
</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

The underlying data for the examples in this topic is a collection of CLR objects. If you are binding to XML data, the fundamental concepts are the same, but there is a slight syntactic difference. For example, instead of having Path=TaskName, you would set [XPath](http://msdn.microsoft.com/en-us/library/system.windows.data.binding.xpath.aspx) to @TaskName (if TaskName is an attribute of your XML node).

Now our [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) looks like the following:



[**Creating the DataTemplate as a Resource**](javascript:void(0))

In the above example, we defined the [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) inline. It is more common to define it in the resources section so it can be a reusable object, as in the following example:

**XAML**

<Window.Resources>

...

<DataTemplate x:Key="myTaskTemplate">

<StackPanel>

<TextBlock Text="{Binding Path=TaskName}" />

<TextBlock Text="{Binding Path=Description}"/>

<TextBlock Text="{Binding Path=Priority}"/>

</StackPanel>

</DataTemplate>

...

</Window.Resources>

Now you can use myTaskTemplate as a resource, as in the following example:

**XAML**

<ListBox Width="400" Margin="10"

ItemsSource="{Binding Source={StaticResource myTodoList}}"

ItemTemplate="{StaticResource myTaskTemplate}"/>

Because myTaskTemplate is a resource, you can now use it on other controls that have a property that takes a[DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) type. As shown above, for [ItemsControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.aspx) objects, such as the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx), it is the [ItemTemplate](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.itemtemplate.aspx) property. For[ContentControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.contentcontrol.aspx) objects, it is the [ContentTemplate](http://msdn.microsoft.com/en-us/library/system.windows.controls.contentcontrol.contenttemplate.aspx) property.

[**The DataType Property**](javascript:void(0))

The [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) class has a [DataType](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.datatype.aspx) property that is very similar to the [TargetType](http://msdn.microsoft.com/en-us/library/system.windows.style.targettype.aspx) property of the [Style](http://msdn.microsoft.com/en-us/library/system.windows.style.aspx) class. Therefore, instead of specifying an **x:Key** for the [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) in the above example, you can do the following:

**XAML**

<DataTemplate DataType="{x:Type local:Task}">

<StackPanel>

<TextBlock Text="{Binding Path=TaskName}" />

<TextBlock Text="{Binding Path=Description}"/>

<TextBlock Text="{Binding Path=Priority}"/>

</StackPanel>

</DataTemplate>

This [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) gets applied automatically to all Task objects. Note that in this case the **x:Key** is set implicitly. Therefore, if you assign this [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) an **x:Key** value, you are overriding the implicit **x:Key** and the [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) would not be applied automatically.

If you are binding a [ContentControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.contentcontrol.aspx) to a collection of Task objects, the [ContentControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.contentcontrol.aspx) does not use the above [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx)automatically. This is because the binding on a [ContentControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.contentcontrol.aspx) needs more information to distinguish whether you want to bind to an entire collection or the individual objects. If your [ContentControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.contentcontrol.aspx) is tracking the selection of an [ItemsControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.aspx) type, you can set the [Path](http://msdn.microsoft.com/en-us/library/system.windows.data.binding.path.aspx) property of the [ContentControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.contentcontrol.aspx) binding to "/" to indicate that you are interested in the current item. For an example, see [How to: Bind to a Collection and Display Information Based on Selection](http://msdn.microsoft.com/en-us/library/aa970558.aspx). Otherwise, you need to specify the[DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) explicitly by setting the [ContentTemplate](http://msdn.microsoft.com/en-us/library/system.windows.controls.contentcontrol.contenttemplate.aspx) property.

The [DataType](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.datatype.aspx) property is particularly useful when you have a [CompositeCollection](http://msdn.microsoft.com/en-us/library/system.windows.data.compositecollection.aspx) of different types of data objects. For an example, see [How to: Implement a CompositeCollection](http://msdn.microsoft.com/en-us/library/ms742405.aspx).

