Zeus UpDownControls Ver. 1.0

Manual

Πίνακας περιεχομένων

Zeus UpDownControls version 1.0

Κλάσεις: ByteUpDown, ShortUpDown, IntegerUpDown, LongUpDown, SingleUpDown, DoubleUpDown, DecimalUpDown, DatePickerUpDown

Βασική κλάση: UpDownBase

Inherits: System.Windows.Controls.Control

Namespace: Zeus.WPF.Controls.UpDownControls

Assembly: ZeusUpDownControls (in ZeusUpDownControls.dll)

Dependencies: -

Περιγραφή

Η βιβλιοθήκη **Zeus UpDownControls** περιλαμβάνει ένα σύνολο από **εξειδεικευμένα UpDown controls**, κυρίως για είσοδο αριθμητικών δεδομένων συγκεριμένου τύπου (π.χ. Byte, Short, Integer κλπ) αλλά και ημερομηνίας (DateTime). Ο έλεγχος των εισαγόμενων δεδομένων πραγματοποιείται μέσω **εξειδικευμένων ValidationRules** που έχουν ενσωματωθεί κατά περίπτωση. Αποτελεί μία πολύτιμη προσθήκη στην εργαλειοθήκη του WPF όπου, παραδόξως, δεν περιλαμβάνει η στάνταρ βιβλιοθήκη.

Παρακάτω, παρατίθενται τα **UpDownControls** ανά τύπο δεδομένων.

Για **Αριθμητικά δεδομένα**:

	ByteUpDown	Επιτρέπει την είσοδο και αυξομείωση μόνο αριθμητικών δεδομένων, τύπου **Byte**. Η τιμή απαιτείται (required).
	ShortUpDown	Επιτρέπει την είσοδο και αυξομείωση μόνο αριθμητικών δεδομένων, τύπου **Short**. Η τιμή απαιτείται (required).
	IntegerUpDown	Επιτρέπει την είσοδο και αυξομείωση μόνο αριθμητικών δεδομένων, τύπου **Integer**. Η τιμή απαιτείται (required).

	LongUpDown	Επιτρέπει την είσοδο και αυξομείωση μόνο αριθμητικών δεδομένων, τύπου **Long**. Η τιμή απαιτείται (required).
	SingleUpDown	Επιτρέπει την είσοδο και αυξομείωση μόνο αριθμητικών δεδομένων, τύπου **Single**. Η τιμή απαιτείται (required).
	DoubleUpDown	Επιτρέπει την είσοδο και αυξομείωση μόνο αριθμητικών δεδομένων, τύπου **Double**. Η τιμή απαιτείται (required).
	DecimalUpDown	Επιτρέπει την είσοδο και αυξομείωση μόνο αριθμητικών δεδομένων, τύπου **Decimal**. Η τιμή απαιτείται (required).

Για **Ημερομηνιακά δεδομένα**:

	DatePickerUpDown	Επιτρέπει την είσοδο και αυξομείωση μόνο ημερομηνιακών δεδομένων, τύπου **Date** ή **DateTime**. Η τιμή δεν απαιτείται (not required).

Οι ορισμοί των κλάσεων

Οι κλάσεις έχουν οριστεί ως εξής:

- Για την **βασική κλάση** **UpDownBase**:

  Σύνταξη:

  VB:

```
Public Class UpDownBase
             Inherits Control
```

- Για την **κλάση** **ByteUpDown**:

  Σύνταξη:

  VB:

```
Public Class ByteUpDown
             Inherits UpDownBase
```

  XAML Object Element Usage:

  Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

  Χρήση:

```
<zeus:ByteUpDown ... />
```

- Για την **κλάση** **ShortUpDown**:

  Σύνταξη:

  VB:

```
Public Class ShortUpDown
             Inherits UpDownBase
```

  XAML Object Element Usage:

Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

Χρήση:

```
<zeus:ShortUpDown ... />
```

- Για την **κλάση** **IntegerUpDown**:

  Σύνταξη:

  VB:

```
Public Class IntegerUpDown
             Inherits UpDownBase
```

  XAML Object Element Usage:

  Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

  Χρήση:

```
<zeus:IntegerUpDown ... />
```

- Για την **κλάση** **LongUpDown**:

  Σύνταξη:

  VB:

```
Public Class LongUpDown
             Inherits UpDownBase
```

  XAML Object Element Usage:

  Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

Χρήση:

```
<zeus:LongUpDown ... />
```

- Για την **κλάση SingleUpDown**:

  Σύνταξη:

  VB:

```
Public Class SingleUpDown
             Inherits UpDownBase
```

  XAML Object Element Usage:

  Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

  Χρήση:

```
<zeus:SingleUpDown ... />
```

- Για την **κλάση DoubleUpDown**:

  Σύνταξη:

  VB:

```
Public Class DoubleUpDown
             Inherits UpDownBase
```

  XAML Object Element Usage:

  Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

  Χρήση:

```
<zeus:DoubleUpDown ... />
```

- Για την **κλάση DecimalUpDown**:

  Σύνταξη:

  VB:

```
Public Class DecimalUpDown
             Inherits UpDownBase
```

  XAML Object Element Usage:

  Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

  Χρήση:

```
<zeus:DecimalUpDown ... />
```

- Για την **κλάση DatePickerUpDown**:

  Σύνταξη:

  VB:

```
Public Class DatePickerUpDown
             Inherits UpDownBase
```

  XAML Object Element Usage:

  Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

  Χρήση:

```
<zeus:DatePickerUpDown ... />
```

Διανομή

Κατά τη διανομή, στο φάκελο της εφαρμογής σας πρέπει να αντιγράψετε το **assembly αρχείο Zeus UpDownControls.dll**.

Επικοινωνία

Για οποιαδήποτε πληροφορία ή διευκρίνηση παρακαλώ επικοινωνήστε στο :

mouratx@yahoo.com ή mouratx@hotmail.com

Χρήστος Μουρατίδης,
Πειραιάς, Σεπτέμβριος 2018

Υ.Γ. Μπορείτε να επικοινωνήσετε μαζί μου για να προμηθευτείτε το **βιβλίο** μου
"Μάθετε το WPF με τη Visual Basic" (1.333 σελίδες, Αυτοέκδοση 2018).

UpDownBase

Η βασική κλάση όλων των **UpDownControls**.

Ιδιότητες

Ιδιότητες

Όνομα	Περιγραφή
ButtonsWidth	Καθορίζει το **πλάτος**, σε pixels, **των buttons αυξομείωσης του control**. (*Το ύψος είναι σταθερά ορισμένο στο μέσον του ύψους του control)*.
ErrorMessageOnIncorrectValueType	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι του συγκεκριμένου τύπου (ανάλογα με την sub-class, π.χ. για το ByteUpDown να είναι Byte κλπ).
ErrorMessageOnValueOutOfRange	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι στο αποδεκτό εύρος (Minimum-Maximum). Οι ιδιότητες Minimum, Maximum ορίζονται στις sub-classes.
ErrorTemplate	Καθορίζει ένα **ControlTemplate** που θα εφαρμόζεται **όταν το control είναι σε κατάσταση λάθους**.
HasError	Αν είναι **True**, τότε **το control είναι σε κατάσταση λάθους**.
IsMinimum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο ελάχιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Minimum σε μία sub-class (π.χ. ByteUpDown).
IsMaximum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο μέγιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Maximum σε μία sub-class (π.χ. ByteUpDown).
ValueFormat	Καθορίζει το **StringFormat** της τιμής του control.

ButtonsWidth

Καθορίζει το **πλάτος**, σε pixels, **των buttons αυξομείωσης του UpDownControl**.

Σύνταξη:

VB:

```
Public Property ButtonsWidth As Double
```

Τύπος: System.Double

Προσδιορίζουμε μία τιμή Double που θα καθορίζει το πλάτος του buttons αυξομείωσης του control.
By default, η τιμή είναι 15.

Παρατηρήσεις:

Η ελάχιστη τιμή είναι 10. Το ύψος είναι σταθερά ορισμένο στο μέσον του ύψους του control.

Dependency Property Information:

Identifier field: `ButtonsWidthProperty`

Παράδειγμα 1:

Στο επόμενο παράδειγμα, καθορίζουμε το πλάτος στο UpDown1 σε 25 pixels.

XAML:

```
<zeus:ByteUpDown x:Name="UpDown1" Minimum="1" Maximum="20" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.ButtonsWidth = 25
```

Default Τιμή (**ButtonsWidth=15**)	**ButtonsWidth=25**
Amount (Byte): 5	Amount (Byte): 5

ErrorMessageOnIncorrectValueType

Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται **όταν ο χρήστης εισάγει τιμή που δεν είναι συγκεκριμένου τύπου** (ανάλογα με την sub-class, π.χ. για ByteUpDown να είναι Byte κλπ).

Σύνταξη:

VB:

```
Public Property ErrorMessageOnIncorrectValueType As String
```

Τύπος: System.String

Προσδιορίζουμε μία τιμή String που θα αποτελεί το μήνυμα για λανθασμένη εισαγωγή τιμής.

Dependency Property Information:

Identifier field: `ErrorMessageOnIncorrectValueTypeProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το μήνυμα "Please, enter a valid value!" να εμφανίζεται όταν ο χρήστης εισάγει μία τιμή που δεν είναι τύπου Byte. Έχουμε ορίσει, στο τμήμα Window.Resources ενός αντικειμένου Window, ένα ControlTemplate για τα σφάλματα κι ένα Style για τα UpDown controls.

XAML

```
<Window.Resources>

    <!-- The Validation Error ControlTemplate -->
    <ControlTemplate x:Key="validationErrorTemplate">

        <StackPanel Orientation="Horizontal">

            <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                    Background="#11FF0000"
                    IsHitTestVisible="False" >
                <AdornedElementPlaceholder x:Name="placeholder" />
            </Border>

            <Grid  Margin="20,0,0,0"  >
                <Ellipse  Width="20" Height="20"  Fill="Red"  />
                <TextBlock Foreground="White" FontSize="14" Text="!"
                           FontWeight="ExtraBold"
                           VerticalAlignment="Center"
                           HorizontalAlignment="Center" />
                <Grid.ToolTip>
                    <TextBlock Text="{Binding Path=/ErrorContent}"/>
```

```
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

    <!-- The style base for the UpDown controls -->
    <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
        <Setter Property="FontSize" Value="14"/>
        <Setter Property="Background" Value="White"/>
        <Setter Property="HorizontalAlignment" Value="Left" />
        <Setter Property="Margin" Value="5,0,0,0"/>

        <Style.Triggers >
            <Trigger Property="IsMinimum" Value="True">
                <Setter Property="Background" Value="Orange" />
                <Setter Property="Foreground"  Value="White" />
            </Trigger>

            <Trigger Property="IsMaximum" Value="True">
                <Setter Property="Background" Value="Green"  />
                <Setter Property="Foreground"  Value="White" />
            </Trigger>

            <Trigger Property="HasError" Value="True">
                <Setter Property="Foreground" Value="Red"/>
                <Setter Property="BorderBrush" Value="Red"/>
                <Setter Property="BorderThickness" Value="2"/>
            </Trigger>

        </Style.Triggers>

      </Style>

</Window.Resources>

...

<zeus:ByteUpDown x:Name="UpDown1" Minimum="1" Maximum="20" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                 Style="{StaticResource UpDownStyle}" />
```

VB:

```
Dim errTemplate As ControlTemplate = _
              CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
UpDown1.ErrorTemplate = errTemplate
UpDown1.ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
```

Βλέπουμε το αποτέλεσμα, όταν ο χρήστης εισάγει μία λανθασμένη (ως προς τον τύπο) τιμή:

ErrorMessageOnValueOutOfRange

Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν **ο χρήστης εισάγει τιμή που δεν είναι μέσα στο αποδεκτό εύρος τιμών, όπως αυτό ορίζεται στις ιδιότητες Minimum και Maximum των sub-classes.**

Σύνταξη:

VB:

```
Public Property ErrorMessageOnValueOutOfRange As String
```

Τύπος: System.String

Προσδιορίζουμε μία τιμή String που θα αποτελεί το μήνυμα για λανθασμένη εισαγωγή τιμής.

Dependency Property Information:

Identifier field: `ErrorMessageOnValueOutOfRangeProperty`

Παρατηρήσεις:

Για το DatePickerUpDown το αποδεκτό εύρος ημερομηνιών προσδιορίζεται μέσω των ιδιοτήτων DisplayDateStart και DisplayDateEnd. Για τα αριθμητικά UpDownControls το αποδεκτό εύρος προσδιορίζεται μέσω των ιδιοτήτων Minimum και Maximum.

Παράδειγμα:

Στο επόμενο παράδειγμα, το UpDown1 έχει αποδεκτό εύρος τιμών από 1 έως 20. Αν ο χρήστης εισάγει μία έγκυρη τιμή Byte αλλά έξω από τα όρια τότε εμφανίζεται το μήνυμα "The number must be between 1 and 20. Please, retry." Έχουμε ορίσει, στο τμήμα Window.Resources ενός αντικειμένου Window, ένα ControlTemplate για τα σφάλματα κι ένα Style για τα UpDown controls.

XAML

```
<Window.Resources>

    <!-- The Validation Error ControlTemplate -->
    <ControlTemplate x:Key="validationErrorTemplate">

        <StackPanel Orientation="Horizontal">

            <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                    Background="#11FF0000"
                    IsHitTestVisible="False" >
```

```
                    <AdornedElementPlaceholder x:Name="placeholder" />
                </Border>

                <Grid  Margin="20,0,0,0"  >
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

    <!-- The style base for the UpDown controls -->
    <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
        <Setter Property="FontSize" Value="14"/>
        <Setter Property="Background" Value="White"/>
        <Setter Property="HorizontalAlignment" Value="Left" />
        <Setter Property="Margin" Value="5,0,0,0"/>

        <Style.Triggers >
            <Trigger Property="IsMinimum" Value="True">
                <Setter Property="Background" Value="Orange" />
                <Setter Property="Foreground"  Value="White" />
            </Trigger>

            <Trigger Property="IsMaximum" Value="True">
                <Setter Property="Background" Value="Green"  />
                <Setter Property="Foreground"  Value="White" />
            </Trigger>

            <Trigger Property="HasError" Value="True">
                <Setter Property="Foreground" Value="Red"/>
                <Setter Property="BorderBrush" Value="Red"/>
                <Setter Property="BorderThickness" Value="2"/>
            </Trigger>

        </Style.Triggers>

      </Style>

</Window.Resources>

...

<zeus:ByteUpDown x:Name="UpDown1" Minimum="1" Maximum="20" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                 ErrorMessageOnValueOutOfRange = "The number must be between 1 and 20.
                                                  Please, retry."
                 Style="{StaticResource UpDownStyle}" />
```

VB:

```
Dim errTemplate As ControlTemplate = _
              CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
UpDown1.ErrorTemplate = errTemplate
UpDown1.ErrorMessageOnValueOutOfRange = "The number must be betweem 1 and 20. Please,
retry."
```

Βλέπουμε το αποτέλεσμα, όταν ο χρήστης εισάγει μία **λανθασμένη (ως προς το εύρος) τιμή**:

ErrorTemplate

Καθορίζει ένα **ControlTemplate** που θα εφαρμόζεται **όταν το control είναι σε κατάσταση λάθους (λανθασμένος τύπος τιμής ή η τιμή δεν είναι μέσα στο αποδεκτό εύρος Minimum-Maximum)**.

Σύνταξη:

VB:

```
Public Property ErrorTemplate As ControlTemplate
```

Τύπος: System.Windows.Control.ControlTemplate

Προσδιορίζουμε ένα αντικείμενο ControlTemplate που θα αποτελεί το template εμφάνισης για την κατάσταση λάθους.

Dependency Property Information:

Identifier field: `ErrorTemplateProperty`

Παρατηρήσεις:

- Για το DatePickerUpDown το αποδεκτό εύρος ημερομηνιών προσδιορίζεται μέσω των ιδιοτήτων DisplayDateStart και DisplayDateEnd. Για τα αριθμητικά UpDownControls το αποδεκτό εύρος προσδιορίζεται μέσω των ιδιοτήτων Minimum και Maximum.

- Συνιστάται η εφαρμογή του ErrorTemplate να γίνεται στο συμβάν Loaded των root elements του προγράμματος (Window ή Page ή UserControl).

Παράδειγμα:

Στο επόμενο παράδειγμα, το UpDown1 έχει αποδεκτό εύρος τιμών από 1 έως 20. Αν ο χρήστης εισάγει μία έγκυρη τιμή Byte αλλά έξω από τα όρια τότε εμφανίζεται το μήνυμα "The number must be between 1 and 20. Please, retry." Έχουμε ορίσει, στο τμήμα Window.Resources ενός αντικειμένου Window, ένα ControlTemplate για τα σφάλματα κι ένα Style για τα UpDown controls.

XAML

```
<Window.Resources>

    <!-- The Validation Error ControlTemplate -->
    <ControlTemplate x:Key="validationErrorTemplate">

        <StackPanel Orientation="Horizontal">
```

```
                <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                        Background="#11FF0000"
                        IsHitTestVisible="False" >
                    <AdornedElementPlaceholder x:Name="placeholder" />
                </Border>

                <Grid  Margin="20,0,0,0"  >
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

    <!-- The style base for the UpDown controls -->
    <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
        <Setter Property="FontSize" Value="14"/>
        <Setter Property="Background" Value="White"/>
        <Setter Property="HorizontalAlignment" Value="Left" />
        <Setter Property="Margin" Value="5,0,0,0"/>

        <Style.Triggers >
            <Trigger Property="IsMinimum" Value="True">
                <Setter Property="Background" Value="Orange" />
                <Setter Property="Foreground"  Value="White" />
            </Trigger>

            <Trigger Property="IsMaximum" Value="True">
                <Setter Property="Background" Value="Green"  />
                <Setter Property="Foreground"  Value="White" />
            </Trigger>

            <Trigger Property="HasError" Value="True">
                <Setter Property="Foreground" Value="Red"/>
                <Setter Property="BorderBrush" Value="Red"/>
                <Setter Property="BorderThickness" Value="2"/>
            </Trigger>

        </Style.Triggers>

      </Style>

</Window.Resources>

...

<zeus:ByteUpDown x:Name="UpDown1" Minimum="1" Maximum="20" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                 ErrorMessageOnValueOutOfRange = "The number must be between 1 and 20.
                                                  Please, retry."
                 Style="{StaticResource UpDownStyle}" />
```

VB:

```
Private Sub Window_Loaded(sender As Object, e As RoutedEventArgs)

    Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
    UpDown1.ErrorTemplate = errTemplate

End Sub
```

Βλέπουμε το αποτέλεσμα, όταν ο χρήστης εισάγει μία λανθασμένη (ως προς το εύρος) τιμή. Εφαρμόζεται το αντικείμενο validationErrorTemplate, το οποίο έχει οριστεί στο XAML κώδικα του παραθύρου (ως resource), στην ιδιότητα ErrorTemplate του UpDown1 στον Loaded event handler του παραθύρου.

HasError

Αν είναι **True**, τότε **το UpDown control είναι σε κατάσταση λάθους**.

Σύνταξη:

VB:

```
Public ReadOnly Property HasError As Boolean
```

Τύπος: System.Boolean

Επιστρέφει True αν το UpDown control είναι σε κατάσταση λάθους.

Dependency Property Information:

Identifier field: `HasErrorProperty`

Παράδειγμα:

Πολύ χρήσιμο είναι να αξιοποιήσουμε την ιδιότητα HasError σε ένα Style Trigger ώστε να θέσουμε κάποιες ιδιότητες μορφοποίησης (π.χ. Foreground) στην περίπτωση που το UpDown control είναι σε κατάσταση λάθους. Στο παρακάτω παράδειγμα, ορίζουμε ένα style για τα UpDown controls όπου σε ένα Style Trigger ορίζουμε ότι αν η ιδιότητα HasError είναι True, το χρώμα να γίνει κόκκινο με κόκκινο περίγραμμα πάχους 2 pixels (Foreground σε Red, BorderBrush σε Red και BorderThickness σε 2).

