<http://msdn.microsoft.com/en-us/library/ms752347.aspx>

If correct in Validate function :

return ValidationResult.ValidResult;

Show validation error message in a textblock

<http://stackoverflow.com/questions/3362563/wpf-display-textblock-with-validation-error-message-below-control>

<TextBlock Text="{Binding ErrorContent}" DockPanel.Dock="Bottom" Foreground="Red"/>

Try alternate syntax for tooltip

<http://stackoverflow.com/questions/2260616/why-does-wpf-style-to-show-validation-errors-in-tooltip-work-for-a-textbox-but-f>

Replace Path=(Validation.Errors)[0].ErrorContent

By Path=(Validation.Errors).CurrentItem.ErrorContent

Generic template

<ControlTemplate x:Key="ValidationErrorTemplate" TargetType="Control">  
    <StackPanel Orientation="Horizontal">  
        <TextBlock Foreground="Red" FontSize="24" Text="\*"   
                   ToolTip="{Binding .CurrentItem}">  
        </TextBlock>  
        <AdornedElementPlaceholder>  
        </AdornedElementPlaceholder>  
    </StackPanel>  
</ControlTemplate>

Validation at form level

<http://msdn.microsoft.com/en-us/library/aa969773.aspx>

WPF does not restrict a user to the invalid control until they have entered valid data. This is good behavior for a dialog box; a user should be able to freely navigate the controls in a dialog box whether or not data is valid. However, this means a user can enter invalid data and press the **OK** button. For this reason, your code also needs to validate all controls in a dialog box when the **OK** button is pressed by handling the [Click](http://msdn.microsoft.com/en-us/library/system.windows.controls.primitives.buttonbase.click.aspx) event.

using System.Windows; // Window, RoutedEventArgs, IInputElement, DependencyObject

using System.Windows.Controls; // Validation

using System.Windows.Input; // Keyboard

namespace SDKSample

{

public partial class MarginsDialogBox : Window

{

...

void okButton\_Click(object sender, RoutedEventArgs e)

{

// Don't accept the dialog box if there is invalid data

if (!IsValid(this)) return;

...

}

// Validate all dependency objects in a window

bool IsValid(DependencyObject node)

{

// Check if dependency object was passed

if (node != null)

{

// Check if dependency object is valid.

// NOTE: Validation.GetHasError works for controls that have validation rules attached

bool isValid = !Validation.GetHasError(node);

if (!isValid)

{

// If the dependency object is invalid, and it can receive the focus,

// set the focus

if (node is IInputElement) Keyboard.Focus((IInputElement)node);

return false;

}

}

// If this dependency object is valid, check all child dependency objects

foreach (object subnode in LogicalTreeHelper.GetChildren(node))

{

if (subnode is DependencyObject)

{

// If a child dependency object is invalid, return false immediately,

// otherwise keep checking

if (IsValid((DependencyObject)subnode) == false) return false;

}

}

// All dependency objects are valid

return true;

}

}

}