[**Adding More to the DataTemplate**](javascript:void(0))

Currently the data appears with the necessary information, but there's definitely room for improvement. Let's improve on the presentation by adding a [Border](http://msdn.microsoft.com/en-us/library/system.windows.controls.border.aspx), a [Grid](http://msdn.microsoft.com/en-us/library/system.windows.controls.grid.aspx), and some [TextBlock](http://msdn.microsoft.com/en-us/library/system.windows.controls.textblock.aspx) elements that describe the data that is being displayed.

**XAML**

<DataTemplate x:Key="myTaskTemplate">

<Border Name="border" BorderBrush="Aqua" BorderThickness="1"

Padding="5" Margin="5">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0" Text="Task Name:"/>

<TextBlock Grid.Row="0" Grid.Column="1" Text="{Binding Path=TaskName}" />

<TextBlock Grid.Row="1" Grid.Column="0" Text="Description:"/>

<TextBlock Grid.Row="1" Grid.Column="1" Text="{Binding Path=Description}"/>

<TextBlock Grid.Row="2" Grid.Column="0" Text="Priority:"/>

<TextBlock Grid.Row="2" Grid.Column="1" Text="{Binding Path=Priority}"/>

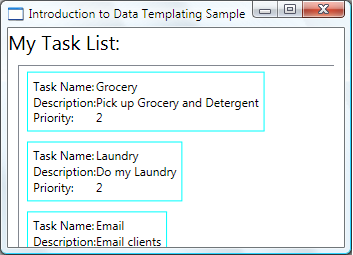
</Grid>

</Border>

...

</DataTemplate>

The following screenshot shows the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) with this modified [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx):



We can set [HorizontalContentAlignment](http://msdn.microsoft.com/en-us/library/system.windows.controls.control.horizontalcontentalignment.aspx) to [Stretch](http://msdn.microsoft.com/en-us/library/system.windows.horizontalalignment.aspx) on the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) to make sure the width of the items takes up the entire space:

**XAML**

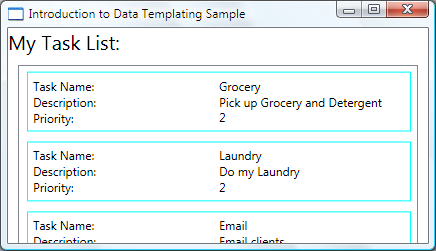
<ListBox Width="400" Margin="10"

ItemsSource="{Binding Source={StaticResource myTodoList}}"

ItemTemplate="{StaticResource myTaskTemplate}"

HorizontalContentAlignment="Stretch"/>

With the [HorizontalContentAlignment](http://msdn.microsoft.com/en-us/library/system.windows.controls.control.horizontalcontentalignment.aspx) property set to [Stretch](http://msdn.microsoft.com/en-us/library/system.windows.horizontalalignment.aspx), the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) now looks like this:



[**Use DataTriggers to Apply Property Values**](javascript:void(0))

The current presentation does not tell us whether a Task is a home task or an office task. Remember that the Task object has a TaskType property of type TaskType, which is an enumeration with values Home and Work.

In the following example, the [DataTrigger](http://msdn.microsoft.com/en-us/library/system.windows.datatrigger.aspx) sets the [BorderBrush](http://msdn.microsoft.com/en-us/library/system.windows.controls.border.borderbrush.aspx) of the element named border to **Yellow** if the TaskTypeproperty is TaskType.Home.

**XAML**

<DataTemplate x:Key="myTaskTemplate">

...

<DataTemplate.Triggers>

<DataTrigger Binding="{Binding Path=TaskType}">

<DataTrigger.Value>

<local:TaskType>Home</local:TaskType>

</DataTrigger.Value>

<Setter TargetName="border" Property="BorderBrush" Value="Yellow"/>

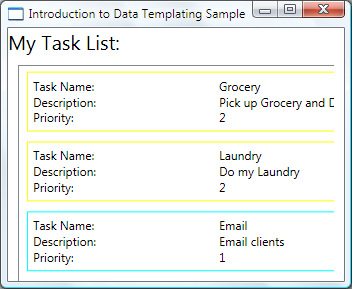
</DataTrigger>

</DataTemplate.Triggers>

...