XAML

```
<Window.Resources>

    <!-- The Validation Error ControlTemplate -->
    <ControlTemplate x:Key="validationErrorTemplate">

        <StackPanel Orientation="Horizontal">

            <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                    Background="#11FF0000"
                    IsHitTestVisible="False" >
                <AdornedElementPlaceholder x:Name="placeholder" />
            </Border>

            <Grid  Margin="20,0,0,0"  >
                <Ellipse  Width="20" Height="20"  Fill="Red"  />
                <TextBlock Foreground="White" FontSize="14" Text="!"
                           FontWeight="ExtraBold"
                           VerticalAlignment="Center"
                           HorizontalAlignment="Center" />
                <Grid.ToolTip>
                    <TextBlock Text="{Binding Path=/ErrorContent}"/>
```

```
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

    <!-- The style base for the UpDown controls -->
    <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
        <Setter Property="FontSize" Value="14"/>
        <Setter Property="Background" Value="White"/>
        <Setter Property="HorizontalAlignment" Value="Left" />
        <Setter Property="Margin" Value="5,0,0,0"/>

        <Style.Triggers >
            <Trigger Property="IsMinimum" Value="True">
                <Setter Property="Background" Value="Orange" />
                <Setter Property="Foreground"  Value="White" />
            </Trigger>

            <Trigger Property="IsMaximum" Value="True">
                <Setter Property="Background" Value="Green"  />
                <Setter Property="Foreground"  Value="White" />
            </Trigger>

            <Trigger Property="HasError" Value="True">
                <Setter Property="Foreground" Value="Red"/>
                <Setter Property="BorderBrush" Value="Red"/>
                <Setter Property="BorderThickness" Value="2"/>
            </Trigger>

        </Style.Triggers>

    </Style>

</Window.Resources>

...

<zeus:ByteUpDown x:Name="UpDown1" Minimum="1" Maximum="20" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                 ErrorMessageOnValueOutOfRange = "The number must be between 1 and 20.
                                                  Please, retry."
                 Style="{StaticResource UpDownStyle}" />
```

VB:

```
Private Sub Window_Loaded(sender As Object, e As RoutedEventArgs)

    Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
    UpDown1.ErrorTemplate = errTemplate

End Sub
```

Βλέπουμε το αποτέλεσμα, όταν ο χρήστης εισάγει μία λανθασμένη (ως προς το εύρος) τιμή. Πέρα από το ErrorTemplate, ενεργοποιείται και το Style Trigger όπου το χρώμα του αριθμού γίνεται κόκκινο, το περίγραμμα κόκκινο με πάχος 2 pixels.

IsMinimum

Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο ελάχιστο όριο,** όπως έχει προσδιοριστεί στην ιδιότητα Minimum σε μία sub-class (π.χ. ByteUpDown).

Σύνταξη:

VB:

```
Public ReadOnly Property IsMinimum As Boolean
```

Τύπος: System.Boolean

Επιστρέφει True αν η τρέχουσα τιμή έχει φτάσει στο ελάχιστο όριο, όπως έχει προσδιοριστεί στην ιδιότητα Minimum σε μία sub-class (π.χ. ByteUpDown).

Dependency Property Information:

Identifier field: `IsMinimumProperty`

Παράδειγμα:

Πολύ χρήσιμο είναι να αξιοποιήσουμε την ιδιότητα IsMinimum σε ένα Style Trigger ώστε να θέσουμε κάποιες ιδιότητες μορφοποίησης (π.χ. Foreground) στην περίπτωση που η τρέχουσα τιμή στο UpDown control έχει φτάσει στο ελάχιστο όριο, όπως αυτό έχει προσδιοριστεί στην ιδιότητα Minimum σε μία sub-class (π.χ. ByteUpDown).
Στο παρακάτω παράδειγμα, ορίζουμε ένα style για τα UpDown controls όπου σε ένα Style Trigger ορίζουμε ότι αν η ιδιότητα IsMinimum είναι True, το χρώμα φόντου να γίνει πορτοκαλί και το χρώμα γραμματοσειράς να γίνει άσπρο (Background σε Orange και Foreground σε White).

XAML

```
<Window.Resources>

    <!-- The Validation Error ControlTemplate -->
    <ControlTemplate x:Key="validationErrorTemplate">

        <StackPanel Orientation="Horizontal">

            <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                    Background="#11FF0000"
                    IsHitTestVisible="False" >
                <AdornedElementPlaceholder x:Name="placeholder" />
            </Border>

            <Grid  Margin="20,0,0,0"  >
                <Ellipse  Width="20" Height="20"  Fill="Red"  />
                <TextBlock Foreground="White" FontSize="14" Text="!"
                           FontWeight="ExtraBold"
```

```
                                VerticalAlignment="Center"
                                HorizontalAlignment="Center" />
                        <Grid.ToolTip>
                            <TextBlock Text="{Binding Path=/ErrorContent}"/>
                        </Grid.ToolTip>
                    </Grid>

                </StackPanel>

            </ControlTemplate>

        <!-- The style base for the UpDown controls -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="HasError" Value="True">
                    <Setter Property="Foreground" Value="Red"/>
                    <Setter Property="BorderBrush" Value="Red"/>
                    <Setter Property="BorderThickness" Value="2"/>
                </Trigger>

            </Style.Triggers>

        </Style>

    </Window.Resources>

    ...

    <zeus:ByteUpDown x:Name="UpDown1" Minimum="1" Maximum="20" Increment="1"
                     HorizontalContentAlignment="Right"
                     Value="{Binding Amount1}"  ValueFormat="N0"
                     ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                     ErrorMessageOnValueOutOfRange = "The number must be between 1 and 20.
                                                      Please, retry."
                     Style="{StaticResource UpDownStyle}" />
```

Το αποτέλεσμα όταν η τρέχουσα τιμή φτάσει στο ελάχιστο (επίσης, το down arrow button απενεργοποείται αυτόματα):

Παράδειγμα 2 :

Εδώ, ορίζουμε **δυναμικά το style**, με όνομα style1 (στοχευμένο στον τύπο UpDownBase), ορίζοντας κάποιες ιδιότητες μορφοποίησης (FontSize, Background, HorizontalAlignment, Margin) και ένα style trigger για την ιδιότητα IsMinimum:

VB:

```
'Define the new style, targeting at the UpDownBase.
Dim style1 As New Style With {.TargetType = GetType(UpDownBase)}
With style1.Setters
    .Add(New Setter(FontSizeProperty, CDbl(16)))
    .Add(New Setter(BackgroundProperty, Brushes.White))
    .Add(New Setter(HorizontalAlignmentProperty, HorizontalAlignment.Left))
    .Add(New Setter(MarginProperty, New Thickness(5, 0, 0, 0)))
End With

'The style trigger for the isMinimum property.
Dim minimumStyleTrigger As New Trigger
With minimumStyleTrigger

    'Condition.
    .Property = UpDownBase.IsMinimumProperty
    .Value = True

    'Format setters.
    .Setters.Add(New Setter(BackgroundProperty, Brushes.Brown))
    .Setters.Add(New Setter(ForegroundProperty, Brushes .Yellow))

End With

'Add the style trigger to the collection.
style1.Triggers.Add(minimumStyleTrigger)

'Set the UpDown1 style to the style1.
UpDown1.Style = style1
```

Βλέπουμε το αποτέλεσμα:

IsMaximum

Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο μέγιστο όριο,** όπως έχει προσδιοριστεί στην ιδιότητα Maximum σε μία sub-class (π.χ. ByteUpDown).

Σύνταξη:

VB:

```
Public ReadOnly Property IsMaximum As Boolean
```

Τύπος: System.Boolean

Επιστρέφει True αν η τρέχουσα τιμή έχει φτάσει στο μέγιστο όριο, όπως έχει προσδιοριστεί στην ιδιότητα Maximum σε μία sub-class (π.χ. ByteUpDown).

Dependency Property Information:

Identifier field: `IsMaximumProperty`

Παράδειγμα:

Πολύ χρήσιμο είναι να αξιοποιήσουμε την ιδιότητα IsMaximum σε ένα Style Trigger ώστε να θέσουμε κάποιες ιδιότητες μορφοποίησης (π.χ. Foreground) στην περίπτωση που η τρέχουσα τιμή στο UpDown control έχει φτάσει στο μέγιστο όριο, όπως έχει προσδιοριστεί στην ιδιότητα Maximum σε μία sub-class (π.χ. ByteUpDown).
Στο παρακάτω παράδειγμα, ορίζουμε ένα style για τα UpDown controls όπου σε ένα Style Trigger ορίζουμε ότι αν η ιδιότητα IsMaximum είναι True, το χρώμα φόντου να γίνει πράσινο και το χρώμα γραμματοσειράς να γίνει άσπρο (Background σε Green και Foreground σε White).

XAML

```
<Window.Resources>

    <!-- The Validation Error ControlTemplate -->
    <ControlTemplate x:Key="validationErrorTemplate">

        <StackPanel Orientation="Horizontal">

            <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                    Background="#11FF0000"
                    IsHitTestVisible="False" >
                <AdornedElementPlaceholder x:Name="placeholder" />
            </Border>

            <Grid  Margin="20,0,0,0"  >
                <Ellipse  Width="20" Height="20"  Fill="Red"  />
                <TextBlock Foreground="White" FontSize="14" Text="!"
                           FontWeight="ExtraBold"
```

```
                                VerticalAlignment="Center"
                                HorizontalAlignment="Center" />
                        <Grid.ToolTip>
                            <TextBlock Text="{Binding Path=/ErrorContent}"/>
                        </Grid.ToolTip>
                    </Grid>

                </StackPanel>

            </ControlTemplate>

        <!-- The style base for the UpDown controls -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="HasError" Value="True">
                    <Setter Property="Foreground" Value="Red"/>
                    <Setter Property="BorderBrush" Value="Red"/>
                    <Setter Property="BorderThickness" Value="2"/>
                </Trigger>

            </Style.Triggers>

        </Style>

    </Window.Resources>

    ...

    <zeus:ByteUpDown x:Name="UpDown1" Minimum="1" Maximum="20" Increment="1"
                     HorizontalContentAlignment="Right"
                     Value="{Binding Amount1}"  ValueFormat="N0"
                     ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                     ErrorMessageOnValueOutOfRange = "The number must be between 1 and 20.
                                                      Please, retry."
                     Style="{StaticResource UpDownStyle}" />
```

Το αποτέλεσμα όταν η τρέχουσα τιμή φτάσει στο μέγιστο (επίσης, το up arrow button απενεργοποείται αυτόματα):

Παράδειγμα 2 :

Εδώ, ορίζουμε **δυναμικά το style**, με όνομα style2 (στοχευμένο στον τύπο UpDownBase), ορίζοντας κάποιες ιδιότητες μορφοποίησης (FontSize, Background, HorizontalAlignment, Margin) και ένα style trigger για την ιδιότητα IsMaximum:

VB:

```
'Define the new style, targeting at the UpDownBase.
Dim style2 As New Style With {.TargetType = GetType(UpDownBase)}
With style2.Setters
    .Add(New Setter(FontSizeProperty, CDbl(16)))
    .Add(New Setter(BackgroundProperty, Brushes.White))
    .Add(New Setter(HorizontalAlignmentProperty, HorizontalAlignment.Left))
    .Add(New Setter(MarginProperty, New Thickness(5, 0, 0, 0)))
End With

'The style trigger for the isMaximum property.
Dim maximumStyleTrigger As New Trigger
With maximumStyleTrigger

    'Condition.
    .Property = UpDownBase.IsMaximumProperty
    .Value = True

    'Format setters.
    .Setters.Add(New Setter(BackgroundProperty, Brushes.BlueViolet))
    .Setters.Add(New Setter(ForegroundProperty, Brushes .Yellow))
   .Setters.Add(New Setter(FontSizeProperty, CDbl(18)))

End With

'Add the style trigger to the collection.
style2.Triggers.Add(maximumStyleTrigger )

'Set the UpDown1 style to the style2.
UpDown1.Style = style2
```

Βλέπουμε το αποτέλεσμα:

ValueFormat

Καθορίζει το **StringFormat** της τιμής του control.

Σύνταξη:

VB:

```
Public Property ValueFormat As String
```

Τύπος: System.String

Προσδιορίζουμε ένα string για την εμφάνιση της τιμής του control.

Dependency Property Information:

Identifier field: `ValueFormatProperty`

Παρατηρήσεις:

- Το string μορφοποίησης της τιμής (αριθμού ή ημερομηνίας) ακολουθεί τους κανόνες που ισχύουν στην ιδιότητα StringFormat της κλάσης BindingBase. Συμβουλευθείτε το Help του Visual Studio για λεπτομέρειες.

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε η μορφή εμφάνισης της τιμής του UpDown1 να είναι "C0", δηλαδή νομισματική με 0 δεκαδικά. Προσέχουμε να ορίσουμε σωστά την ιδιότητα Language, αν αυτό είναι απαραίτητο.

XAML

```
<zeus:ByteUpDown x:Name="UpDown1" Minimum="1" Maximum="20" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="C0" Language="el"
                 ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                 ErrorMessageOnValueOutOfRange = "The amount must be between 1 and 20.
                                                  Please, retry."
                 Style="{StaticResource UpDownStyle }"  />
```

VB:

```
UpDown1.ValueFormat = "C0"
```

Βλέπουμε το αποτέλεσμα:

Αριθμητικοί τύποι

Στα επόμενα, παρουσιάζονται τα **UpDownControls** για τους βασικούς **αριθμητικούς τύπους** (**Byte, Short, Integer, Long, Single, Double** και **Decimal**).

- ByteUpDown
- ShortUpDown
- IntegerUpDown
- LongUpDown
- SingleUpDown
- DoubleUpDown
- DecimalUpDown

ByteUpDown

Ένα **UpDown control** που επιτρέπει την **είσοδο μόνο αριθμητικών δεδομένων, τύπου Byte**. Η τιμή απαιτείται (required).

Σύνταξη:

VB:

```
<TemplateVisualState(Name:="Normal", GroupName:="CommonStates"),
TemplateVisualState(Name:="MouseOver", GroupName:="CommonStates"),
TemplateVisualState(Name:="Disabled", GroupName:="CommonStates"),
TemplateVisualState(Name:="Focused", GroupName:="FocusStates"),
TemplateVisualState(Name:="Unfocused", GroupName:="FocusStates"),
TemplatePart(Name:="PART_Value", Type:=GetType(TextBox)),
TemplatePart(Name:="PART_IncrementUp", Type:=GetType(RepeatButton)),
TemplatePart(Name:="PART_IncrementDown", Type:=GetType(RepeatButton))>
<DefaultProperty("Value")>
Public Class ByteUpDown
             Inherits UpDownBase
```

XAML Object Element Usage:

Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

Χρήση:

```
<zeus:ByteUpDown ... />
```

Παράδειγμα:

Στο παράδειγμα που ακολουθεί, τοποθετούμε ένα **ByteUpDown** μέσα σε ένα **Grid** ενός παραθύρου. Συνδέουμε την **ιδιότητα Value** με την **ιδιότητα Amount1**, τύπου **Byte**, ενός **Customer** αντικειμένου (source object). Η **κλάση Customer** ορίζεται στο project και συνεπώς πρέπει να εισάγουμε και το αντίστοιχο namespace με alias p. Επίσης, ορίζουμε ένα **ErrorTemplate** για το **Validation** (όπου μέσω αυτού θα εμφανιστεί το μήνυμα λάθους που ορίζουμε στην **ιδιότητα ErrorMessageOnIncorrectValueType** ή στην **ιδιότητα ErrorMessageOnValueOutOfRange**). Επιπλέον, ορίζουμε ένα **Style**, στοχευμένο στην κλάση UpDownBase, για την ομοιόμορφη εμφάνιση των UpDown controls στο παράθυρο.

```
<Window x:Class="MainWindow"
         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:p="clr-namespace:UpDownControlsTestProject"
        xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
        mc:Ignorable="d"
        Title="UpDownControls Sample Project"
        Height="350" Width="525" Loaded="Window_Loaded" >

    <Window.Resources>

        <!-- The Validation Error ControlTemplate -->
        <ControlTemplate x:Key="validationErrorTemplate">

            <StackPanel Orientation="Horizontal">

                <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                        Background="#11FF0000"
                        IsHitTestVisible="False" >
                    <AdornedElementPlaceholder x :Name="placeholder"  />
                </Border>

                <Grid  Margin="20,0,0,0"  >
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

        <Style TargetType="{x:Type TextBlock }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="HorizontalAlignment" Value="Right" />
            <Setter Property="Margin" Value="5"/>
        </Style>

        <!-- The style base for the UpDown controls. -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>
```

```
                    <Trigger Property="HasError" Value="True">
                        <Setter Property="Foreground" Value="Red"/>
                        <Setter Property="BorderBrush" Value="Red"/>
                        <Setter Property="BorderThickness" Value="2"/>
                    </Trigger>

                </Style.Triggers>

            </Style>

        <!-- A sample Customer resource object for the Bindings -->
        <p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
                                     Amount1="12"/>


    </Window.Resources>

    <Grid  Margin="10,40,10,10"
           DataContext="{Binding Source={StaticResource customer}}">

        ...

        <TextBlock Text="Amount (Byte):" />
        <zeus:ByteUpDown Name="UpDown1" Grid.Column="1" Minimum="1" Maximum="20"
                         Increment="1" Language="el"
                         HorizontalContentAlignment="Right"
                  ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                  ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                   20. Please, retry."
                         Value="{Binding Amount1}"  ValueFormat="N0"
                        ValueChanged="UpDown1_ValueChanged"
                         Style="{StaticResource UpDownStyle}"/>

       ...

    </Grid>


</Window>
```

VB:

```
Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
UpDown1.ErrorTemplate = errTemplate
```

- Για την **κλάση Customer** δείτε το Παράρτημα 1.

Μία κανονική κατάσταση του **ByteUpDown** :

Παρακάτω, βλέπουμε την **ένδειξη λάθους** όταν ο χρήστης εισάγει μία **λανθασμένη (ως προς το εύρος) τιμή**. Αν πάει το δείκτη του ποντικιού πάνω στο **θαυμαστικό** θα εμφανιστεί το μήνυμα λάθους "`The number must be between 1 and 20. Please, retry.`"

Ιδιότητες

Όνομα	Περιγραφή
ButtonsWidth	Καθορίζει το **πλάτος**, σε pixels, **των buttons αυξομείωσης του control**. (*Το ύψος είναι σταθερά ορισμένο στο μέσον του ύψους του control)*. (Κληρονομείται από την UpDownBase).
ErrorMessageOnIncorrectValueType	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι τύπου Short. (Κληρονομείται από την UpDownBase).
ErrorMessageOnValueOutOfRange	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι στο αποδεκτό εύρος (Minimum-Maximum). (Κληρονομείται από την UpDownBase).
ErrorTemplate	Καθορίζει ένα **ControlTemplate** που θα εφαρμόζεται **όταν το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
HasError	Αν είναι **True**, τότε **το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
Increment	Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Byte. Η default τιμή είναι 1.
IsMinimum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο ελάχιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Minimum. (Κληρονομείται από την UpDownBase).
IsMaximum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο μέγιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Maximum. (Κληρονομείται από την UpDownBase).
Minimum	Καθορίζει το **ελάχιστο όριο** στο control, τύπου Byte. Η default τιμή είναι Byte.MinValue.
Maximum	Καθορίζει το **μέγιστο όριο** στο control, τύπου Byte. Η default τιμή είναι Byte.MaxValue.

Value	Η **τρέχουσα τιμή** στο control, τύπου Byte. Η default τιμή είναι 0.
ValueFormat	Καθορίζει το **StringFormat** της τιμής του control. (Κληρονομείται από την UpDownBase).

Increment

Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Byte.

Σύνταξη:

VB:

```
Public Property Increment As Byte
```

Τύπος: System.Byte

Προσδιορίζουμε μία θετική τιμή για το βήμα αύξησης της τιμής. Η default τιμή είναι 1.

Dependency Property Information:

Identifier field: `IncrementProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το βήμα αύξησης στο UpDown1 σε 2.

XAML:

```
<zeus:ByteUpDown x:Name="UpDown1" Minimum="1" Maximum="20" Increment="2"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Increment = 2
```

Minimum

Καθορίζει το **ελάχιστο όριο** στο control, τύπου Byte.

Σύνταξη:

VB:

```
Public Property Minimum As Byte
```

Τύπος: System.Byte

Προσδιορίζουμε μία τιμή, ως το ελάχιστο αποδεκτό όριο. Η default τιμή είναι Byte.MinValue.

Dependency Property Information:

Identifier field: `MinimumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το ελάχιστο όριο στο UpDown1 σε 10.

XAML:

```
<zeus:ByteUpDown x:Name="UpDown1" Minimum="10" Maximum="20" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Minimum = 10
```

Maximum

Καθορίζει το **μέγιστο όριο** στο control, τύπου Byte.

Σύνταξη:

VB:

```
Public Property Maximum As Byte
```

Τύπος: System.Byte

Προσδιορίζουμε μία τιμή, ως το μέγιστο αποδεκτό όριο. Η default τιμή είναι Byte.MaxValue.

Dependency Property Information:

Identifier field: `MaximumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το μέγιστο όριο στο UpDown1 σε 50.

XAML:

```
<zeus:ByteUpDown x:Name="UpDown1" Minimum="10" Maximum="50" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Maximum = 50
```

Value

Η **τρέχουσα τιμή** στο control, τύπου Byte.