</DataTemplate>

Our application now looks like the following. Home tasks appear with a yellow border and office tasks appear with an aqua border:



In this example the [DataTrigger](http://msdn.microsoft.com/en-us/library/system.windows.datatrigger.aspx) uses a [Setter](http://msdn.microsoft.com/en-us/library/system.windows.setter.aspx) to set a property value. The trigger classes also have the [EnterActions](http://msdn.microsoft.com/en-us/library/system.windows.triggerbase.enteractions.aspx) and[ExitActions](http://msdn.microsoft.com/en-us/library/system.windows.triggerbase.exitactions.aspx) properties that allow you to start a set of actions such as animations. In addition, there is also a [MultiDataTrigger](http://msdn.microsoft.com/en-us/library/system.windows.multidatatrigger.aspx)class that allows you to apply changes based on multiple data-bound property values.

An alternative way to achieve the same effect is to bind the [BorderBrush](http://msdn.microsoft.com/en-us/library/system.windows.controls.border.borderbrush.aspx) property to the TaskType property and use a value converter to return the color based on the TaskType value. Creating the above effect using a converter is slightly more efficient in terms of performance. Additionally, creating your own converter gives you more flexibility because you are supplying your own logic. Ultimately, which technique you choose depends on your scenario and your preference. For information about how to write a converter, see [IValueConverter](http://msdn.microsoft.com/en-us/library/system.windows.data.ivalueconverter.aspx).

[**What Belongs in a DataTemplate?**](javascript:void(0))

In the previous example, we placed the trigger within the [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) using the [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx).[Triggers](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.triggers.aspx) property. The [Setter](http://msdn.microsoft.com/en-us/library/system.windows.setter.aspx) of the trigger sets the value of a property of an element (the [Border](http://msdn.microsoft.com/en-us/library/system.windows.controls.border.aspx) element) that is within the [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx). However, if the properties that your **Setters** are concerned with are not properties of elements that are within the current [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx), it may be more suitable to set the properties using a [Style](http://msdn.microsoft.com/en-us/library/system.windows.style.aspx) that is for the [ListBoxItem](http://msdn.microsoft.com/en-us/library/system.windows.controls.listboxitem.aspx) class (if the control you are binding is a [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx)). For example, if you want your [Trigger](http://msdn.microsoft.com/en-us/library/system.windows.trigger.aspx) to animate the [Opacity](http://msdn.microsoft.com/en-us/library/system.windows.uielement.opacity.aspx) value of the item when a mouse points to an item, you define triggers within a [ListBoxItem](http://msdn.microsoft.com/en-us/library/system.windows.controls.listboxitem.aspx) style. For an example, see the [Introduction to Styling and Templating Sample](http://go.microsoft.com/fwlink/?LinkID=160010).

In general, keep in mind that the [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) is being applied to each of the generated [ListBoxItem](http://msdn.microsoft.com/en-us/library/system.windows.controls.listboxitem.aspx) (for more information about how and where it is actually applied, see the [ItemTemplate](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.itemtemplate.aspx) page.). Your [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) is concerned with only the presentation and appearance of the data objects. In most cases, all other aspects of presentation, such as what an item looks like when it is selected or how the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) lays out the items, do not belong in the definition of a [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx). For an example, see the [Styling and Templating an ItemsControl](http://msdn.microsoft.com/en-us/library/ms742521.aspx#DataTemplating_ItemsControl) section.

[**Choosing a DataTemplate Based on Properties of the Data Object**](javascript:void(0))

In [The DataType Property](http://msdn.microsoft.com/en-us/library/ms742521.aspx#Styling_DataType) section, we discussed that you can define different data templates for different data objects. That is especially useful when you have a [CompositeCollection](http://msdn.microsoft.com/en-us/library/system.windows.data.compositecollection.aspx) of different types or collections with items of different types. In the [Use DataTriggers to Apply Property Values](http://msdn.microsoft.com/en-us/library/ms742521.aspx#DataTrigger_to_Apply_Property_Values) section, we have shown that if you have a collection of the same type of data objects you can create a [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) and then use triggers to apply changes based on the property values of each data object. However, triggers allow you to apply property values or start animations but they don't give you the flexibility to reconstruct the structure of your data objects. Some scenarios may require you to create a different [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) for data objects that are of the same type but have different properties.