Σύνταξη:

VB:

```
Public Property Value As Byte
```

Τύπος: System.Byte

Η τρέχουσα τιμή. Η default τιμή είναι 0.

Dependency Property Information:

Identifier field: `ValueProperty`

Παρατηρήσεις:

Σε μία WPF εφαρμογή, η λογική χρήση της ιδιότητας αυτής είναι μέσω μίας Binding έκφρασης. Με αυτόν τον τρόπο, η τρέχουσα τιμή συνδέεται με μία source πηγή (π.χ. μία public ιδιότητα ενός data object). Η τιμή προέρχεται από το source object και οποιαδήποτε μεταβολή της επιστρέφει στο source object (σε ένα two-way mode, που είναι και το default). Συμπερασματικά, **συνιστάται η χρήση της με Binding έφραση**.

Παράδειγμα:

Στο επόμενο παράδειγμα, η τιμή της **ιδιότητας Value** στο **UpDown1** συνδέεται, **μέσω Binding έκφρασης**, με την **ιδιότητα Amount1** του **source object**. Το τελευταίο (ένα **αντικείμενο Customer**) τίθεται ως πηγή δεδομένων στην **ιδιότητα DataContext** σε ένα container, που εδώ είναι το **Grid** panel. Οι τιμές των ιδιοτήτων του αντικειμένου Customer τίθενται στο τμήμα Window.Resources, αλλά θα μπορούσαν να τεθούν και στο VB κώδικα, σε κάποιον event handler.

```
<Window x:Class="MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:p="clr-namespace:UpDownControlsTestProject"
        xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
        mc:Ignorable="d"
        Title="UpDownControls Sample Project"
        Height="350" Width="525" Loaded="Window_Loaded" >

    <Window.Resources>
```

```
<!-- The Validation Error ControlTemplate -->
<ControlTemplate x:Key="validationErrorTemplate">

    <StackPanel Orientation="Horizontal">

        <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                Background="#11FF0000"
                IsHitTestVisible="False" >
            <AdornedElementPlaceholder x :Name="placeholder"  />
        </Border>

        <Grid  Margin="20,0,0,0"  >
            <Ellipse  Width="20" Height="20"  Fill="Red"  />
            <TextBlock Foreground="White" FontSize="14" Text="!"
                       FontWeight="ExtraBold"
                       VerticalAlignment="Center"
                       HorizontalAlignment="Center" />
            <Grid.ToolTip>
                <TextBlock Text="{Binding Path=/ErrorContent}"/>
            </Grid.ToolTip>
        </Grid>

    </StackPanel>

</ControlTemplate>

<Style TargetType="{x:Type TextBlock }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="HorizontalAlignment" Value="Right" />
    <Setter Property="Margin" Value="5"/>
</Style>

<!-- The style base for the UpDown controls. -->
<Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Background" Value="White"/>
    <Setter Property="HorizontalAlignment" Value="Left" />
    <Setter Property="Margin" Value="5,0,0,0"/>

    <Style.Triggers >
        <Trigger Property="IsMinimum" Value="True">
            <Setter Property="Background" Value="Orange" />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="IsMaximum" Value="True">
            <Setter Property="Background" Value="Green"  />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="HasError" Value="True">
            <Setter Property="Foreground" Value="Red"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
        </Trigger>

    </Style.Triggers>

  </Style>

<!-- A sample Customer resource object for the Bindings -->
<p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
```

```
                                        Amount1="12"/>


        </Window.Resources>

        <Grid  Margin="10,40,10,10"
               DataContext="{Binding Source={StaticResource customer}}">

            ...

            <TextBlock Text="Amount (Byte):" />
            <zeus:ByteUpDown Name="UpDown1" Grid.Column="1" Minimum="1" Maximum="20"
                             Increment="1" Language="el"
                             HorizontalContentAlignment="Right"
                        ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                        ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                                    20. Please, retry."
                             Value="{Binding Amount1}"  ValueFormat="N0"
                            ValueChanged="UpDown1_ValueChanged"
                             Style="{StaticResource UpDownStyle}"/>

           ...

        </Grid>


</Window>
```

VB:

```
Private Sub Window_Loaded(sender As Object, e As RoutedEventArgs)

    Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
    UpDown1.ErrorTemplate = errTemplate

End Sub
```

Βλέπουμε την αρχική κατάσταση του UpDown1. Η τιμή 12 προέρχεται από την ιδιότητα Amount1 του αντικειμένου Customer:

Αν ο χρήστης φτάσει στη μέγιστη τιμή (20) τότε, λόγω του style trigger, το ByteUpDown μορφοποιείται ως εξής:

Παράλληλα, απενεργοποιείται και το up arrow button. Αν ο χρήστης, πληκτρολογήσει απ' ευθείας μία **τιμή εκτός ορίων**, τότε το control εισέρχεται σε **κατάσταση λάθους**:

Επίσης, αν ο χρήστης, πληκτρολογήσει μία **τιμή λανθασμένου τύπου**, τότε το control πάλι εισέρχεται σε **κατάσταση λάθους**:

Συμβάντα

Όνομα	Περιγραφή
MinimumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.
MaximumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.
ValueChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

MinimumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MinimumChanged As RoutedPropertyChangedEventHandler(Of Byte)
```

XAML attribute usage:

```
<zeus:ByteUpDown MinimumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MinimumChanged του UpDown1:

XAML:

```
<zeus:ByteUpDown x:Name="UpDown1" Minimum="0" Maximum="20" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 MinimumChanged="UpDown1_MinimumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MinimumChanged(sender As  Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Byte))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new minimum value is {0}", e.NewValue))

     End If

End Sub
```

MaximumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MaximumChanged As RoutedPropertyChangedEventHandler(Of Byte)
```

XAML attribute usage:

```
<zeus:ByteUpDown MaximumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MaximumChanged του UpDown1:

XAML:

```
<zeus:ByteUpDown x:Name="UpDown1" Minimum="0" Maximum="20" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 MaximumChanged="UpDown1_MaximumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MaximumChanged(sender As Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Byte))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new maximum value is {0}", e.NewValue))

     End If

End Sub
```

ValueChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event ValueChanged As RoutedPropertyChangedEventHandler(Of Byte)
```

XAML attribute usage:

```
<zeus:ByteUpDown ValueChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν ValueChanged του UpDown1:

XAML:

```
<zeus:ByteUpDown x:Name="UpDown1" Minimum="0" Maximum="20" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ValueChanged="UpDown1_ValueChanged" ... />
```

VB:

```
 Private Sub UpDown1_ValueChanged(sender As Object, _
                                  e As RoutedPropertyChangedEventArgs(Of Byte))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The current value is {0}", e.NewValue))

     End If

End Sub
```

Styles και Templates

- Parts και States
- Το default Style και ControlTemplate
- Παράδειγμα custom Style και ControlTemplate

Parts και States

Το default ControlTemplate περιλαμβάνει κάποια **part names** και **visual states**. Μπορείτε να τροποποιήσετε το default ControlTemplate ώστε να δώσετε στο control μία μοναδική εμφάνιση. Το default ControlTemplate μπορείτε να το δείτε εδώ ώστε να πάρετε μία ιδέα και με βάση αυτό να δημιουργήσετε ένα δικό σας. Ένα παράδειγμα custom ControlTemplate μπορείτε να δείτε εδώ.

Ο παρακάτω πίνακας εμφανίζει τα **part names** του **ByteUpDown** control:

Part	Τύπος	Περιγραφή
PART_Value	TextBox	Το standard TextBox control.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την αύξηση της τιμής.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την μείωση της τιμής.

Ο παρακάτω πίνακας εμφανίζει τα **visual states** του **ByteUpDown** control:

VisualState	VisualStateGroup	Περιγραφή
Normal	CommonStates	Το default state.
MouseOver	CommonStates	Το state όταν ο δείκτης του ποντικιού είναι πάνω από το control.
Disabled	CommonStates	Το state όταν το control είναι disabled.
Focused	FocusStates	Το state όταν το control έχει το focus.
Unfocused	FocusStates	Το state όταν το control δεν έχει το focus.

Το default ControlTemplate έχει καθορισμένο μόνο το Disabled state. Μπορείτε να δημιουργήσετε ένα custom ControlTemplate για να το αλλάξετε ή/και για να καθορίσετε τα υπόλοιπα.

Το default Style και ControlTemplate

Ο XAML κώδικας για το **default Style** και **ControlTemplate** φαίνεται παρακάτω. Μπορείτε να βασιστείτε σε αυτόν για να δημιουργήσετε μία μικρή ή μεγάλη παραλλαγή του δικού σας custom Style και ControlTemplate:

```
    xmlns:z="clr-namespace:Zeus.WPF.Controls.UpDownControls">

    <Style TargetType="{x:Type z:ByteUpDown}" >

        <Setter Property="Focusable" Value="True"/>
        <Setter Property="BorderBrush" Value="Black" />
        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="Background"  Value="Transparent" />
        <Setter Property="Width" Value ="120"/>
        <Setter Property="Height" Value="23"/>

        <Setter Property="Template">

            <Setter.Value>

                <ControlTemplate TargetType="{x:Type z:ByteUpDown}">

                    <ControlTemplate.Resources >
                        <z:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                    </ControlTemplate.Resources>

                    <!-- Root element -->
                    <Border Name="Border"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">

                        <VisualStateManager.VisualStateGroups >

                            <VisualStateGroup Name="CommonStates">

                                <VisualState Name="Normal"/>

                                <VisualState Name="Disabled">

                                    <Storyboard >

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Background.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                     Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Foreground.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                    Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ObjectAnimationUsingKeyFrames
```

```
                                        Storyboard.TargetName="PART_Value"
                                    Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                        <DiscreteObjectKeyFrame KeyTime="0"
                                 Value="{Binding Source={x:Static FontWeights.Normal}}" />
                                        </ObjectAnimationUsingKeyFrames>

                                    </Storyboard>

                                </VisualState>


                            </VisualStateGroup>

                            <VisualStateGroup Name="FocusStates">

                                <VisualState Name="Focused" />
                                <VisualState Name="Unfocused"/>

                            </VisualStateGroup>

                        </VisualStateManager.VisualStateGroups>

                        <!-- Content -->
                        <Grid Name="mainGrid" Background="Transparent" >

                            <Grid.RowDefinitions >
                                <RowDefinition Height="*" />
                            </Grid.RowDefinitions>

                            <!-- The width of the second Grid column is about for the
                                 RepeatButtons and depends on ButtonsWidth property -->
                            <Grid.ColumnDefinitions >
                                <ColumnDefinition Width="*" />
                                <ColumnDefinition
                                      Width="{Binding RelativeSource={Relative Source
                              Mode=FindAncestor, AncestorType={x:Type z:ByteUpDown}},
        Path=ButtonsWidth, Converter={StaticResource doubleToGridLengthConverter}}" />
                            </Grid.ColumnDefinitions>

                             <TextBox Name="PART_Value"
                HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                    VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                                Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                            </TextBox>

                            <Viewbox Stretch="Fill"   Grid.Column="1">

                                <StackPanel Name="stkRepeatButtons"
                                            VerticalAlignment="Center"
                                            HorizontalAlignment="Left"  Focusable="False">

                                    <RepeatButton Name="PART_IncrementUp"
                                                  Focusable="False"  >
                                        <RepeatButton.Content>
                                            <Path Fill="Black" Stroke="Green"
                                                  Data="M 1,10 10,1 20,10 Z" />
                                        </RepeatButton.Content>
                                    </RepeatButton>
```

```
                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                    <RepeatButton.Content>
                                        <Path Fill="Black" Stroke="Green"
                                              Data="M 10,10 1,1 20,1 Z" />
                                    </RepeatButton.Content>
                                </RepeatButton>

                            </StackPanel>

                        </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

Η δεύτερη στήλη του Grid περιέχει τα repeat buttons μέσα σε ένα StackPanel. Το πλάτος της δεύτερης στήλης του Grid προσδιορίζεται με βάση την τιμή της ιδιότητας ButtonsWidth. Για να μετατραπεί η τιμή Double της ιδιότητας ButtonsWidth σε τιμή τύπου GridLength χρησιμοποιούμε μία κλάση μετατροπής τιμής (ValueConverter). Θα τη βρείτε στο Παράρτημα 2.

Παράδειγμα custom Style και ControlTemplate

Στο επόμενο παράδειγμα, δημιουργούμε ένα custom Style και ControlTemplate. Συγκεκριμένα:

- Ορίζουμε ότι **όταν λάβει την εστίαση, το background να γίνει κίτρινο**. Αυτό γίνεται, θέτοντας στο **Focused state** το σχετικό Storyboard που περιέχει ένα ColorAnimation που κάνει animate το background color σε Yellow σε χρόνο 0.2 sec.

- Στη δομή του ControlTemplate, **αλλάζουμε θέση στα up-down buttons**. Το up-button βρίσκεται πάνω δεξιά από το TextBox ενώ το down-button βρίσκεται από κάτω δεξιά (3 γραμμές). Για να επιτευχθεί αυτό αλλάζει λίγο η διάταξη του Grid container, που τώρα περιέχει 3 RowDefinintions. Η λειτουργικότητα του control παραμένει η ίδια.

Η εικόνα του custom ByteUpDown σε κανονική κατάσταση - **Normal state**:

Η εικόνα του custom ByteUpDown όταν έχει την εστίαση - **Focused state**:

Ο XAML κώδικας για το custom Style και ControlTemplate του παραπάνω ByteUpDown είναι ο εξής:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownCont
                                                                              rols"

<Style TargetType="{x:Type zeus:ByteUpDown}" >

    <Setter Property="Focusable" Value="True"/>
    <Setter Property="BorderBrush" Value="Black" />
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="Background"  Value="Transparent" />
    <Setter Property="Width" Value ="120"/>
    <Setter Property="Height" Value="23"/>

    <Setter Property="Template">

        <Setter.Value>

            <ControlTemplate TargetType="{x:Type zeus:ByteUpDown}">

                <ControlTemplate.Resources >
                    <zeus:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                </ControlTemplate.Resources>
```

```
        <!-- Root element -->
        <Border Name="Border"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="{TemplateBinding BorderThickness}">

            <VisualStateManager.VisualStateGroups >

                <VisualStateGroup Name="CommonStates">

                    <VisualState Name="Normal"/>

                    <VisualState Name="Disabled">

                        <Storyboard>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Background.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
 Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Foreground.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ObjectAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                            Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                <DiscreteObjectKeyFrame KeyTime="0"
                     Value="{Binding Source={x:Static FontWeights.Normal}}" />
                            </ObjectAnimationUsingKeyFrames>

                        </Storyboard>

                    </VisualState>


                </VisualStateGroup>

                <VisualStateGroup Name="FocusStates">

                   <VisualState Name="Focused" >

                        <Storyboard >

                            ColorAnimation Storyboard.TargetName="PART_Value"
                                 Storyboard.TargetProperty="Background.Color"
                                           To="Yellow" Duration="0:0:0.2"/>

                        </Storyboard>

                    </VisualState>

                    <VisualState Name="Unfocused"/>

                </VisualStateGroup>
```

```
                    </VisualStateManager.VisualStateGroups>

                    <!-- Content -->
                   <Grid Name="mainGrid" Background="Transparent"
                                         Margin="{TemplateBinding Padding}">

                            <Grid.RowDefinitions >
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                            </Grid.RowDefinitions>

                            <TextBox Name="PART_Value" Grid.Row="1"
              HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                  VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                               Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                           </TextBox>

                           <Viewbox Stretch="Uniform" Grid.Row="0" Height="12"
                                    HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementUp"
                                              Focusable="False"  >
                                     <RepeatButton.Content>
                                           <Path Fill="Black" Stroke="Green"
                                                 Data="M 1,10 10,1 20,10 Z" />
                                      </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                           <Viewbox  Stretch="Uniform" Grid.Row="2" Height="12"
                                     HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                     <RepeatButton.Content>
                                          <Path Fill="Black" Stroke="Green"
                                                Data="M 10,10 1,1 20,1 Z" />
                                     </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

ShortUpDown

Ένα **UpDown control** που επιτρέπει την **είσοδο μόνο αριθμητικών δεδομένων, τύπου Short**. Η τιμή απαιτείται (required).

Σύνταξη:

VB:

```
<TemplateVisualState(Name:="Normal", GroupName:="CommonStates"),
TemplateVisualState(Name:="MouseOver", GroupName:="CommonStates"),
TemplateVisualState(Name:="Disabled", GroupName:="CommonStates"),
TemplateVisualState(Name:="Focused", GroupName:="FocusStates"),
TemplateVisualState(Name:="Unfocused", GroupName:="FocusStates"),
TemplatePart(Name:="PART_Value", Type:=GetType(TextBox)),
TemplatePart(Name:="PART_IncrementUp", Type:=GetType(RepeatButton)),
TemplatePart(Name:="PART_IncrementDown", Type:=GetType(RepeatButton))>
<DefaultProperty("Value")>
Public Class ShortUpDown
             Inherits UpDownBase
```

XAML Object Element Usage:

Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

Χρήση:

```
<zeus:ShortUpDown ... />
```

Παράδειγμα:

Στο παράδειγμα που ακολουθεί, τοποθετούμε ένα **ShortUpDown** μέσα σε ένα **Grid** ενός παραθύρου. Συνδέουμε την **ιδιότητα Value** με την **ιδιότητα Amount1**, τύπου **Short**, ενός **Customer** αντικειμένου (source object). Η **κλάση Customer** ορίζεται στο project και συνεπώς πρέπει να εισάγουμε και το αντίστοιχο namespace με alias p. Επίσης, ορίζουμε ένα **ErrorTemplate** για το **Validation** (όπου μέσω αυτού θα εμφανιστεί το μήνυμα λάθους που ορίζουμε στην **ιδιότητα ErrorMessageOnIncorrectValueType** ή στην **ιδιότητα ErrorMessageOnValueOutOfRange**). Επιπλέον, ορίζουμε ένα **Style**, στοχευμένο στην κλάση UpDownBase, για την ομοιόμορφη εμφάνιση των UpDown controls στο παράθυρο.

```
<Window x:Class="MainWindow"
```

```
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:p="clr-namespace:UpDownControlsTestProject"
        xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
        mc:Ignorable="d"
        Title="UpDownControls Sample Project"
        Height="350" Width="525" Loaded="Window_Loaded" >

    <Window.Resources>

        <!-- The Validation Error ControlTemplate -->
        <ControlTemplate x:Key="validationErrorTemplate">

            <StackPanel Orientation="Horizontal">

                <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                        Background="#11FF0000"
                        IsHitTestVisible="False" >
                    <AdornedElementPlaceholder x :Name="placeholder"  />
                </Border>

                <Grid  Margin="20,0,0,0"  >
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

        <Style TargetType="{x:Type TextBlock }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="HorizontalAlignment" Value="Right" />
            <Setter Property="Margin" Value="5"/>
        </Style>

        <!-- The style base for the UpDown controls. -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>
```

```
                    <Trigger Property="HasError" Value="True">
                        <Setter Property="Foreground" Value="Red"/>
                        <Setter Property="BorderBrush" Value="Red"/>
                        <Setter Property="BorderThickness" Value="2"/>
                    </Trigger>

                </Style.Triggers>

            </Style>

        <!-- A sample Customer resource object for the Bindings -->
        <p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
                                     Amount1="12"/>


    </Window.Resources>

    <Grid  Margin="10,40,10,10"
           DataContext="{Binding Source={StaticResource customer}}">

        ...

        <TextBlock Text="Amount (Short):" />
        <zeus:ShortUpDown Name="UpDown1" Grid.Column="1"
                          Minimum="1" Maximum="800"
                          Increment="1" Language="el"
                          HorizontalContentAlignment="Right"
                  ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                  ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                   800. Please, retry."
                          Value="{Binding Amount1}"  ValueFormat="N0"
                         ValueChanged="UpDown1_ValueChanged"
                          Style="{StaticResource UpDownStyle}"/>

       ...