For example, when a Task object has a Priority value of 1, you may want to give it a completely different look to serve as an alert for yourself. In that case, you create a [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) for the display of the high-priority Task objects. Let's add the following[DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) to the resources section:

**XAML**

<DataTemplate x:Key="importantTaskTemplate">

<DataTemplate.Resources>

<Style TargetType="TextBlock">

<Setter Property="FontSize" Value="20"/>

</Style>

</DataTemplate.Resources>

<Border Name="border" BorderBrush="Red" BorderThickness="1"

Padding="5" Margin="5">

<DockPanel HorizontalAlignment="Center">

<TextBlock Text="{Binding Path=Description}" />

<TextBlock>!</TextBlock>

</DockPanel>

</Border>

</DataTemplate>

Notice this example uses the [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx).[Resources](http://msdn.microsoft.com/en-us/library/system.windows.frameworktemplate.resources.aspx) property. Resources defined in that section are shared by the elements within the [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx).

To supply logic to choose which [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) to use based on the Priority value of the data object, create a subclass of[DataTemplateSelector](http://msdn.microsoft.com/en-us/library/system.windows.controls.datatemplateselector.aspx) and override the [SelectTemplate](http://msdn.microsoft.com/en-us/library/system.windows.controls.datatemplateselector.selecttemplate.aspx) method. In the following example, the [SelectTemplate](http://msdn.microsoft.com/en-us/library/system.windows.controls.datatemplateselector.selecttemplate.aspx) method provides logic to return the appropriate template based on the value of the Priority property. The template to return is found in the resources of the enveloping [Window](http://msdn.microsoft.com/en-us/library/system.windows.window.aspx) element.

**C#**

[**VB**](javascript:%20CodeSnippet_SetLanguage('CodeSnippetContainerLang',%20'Programming',%20'Visual%20Basic');)

using System.Windows;

using System.Windows.Controls;

namespace SDKSample

{

public class TaskListDataTemplateSelector : DataTemplateSelector

{

public override DataTemplate

SelectTemplate(object item, DependencyObject container)

{

FrameworkElement element = container as FrameworkElement;

if (element != null && item != null && item is Task)

{

Task taskitem = item as Task;

if (taskitem.Priority == 1)

return

element.FindResource("importantTaskTemplate") as DataTemplate;

else

return

element.FindResource("myTaskTemplate") as DataTemplate;

}

return null;

}

}

}

We can then declare the TaskListDataTemplateSelector as a resource:

**XAML**

<Window.Resources>

...

<local:TaskListDataTemplateSelector x:Key="myDataTemplateSelector"/>

...

</Window.Resources>

To use the template selector resource, assign it to the [ItemTemplateSelector](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.itemtemplateselector.aspx) property of the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx). The [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) calls the[SelectTemplate](http://msdn.microsoft.com/en-us/library/system.windows.controls.datatemplateselector.selecttemplate.aspx) method of the TaskListDataTemplateSelector for each of the items in the underlying collection. The call passes the data object as the item parameter. The [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) that is returned by the method is then applied to that data object.

**XAML**

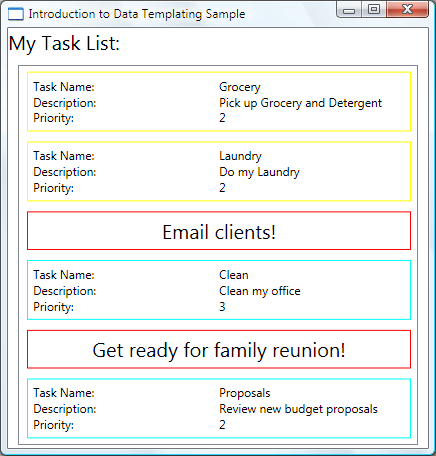
<ListBox Width="400" Margin="10"

ItemsSource="{Binding Source={StaticResource myTodoList}}"

ItemTemplateSelector="{StaticResource myDataTemplateSelector}"

HorizontalContentAlignment="Stretch"/>

With the template selector in place, the [ListBox](http://msdn.microsoft.com/en-us/library/system.windows.controls.listbox.aspx) now appears as follows:



This concludes our discussion of this example. For the complete sample, see [Introduction to Data Templating Sample](http://go.microsoft.com/fwlink/?LinkID=160009).