    </Grid>


</Window>
```

VB:

```
Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
UpDown1.ErrorTemplate = errTemplate
```

- Για την **κλάση Customer** δείτε το Παράρτημα 1

Μία κανονική κατάσταση του **ShortUpDown** :

Παρακάτω, βλέπουμε την **ένδειξη λάθους** όταν ο χρήστης εισάγει μία **λανθασμένη (ως προς το εύρος) τιμή**. Αν πάει το δείκτη του ποντικιού πάνω στο **θαυμαστικό** θα εμφανιστεί το μήνυμα λάθους "`The number must be between 1 and 800. Please, retry.`"

Ιδιότητες

Όνομα	Περιγραφή
ButtonsWidth	Καθορίζει το **πλάτος**, σε pixels, **των buttons αυξομείωσης του control**. (*Το ύψος είναι σταθερά ορισμένο στο μέσον του ύψους του control)*. (Κληρονομείται από την UpDownBase).
ErrorMessageOnIncorrectValueType	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι τύπου Short. (Κληρονομείται από την UpDownBase).
ErrorMessageOnValueOutOfRange	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι στο αποδεκτό εύρος (Minimum-Maximum). (Κληρονομείται από την UpDownBase).
ErrorTemplate	Καθορίζει ένα **ControlTemplate** που θα εφαρμόζεται **όταν το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
HasError	Αν είναι **True**, τότε **το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
Increment	Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Short. Η default τιμή είναι 1.
IsMinimum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο ελάχιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Minimum. (Κληρονομείται από την UpDownBase).
IsMaximum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο μέγιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Maximum. (Κληρονομείται από την UpDownBase).
Minimum	Καθορίζει το **ελάχιστο όριο** στο control, τύπου Short. Η default τιμή είναι Short.MinValue.
Maximum	Καθορίζει το **μέγιστο όριο** στο control, τύπου Short. Η default τιμή είναι Short.MaxValue.

Value	Η **τρέχουσα τιμή** στο control, τύπου Short. Η default τιμή είναι 0.
ValueFormat	Καθορίζει το **StringFormat** της τιμής του control. (Κληρονομείται από την UpDownBase).

Increment

Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Short.

Σύνταξη:

VB:

```
Public Property Increment As Short
```

Τύπος: System.Short

Προσδιορίζουμε μία θετική τιμή για το βήμα αύξησης της τιμής. Η default τιμή είναι 1.

Dependency Property Information:

Identifier field: `IncrementProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το βήμα αύξησης στο UpDown1 σε 2.

XAML:

```
<zeus:ShortUpDown x:Name="UpDown1" Minimum="1" Maximum="800" Increment="2"
                  HorizontalContentAlignment="Right"
                  Value="{Binding Amount1}"  ValueFormat="N0"
                  ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Increment = 2
```

Minimum

Καθορίζει το **ελάχιστο όριο** στο control, τύπου Short.

Σύνταξη:

VB:

```
Public Property Minimum As Short
```

Τύπος: System.Short

Προσδιορίζουμε μία τιμή, ως το ελάχιστο αποδεκτό όριο. Η default τιμή είναι Short.MinValue.

Dependency Property Information:

Identifier field: `MinimumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το ελάχιστο όριο στο UpDown1 σε 10.

XAML:

```
<zeus:ShortUpDown x:Name="UpDown1" Minimum="10" Maximum="800" Increment="1"
                  HorizontalContentAlignment="Right"
                  Value="{Binding Amount1}"  ValueFormat="N0"
                  ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Minimum = 10
```

Maximum

Καθορίζει το **μέγιστο όριο** στο control, τύπου Short.

Σύνταξη:

VB:

```
Public Property Maximum As Short
```

Τύπος: System.Short

Προσδιορίζουμε μία τιμή, ως το μέγιστο αποδεκτό όριο. Η default τιμή είναι Short.MaxValue.

Dependency Property Information:

Identifier field: `MaximumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το μέγιστο όριο στο UpDown1 σε 900.

XAML:

```
<zeus:ShortUpDown x:Name="UpDown1" Minimum="10" Maximum="900" Increment="1"
                  HorizontalContentAlignment="Right"
                  Value="{Binding Amount1}"  ValueFormat="N0"
                  ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Maximum = 900
```

Value

Η **τρέχουσα τιμή** στο control, τύπου Short.

Σύνταξη:

VB:

```
Public Property Value As Short
```

Τύπος: System.Short

Η τρέχουσα τιμή. Η default τιμή είναι 0.

Dependency Property Information:

Identifier field: `ValueProperty`

Παρατηρήσεις:

Σε μία WPF εφαρμογή, η λογική χρήση της ιδιότητας αυτής είναι μέσω μίας Binding έκφρασης. Με αυτόν τον τρόπο, η τρέχουσα τιμή συνδέεται με μία source πηγή (π.χ. μία public ιδιότητα ενός data object). Η τιμή προέρχεται από το source object και οποιαδήποτε μεταβολή της επιστρέφει στο source object (σε ένα two-way mode, που είναι και το default). Συμπερασματικά, **συνιστάται η χρήση της με Binding έφραση**.

Παράδειγμα:

Στο επόμενο παράδειγμα, η τιμή της **ιδιότητας Value** στο **UpDown1** συνδέεται, **μέσω Binding έκφρασης**, με την **ιδιότητα Amount1** του **source object**. Το τελευταίο (ένα **αντικείμενο Customer**) τίθεται ως πηγή δεδομένων στην **ιδιότητα DataContext** σε ένα container, που εδώ είναι το **Grid** panel. Οι τιμές των ιδιοτήτων του αντικειμένου Customer τίθενται στο τμήμα Window.Resources, αλλά θα μπορούσαν να τεθούν και στο VB κώδικα, σε κάποιον event handler.

```
<Window x:Class="MainWindow"
          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
          xmlns:p="clr-namespace:UpDownControlsTestProject"
          xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
          mc:Ignorable="d"
          Title="UpDownControls Sample Project"
          Height="350" Width="525" Loaded="Window_Loaded" >

        <Window.Resources>
```

```
<!-- The Validation Error ControlTemplate -->
<ControlTemplate x:Key="validationErrorTemplate">

    <StackPanel Orientation="Horizontal">

        <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                Background="#11FF0000"
                IsHitTestVisible="False" >
            <AdornedElementPlaceholder x :Name="placeholder"  />
        </Border>

        <Grid  Margin="20,0,0,0"  >
            <Ellipse  Width="20" Height="20"  Fill="Red"  />
            <TextBlock Foreground="White" FontSize="14" Text="!"
                       FontWeight="ExtraBold"
                       VerticalAlignment="Center"
                       HorizontalAlignment="Center" />
            <Grid.ToolTip>
                <TextBlock Text="{Binding Path=/ErrorContent}"/>
            </Grid.ToolTip>
        </Grid>

    </StackPanel>

</ControlTemplate>

<Style TargetType="{x:Type TextBlock }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="HorizontalAlignment" Value="Right" />
    <Setter Property="Margin" Value="5"/>
</Style>

<!-- The style base for the UpDown controls. -->
<Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Background" Value="White"/>
    <Setter Property="HorizontalAlignment" Value="Left" />
    <Setter Property="Margin" Value="5,0,0,0"/>

    <Style.Triggers >
        <Trigger Property="IsMinimum" Value="True">
            <Setter Property="Background" Value="Orange" />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="IsMaximum" Value="True">
            <Setter Property="Background" Value="Green"  />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="HasError" Value="True">
            <Setter Property="Foreground" Value="Red"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
        </Trigger>

    </Style.Triggers>

  </Style>

<!-- A sample Customer resource object for the Bindings -->
<p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
```

```
                                        Amount1="12"/>

        </Window.Resources>

        <Grid  Margin="10,40,10,10"
               DataContext="{Binding Source={StaticResource customer}}">

            ...

            <TextBlock Text="Amount (Short):" />
            <zeus:ShortUpDown Name="UpDown1" Grid.Column="1"
                              Minimum="1" Maximum="800"
                              Increment="1" Language="el"
                              HorizontalContentAlignment="Right"
                        ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                        ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                         800. Please, retry."
                              Value="{Binding Amount1}"  ValueFormat="N0"
                              ValueChanged="UpDown1_ValueChanged"
                              Style="{StaticResource UpDownStyle}"/>

           ...

        </Grid>


</Window>
```

VB:

```
Private Sub Window_Loaded(sender As Object, e As RoutedEventArgs)

    Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
    UpDown1.ErrorTemplate = errTemplate

End Sub
```

Βλέπουμε την αρχική κατάσταση του UpDown1. Η τιμή 12 προέρχεται από την ιδιότητα Amount1 του αντικειμένου Customer:

Αν ο χρήστης φτάσει στη μέγιστη τιμή (800) τότε, λόγω του style trigger, το ShortUpDown μορφοποιείται ως εξής:

Παράλληλα, απενεργοποιείται και το up arrow button. Αν ο χρήστης, πληκτρολογήσει απ' ευθείας

μία **τιμή εκτός ορίων**, τότε το control εισέρχεται σε **κατάσταση λάθους**:

Επίσης, αν ο χρήστης, πληκτρολογήσει μία **τιμή λανθασμένου τύπου**, τότε το control πάλι εισέρχεται σε **κατάσταση λάθους**:

Συμβάντα

Όνομα	Περιγραφή
MinimumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.
MaximumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.
ValueChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

MinimumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MinimumChanged As RoutedPropertyChangedEventHandler(Of Short)
```

XAML attribute usage:

```
<zeus:ShortUpDown MinimumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MinimumChanged του UpDown1:

XAML:

```
<zeus:ShortUpDown x:Name="UpDown1" Minimum="0" Maximum="800" Increment="1"
                  HorizontalContentAlignment="Right"
                  Value="{Binding Amount1}"  ValueFormat="N0"
                  MinimumChanged="UpDown1_MinimumChanged" ... />
```

VB:

```
Private Sub UpDown1_MinimumChanged(sender As  Object, _
                                   e As RoutedPropertyChangedEventArgs(Of Short))

    If e IsNot Nothing Then

        MessageBox.Show(String.Format("The new minimum value is {0}", e.NewValue))

    End If

End Sub
```

MaximumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MaximumChanged As RoutedPropertyChangedEventHandler(Of Short)
```

XAML attribute usage:

```
<zeus:ShortUpDown MaximumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MaximumChanged του UpDown1:

XAML:

```
<zeus:ShortUpDown x:Name="UpDown1" Minimum="0" Maximum="800" Increment="1"
                  HorizontalContentAlignment="Right"
                  Value="{Binding Amount1}"  ValueFormat="N0"
                  MaximumChanged="UpDown1_MaximumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MaximumChanged(sender As Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Short))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new maximum value is {0}", e.NewValue))

     End If

End Sub
```

ValueChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event ValueChanged As RoutedPropertyChangedEventHandler(Of Short)
```

XAML attribute usage:

```
<zeus:ShortUpDown ValueChanged ="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν ValueChanged του UpDown1:

XAML:

```
<zeus:ShortUpDown x:Name="UpDown1" Minimum="0" Maximum="800" Increment="1"
                  HorizontalContentAlignment="Right"
                  Value="{Binding Amount1}"  ValueFormat="N0"
                  ValueChanged="UpDown1_ValueChanged" ... />
```

VB:

```
 Private Sub UpDown1_ValueChanged(sender As Object, _
                                  e As RoutedPropertyChangedEventArgs(Of Short))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The current value is {0}", e.NewValue))

     End If

End Sub
```

Styles και Templates

Parts και States

Το default ControlTemplate περιλαμβάνει κάποια **part names** και **visual states**. Μπορείτε να τροποποιήσετε το default ControlTemplate ώστε να δώσετε στο control μία μοναδική εμφάνιση. Το default ControlTemplate μπορείτε να το δείτε εδώ ώστε να πάρετε μία ιδέα και με βάση αυτό να δημιουργήσετε ένα δικό σας. Ένα παράδειγμα custom ControlTemplate μπορείτε να δείτε εδώ.

Ο παρακάτω πίνακας εμφανίζει τα **part names** του **ShortUpDown** control:

Part	Τύπος	Περιγραφή
PART_Value	TextBox	Το standard TextBox control.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την αύξηση της τιμής.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την μείωση της τιμής.

Ο παρακάτω πίνακας εμφανίζει τα **visual states** του **ShortUpDown** control:

VisualState	VisualStateGroup	Περιγραφή
Normal	CommonStates	Το default state.
MouseOver	CommonStates	Το state όταν ο δείκτης του ποντικιού είναι πάνω από το control.
Disabled	CommonStates	Το state όταν το control είναι disabled.
Focused	FocusStates	Το state όταν το control έχει το focus.
Unfocused	FocusStates	Το state όταν το control δεν έχει το focus.

Το default ControlTemplate έχει καθορισμένο μόνο το Disabled state. Μπορείτε να δημιουργήσετε ένα custom ControlTemplate για να το αλλάξετε ή/και για να καθορίσετε τα υπόλοιπα.

Το default Style και ControlTemplate

Ο XAML κώδικας για το **default Style** και **ControlTemplate** φαίνεται παρακάτω. Μπορείτε να βασιστείτε σε αυτόν για να δημιουργήσετε μία μικρή ή μεγάλη παραλλαγή του δικού σας custom Style και ControlTemplate:

```
    xmlns:z="clr-namespace:Zeus.WPF.Controls.UpDownControls">

    <Style TargetType="{x:Type z:ShortUpDown}" >

        <Setter Property="Focusable" Value="True"/>
        <Setter Property="BorderBrush" Value="Black" />
        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="Background"  Value="Transparent" />
        <Setter Property="Width" Value ="120"/>
        <Setter Property="Height" Value="23"/>

        <Setter Property="Template">

            <Setter.Value>

                <ControlTemplate TargetType="{x:Type z:ShortUpDown}">

                    <ControlTemplate.Resources >
                        <z:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                    </ControlTemplate.Resources>

                    <!-- Root element -->
                    <Border Name="Border"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">

                        <VisualStateManager.VisualStateGroups >

                            <VisualStateGroup Name="CommonStates">

                                <VisualState Name="Normal"/>

                                <VisualState Name="Disabled">

                                    <Storyboard >

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Background.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                     Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Foreground.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                    Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ObjectAnimationUsingKeyFrames
```

```
                                        Storyboard.TargetName="PART_Value"
                                    Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                        <DiscreteObjectKeyFrame KeyTime="0"
                                 Value="{Binding Source={x:Static FontWeights.Normal}}" />
                                        </ObjectAnimationUsingKeyFrames>

                                    </Storyboard>

                                </VisualState>


                            </VisualStateGroup>

                            <VisualStateGroup Name="FocusStates">

                                <VisualState Name="Focused" />
                                <VisualState Name="Unfocused"/>

                            </VisualStateGroup>

                        </VisualStateManager.VisualStateGroups>

                        <!-- Content -->
                        <Grid Name="mainGrid" Background="Transparent" >

                            <Grid.RowDefinitions >
                                <RowDefinition Height="*" />
                            </Grid.RowDefinitions>

                            <!-- The width of the second Grid column is about for the
                                 RepeatButtons and depends on ButtonsWidth property -->
                            <Grid.ColumnDefinitions >
                                <ColumnDefinition Width="*" />
                                <ColumnDefinition
                                      Width="{Binding RelativeSource={Relative Source
                              Mode=FindAncestor, AncestorType={x:Type z:ShortUpDown}},
Path=ButtonsWidth, Converter={StaticResource doubleToGridLengthConverter}}" />
                            </Grid.ColumnDefinitions>

                             <TextBox Name="PART_Value"
                HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                    VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                                Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                            </TextBox>

                            <Viewbox Stretch="Fill"   Grid.Column="1">

                                <StackPanel Name="stkRepeatButtons"
                                            VerticalAlignment="Center"
                                            HorizontalAlignment="Left"  Focusable="False">

                                    <RepeatButton Name="PART_IncrementUp"
                                                  Focusable="False"  >
                                        <RepeatButton.Content>
                                            <Path Fill="Black" Stroke="Green"
                                                  Data="M 1,10 10,1 20,10 Z" />
                                        </RepeatButton.Content>
                                    </RepeatButton>
```

```
                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                    <RepeatButton.Content>
                                        <Path Fill="Black" Stroke="Green"
                                              Data="M 10,10 1,1 20,1 Z" />
                                    </RepeatButton.Content>
                                </RepeatButton>

                            </StackPanel>

                        </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

Η δεύτερη στήλη του Grid περιέχει τα repeat buttons μέσα σε ένα StackPanel. Το πλάτος της δεύτερης στήλης του Grid προσδιορίζεται με βάση την τιμή της ιδιότητας ButtonsWidth. Για να μετατραπεί η τιμή Double της ιδιότητας ButtonsWidth σε τιμή τύπου GridLength χρησιμοποιούμε μία κλάση μετατροπής τιμής (ValueConverter). Θα τη βρείτε στο Παράρτημα 2.

Παράδειγμα custom Style και ControlTemplate

Στο επόμενο παράδειγμα, δημιουργούμε ένα custom Style και ControlTemplate. Συγκεκριμένα:

- Ορίζουμε ότι **όταν λάβει την εστίαση, το background να γίνει κίτρινο**. Αυτό γίνεται, θέτοντας στο **Focused state** το σχετικό Storyboard που περιέχει ένα ColorAnimation που κάνει animate το background color σε Yellow σε χρόνο 0.2 sec.

- Στη δομή του ControlTemplate, **αλλάζουμε θέση στα up-down buttons**. Το up-button βρίσκεται πάνω δεξιά από το TextBox ενώ το down-button βρίσκεται από κάτω δεξιά (3 γραμμές). Για να επιτευχθεί αυτό αλλάζει λίγο η διάταξη του Grid container, που τώρα περιέχει 3 RowDefinintions. Η λειτουργικότητα του control παραμένει η ίδια.

Η εικόνα του custom ShortUpDown σε κανονική κατάσταση - **Normal state**:

Η εικόνα του custom ShortUpDown όταν έχει την εστίαση - **Focused state**:

Ο XAML κώδικας για το custom Style και ControlTemplate του παραπάνω ShortUpDown είναι ο εξής:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownCont
                                                                              rols"

<Style TargetType="{x:Type zeus:ShortUpDown}" >

    <Setter Property="Focusable" Value="True"/>
    <Setter Property="BorderBrush" Value="Black" />
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="Background"  Value="Transparent" />
    <Setter Property="Width" Value ="120"/>
    <Setter Property="Height" Value="23"/>

    <Setter Property="Template">

        <Setter.Value>

            <ControlTemplate TargetType="{x:Type zeus:ShortUpDown}">

                <ControlTemplate.Resources >
                    <zeus:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                </ControlTemplate.Resources>
```

```
        <!-- Root element -->
        <Border Name="Border"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="{TemplateBinding BorderThickness}">

            <VisualStateManager.VisualStateGroups >

                <VisualStateGroup Name="CommonStates">

                    <VisualState Name="Normal"/>

                    <VisualState Name="Disabled">

                        <Storyboard>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Background.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
 Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Foreground.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ObjectAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                            Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                <DiscreteObjectKeyFrame KeyTime="0"
                     Value="{Binding Source={x:Static FontWeights.Normal}}" />
                            </ObjectAnimationUsingKeyFrames>

                        </Storyboard>

                    </VisualState>


                </VisualStateGroup>

                <VisualStateGroup Name="FocusStates">

                   <VisualState Name="Focused" >

                        <Storyboard >

                            ColorAnimation Storyboard.TargetName="PART_Value"
                                 Storyboard.TargetProperty="Background.Color"
                                           To="Yellow" Duration="0:0:0.2"/>

                        </Storyboard>

                    </VisualState>

                    <VisualState Name="Unfocused"/>

                </VisualStateGroup>
```

```
                    </VisualStateManager.VisualStateGroups>

                    <!-- Content -->
                   <Grid Name="mainGrid" Background="Transparent"
                                         Margin="{TemplateBinding Padding}">

                            <Grid.RowDefinitions >
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                            </Grid.RowDefinitions>

                            <TextBox Name="PART_Value" Grid.Row="1"
              HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                  VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                               Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                           </TextBox>

                           <Viewbox Stretch="Uniform" Grid.Row="0" Height="12"
                                    HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementUp"
                                              Focusable="False"  >
                                     <RepeatButton.Content>
                                           <Path Fill="Black" Stroke="Green"
                                                 Data="M 1,10 10,1 20,10 Z" />
                                      </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                           <Viewbox  Stretch="Uniform" Grid.Row="2" Height="12"
                                     HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                     <RepeatButton.Content>
                                          <Path Fill="Black" Stroke="Green"
                                                Data="M 10,10 1,1 20,1 Z" />
                                     </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

IntegerUpDown

Ένα **UpDown control** που επιτρέπει την **είσοδο μόνο αριθμητικών δεδομένων, τύπου Integer**. Η τιμή απαιτείται (required).