[**Styling and Templating an ItemsControl**](javascript:void(0))

Even though the [ItemsControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.aspx) is not the only control type that you can use a [DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) with, it is a very common scenario to bind an [ItemsControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.aspx) to a collection. In the [What Belongs in a DataTemplate](http://msdn.microsoft.com/en-us/library/ms742521.aspx#what_belongs_in_datatemplate) section we discussed that the definition of your[DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) should only be concerned with the presentation of data. In order to know when it is not suitable to use a[DataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.datatemplate.aspx) it is important to understand the different style and template properties provided by the [ItemsControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.aspx). The following example is designed to illustrate the function of each of these properties. The [ItemsControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.aspx) in this example is bound to the same Tasks collection as in the previous example. For demonstration purposes, the styles and templates in this example are all declared inline.

**XAML**

<ItemsControl Margin="10"

ItemsSource="{Binding Source={StaticResource myTodoList}}">

<!--The ItemsControl has no default visual appearance.

Use the Template property to specify a ControlTemplate to define

the appearance of an ItemsControl. The ItemsPresenter uses the specified

ItemsPanelTemplate (see below) to layout the items. If an

ItemsPanelTemplate is not specified, the default is used. (For ItemsControl,

the default is an ItemsPanelTemplate that specifies a StackPanel.-->

<ItemsControl.Template>

<ControlTemplate TargetType="ItemsControl">

<Border BorderBrush="Aqua" BorderThickness="1" CornerRadius="15">

<ItemsPresenter/>

</Border>

</ControlTemplate>

</ItemsControl.Template>

<!--Use the ItemsPanel property to specify an ItemsPanelTemplate

that defines the panel that is used to hold the generated items.

In other words, use this property if you want to affect

how the items are laid out.-->

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel />

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<!--Use the ItemTemplate to set a DataTemplate to define

the visualization of the data objects. This DataTemplate

specifies that each data object appears with the Proriity

and TaskName on top of a silver ellipse.-->

<ItemsControl.ItemTemplate>

<DataTemplate>

<DataTemplate.Resources>

<Style TargetType="TextBlock">

<Setter Property="FontSize" Value="18"/>

<Setter Property="HorizontalAlignment" Value="Center"/>

</Style>

</DataTemplate.Resources>

<Grid>

<Ellipse Fill="Silver"/>

<StackPanel>

<TextBlock Margin="3,3,3,0"

Text="{Binding Path=Priority}"/>

<TextBlock Margin="3,0,3,7"

Text="{Binding Path=TaskName}"/>

</StackPanel>

</Grid>

</DataTemplate>

</ItemsControl.ItemTemplate>

<!--Use the ItemContainerStyle property to specify the appearance

of the element that contains the data. This ItemContainerStyle

gives each item container a margin and a width. There is also

a trigger that sets a tooltip that shows the description of

the data object when the mouse hovers over the item container.-->

<ItemsControl.ItemContainerStyle>

<Style>

<Setter Property="Control.Width" Value="100"/>

<Setter Property="Control.Margin" Value="5"/>

<Style.Triggers>

<Trigger Property="Control.IsMouseOver" Value="True">

<Setter Property="Control.ToolTip"

Value="{Binding RelativeSource={x:Static RelativeSource.Self},

Path=Content.Description}"/>

</Trigger>

</Style.Triggers>

</Style>

</ItemsControl.ItemContainerStyle>

</ItemsControl>

The following is a screenshot of the example when it is rendered:



Note that instead of using the [ItemTemplate](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.itemtemplate.aspx), you can use the [ItemTemplateSelector](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.itemtemplateselector.aspx). Refer to the previous section for an example. Similarly, instead of using the [ItemContainerStyle](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.itemcontainerstyle.aspx), you have the option to use the [ItemContainerStyleSelector](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.itemcontainerstyleselector.aspx).