Σύνταξη:

VB:

```
<TemplateVisualState(Name:="Normal", GroupName:="CommonStates"),
TemplateVisualState(Name:="MouseOver", GroupName:="CommonStates"),
TemplateVisualState(Name:="Disabled", GroupName:="CommonStates"),
TemplateVisualState(Name:="Focused", GroupName:="FocusStates"),
TemplateVisualState(Name:="Unfocused", GroupName:="FocusStates"),
TemplatePart(Name:="PART_Value", Type:=GetType(TextBox)),
TemplatePart(Name:="PART_IncrementUp", Type:=GetType(RepeatButton)),
TemplatePart(Name:="PART_IncrementDown", Type:=GetType(RepeatButton))>
<DefaultProperty("Value")>
Public Class IntegerUpDown
             Inherits UpDownBase
```

XAML Object Element Usage:

Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

Χρήση:

```
<zeus:IntegerUpDown ... />
```

Παράδειγμα:

Στο παράδειγμα που ακολουθεί, τοποθετούμε ένα **IntegerDown** μέσα σε ένα **Grid** ενός παραθύρου. Συνδέουμε την **ιδιότητα Value** με την **ιδιότητα Amount1**, τύπου **Integer**, ενός **Customer** αντικειμένου (source object). Η **κλάση Customer** ορίζεται στο project και συνεπώς πρέπει να εισάγουμε και το αντίστοιχο namespace με alias p. Επίσης, ορίζουμε ένα **ErrorTemplate** για το **Validation** (όπου μέσω αυτού θα εμφανιστεί το μήνυμα λάθους που ορίζουμε στην **ιδιότητα ErrorMessageOnIncorrectValueType** ή στην **ιδιότητα ErrorMessageOnValueOutOfRange**). Επιπλέον, ορίζουμε ένα **Style**, στοχευμένο στην κλάση UpDownBase, για την ομοιόμορφη εμφάνιση των UpDown controls στο παράθυρο.

```
<Window x:Class="MainWindow"
         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:p="clr-namespace:UpDownControlsTestProject"
        xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
        mc:Ignorable="d"
        Title="UpDownControls Sample Project"
        Height="350" Width="525" Loaded="Window_Loaded" >

    <Window.Resources>

        <!-- The Validation Error ControlTemplate -->
        <ControlTemplate x:Key="validationErrorTemplate">

            <StackPanel Orientation="Horizontal">

                <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                        Background="#11FF0000"
                        IsHitTestVisible="False" >
                    <AdornedElementPlaceholder x :Name="placeholder"  />
                </Border>

                <Grid  Margin="20,0,0,0"  >
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

        <Style TargetType="{x:Type TextBlock }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="HorizontalAlignment" Value="Right" />
            <Setter Property="Margin" Value="5"/>
        </Style>

        <!-- The style base for the UpDown controls. -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>
```

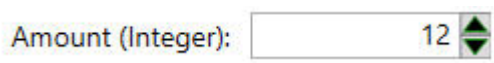
```
                    <Trigger Property="HasError" Value="True">
                        <Setter Property="Foreground" Value="Red"/>
                        <Setter Property="BorderBrush" Value="Red"/>
                        <Setter Property="BorderThickness" Value="2"/>
                    </Trigger>

                </Style.Triggers>

            </Style>

        <!-- A sample Customer resource object for the Bindings -->
        <p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
                                     Amount1="12"/>


    </Window.Resources>

    <Grid  Margin="10,40,10,10"
           DataContext="{Binding Source={StaticResource customer}}">

        ...

        <TextBlock Text="Amount (Integer):" />
        <zeus:IntegerUpDown Name="UpDown1" Grid.Column="1"
                            Minimum="1" Maximum="800"
                            Increment="1" Language="el"
                            HorizontalContentAlignment="Right"
                 ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                 ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                  800. Please, retry."
                            Value="{Binding Amount1}"  ValueFormat="N0"
                           ValueChanged="UpDown1_ValueChanged"
                            Style="{StaticResource UpDownStyle}"/>

       ...

    </Grid>


</Window>
```

VB:

```
Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
UpDown1.ErrorTemplate = errTemplate
```

- Για την **κλάση Customer** δείτε το Παράρτημα 1

Μία κανονική κατάσταση του **IntegerUpDown** :



Παρακάτω, βλέπουμε την **ένδειξη λάθους** όταν ο χρήστης εισάγει μία **λανθασμένη (ως προς το εύρος) τιμή**. Αν πάει το δείκτη του ποντικιού πάνω στο **θαυμαστικό** θα εμφανιστεί το μήνυμα λάθους "`The number must be between 1 and 800. Please, retry.`"

Ιδιότητες

Όνομα	Περιγραφή
ButtonsWidth	Καθορίζει το **πλάτος**, σε pixels, **των buttons αυξομείωσης του control**. (*Το ύψος είναι σταθερά ορισμένο στο μέσον του ύψους του control)*. (Κληρονομείται από την UpDownBase).
ErrorMessageOnIncorrectValueType	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι τύπου Integer. (Κληρονομείται από την UpDownBase).
ErrorMessageOnValueOutOfRange	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι στο αποδεκτό εύρος (Minimum-Maximum). (Κληρονομείται από την UpDownBase).
ErrorTemplate	Καθορίζει ένα **ControlTemplate** που θα εφαρμόζεται **όταν το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
HasError	Αν είναι **True**, τότε **το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
Increment	Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Integer. Η default τιμή είναι 1.
IsMinimum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο ελάχιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Minimum. (Κληρονομείται από την UpDownBase).
IsMaximum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο μέγιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Maximum. (Κληρονομείται από την UpDownBase).
Minimum	Καθορίζει το **ελάχιστο όριο** στο control, τύπου Integer. Η default τιμή είναι Integer.MinValue.

Maximum	Καθορίζει το **μέγιστο όριο** στο control, τύπου Integer. Η default τιμή είναι Integer.MaxValue.
Value	Η **τρέχουσα τιμή** στο control, τύπου Integer. Η default τιμή είναι 0.
ValueFormat	Καθορίζει το **StringFormat** της τιμής του control. (Κληρονομείται από την UpDownBase).

Increment

Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Integer.

Σύνταξη:

VB:

```
Public Property Increment As Integer
```

Τύπος: System.Integer

Προσδιορίζουμε μία θετική τιμή για το βήμα αύξησης της τιμής. Η default τιμή είναι 1.

Dependency Property Information:

Identifier field: `IncrementProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το βήμα αύξησης στο UpDown1 σε 2.

XAML:

```
<zeus:IntegerUpDown x:Name="UpDown1" Minimum="1" Maximum="800" Increment="2"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="N0"
                    ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Increment = 2
```

Minimum

Καθορίζει το **ελάχιστο όριο** στο control, τύπου Integer.

Σύνταξη:

VB:

```
Public Property Minimum As Integer
```

Τύπος: System.Integer

Προσδιορίζουμε μία τιμή, ως το ελάχιστο αποδεκτό όριο. Η default τιμή είναι Integer.MinValue.

Dependency Property Information:

Identifier field: `MinimumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το ελάχιστο όριο στο UpDown1 σε 10.

XAML:

```
<zeus:IntegerUpDown x:Name="UpDown1" Minimum="10" Maximum="800" Increment="1"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="N0"
                    ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Minimum = 10
```

Maximum

Καθορίζει το **μέγιστο όριο** στο control, τύπου Integer.

Σύνταξη:

VB:

```
Public Property Maximum As Integer
```

Τύπος: System.Integer

Προσδιορίζουμε μία τιμή, ως το μέγιστο αποδεκτό όριο. Η default τιμή είναι Integer.MaxValue.

Dependency Property Information:

Identifier field: `MaximumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το μέγιστο όριο στο UpDown1 σε 900.

XAML:

```
<zeus:IntegerUpDown x:Name="UpDown1" Minimum="10" Maximum="900" Increment="1"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="N0"
                    ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Maximum = 900
```

Value

Η **τρέχουσα τιμή** στο control, τύπου Integer.

Σύνταξη:

VB:

```
Public Property Value As Integer
```

Τύπος: System.Integer

Η τρέχουσα τιμή. Η default τιμή είναι 0.

Dependency Property Information:

Identifier field: `ValueProperty`

Παρατηρήσεις:

Σε μία WPF εφαρμογή, η λογική χρήση της ιδιότητας αυτής είναι μέσω μίας Binding έκφρασης. Με αυτόν τον τρόπο, η τρέχουσα τιμή συνδέεται με μία source πηγή (π.χ. μία public ιδιότητα ενός data object). Η τιμή προέρχεται από το source object και οποιαδήποτε μεταβολή της επιστρέφει στο source object (σε ένα two-way mode, που είναι και το default). Συμπερασματικά, **συνιστάται η χρήση της με Binding έφραση**.

Παράδειγμα:

Στο επόμενο παράδειγμα, η τιμή της **ιδιότητας Value** στο **UpDown1** συνδέεται, **μέσω Binding έκφρασης**, με την **ιδιότητα Amount1** του **source object**. Το τελευταίο (ένα **αντικείμενο Customer**) τίθεται ως πηγή δεδομένων στην **ιδιότητα DataContext** σε ένα container, που εδώ είναι το **Grid** panel. Οι τιμές των ιδιοτήτων του αντικειμένου Customer τίθενται στο τμήμα Window.Resources, αλλά θα μπορούσαν να τεθούν και στο VB κώδικα, σε κάποιον event handler.

```
<Window x:Class="MainWindow"
          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
          xmlns:p="clr-namespace:UpDownControlsTestProject"
          xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
          mc:Ignorable="d"
          Title="UpDownControls Sample Project"
          Height="350" Width="525" Loaded="Window_Loaded" >

        <Window.Resources>
```

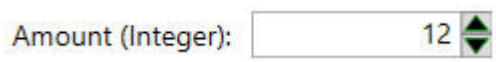
```
<!-- The Validation Error ControlTemplate -->
<ControlTemplate x:Key="validationErrorTemplate">

    <StackPanel Orientation="Horizontal">

        <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                Background="#11FF0000"
                IsHitTestVisible="False" >
            <AdornedElementPlaceholder x :Name="placeholder"  />
        </Border>

        <Grid  Margin="20,0,0,0"  >
            <Ellipse  Width="20" Height="20"  Fill="Red"  />
            <TextBlock Foreground="White" FontSize="14" Text="!"
                       FontWeight="ExtraBold"
                       VerticalAlignment="Center"
                       HorizontalAlignment="Center" />
            <Grid.ToolTip>
                <TextBlock Text="{Binding Path=/ErrorContent}"/>
            </Grid.ToolTip>
        </Grid>

    </StackPanel>

</ControlTemplate>

<Style TargetType="{x:Type TextBlock }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="HorizontalAlignment" Value="Right" />
    <Setter Property="Margin" Value="5"/>
</Style>

<!-- The style base for the UpDown controls. -->
<Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Background" Value="White"/>
    <Setter Property="HorizontalAlignment" Value="Left" />
    <Setter Property="Margin" Value="5,0,0,0"/>

    <Style.Triggers >
        <Trigger Property="IsMinimum" Value="True">
            <Setter Property="Background" Value="Orange" />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="IsMaximum" Value="True">
            <Setter Property="Background" Value="Green"  />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="HasError" Value="True">
            <Setter Property="Foreground" Value="Red"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
        </Trigger>

    </Style.Triggers>

  </Style>

<!-- A sample Customer resource object for the Bindings -->
<p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
```

```
                                        Amount1="12"/>

        </Window.Resources>

        <Grid  Margin="10,40,10,10"
               DataContext="{Binding Source={StaticResource customer}}">

            ...

            <TextBlock Text="Amount (Integer):" />
            <zeus:IntegerUpDown Name="UpDown1" Grid.Column="1"
                                Minimum="1" Maximum="800"
                                Increment="1" Language="el"
                                HorizontalContentAlignment="Right"
                        ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                        ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                         800. Please, retry."
                                 Value="{Binding Amount1}"  ValueFormat="N0"
                                ValueChanged="UpDown1_ValueChanged"
                                 Style="{StaticResource UpDownStyle}"/>

           ...

        </Grid>


</Window>
```

VB:

```
Private Sub Window_Loaded(sender As Object, e As RoutedEventArgs)

    Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
    UpDown1.ErrorTemplate = errTemplate

End Sub
```

Βλέπουμε την αρχική κατάσταση του UpDown1. Η τιμή 12 προέρχεται από την ιδιότητα Amount1 του αντικειμένου Customer:



Αν ο χρήστης φτάσει στη μέγιστη τιμή (800) τότε, λόγω του style trigger, το IntegerUpDown μορφοποιείται ως εξής:

Παράλληλα, απενεργοποιείται και το up arrow button. Αν ο χρήστης, πληκτρολογήσει απ' ευθείας

μία **τιμή εκτός ορίων**, τότε το control εισέρχεται σε **κατάσταση λάθους**:

Επίσης, αν ο χρήστης, πληκτρολογήσει μία **τιμή λανθασμένου τύπου**, τότε το control πάλι εισέρχεται σε **κατάσταση λάθους**:

Συμβάντα

Όνομα	Περιγραφή
MinimumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.
MaximumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.
ValueChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

MinimumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MinimumChanged As RoutedPropertyChangedEventHandler(Of Integer)
```

XAML attribute usage:

```
<zeus:IntegerUpDown MinimumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MinimumChanged του UpDown1:

XAML:

```
<zeus:IntegerUpDown x:Name="UpDown1" Minimum="0" Maximum="800" Increment="1"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="N0"
                    MinimumChanged="UpDown1_MinimumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MinimumChanged(sender As  Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Integer))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new minimum value is {0}", e.NewValue))

     End If

End Sub
```

MaximumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MaximumChanged As RoutedPropertyChangedEventHandler(Of Integer)
```

XAML attribute usage:

```
<zeus:IntegerUpDown MaximumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MaximumChanged του UpDown1:

XAML:

```
<zeus:IntegerUpDown x:Name="UpDown1" Minimum="0" Maximum="800" Increment="1"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="N0"
                    MaximumChanged="UpDown1_MaximumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MaximumChanged(sender As Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Integer))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new maximum value is {0}", e.NewValue))

     End If

End Sub
```

ValueChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event ValueChanged As RoutedPropertyChangedEventHandler(Of Integer)
```

XAML attribute usage:

```
<zeus: IntegerUpDown ValueChanged ="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν ValueChanged του UpDown1:

XAML:

```
<zeus:IntegerUpDown x:Name="UpDown1" Minimum="0" Maximum="800" Increment="1"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="N0"
                    ValueChanged="UpDown1_ValueChanged" ... />
```

VB:

```
 Private Sub UpDown1_ValueChanged(sender As Object, _
                                  e As RoutedPropertyChangedEventArgs(Of Integer))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The current value is {0}", e.NewValue))

     End If

End Sub
```

Styles και Templates

- Parts και States
- Το default Style και ControlTemplate
- Παράδειγμα custom Style και ControlTemplate

Parts και States

Το default ControlTemplate περιλαμβάνει κάποια **part names** και **visual states**. Μπορείτε να τροποποιήσετε το default ControlTemplate ώστε να δώσετε στο control μία μοναδική εμφάνιση. Το default ControlTemplate μπορείτε να το δείτε εδώ ώστε να πάρετε μία ιδέα και με βάση αυτό να δημιουργήσετε ένα δικό σας. Ένα παράδειγμα custom ControlTemplate μπορείτε να δείτε εδώ.

Ο παρακάτω πίνακας εμφανίζει τα **part names** του **IntegerUpDown** control:

Part	Τύπος	Περιγραφή
PART_Value	TextBox	Το standard TextBox control.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την αύξηση της τιμής.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την μείωση της τιμής.

Ο παρακάτω πίνακας εμφανίζει τα **visual states** του **IntegerUpDown** control:

VisualState	VisualStateGroup	Περιγραφή
Normal	CommonStates	Το default state.
MouseOver	CommonStates	Το state όταν ο δείκτης του ποντικιού είναι πάνω από το control.
Disabled	CommonStates	Το state όταν το control είναι disabled.
Focused	FocusStates	Το state όταν το control έχει το focus.
Unfocused	FocusStates	Το state όταν το control δεν έχει το focus.

Το default ControlTemplate έχει καθορισμένο μόνο το Disabled state. Μπορείτε να δημιουργήσετε ένα custom ControlTemplate για να το αλλάξετε ή/και για να καθορίσετε τα υπόλοιπα.

Το default Style και ControlTemplate

Ο XAML κώδικας για το **default Style** και **ControlTemplate** φαίνεται παρακάτω. Μπορείτε να βασιστείτε σε αυτόν για να δημιουργήσετε μία μικρή ή μεγάλη παραλλαγή του δικού σας custom Style και ControlTemplate:

```
    xmlns:z="clr-namespace:Zeus.WPF.Controls.UpDownControls">

    <Style TargetType="{x:Type z:IntegerUpDown}" >

        <Setter Property="Focusable" Value="True"/>
        <Setter Property="BorderBrush" Value="Black" />
        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="Background"  Value="Transparent" />
        <Setter Property="Width" Value ="120"/>
        <Setter Property="Height" Value="23"/>

        <Setter Property="Template">

            <Setter.Value>

                <ControlTemplate TargetType="{x:Type z:IntegerUpDown}">

                    <ControlTemplate.Resources >
                        <z:DoubleToGridLengthConverter
                                            x:Key="doubleToGridLengthConverter"/>
                    </ControlTemplate.Resources>

                    <!-- Root element -->
                    <Border Name="Border"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">

                        <VisualStateManager.VisualStateGroups >

                            <VisualStateGroup Name="CommonStates">

                                <VisualState Name="Normal"/>

                                <VisualState Name="Disabled">

                                    <Storyboard >

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Background.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                     Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Foreground.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                    Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ObjectAnimationUsingKeyFrames
```

```
                                        Storyboard.TargetName="PART_Value"
                                    Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                        <DiscreteObjectKeyFrame KeyTime="0"
                                 Value="{Binding Source={x:Static FontWeights.Normal}}" />
                                        </ObjectAnimationUsingKeyFrames>

                                    </Storyboard>

                                </VisualState>


                            </VisualStateGroup>

                            <VisualStateGroup Name="FocusStates">

                                <VisualState Name="Focused" />
                                <VisualState Name="Unfocused"/>

                            </VisualStateGroup>

                        </VisualStateManager.VisualStateGroups>

                        <!-- Content -->
                        <Grid Name="mainGrid" Background="Transparent" >

                            <Grid.RowDefinitions >
                                <RowDefinition Height="*" />
                            </Grid.RowDefinitions>

                            <!-- The width of the second Grid column is about for the
                                 RepeatButtons and depends on ButtonsWidth property -->
                            <Grid.ColumnDefinitions >
                                <ColumnDefinition Width="*" />
                                <ColumnDefinition
                                      Width="{Binding RelativeSource={Relative Source
                              Mode=FindAncestor, AncestorType={x:Type z:IntegerUpDown}},
Path=ButtonsWidth, Converter={StaticResource doubleToGridLengthConverter}}" />
                            </Grid.ColumnDefinitions>

                             <TextBox Name="PART_Value"
                HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                    VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                                Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                            </TextBox>

                            <Viewbox Stretch="Fill"   Grid.Column="1">

                                <StackPanel Name="stkRepeatButtons"
                                            VerticalAlignment="Center"
                                            HorizontalAlignment="Left"  Focusable="False">

                                    <RepeatButton Name="PART_IncrementUp"
                                                  Focusable="False"  >
                                        <RepeatButton.Content>
                                            <Path Fill="Black" Stroke="Green"
                                                  Data="M 1,10 10,1 20,10 Z" />
                                        </RepeatButton.Content>
                                    </RepeatButton>
```

```
                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                    <RepeatButton.Content>
                                        <Path Fill="Black" Stroke="Green"
                                              Data="M 10,10 1,1 20,1 Z" />
                                    </RepeatButton.Content>
                                </RepeatButton>

                            </StackPanel>

                        </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

Η δεύτερη στήλη του Grid περιέχει τα repeat buttons μέσα σε ένα StackPanel. Το πλάτος της δεύτερης στήλης του Grid προσδιορίζεται με βάση την τιμή της ιδιότητας ButtonsWidth. Για να μετατραπεί η τιμή Double της ιδιότητας ButtonsWidth σε τιμή τύπου GridLength χρησιμοποιούμε μία κλάση μετατροπής τιμής (ValueConverter). Θα τη βρείτε στο Παράρτημα 2.

Παράδειγμα custom Style και ControlTemplate

Στο επόμενο παράδειγμα, δημιουργούμε ένα custom Style και ControlTemplate. Συγκεκριμένα:

- Ορίζουμε ότι **όταν λάβει την εστίαση, το background να γίνει κίτρινο**. Αυτό γίνεται, θέτοντας στο **Focused state** το σχετικό Storyboard που περιέχει ένα ColorAnimation που κάνει animate το background color σε Yellow σε χρόνο 0.2 sec.

- Στη δομή του ControlTemplate, **αλλάζουμε θέση στα up-down buttons**. Το up-button βρίσκεται πάνω δεξιά από το TextBox ενώ το down-button βρίσκεται από κάτω δεξιά (3 γραμμές). Για να επιτευχθεί αυτό αλλάζει λίγο η διάταξη του Grid container, που τώρα περιέχει 3 RowDefinintions. Η λειτουργικότητα του control παραμένει η ίδια.

Η εικόνα του custom IntegerUpDown σε κανονική κατάσταση - **Normal state**:

Η εικόνα του custom IntegerUpDown όταν έχει την εστίαση - **Focused state**:

Ο XAML κώδικας για το custom Style και ControlTemplate του παραπάνω IntegerUpDown είναι ο εξής:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownCont
                                                                              rols"

<Style TargetType="{x:Type zeus:IntegerUpDown}" >

    <Setter Property="Focusable" Value="True"/>
    <Setter Property="BorderBrush" Value="Black" />
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="Background"  Value="Transparent" />
    <Setter Property="Width" Value ="120"/>
    <Setter Property="Height" Value="23"/>

    <Setter Property="Template">

        <Setter.Value>

            <ControlTemplate TargetType="{x:Type zeus:IntegerUpDown}">

                <ControlTemplate.Resources >
                    <zeus:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                </ControlTemplate.Resources>
```

```
        <!-- Root element -->
        <Border Name="Border"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="{TemplateBinding BorderThickness}">

            <VisualStateManager.VisualStateGroups >

                <VisualStateGroup Name="CommonStates">

                    <VisualState Name="Normal"/>

                    <VisualState Name="Disabled">

                        <Storyboard>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Background.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
 Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Foreground.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ObjectAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                            Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                <DiscreteObjectKeyFrame KeyTime="0"
                     Value="{Binding Source={x:Static FontWeights.Normal}}" />
                            </ObjectAnimationUsingKeyFrames>

                        </Storyboard>

                    </VisualState>


                </VisualStateGroup>

                <VisualStateGroup Name="FocusStates">

                   <VisualState Name="Focused" >

                        <Storyboard >

                            ColorAnimation Storyboard.TargetName="PART_Value"
                                 Storyboard.TargetProperty="Background.Color"
                                           To="Yellow" Duration="0:0:0.2"/>

                        </Storyboard>

                    </VisualState>

                    <VisualState Name="Unfocused"/>

                </VisualStateGroup>
```

```
                    </VisualStateManager.VisualStateGroups>

                    <!-- Content -->
                   <Grid Name="mainGrid" Background="Transparent"
                                         Margin="{TemplateBinding Padding}">

                            <Grid.RowDefinitions >
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                            </Grid.RowDefinitions>

                            <TextBox Name="PART_Value" Grid.Row="1"
              HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                  VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                               Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                           </TextBox>

                           <Viewbox Stretch="Uniform" Grid.Row="0" Height="12"
                                    HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementUp"
                                              Focusable="False"  >
                                     <RepeatButton.Content>
                                           <Path Fill="Black" Stroke="Green"
                                                 Data="M 1,10 10,1 20,10 Z" />
                                      </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                           <Viewbox  Stretch="Uniform" Grid.Row="2" Height="12"
                                     HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                     <RepeatButton.Content>
                                          <Path Fill="Black" Stroke="Green"
                                                Data="M 10,10 1,1 20,1 Z" />
                                     </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

LongUpDown

Ένα **UpDown control** που επιτρέπει την **είσοδο μόνο αριθμητικών δεδομένων, τύπου Long**. Η τιμή απαιτείται (required).

Σύνταξη:

VB:

```
<TemplateVisualState(Name:="Normal", GroupName:="CommonStates"),
TemplateVisualState(Name:="MouseOver", GroupName:="CommonStates"),
TemplateVisualState(Name:="Disabled", GroupName:="CommonStates"),
TemplateVisualState(Name:="Focused", GroupName:="FocusStates"),
TemplateVisualState(Name:="Unfocused", GroupName:="FocusStates"),
TemplatePart(Name:="PART_Value", Type:=GetType(TextBox)),
TemplatePart(Name:="PART_IncrementUp", Type:=GetType(RepeatButton)),
TemplatePart(Name:="PART_IncrementDown", Type:=GetType(RepeatButton))>
<DefaultProperty("Value")>
Public Class LongUpDown
             Inherits UpDownBase
```

XAML Object Element Usage:

Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

Χρήση:

```
<zeus:LongUpDown ... />
```

Παράδειγμα:

Στο παράδειγμα που ακολουθεί, τοποθετούμε ένα **LongUpDown** μέσα σε ένα **Grid** ενός παραθύρου. Συνδέουμε την **ιδιότητα Value** με την **ιδιότητα Amount1**, τύπου **Long**, ενός **Customer** αντικειμένου (source object). Η **κλάση Customer** ορίζεται στο project και συνεπώς πρέπει να εισάγουμε και το αντίστοιχο namespace με alias p. Επίσης, ορίζουμε ένα **ErrorTemplate** για το **Validation** (όπου μέσω αυτού θα εμφανιστεί το μήνυμα λάθους που ορίζουμε στην **ιδιότητα ErrorMessageOnIncorrectValueType** ή στην **ιδιότητα ErrorMessageOnValueOutOfRange**). Επιπλέον, ορίζουμε ένα **Style**, στοχευμένο στην κλάση UpDownBase, για την ομοιόμορφη εμφάνιση των UpDown controls στο παράθυρο.

```
<Window x:Class="MainWindow"
         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:p="clr-namespace:UpDownControlsTestProject"
        xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
        mc:Ignorable="d"
        Title="UpDownControls Sample Project"
        Height="350" Width="525" Loaded="Window_Loaded" >

    <Window.Resources>

        <!-- The Validation Error ControlTemplate -->
        <ControlTemplate x:Key="validationErrorTemplate">

            <StackPanel Orientation="Horizontal">

                <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                        Background="#11FF0000"
                        IsHitTestVisible="False" >
                    <AdornedElementPlaceholder x :Name="placeholder"  />
                </Border>

                <Grid  Margin="20,0,0,0"  >
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

        <Style TargetType="{x:Type TextBlock }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="HorizontalAlignment" Value="Right" />
            <Setter Property="Margin" Value="5"/>
        </Style>

        <!-- The style base for the UpDown controls. -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>
```

```
                    <Trigger Property="HasError" Value="True">
                        <Setter Property="Foreground" Value="Red"/>
                        <Setter Property="BorderBrush" Value="Red"/>
                        <Setter Property="BorderThickness" Value="2"/>
                    </Trigger>

                </Style.Triggers>

            </Style>

            <!-- A sample Customer resource object for the Bindings -->
            <p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
                                         Amount1="12"/>


        </Window.Resources>

        <Grid  Margin="10,40,10,10"
               DataContext="{Binding Source={StaticResource customer}}">

            ...

            <TextBlock Text="Amount (Long):" />
            <zeus:LongUpDown Name="UpDown1" Grid.Column="1"
                             Minimum="1" Maximum="800"
                             Increment="1" Language="el"
                             HorizontalContentAlignment="Right"
                 ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                 ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                  800. Please, retry."
                              Value="{Binding Amount1}"  ValueFormat="N0"
                             ValueChanged="UpDown1_ValueChanged"
                              Style="{StaticResource UpDownStyle}"/>

           ...

        </Grid>


    </Window>
```

VB:

```
Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
UpDown1.ErrorTemplate = errTemplate
```

- Για την **κλάση Customer** δείτε το Παράρτημα 1.

Μία κανονική κατάσταση του **LongUpDown** :

Παρακάτω, βλέπουμε την **ένδειξη λάθους** όταν ο χρήστης εισάγει μία **λανθασμένη (ως προς το εύρος) τιμή**. Αν πάει το δείκτη του ποντικιού πάνω στο **θαυμαστικό** θα εμφανιστεί το μήνυμα λάθους "`The number must be between 1 and 800. Please, retry.`"

Ιδιότητες

Όνομα	Περιγραφή
ButtonsWidth	Καθορίζει το **πλάτος**, σε pixels, **των buttons αυξομείωσης του control**. (*Το ύψος είναι σταθερά ορισμένο στο μέσον του ύψους του control)*. (Κληρονομείται από την UpDownBase).
ErrorMessageOnIncorrectValueType	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι τύπου Long. (Κληρονομείται από την UpDownBase).
ErrorMessageOnValueOutOfRange	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι στο αποδεκτό εύρος (Minimum-Maximum). (Κληρονομείται από την UpDownBase).
ErrorTemplate	Καθορίζει ένα **ControlTemplate** που θα εφαρμόζεται **όταν το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
HasError	Αν είναι **True**, τότε **το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
Increment	Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Long. Η default τιμή είναι 1.
IsMinimum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο ελάχιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Minimum. (Κληρονομείται από την UpDownBase).
IsMaximum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο μέγιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Maximum. (Κληρονομείται από την UpDownBase).
Minimum	Καθορίζει το **ελάχιστο όριο** στο control, τύπου Long. Η default τιμή είναι Long.MinValue.

Maximum	Καθορίζει το **μέγιστο όριο** στο control, τύπου Long. Η default τιμή είναι Long.MaxValue.
Value	Η **τρέχουσα τιμή** στο control, τύπου Long. Η default τιμή είναι 0.
ValueFormat	Καθορίζει το **StringFormat** της τιμής του control. (Κληρονομείται από την UpDownBase).

Increment

Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Long.

Σύνταξη:

VB:

```
Public Property Increment As Long
```

Τύπος: System.Long

Προσδιορίζουμε μία θετική τιμή για το βήμα αύξησης της τιμής. Η default τιμή είναι 1.

Dependency Property Information:

Identifier field: `IncrementProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το βήμα αύξησης στο UpDown1 σε 2.

XAML:

```
<zeus:LongUpDown x:Name="UpDown1" Minimum="1" Maximum="800" Increment="2"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Increment = 2
```

Minimum

Καθορίζει το **ελάχιστο όριο** στο control, τύπου Long.

Σύνταξη:

VB:

```
Public Property Minimum As Long
```

Τύπος: System.Long

Προσδιορίζουμε μία τιμή, ως το ελάχιστο αποδεκτό όριο. Η default τιμή είναι Long.MinValue.

Dependency Property Information:

Identifier field: `MinimumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το ελάχιστο όριο στο UpDown1 σε 10.

XAML:

```
<zeus:LongUpDown x:Name="UpDown1" Minimum="10" Maximum="800" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Minimum = 10
```

Maximum

Καθορίζει το **μέγιστο όριο** στο control, τύπου Long.

Σύνταξη:

VB:

```
Public Property Maximum As Long
```

Τύπος: System.Long

Προσδιορίζουμε μία τιμή, ως το μέγιστο αποδεκτό όριο. Η default τιμή είναι Long.MaxValue.

Dependency Property Information:

Identifier field: `MaximumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το μέγιστο όριο στο UpDown1 σε 900.

XAML:

```
<zeus:LongUpDown x:Name="UpDown1" Minimum="10" Maximum="900" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Maximum = 900
```

Value

Η **τρέχουσα τιμή** στο control, τύπου Long.

Σύνταξη:

VB:

```
Public Property Value As Long
```

Τύπος: System.Long

Η τρέχουσα τιμή. Η default τιμή είναι 0.

Dependency Property Information:

Identifier field: `ValueProperty`

Παρατηρήσεις:

Σε μία WPF εφαρμογή, η λογική χρήση της ιδιότητας αυτής είναι μέσω μίας Binding έκφρασης. Με αυτόν τον τρόπο, η τρέχουσα τιμή συνδέεται με μία source πηγή (π.χ. μία public ιδιότητα ενός data object). Η τιμή προέρχεται από το source object και οποιαδήποτε μεταβολή της επιστρέφει στο source object (σε ένα two-way mode, που είναι και το default). Συμπερασματικά, **συνιστάται η χρήση της με Binding έφραση**.

Παράδειγμα:

Στο επόμενο παράδειγμα, η τιμή της **ιδιότητας Value** στο **UpDown1** συνδέεται, **μέσω Binding έκφρασης**, με την **ιδιότητα Amount1** του **source object**. Το τελευταίο (ένα **αντικείμενο Customer**) τίθεται ως πηγή δεδομένων στην **ιδιότητα DataContext** σε ένα container, που εδώ είναι το **Grid** panel. Οι τιμές των ιδιοτήτων του αντικειμένου Customer τίθενται στο τμήμα Window.Resources, αλλά θα μπορούσαν να τεθούν και στο VB κώδικα, σε κάποιον event handler.

```
<Window x:Class="MainWindow"
          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
          xmlns:p="clr-namespace:UpDownControlsTestProject"
          xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                      DownControls"
          mc:Ignorable="d"
          Title="UpDownControls Sample Project"
          Height="350" Width="525" Loaded="Window_Loaded" >

        <Window.Resources>
```

```
<!-- The Validation Error ControlTemplate -->
<ControlTemplate x:Key="validationErrorTemplate">

    <StackPanel Orientation="Horizontal">

        <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                Background="#11FF0000"
                IsHitTestVisible="False" >
            <AdornedElementPlaceholder x :Name="placeholder"  />
        </Border>

        <Grid  Margin="20,0,0,0"  >
            <Ellipse  Width="20" Height="20"  Fill="Red"  />
            <TextBlock Foreground="White" FontSize="14" Text="!"
                       FontWeight="ExtraBold"
                       VerticalAlignment="Center"
                       HorizontalAlignment="Center" />
            <Grid.ToolTip>
                <TextBlock Text="{Binding Path=/ErrorContent}"/>
            </Grid.ToolTip>
        </Grid>

    </StackPanel>

</ControlTemplate>

<Style TargetType="{x:Type TextBlock }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="HorizontalAlignment" Value="Right" />
    <Setter Property="Margin" Value="5"/>
</Style>

<!-- The style base for the UpDown controls. -->
<Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Background" Value="White"/>
    <Setter Property="HorizontalAlignment" Value="Left" />
    <Setter Property="Margin" Value="5,0,0,0"/>

    <Style.Triggers >
        <Trigger Property="IsMinimum" Value="True">
            <Setter Property="Background" Value="Orange" />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="IsMaximum" Value="True">
            <Setter Property="Background" Value="Green"  />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="HasError" Value="True">
            <Setter Property="Foreground" Value="Red"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
        </Trigger>

    </Style.Triggers>

  </Style>

<!-- A sample Customer resource object for the Bindings -->
<p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
```

```
                                        Amount1="12"/>

        </Window.Resources>

        <Grid  Margin="10,40,10,10"
               DataContext="{Binding Source={StaticResource customer}}">

            ...

            <TextBlock Text="Amount (Long):" />
            <zeus:LongUpDown Name="UpDown1" Grid.Column="1"
                             Minimum="1" Maximum="800"
                             Increment="1" Language="el"
                             HorizontalContentAlignment="Right"
                    ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                    ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                     800. Please, retry."
                             Value="{Binding Amount1}"  ValueFormat="N0"
                            ValueChanged="UpDown1_ValueChanged"
                             Style="{StaticResource UpDownStyle}"/>

           ...

        </Grid>


</Window>
```

VB:

```
Private Sub Window_Loaded(sender As Object, e As RoutedEventArgs)

    Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
    UpDown1.ErrorTemplate = errTemplate

End Sub
```

Βλέπουμε την αρχική κατάσταση του UpDown1. Η τιμή 12 προέρχεται από την ιδιότητα Amount1 του αντικειμένου Customer:

Αν ο χρήστης φτάσει στη μέγιστη τιμή (800) τότε, λόγω του style trigger, το LongUpDown μορφοποιείται ως εξής:

Παράλληλα, απενεργοποείται και το up arrow button. Αν ο χρήστης, πληκτρολογήσει απ' ευθείας

μία **τιμή εκτός ορίων**, τότε το control εισέρχεται σε **κατάσταση λάθους**:

Επίσης, αν ο χρήστης, πληκτρολογήσει μία **τιμή λανθασμένου τύπου**, τότε το control πάλι εισέρχεται σε **κατάσταση λάθους**:

Συμβάντα

Όνομα	Περιγραφή
MinimumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.
MaximumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.
ValueChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

MinimumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MinimumChanged As RoutedPropertyChangedEventHandler(Of Long)
```

XAML attribute usage:

```
<zeus:LongUpDown MinimumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MinimumChanged του UpDown1:

XAML:

```
<zeus:LongUpDown x:Name="UpDown1" Minimum="0" Maximum="800" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 MinimumChanged="UpDown1_MinimumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MinimumChanged(sender As  Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Long))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new minimum value is {0}", e.NewValue))

     End If

End Sub
```

MaximumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MaximumChanged As RoutedPropertyChangedEventHandler(Of Long)
```

XAML attribute usage:

```
<zeus:LongUpDown MaximumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MaximumChanged του UpDown1:

XAML:

```
<zeus:LongUpDown x:Name="UpDown1" Minimum="0" Maximum="800" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 MaximumChanged="UpDown1_MaximumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MaximumChanged(sender As Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Long))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new maximum value is {0}", e.NewValue))

     End If

End Sub
```

ValueChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event ValueChanged As RoutedPropertyChangedEventHandler(Of Long)
```

XAML attribute usage:

```
<zeus:LongUpDown ValueChanged ="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν ValueChanged του UpDown1:

XAML:

```
<zeus:LongUpDown x:Name="UpDown1" Minimum="0" Maximum="800" Increment="1"
                 HorizontalContentAlignment="Right"
                 Value="{Binding Amount1}"  ValueFormat="N0"
                 ValueChanged="UpDown1_ValueChanged" ... />
```

VB:

```
 Private Sub UpDown1_ValueChanged(sender As Object, _
                                  e As RoutedPropertyChangedEventArgs(Of Long))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The current value is {0}", e.NewValue))

     End If

End Sub
```

Styles και Templates

Parts και States

Το default ControlTemplate περιλαμβάνει κάποια **part names** και **visual states**. Μπορείτε να τροποποιήσετε το default ControlTemplate ώστε να δώσετε στο control μία μοναδική εμφάνιση. Το default ControlTemplate μπορείτε να το δείτε εδώ ώστε να πάρετε μία ιδέα και με βάση αυτό να δημιουργήσετε ένα δικό σας. Ένα παράδειγμα custom ControlTemplate μπορείτε να δείτε εδώ.

Ο παρακάτω πίνακας εμφανίζει τα **part names** του **LongUpDown** control:

Part	Τύπος	Περιγραφή
PART_Value	TextBox	Το standard TextBox control.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την αύξηση της τιμής.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την μείωση της τιμής.

Ο παρακάτω πίνακας εμφανίζει τα **visual states** του **LongUpDown** control:

VisualState	VisualStateGroup	Περιγραφή
Normal	CommonStates	Το default state.
MouseOver	CommonStates	Το state όταν ο δείκτης του ποντικιού είναι πάνω από το control.
Disabled	CommonStates	Το state όταν το control είναι disabled.
Focused	FocusStates	Το state όταν το control έχει το focus.
Unfocused	FocusStates	Το state όταν το control δεν έχει το focus.

Το default ControlTemplate έχει καθορισμένο μόνο το Disabled state. Μπορείτε να δημιουργήσετε ένα custom ControlTemplate για να το αλλάξετε ή/και για να καθορίσετε τα υπόλοιπα.

Το default Style και ControlTemplate

Ο XAML κώδικας για το **default Style** και **ControlTemplate** φαίνεται παρακάτω. Μπορείτε να βασιστείτε σε αυτόν για να δημιουργήσετε μία μικρή ή μεγάλη παραλλαγή του δικού σας custom Style και ControlTemplate:

```
    xmlns:z="clr-namespace:Zeus.WPF.Controls.UpDownControls">

    <Style TargetType="{x:Type z:LongUpDown}" >

        <Setter Property="Focusable" Value="True"/>
        <Setter Property="BorderBrush" Value="Black" />
        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="Background"  Value="Transparent" />
        <Setter Property="Width" Value ="120"/>
        <Setter Property="Height" Value="23"/>

        <Setter Property="Template">

            <Setter.Value>

                <ControlTemplate TargetType="{x:Type z:LongUpDown}">

                    <ControlTemplate.Resources >
                        <z:DoubleToGridLengthConverter
                                            x:Key="doubleToGridLengthConverter"/>
                    </ControlTemplate.Resources>

                    <!-- Root element -->
                    <Border Name="Border"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">

                        <VisualStateManager.VisualStateGroups >

                            <VisualStateGroup Name="CommonStates">

                                <VisualState Name="Normal"/>

                                <VisualState Name="Disabled">

                                    <Storyboard >

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Background.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                     Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Foreground.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                    Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ObjectAnimationUsingKeyFrames
```

```
                                        Storyboard.TargetName="PART_Value"
                                    Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                        <DiscreteObjectKeyFrame KeyTime="0"
                                 Value="{Binding Source={x:Static FontWeights.Normal}}" />
                                        </ObjectAnimationUsingKeyFrames>

                                    </Storyboard>

                                </VisualState>


                            </VisualStateGroup>

                            <VisualStateGroup Name="FocusStates">

                                <VisualState Name="Focused" />
                                <VisualState Name="Unfocused"/>

                            </VisualStateGroup>

                        </VisualStateManager.VisualStateGroups>

                        <!-- Content -->
                        <Grid Name="mainGrid" Background="Transparent" >

                            <Grid.RowDefinitions >
                                <RowDefinition Height="*" />
                            </Grid.RowDefinitions>

                            <!-- The width of the second Grid column is about for the
                                 RepeatButtons and depends on ButtonsWidth property -->
                            <Grid.ColumnDefinitions >
                                <ColumnDefinition Width="*" />
                                <ColumnDefinition
                                      Width="{Binding RelativeSource={Relative Source
                              Mode=FindAncestor, AncestorType={x:Type z:LongUpDown}},
Path=ButtonsWidth, Converter={StaticResource doubleToGridLengthConverter}}" />
                            </Grid.ColumnDefinitions>

                             <TextBox Name="PART_Value"
                HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                    VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                                Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                            </TextBox>

                            <Viewbox Stretch="Fill"   Grid.Column="1">

                                <StackPanel Name="stkRepeatButtons"
                                            VerticalAlignment="Center"
                                            HorizontalAlignment="Left"  Focusable="False">

                                    <RepeatButton Name="PART_IncrementUp"
                                                  Focusable="False"  >
                                        <RepeatButton.Content>
                                            <Path Fill="Black" Stroke="Green"
                                                  Data="M 1,10 10,1 20,10 Z" />
                                        </RepeatButton.Content>
                                    </RepeatButton>
```

```
                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                    <RepeatButton.Content>
                                        <Path Fill="Black" Stroke="Green"
                                              Data="M 10,10 1,1 20,1 Z" />
                                    </RepeatButton.Content>
                                </RepeatButton>

                            </StackPanel>

                        </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

Η δεύτερη στήλη του Grid περιέχει τα repeat buttons μέσα σε ένα StackPanel. Το πλάτος της δεύτερης στήλης του Grid προσδιορίζεται με βάση την τιμή της ιδιότητας ButtonsWidth. Για να μετατραπεί η τιμή Double της ιδιότητας ButtonsWidth σε τιμή τύπου GridLength χρησιμοποιούμε μία κλάση μετατροπής τιμής (ValueConverter). Θα τη βρείτε στο Παράρτημα 2.