Two other style-related properties of the [ItemsControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.aspx) that are not shown here are [GroupStyle](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.groupstyle.aspx) and [GroupStyleSelector](http://msdn.microsoft.com/en-us/library/system.windows.controls.itemscontrol.groupstyleselector.aspx).

[**Support for Hierarchical Data**](javascript:void(0))

So far we have only looked at how to bind to and display a single collection. Sometimes you have a collection that contains other collections. The [HierarchicalDataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.hierarchicaldatatemplate.aspx) class is designed to be used with [HeaderedItemsControl](http://msdn.microsoft.com/en-us/library/system.windows.controls.headereditemscontrol.aspx) types to display such data. In the following example, ListLeagueList is a list of League objects. Each League object has a Name and a collection ofDivision objects. Each Division has a Name and a collection of Team objects, and each Team object has a Name.

**XAML**

<Window x:Class="SDKSample.Window1"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="HierarchicalDataTemplate Sample"

xmlns:src="clr-namespace:SDKSample">

<DockPanel>

<DockPanel.Resources>

<src:ListLeagueList x:Key="MyList"/>

<HierarchicalDataTemplate DataType = "{x:Type src:League}"

ItemsSource = "{Binding Path=Divisions}">

<TextBlock Text="{Binding Path=Name}"/>

</HierarchicalDataTemplate>

<HierarchicalDataTemplate DataType = "{x:Type src:Division}"

ItemsSource = "{Binding Path=Teams}">

<TextBlock Text="{Binding Path=Name}"/>

</HierarchicalDataTemplate>

<DataTemplate DataType="{x:Type src:Team}">

<TextBlock Text="{Binding Path=Name}"/>

</DataTemplate>

</DockPanel.Resources>

<Menu Name="menu1" DockPanel.Dock="Top" Margin="10,10,10,10">

<MenuItem Header="My Soccer Leagues"

ItemsSource="{Binding Source={StaticResource MyList}}" />

</Menu>

<TreeView>

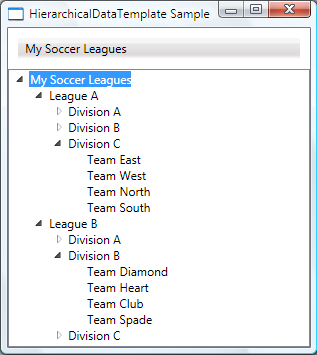
<TreeViewItem ItemsSource="{Binding Source={StaticResource MyList}}" Header="My Soccer Leagues" />

</TreeView>

</DockPanel>

</Window>

The example shows that with the use of [HierarchicalDataTemplate](http://msdn.microsoft.com/en-us/library/system.windows.hierarchicaldatatemplate.aspx), you can easily display list data that contains other lists. The following is a screenshot of the example.



# Data Templates

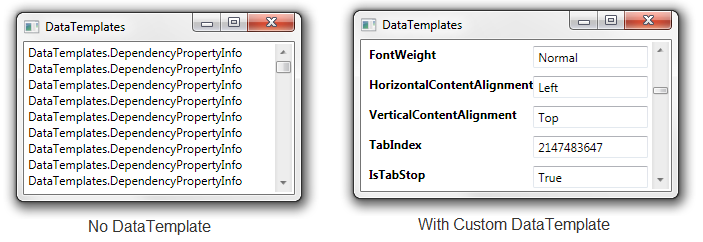
## Introduction

Data Template are a similar concept as [Control Templates](http://wpftutorial.net/templates.html). They give you a very flexible and powerful solution to **replace the visual appearance of a data item** in a control like ListBox, ComboBox or ListView. In my opinion this is one of the key success factory of WPF.

If you don't specify a data template, WPF takes the default template that is just a TextBlock. If you bind complex objects to the control, it just calls ToString() on it. Within a DataTemplate, the DataContext is set the data object. So you can easily bind against the data context to display various members of your data object

## DataTemplates in Action: Building a simple PropertyGrid

Whereas it was really hard to display complex data in a ListBox with WinForms, its super easy with WPF. The following example shows a ListBox with a list of DependencyPropertyInfo instances bound to it. Without a DataTemplate you just see the result of calling ToString() on the object. With the data template we see the name of the property and a TextBox that even allows us to edit the value.