Παράδειγμα custom Style και ControlTemplate

Στο επόμενο παράδειγμα, δημιουργούμε ένα custom Style και ControlTemplate. Συγκεκριμένα:

- Ορίζουμε ότι **όταν λάβει την εστίαση, το background να γίνει κίτρινο**. Αυτό γίνεται, θέτοντας στο **Focused state** το σχετικό Storyboard που περιέχει ένα ColorAnimation που κάνει animate το background color σε Yellow σε χρόνο 0.2 sec.

- Στη δομή του ControlTemplate, **αλλάζουμε θέση στα up-down buttons**. Το up-button βρίσκεται πάνω δεξιά από το TextBox ενώ το down-button βρίσκεται από κάτω δεξιά (3 γραμμές). Για να επιτευχθεί αυτό αλλάζει λίγο η διάταξη του Grid container, που τώρα περιέχει 3 RowDefinintions. Η λειτουργικότητα του control παραμένει η ίδια.

Η εικόνα του custom LongUpDown σε κανονική κατάσταση - **Normal state**:

Η εικόνα του custom LongUpDown όταν έχει την εστίαση - **Focused state**:

Ο XAML κώδικας για το custom Style και ControlTemplate του παραπάνω LongUpDown είναι ο εξής:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownCont
                                                                              rols"

<Style TargetType="{x:Type zeus:LongUpDown}" >

    <Setter Property="Focusable" Value="True"/>
    <Setter Property="BorderBrush" Value="Black" />
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="Background"  Value="Transparent" />
    <Setter Property="Width" Value ="120"/>
    <Setter Property="Height" Value="23"/>

    <Setter Property="Template">

        <Setter.Value>

            <ControlTemplate TargetType="{x:Type zeus:LongUpDown}">

                <ControlTemplate.Resources >
                    <zeus:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                </ControlTemplate.Resources>
```

```
        <!-- Root element -->
        <Border Name="Border"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="{TemplateBinding BorderThickness}">

            <VisualStateManager.VisualStateGroups >

                <VisualStateGroup Name="CommonStates">

                    <VisualState Name="Normal"/>

                    <VisualState Name="Disabled">

                        <Storyboard>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Background.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
 Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Foreground.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ObjectAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                            Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                <DiscreteObjectKeyFrame KeyTime="0"
                     Value="{Binding Source={x:Static FontWeights.Normal}}" />
                            </ObjectAnimationUsingKeyFrames>

                        </Storyboard>

                    </VisualState>


                </VisualStateGroup>

                <VisualStateGroup Name="FocusStates">

                   <VisualState Name="Focused" >

                        <Storyboard >

                            ColorAnimation Storyboard.TargetName="PART_Value"
                                 Storyboard.TargetProperty="Background.Color"
                                           To="Yellow" Duration="0:0:0.2"/>

                            </Storyboard>

                        </VisualState>

                    <VisualState Name="Unfocused"/>

                </VisualStateGroup>
```

```
                    </VisualStateManager.VisualStateGroups>

                    <!-- Content -->
                   <Grid Name="mainGrid" Background="Transparent"
                                         Margin="{TemplateBinding Padding}">

                            <Grid.RowDefinitions >
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                            </Grid.RowDefinitions>

                            <TextBox Name="PART_Value" Grid.Row="1"
              HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                  VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                               Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                           </TextBox>

                           <Viewbox Stretch="Uniform" Grid.Row="0" Height="12"
                                    HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementUp"
                                              Focusable="False"  >
                                     <RepeatButton.Content>
                                           <Path Fill="Black" Stroke="Green"
                                                 Data="M 1,10 10,1 20,10 Z" />
                                      </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                           <Viewbox  Stretch="Uniform" Grid.Row="2" Height="12"
                                     HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                     <RepeatButton.Content>
                                          <Path Fill="Black" Stroke="Green"
                                                Data="M 10,10 1,1 20,1 Z" />
                                     </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

SingleUpDown

Ένα **UpDown control** που επιτρέπει την **είσοδο μόνο αριθμητικών δεδομένων, τύπου Single**. Η τιμή απαιτείται (required).

Σύνταξη:

VB:

```
<TemplateVisualState(Name:="Normal", GroupName:="CommonStates"),
TemplateVisualState(Name:="MouseOver", GroupName:="CommonStates"),
TemplateVisualState(Name:="Disabled", GroupName:="CommonStates"),
TemplateVisualState(Name:="Focused", GroupName:="FocusStates"),
TemplateVisualState(Name:="Unfocused", GroupName:="FocusStates"),
TemplatePart(Name:="PART_Value", Type:=GetType(TextBox)),
TemplatePart(Name:="PART_IncrementUp", Type:=GetType(RepeatButton)),
TemplatePart(Name:="PART_IncrementDown", Type:=GetType(RepeatButton))>
<DefaultProperty("Value")>
Public Class SingleUpDown
             Inherits UpDownBase
```

XAML Object Element Usage:

Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

Χρήση:

```
<zeus:SingleUpDown ... />
```

Παράδειγμα:

Στο παράδειγμα που ακολουθεί, τοποθετούμε ένα **SingleUpDown** μέσα σε ένα **Grid** ενός παραθύρου. Συνδέουμε την **ιδιότητα Value** με την **ιδιότητα Amount1**, τύπου **Single**, ενός **Customer** αντικειμένου (source object). Η **κλάση Customer** ορίζεται στο project και συνεπώς πρέπει να εισάγουμε και το αντίστοιχο namespace με alias p. Επίσης, ορίζουμε ένα **ErrorTemplate** για το **Validation** (όπου μέσω αυτού θα εμφανιστεί το μήνυμα λάθους που ορίζουμε στην **ιδιότητα ErrorMessageOnIncorrectValueType** ή στην **ιδιότητα ErrorMessageOnValueOutOfRange**). Επιπλέον, ορίζουμε ένα **Style**, στοχευμένο στην κλάση UpDownBase, για την ομοιόμορφη εμφάνιση των UpDown controls στο παράθυρο.

```
<Window x:Class="MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:p="clr-namespace:UpDownControlsTestProject"
        xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
        mc:Ignorable="d"
        Title="UpDownControls Sample Project"
        Height="350" Width="525" Loaded="Window_Loaded" >

    <Window.Resources>

        <!-- The Validation Error ControlTemplate -->
        <ControlTemplate x:Key="validationErrorTemplate">

            <StackPanel Orientation="Horizontal">

                <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                        Background="#11FF0000"
                        IsHitTestVisible="False" >
                    <AdornedElementPlaceholder x :Name="placeholder"  />
                </Border>

                <Grid  Margin="20,0,0,0"  >
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

        <Style TargetType="{x:Type TextBlock }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="HorizontalAlignment" Value="Right" />
            <Setter Property="Margin" Value="5"/>
        </Style>

        <!-- The style base for the UpDown controls. -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
```

```
                    </Trigger>

                    <Trigger Property="HasError" Value="True">
                        <Setter Property="Foreground" Value="Red"/>
                        <Setter Property="BorderBrush" Value="Red"/>
                        <Setter Property="BorderThickness" Value="2"/>
                    </Trigger>

                </Style.Triggers>

            </Style>

            <!-- A sample Customer resource object for the Bindings -->
            <p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
                                         Amount1="12"/>


        </Window.Resources>

        <Grid  Margin="10,40,10,10"
               DataContext="{Binding Source={StaticResource customer}}">

            ...

            <TextBlock Text="Amount (Single):" />
            <zeus:SingleUpDown Name="UpDown1" Grid.Column="1"
                               Minimum="1" Maximum="500"
                               Increment="0.50" Language="el"
                               HorizontalContentAlignment="Right"
                        ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                        ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                         500. Please, retry."
                                 Value="{Binding Amount1}"  ValueFormat="N2"
                                ValueChanged="UpDown1_ValueChanged"
                                 Style="{StaticResource UpDownStyle}"/>

           ...

        </Grid>


    </Window>
```

VB:

```
Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
UpDown1.ErrorTemplate = errTemplate
```

- Για την **κλάση Customer** δείτε το Παράρτημα 1.

Μία κανονική κατάσταση του **SingleUpDown**:

Παρακάτω, βλέπουμε την **ένδειξη λάθους** όταν ο χρήστης εισάγει μία **λανθασμένη (ως προς το εύρος) τιμή**. Αν πάει το δείκτη του ποντικιού πάνω στο **θαυμαστικό** θα εμφανιστεί το μήνυμα λάθους "`The number must be between 1 and 500. Please, retry.`"

Ιδιότητες

Όνομα	Περιγραφή
ButtonsWidth	Καθορίζει το **πλάτος**, σε pixels, **των buttons αυξομείωσης του control**. (*Το ύψος είναι σταθερά ορισμένο στο μέσον του ύψους του control)*. (Κληρονομείται από την UpDownBase).
ErrorMessageOnIncorrectValueType	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι τύπου Single. (Κληρονομείται από την UpDownBase).
ErrorMessageOnValueOutOfRange	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι στο αποδεκτό εύρος (Minimum-Maximum). (Κληρονομείται από την UpDownBase).
ErrorTemplate	Καθορίζει ένα **ControlTemplate** που θα εφαρμόζεται **όταν το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
HasError	Αν είναι **True**, τότε **το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
Increment	Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Single. Η default τιμή είναι 1.
IsMinimum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο ελάχιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Minimum. (Κληρονομείται από την UpDownBase).
IsMaximum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο μέγιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Maximum. (Κληρονομείται από την UpDownBase).
Minimum	Καθορίζει το **ελάχιστο όριο** στο control, τύπου Single. Η default τιμή είναι Single.MinValue.

Maximum	Καθορίζει το **μέγιστο όριο** στο control, τύπου Single. Η default τιμή είναι Single.MaxValue.
Value	Η **τρέχουσα τιμή** στο control, τύπου Single. Η default τιμή είναι 0.
ValueFormat	Καθορίζει το **StringFormat** της τιμής του control. (Κληρονομείται από την UpDownBase).

Increment

Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Single.

Σύνταξη:

VB:

```
Public Property Increment As Single
```

Τύπος: System.Single

Προσδιορίζουμε μία θετική τιμή για το βήμα αύξησης της τιμής. Η default τιμή είναι 1.

Dependency Property Information:

Identifier field: `IncrementProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το βήμα αύξησης στο UpDown1 σε 1.50.

XAML:

```
<zeus:SingleUpDown x:Name="UpDown1" Minimum="1" Maximum="500" Increment="1.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Increment = 1.50
```

Minimum

Καθορίζει το **ελάχιστο όριο** στο control, τύπου Single.

Σύνταξη:

VB:

```
Public Property Minimum As Single
```

Τύπος: System.Single

Προσδιορίζουμε μία τιμή, ως το ελάχιστο αποδεκτό όριο. Η default τιμή είναι Single.MinValue.

Dependency Property Information:

Identifier field: `MinimumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το ελάχιστο όριο στο UpDown1 σε 10.50.

XAML:

```
<zeus:SingleUpDown x:Name="UpDown1" Minimum="10.50" Maximum="500" Increment="0.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Minimum = 10.50
```

Maximum

Καθορίζει το **μέγιστο όριο** στο control, τύπου Single.

Σύνταξη:

VB:

```
Public Property Maximum As Single
```

Τύπος: System.Single

Προσδιορίζουμε μία τιμή, ως το μέγιστο αποδεκτό όριο. Η default τιμή είναι Single.MaxValue.

Dependency Property Information:

Identifier field: `MaximumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το μέγιστο όριο στο UpDown1 σε 900.

XAML:

```
<zeus:SingleUpDown x:Name="UpDown1" Minimum="10.50" Maximum="900" Increment="0.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Maximum = 900
```

Value

Η **τρέχουσα τιμή** στο control, τύπου Single.

Σύνταξη:

VB:

```
Public Property Value As Single
```

Τύπος: System.Single

Η τρέχουσα τιμή. Η default τιμή είναι 0.

Dependency Property Information:

Identifier field: `ValueProperty`

Παρατηρήσεις:

Σε μία WPF εφαρμογή, η λογική χρήση της ιδιότητας αυτής είναι μέσω μίας Binding έκφρασης. Με αυτόν τον τρόπο, η τρέχουσα τιμή συνδέεται με μία source πηγή (π.χ. μία public ιδιότητα ενός data object). Η τιμή προέρχεται από το source object και οποιαδήποτε μεταβολή της επιστρέφει στο source object (σε ένα two-way mode, που είναι και το default). Συμπερασματικά, **συνιστάται η χρήση της με Binding έφραση**.

Παράδειγμα:

Στο επόμενο παράδειγμα, η τιμή της **ιδιότητας Value** στο **UpDown1** συνδέεται, **μέσω Binding έκφρασης**, με την **ιδιότητα Amount1** του **source object**. Το τελευταίο (ένα **αντικείμενο Customer**) τίθεται ως πηγή δεδομένων στην **ιδιότητα DataContext** σε ένα container, που εδώ είναι το **Grid** panel. Οι τιμές των ιδιοτήτων του αντικειμένου Customer τίθενται στο τμήμα Window.Resources, αλλά θα μπορούσαν να τεθούν και στο VB κώδικα, σε κάποιον event handler.

```
<Window x:Class="MainWindow"
          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
          xmlns:p="clr-namespace:UpDownControlsTestProject"
          xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
          mc:Ignorable="d"
          Title="UpDownControls Sample Project"
          Height="350" Width="525" Loaded="Window_Loaded" >

        <Window.Resources>
```

```
<!-- The Validation Error ControlTemplate -->
<ControlTemplate x:Key="validationErrorTemplate">

    <StackPanel Orientation="Horizontal">

        <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                Background="#11FF0000"
                IsHitTestVisible="False" >
            <AdornedElementPlaceholder x :Name="placeholder"  />
        </Border>

        <Grid  Margin="20,0,0,0"  >
            <Ellipse  Width="20" Height="20"  Fill="Red"  />
            <TextBlock Foreground="White" FontSize="14" Text="!"
                       FontWeight="ExtraBold"
                       VerticalAlignment="Center"
                       HorizontalAlignment="Center" />
            <Grid.ToolTip>
                <TextBlock Text="{Binding Path=/ErrorContent}"/>
            </Grid.ToolTip>
        </Grid>

    </StackPanel>

</ControlTemplate>

<Style TargetType="{x:Type TextBlock }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="HorizontalAlignment" Value="Right" />
    <Setter Property="Margin" Value="5"/>
</Style>

<!-- The style base for the UpDown controls. -->
<Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Background" Value="White"/>
    <Setter Property="HorizontalAlignment" Value="Left" />
    <Setter Property="Margin" Value="5,0,0,0"/>

    <Style.Triggers >
        <Trigger Property="IsMinimum" Value="True">
            <Setter Property="Background" Value="Orange" />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="IsMaximum" Value="True">
            <Setter Property="Background" Value="Green"  />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="HasError" Value="True">
            <Setter Property="Foreground" Value="Red"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
        </Trigger>

    </Style.Triggers>

  </Style>

<!-- A sample Customer resource object for the Bindings -->
<p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
```

```
                                        Amount1="12"/>

        </Window.Resources>

        <Grid  Margin="10,40,10,10"
               DataContext="{Binding Source={StaticResource customer}}">

            ...

            <TextBlock Text="Amount (Single):" />
            <zeus:SingleUpDown  Name="UpDown1" Grid.Column="1"
                                Minimum="1" Maximum="500"
                                Increment="0.50" Language="el"
                                HorizontalContentAlignment="Right"
                    ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                    ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                              500. Please, retry."
                             Value="{Binding Amount1}"  ValueFormat="N2"
                            ValueChanged="UpDown1_ValueChanged"
                             Style="{StaticResource UpDownStyle}"/>

           ...

        </Grid>


</Window>
```

VB:

```
Private Sub Window_Loaded(sender As Object, e As RoutedEventArgs)

    Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
    UpDown1.ErrorTemplate = errTemplate

End Sub
```

Βλέπουμε την αρχική κατάσταση του UpDown1. Η τιμή 12.50 προέρχεται από την ιδιότητα Amount1 του αντικειμένου Customer:

Αν ο χρήστης φτάσει στη μέγιστη τιμή (500) τότε, λόγω του style trigger, το SingleUpDown μορφοποιείται ως εξής:

Παράλληλα, απενεργοποιείται και το up arrow button. Αν ο χρήστης, πληκτρολογήσει απ' ευθείας

μία **τιμή εκτός ορίων**, τότε το control εισέρχεται σε **κατάσταση λάθους**:

Επίσης, αν ο χρήστης, πληκτρολογήσει μία **τιμή λανθασμένου τύπου**, τότε το control πάλι εισέρχεται σε **κατάσταση λάθους**:

Συμβάντα

Όνομα	Περιγραφή
MinimumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.
MaximumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.
ValueChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

MinimumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MinimumChanged As RoutedPropertyChangedEventHandler(Of Single)
```

XAML attribute usage:

```
<zeus:SingleUpDown MinimumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MinimumChanged του UpDown1:

XAML:

```
<zeus:SingleUpDown x:Name="UpDown1" Minimum="0" Maximum="500" Increment="0.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   MinimumChanged="UpDown1_MinimumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MinimumChanged(sender As  Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Single))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new minimum value is {0}", e.NewValue))

     End If

End Sub
```

MaximumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MaximumChanged As RoutedPropertyChangedEventHandler(Of Single)
```

XAML attribute usage:

```
<zeus:SingleUpDown MaximumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MaximumChanged του UpDown1:

XAML:

```
<zeus:SingleUpDown x:Name="UpDown1" Minimum="0" Maximum="500" Increment="0.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   MaximumChanged="UpDown1_MaximumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MaximumChanged(sender As Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Single))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new maximum value is {0}", e.NewValue))

     End If

End Sub
```

ValueChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event ValueChanged As RoutedPropertyChangedEventHandler(Of Single)
```

XAML attribute usage:

```
<zeus:SingleUpDown ValueChanged ="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν ValueChanged του UpDown1:

XAML:

```
<zeus:SingleUpDown x:Name="UpDown1" Minimum="0" Maximum="500" Increment="0.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   ValueChanged="UpDown1_ValueChanged" ... />
```

VB:

```
 Private Sub UpDown1_ValueChanged(sender As Object, _
                                  e As RoutedPropertyChangedEventArgs(Of Single))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The current value is {0}", e.NewValue))

     End If

End Sub
```

Styles και Templates

Parts και States

Το default ControlTemplate περιλαμβάνει κάποια **part names** και **visual states**. Μπορείτε να τροποποιήσετε το default ControlTemplate ώστε να δώσετε στο control μία μοναδική εμφάνιση. Το default ControlTemplate μπορείτε να το δείτε εδώ ώστε να πάρετε μία ιδέα και με βάση αυτό να δημιουργήσετε ένα δικό σας. Ένα παράδειγμα custom ControlTemplate μπορείτε να δείτε εδώ.

Ο παρακάτω πίνακας εμφανίζει τα **part names** του **SingleUpDown** control:

Part	Τύπος	Περιγραφή
PART_Value	TextBox	Το standard TextBox control.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την αύξηση της τιμής.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την μείωση της τιμής.