*<!-- Without DataTemplate -->*

<ListBox ItemsSource="{Binding}" />

*<!-- With DataTemplate -->*

<ListBox ItemsSource="{Binding}" BorderBrush="Transparent"

Grid.IsSharedSizeScope="True"

HorizontalContentAlignment="Stretch">

<ListBox.ItemTemplate>

<DataTemplate>

<Grid Margin="4">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" SharedSizeGroup="Key" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<TextBlock Text="{Binding Name}" FontWeight="Bold" />

<TextBox Grid.Column="1" Text="{Binding Value }" />

</Grid>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

## How to use a DataTemplateSelector to switch the Template depending on the data

Our property grid looks nice so far, but it would be much more usable if we could switch the editor depending on the type of the property.

The simplest way to do this is to use a DataTemplateSelector. The DataTemplateSelector has a single method to override: SelectTemplate(object item, DependencyObject container). In this method we decide on the provided item which DataTemplate to choose.

The following exmple shows an DataTemplateSelector that decides between tree data templates:

**public** **class** PropertyDataTemplateSelector : DataTemplateSelector

{

**public** DataTemplate DefaultnDataTemplate { get; set; }

**public** DataTemplate BooleanDataTemplate { get; set; }

**public** DataTemplate EnumDataTemplate { get; set; }

**public** **override** DataTemplate SelectTemplate(**object** item,

DependencyObject container)

{

DependencyPropertyInfo dpi = item **as** DependencyPropertyInfo;

**if** (dpi.PropertyType == [**typeof**](http://www.google.com/search?q=typeof+msdn.microsoft.com)(**bool**))

{

**return** BooleanDataTemplate;

}

**if** (dpi.PropertyType.IsEnum)

{

**return** EnumDataTemplate;

}

**return** DefaultnDataTemplate;

}

}

<Window x:Class="DataTemplates.Window1"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:l="clr-namespace:DataTemplates"

xmlns:sys="clr-namespace:System;assembly=mscorlib">

<Window.Resources>

*<!-- Default DataTemplate -->*

<DataTemplate x:Key="DefaultDataTemplate">

...

</DataTemplate>

*<!-- DataTemplate for Booleans -->*

<DataTemplate x:Key="BooleanDataTemplate">

...

</DataTemplate>

*<!-- DataTemplate for Enums -->*

<DataTemplate x:Key="EnumDataTemplate">

...

</DataTemplate>

*<!-- DataTemplate Selector -->*

<l:PropertyDataTemplateSelector x:Key="templateSelector"

DefaultnDataTemplate="{StaticResource DefaultDataTemplate}"

BooleanDataTemplate="{StaticResource BooleanDataTemplate}"

EnumDataTemplate="{StaticResource EnumDataTemplate}"/>

</Window.Resources>

<Grid>

<ListBox ItemsSource="{Binding}" Grid.IsSharedSizeScope="True"

HorizontalContentAlignment="Stretch"

ItemTemplateSelector="{StaticResource templateSelector}"/>

</Grid>

</Window>

## How to react to IsSelected in the DataTemplate

If you want to change the appearance of a ListBoxItem when it is selected, you have to bind the IsSelected property of the ListBoxItem. But this is a bit tricky, you have to use a relative source with FindAcestor to navigate up the visual tree until you reach the ListBoxItem.

<DataTemplate x:Key="DefaultDataTemplate">

<Border x:Name="border" Height="50">

...

</Border>

<DataTemplate.Triggers>

<DataTrigger Binding="{Binding RelativeSource=

{RelativeSource Mode=FindAncestor, AncestorType=

{x:Type ListBoxItem}},Path=IsSelected}" Value="True">

<Setter TargetName="border" Property="Height" Value="100"/>

</DataTrigger>

</DataTemplate.Triggers>

</DataTemplate>

**HierarchicalDataTemplate**

<http://complexdatatemplates.codeplex.com/>

http://blogs.msdn.com/b/mikehillberg/archive/2009/10/30/treeview-and-hierarchicaldatatemplate-step-by-step.aspx