Ο παρακάτω πίνακας εμφανίζει τα **visual states** του **SingleUpDown** control:

VisualState	VisualStateGroup	Περιγραφή
Normal	CommonStates	Το default state.
MouseOver	CommonStates	Το state όταν ο δείκτης του ποντικιού είναι πάνω από το control.
Disabled	CommonStates	Το state όταν το control είναι disabled.
Focused	FocusStates	Το state όταν το control έχει το focus.
Unfocused	FocusStates	Το state όταν το control δεν έχει το focus.

Το default ControlTemplate έχει καθορισμένο μόνο το Disabled state. Μπορείτε να δημιουργήσετε ένα custom ControlTemplate για να το αλλάξετε ή/και για να καθορίσετε τα υπόλοιπα.

Το default Style και ControlTemplate

Ο XAML κώδικας για το **default Style** και **ControlTemplate** φαίνεται παρακάτω. Μπορείτε να βασιστείτε σε αυτόν για να δημιουργήσετε μία μικρή ή μεγάλη παραλλαγή του δικού σας custom Style και ControlTemplate:

```
    xmlns:z="clr-namespace:Zeus.WPF.Controls.UpDownControls">

    <Style TargetType="{x:Type z:SingleUpDown}" >

        <Setter Property="Focusable" Value="True"/>
        <Setter Property="BorderBrush" Value="Black" />
        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="Background"  Value="Transparent" />
        <Setter Property="Width" Value ="120"/>
        <Setter Property="Height" Value="23"/>

        <Setter Property="Template">

            <Setter.Value>

                <ControlTemplate TargetType="{x:Type z:SingleUpDown}">

                    <ControlTemplate.Resources >
                        <z:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                    </ControlTemplate.Resources>

                    <!-- Root element -->
                    <Border Name="Border"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">

                        <VisualStateManager.VisualStateGroups >

                            <VisualStateGroup Name="CommonStates">

                                <VisualState Name="Normal"/>

                                <VisualState Name="Disabled">

                                    <Storyboard >

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Background.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                     Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Foreground.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                    Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ObjectAnimationUsingKeyFrames
```

```
                                        Storyboard.TargetName="PART_Value"
                                    Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                        <DiscreteObjectKeyFrame KeyTime="0"
                                 Value="{Binding Source={x:Static FontWeights.Normal}}" />
                                        </ObjectAnimationUsingKeyFrames>

                                    </Storyboard>

                                </VisualState>


                            </VisualStateGroup>

                            <VisualStateGroup Name="FocusStates">

                                <VisualState Name="Focused" />
                                <VisualState Name="Unfocused"/>

                            </VisualStateGroup>

                        </VisualStateManager.VisualStateGroups>

                        <!-- Content -->
                        <Grid Name="mainGrid" Background="Transparent" >

                            <Grid.RowDefinitions >
                                <RowDefinition Height="*" />
                            </Grid.RowDefinitions>

                            <!-- The width of the second Grid column is about for the
                                 RepeatButtons and depends on ButtonsWidth property -->
                            <Grid.ColumnDefinitions >
                                <ColumnDefinition Width="*" />
                                <ColumnDefinition
                                      Width="{Binding RelativeSource={Relative Source
                              Mode=FindAncestor, AncestorType={x:Type z:SingleUpDown}},
Path=ButtonsWidth, Converter={StaticResource doubleToGridLengthConverter}}" />
                            </Grid.ColumnDefinitions>

                             <TextBox Name="PART_Value"
                HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                    VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                                Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                            </TextBox>

                            <Viewbox Stretch="Fill"   Grid.Column="1">

                                <StackPanel Name="stkRepeatButtons"
                                            VerticalAlignment="Center"
                                            HorizontalAlignment="Left"  Focusable="False">

                                    <RepeatButton Name="PART_IncrementUp"
                                                  Focusable="False"  >
                                        <RepeatButton.Content>
                                            <Path Fill="Black" Stroke="Green"
                                                  Data="M 1,10 10,1 20,10 Z" />
                                        </RepeatButton.Content>
                                    </RepeatButton>
```

```
                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                    <RepeatButton.Content>
                                        <Path Fill="Black" Stroke="Green"
                                              Data="M 10,10 1,1 20,1 Z" />
                                    </RepeatButton.Content>
                                </RepeatButton>

                            </StackPanel>

                        </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

Η δεύτερη στήλη του Grid περιέχει τα repeat buttons μέσα σε ένα StackPanel. Το πλάτος της δεύτερης στήλης του Grid προσδιορίζεται με βάση την τιμή της ιδιότητας ButtonsWidth. Για να μετατραπεί η τιμή Double της ιδιότητας ButtonsWidth σε τιμή τύπου GridLength χρησιμοποιούμε μία κλάση μετατροπής τιμής (ValueConverter). Θα τη βρείτε στο Παράρτημα 2.

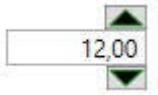
Παράδειγμα custom Style και ControlTemplate

Στο επόμενο παράδειγμα, δημιουργούμε ένα custom Style και ControlTemplate. Συγκεκριμένα:

- Ορίζουμε ότι **όταν λάβει την εστίαση, το background να γίνει κίτρινο**. Αυτό γίνεται, θέτοντας στο **Focused state** το σχετικό Storyboard που περιέχει ένα ColorAnimation που κάνει animate το background color σε Yellow σε χρόνο 0.2 sec.

- Στη δομή του ControlTemplate, **αλλάζουμε θέση στα up-down buttons**. Το up-button βρίσκεται πάνω δεξιά από το TextBox ενώ το down-button βρίσκεται από κάτω δεξιά (3 γραμμές). Για να επιτευχθεί αυτό αλλάζει λίγο η διάταξη του Grid container, που τώρα περιέχει 3 RowDefinintions. Η λειτουργικότητα του control παραμένει η ίδια.

Η εικόνα του custom SingleUpDown σε κανονική κατάσταση - **Normal state**:



Η εικόνα του custom SingleUpDown όταν έχει την εστίαση - **Focused state**:

Ο XAML κώδικας για το custom Style και ControlTemplate του παραπάνω SingleUpDown είναι ο εξής:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownCont
                                                                              rols"

<Style TargetType="{x:Type zeus:SingleUpDown}" >

    <Setter Property="Focusable" Value="True"/>
    <Setter Property="BorderBrush" Value="Black" />
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="Background"  Value="Transparent" />
    <Setter Property="Width" Value ="120"/>
    <Setter Property="Height" Value="23"/>

    <Setter Property="Template">

        <Setter.Value>

            <ControlTemplate TargetType="{x:Type zeus:SingleUpDown}">

                <ControlTemplate.Resources >
                    <zeus:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                </ControlTemplate.Resources>
```

```
        <!-- Root element -->
        <Border Name="Border"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="{TemplateBinding BorderThickness}">

            <VisualStateManager.VisualStateGroups >

                <VisualStateGroup Name="CommonStates">

                    <VisualState Name="Normal"/>

                    <VisualState Name="Disabled">

                        <Storyboard>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Background.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
 Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Foreground.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ObjectAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                            Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                <DiscreteObjectKeyFrame KeyTime="0"
                     Value="{Binding Source={x:Static FontWeights.Normal}}" />
                            </ObjectAnimationUsingKeyFrames>

                        </Storyboard>

                    </VisualState>


                </VisualStateGroup>

                <VisualStateGroup Name="FocusStates">

                   <VisualState Name="Focused" >

                        <Storyboard >

                            ColorAnimation Storyboard.TargetName="PART_Value"
                                 Storyboard.TargetProperty="Background.Color"
                                           To="Yellow" Duration="0:0:0.2"/>

                        </Storyboard>

                    </VisualState>

                    <VisualState Name="Unfocused"/>

                </VisualStateGroup>
```

```
                    </VisualStateManager.VisualStateGroups>

                    <!-- Content -->
                   <Grid Name="mainGrid" Background="Transparent"
                                         Margin="{TemplateBinding Padding}">

                            <Grid.RowDefinitions >
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                            </Grid.RowDefinitions>

                            <TextBox Name="PART_Value" Grid.Row="1"
              HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                  VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                               Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                           </TextBox>

                           <Viewbox Stretch="Uniform" Grid.Row="0" Height="12"
                                    HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementUp"
                                              Focusable="False"  >
                                     <RepeatButton.Content>
                                           <Path Fill="Black" Stroke="Green"
                                                 Data="M 1,10 10,1 20,10 Z" />
                                      </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                           <Viewbox  Stretch="Uniform" Grid.Row="2" Height="12"
                                     HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                     <RepeatButton.Content>
                                          <Path Fill="Black" Stroke="Green"
                                                Data="M 10,10 1,1 20,1 Z" />
                                     </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

DoubleUpDown

Ένα **UpDown control** που επιτρέπει την **είσοδο μόνο αριθμητικών δεδομένων, τύπου Double**. Η τιμή απαιτείται (required).

Σύνταξη:

VB:

```
<TemplateVisualState(Name:="Normal", GroupName:="CommonStates"),
TemplateVisualState(Name:="MouseOver", GroupName:="CommonStates"),
TemplateVisualState(Name:="Disabled", GroupName:="CommonStates"),
TemplateVisualState(Name:="Focused", GroupName:="FocusStates"),
TemplateVisualState(Name:="Unfocused", GroupName:="FocusStates"),
TemplatePart(Name:="PART_Value", Type:=GetType(TextBox)),
TemplatePart(Name:="PART_IncrementUp", Type:=GetType(RepeatButton)),
TemplatePart(Name:="PART_IncrementDown", Type:=GetType(RepeatButton))>
<DefaultProperty("Value")>
Public Class DoubleUpDown
             Inherits UpDownBase
```

XAML Object Element Usage:

Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

Χρήση:

```
<zeus:DoubleUpDown ... />
```

Παράδειγμα:

Στο παράδειγμα που ακολουθεί, τοποθετούμε ένα **DoubleUpDown** μέσα σε ένα **Grid** ενός παραθύρου. Συνδέουμε την **ιδιότητα Value** με την **ιδιότητα Amount1**, τύπου **Double**, ενός **Customer** αντικειμένου (source object). Η **κλάση Customer** ορίζεται στο project και συνεπώς πρέπει να εισάγουμε και το αντίστοιχο namespace με alias p. Επίσης, ορίζουμε ένα **ErrorTemplate** για το **Validation** (όπου μέσω αυτού θα εμφανιστεί το μήνυμα λάθους που ορίζουμε στην **ιδιότητα ErrorMessageOnIncorrectValueType** ή στην **ιδιότητα ErrorMessageOnValueOutOfRange**). Επιπλέον, ορίζουμε ένα **Style**, στοχευμένο στην κλάση UpDownBase, για την ομοιόμορφη εμφάνιση των UpDown controls στο παράθυρο.

```
<Window x:Class="MainWindow"
         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:p="clr-namespace:UpDownControlsTestProject"
        xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
        mc:Ignorable="d"
        Title="UpDownControls Sample Project"
        Height="350" Width="525" Loaded="Window_Loaded" >

    <Window.Resources>

        <!-- The Validation Error ControlTemplate -->
        <ControlTemplate x:Key="validationErrorTemplate">

            <StackPanel Orientation="Horizontal">

                <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                        Background="#11FF0000"
                        IsHitTestVisible="False" >
                    <AdornedElementPlaceholder x :Name="placeholder"  />
                </Border>

                <Grid  Margin="20,0,0,0"  >
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

        <Style TargetType="{x:Type TextBlock }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="HorizontalAlignment" Value="Right" />
            <Setter Property="Margin" Value="5"/>
        </Style>

        <!-- The style base for the UpDown controls. -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>
```

```
                    <Trigger Property="HasError" Value="True">
                        <Setter Property="Foreground" Value="Red"/>
                        <Setter Property="BorderBrush" Value="Red"/>
                        <Setter Property="BorderThickness" Value="2"/>
                    </Trigger>

                </Style.Triggers>

            </Style>

        <!-- A sample Customer resource object for the Bindings -->
        <p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
                                     Amount1="12"/>


    </Window.Resources>

    <Grid  Margin="10,40,10,10"
           DataContext="{Binding Source={StaticResource customer}}">

        ...

        <TextBlock Text="Amount (Double):" />
        <zeus:DoubleUpDown Name="UpDown1" Grid.Column="1"
                           Minimum="1" Maximum="500"
                           Increment="0.50" Language="el"
                           HorizontalContentAlignment="Right"
                 ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                 ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                  500. Please, retry."
                           Value="{Binding Amount1}"  ValueFormat="N2"
                          ValueChanged="UpDown1_ValueChanged"
                           Style="{StaticResource UpDownStyle}"/>

       ...

    </Grid>


</Window>
```

VB:

```
Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
UpDown1.ErrorTemplate = errTemplate
```

- Για την **κλάση Customer** δείτε το Παράρτημα 1.

Μία κανονική κατάσταση του **DoubleUpDown** :

Παρακάτω, βλέπουμε την **ένδειξη λάθους** όταν ο χρήστης εισάγει μία **λανθασμένη (ως προς το εύρος) τιμή**. Αν πάει το δείκτη του ποντικιού πάνω στο **θαυμαστικό** θα εμφανιστεί το μήνυμα λάθους "`The number must be between 1 and 500. Please, retry.`"

Ιδιότητες

Όνομα	Περιγραφή
ButtonsWidth	Καθορίζει το **πλάτος**, σε pixels, **των buttons αυξομείωσης του control**. (*Το ύψος είναι σταθερά ορισμένο στο μέσον του ύψους του control)*. (Κληρονομείται από την UpDownBase).
ErrorMessageOnIncorrectValueType	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι τύπου Double. (Κληρονομείται από την UpDownBase).
ErrorMessageOnValueOutOfRange	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι στο αποδεκτό εύρος (Minimum-Maximum). (Κληρονομείται από την UpDownBase).
ErrorTemplate	Καθορίζει ένα **ControlTemplate** που θα εφαρμόζεται **όταν το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
HasError	Αν είναι **True**, τότε **το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
Increment	Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Double. Η default τιμή είναι 1.
IsMinimum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο ελάχιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Minimum. (Κληρονομείται από την UpDownBase).
IsMaximum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο μέγιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Maximum. (Κληρονομείται από την UpDownBase).
Minimum	Καθορίζει το **ελάχιστο όριο** στο control, τύπου Double. Η default τιμή είναι Double.MinValue.

Maximum	Καθορίζει το **μέγιστο όριο** στο control, τύπου Double. Η default τιμή είναι Double.MaxValue.
Value	Η **τρέχουσα τιμή** στο control, τύπου Double. Η default τιμή είναι 0.
ValueFormat	Καθορίζει το **StringFormat** της τιμής του control. (Κληρονομείται από την UpDownBase).

Increment

Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Double.

Σύνταξη:

VB:

```
Public Property Increment As Double
```

Τύπος: System.Double

Προσδιορίζουμε μία θετική τιμή για το βήμα αύξησης της τιμής. Η default τιμή είναι 1.

Dependency Property Information:

Identifier field: `IncrementProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το βήμα αύξησης στο UpDown1 σε 1.50.

XAML:

```
<zeus:DoubleUpDown x:Name="UpDown1" Minimum="1" Maximum="500" Increment="1.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Increment = 1.50
```

Minimum

Καθορίζει το **ελάχιστο όριο** στο control, τύπου Double.

Σύνταξη:

VB:

```
Public Property Minimum As Double
```

Τύπος: System.Double

Προσδιορίζουμε μία τιμή, ως το ελάχιστο αποδεκτό όριο. Η default τιμή είναι Double.MinValue.

Dependency Property Information:

Identifier field: `MinimumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το ελάχιστο όριο στο UpDown1 σε 10.50.

XAML:

```
<zeus:DoubleUpDown x:Name="UpDown1" Minimum="10.50" Maximum="500" Increment="0.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Minimum = 10.50
```

Maximum

Καθορίζει το **μέγιστο όριο** στο control, τύπου Double.

Σύνταξη:

VB:

```
Public Property Maximum As Double
```

Τύπος: System.Double

Προσδιορίζουμε μία τιμή, ως το μέγιστο αποδεκτό όριο. Η default τιμή είναι Double.MaxValue.

Dependency Property Information:

Identifier field: `MaximumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το μέγιστο όριο στο UpDown1 σε 900.

XAML:

```
<zeus:DoubleUpDown x:Name="UpDown1" Minimum="10.50" Maximum="900" Increment="0.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Maximum = 900
```

Value

Η **τρέχουσα τιμή** στο control, τύπου Double.

Σύνταξη:

VB:

```
Public Property Value As Double
```

Τύπος: System.Double

Η τρέχουσα τιμή. Η default τιμή είναι 0.

Dependency Property Information:

Identifier field: `ValueProperty`

Παρατηρήσεις:

Σε μία WPF εφαρμογή, η λογική χρήση της ιδιότητας αυτής είναι μέσω μίας Binding έκφρασης. Με αυτόν τον τρόπο, η τρέχουσα τιμή συνδέεται με μία source πηγή (π.χ. μία public ιδιότητα ενός data object). Η τιμή προέρχεται από το source object και οποιαδήποτε μεταβολή της επιστρέφει στο source object (σε ένα two-way mode, που είναι και το default). Συμπερασματικά, **συνιστάται η χρήση της με Binding έφραση**.

Παράδειγμα:

Στο επόμενο παράδειγμα, η τιμή της **ιδιότητας Value** στο **UpDown1** συνδέεται, **μέσω Binding έκφρασης**, με την **ιδιότητα Amount1** του **source object**. Το τελευταίο (ένα **αντικείμενο Customer**) τίθεται ως πηγή δεδομένων στην **ιδιότητα DataContext** σε ένα container, που εδώ είναι το **Grid** panel. Οι τιμές των ιδιοτήτων του αντικειμένου Customer τίθενται στο τμήμα Window.Resources, αλλά θα μπορούσαν να τεθούν και στο VB κώδικα, σε κάποιον event handler.

```
<Window x:Class="MainWindow"
          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
          xmlns:p="clr-namespace:UpDownControlsTestProject"
          xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
          mc:Ignorable="d"
          Title="UpDownControls Sample Project"
          Height="350" Width="525" Loaded="Window_Loaded" >

        <Window.Resources>
```

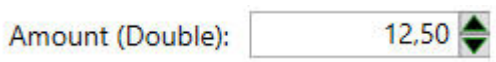
```
<!-- The Validation Error ControlTemplate -->
<ControlTemplate x:Key="validationErrorTemplate">

    <StackPanel Orientation="Horizontal">

        <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                Background="#11FF0000"
                IsHitTestVisible="False" >
            <AdornedElementPlaceholder x :Name="placeholder"  />
        </Border>

        <Grid  Margin="20,0,0,0"  >
            <Ellipse  Width="20" Height="20"  Fill="Red"  />
            <TextBlock Foreground="White" FontSize="14" Text="!"
                       FontWeight="ExtraBold"
                       VerticalAlignment="Center"
                       HorizontalAlignment="Center" />
            <Grid.ToolTip>
                <TextBlock Text="{Binding Path=/ErrorContent}"/>
            </Grid.ToolTip>
        </Grid>

    </StackPanel>

</ControlTemplate>

<Style TargetType="{x:Type TextBlock }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="HorizontalAlignment" Value="Right" />
    <Setter Property="Margin" Value="5"/>
</Style>

<!-- The style base for the UpDown controls. -->
<Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Background" Value="White"/>
    <Setter Property="HorizontalAlignment" Value="Left" />
    <Setter Property="Margin" Value="5,0,0,0"/>

    <Style.Triggers >
        <Trigger Property="IsMinimum" Value="True">
            <Setter Property="Background" Value="Orange" />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="IsMaximum" Value="True">
            <Setter Property="Background" Value="Green"  />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="HasError" Value="True">
            <Setter Property="Foreground" Value="Red"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
        </Trigger>

    </Style.Triggers>

  </Style>

<!-- A sample Customer resource object for the Bindings -->
<p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
```

```
                                        Amount1="12"/>

        </Window.Resources>

        <Grid  Margin="10,40,10,10"
               DataContext="{Binding Source={StaticResource customer}}">

            ...

            <TextBlock Text="Amount (Double):" />
            <zeus:DoubleUpDown Name="UpDown1" Grid.Column="1"
                               Minimum="1" Maximum="500"
                               Increment="0.50" Language="el"
                               HorizontalContentAlignment="Right"
                    ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                    ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                     500. Please, retry."
                             Value="{Binding Amount1}"  ValueFormat="N2"
                            ValueChanged="UpDown1_ValueChanged"
                             Style="{StaticResource UpDownStyle}"/>

           ...

        </Grid>


</Window>
```

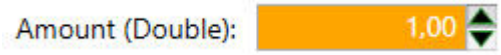
VB:

```
Private Sub Window_Loaded(sender As Object, e As RoutedEventArgs)

    Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
    UpDown1.ErrorTemplate = errTemplate

End Sub
```

Βλέπουμε την αρχική κατάσταση του UpDown1. Η τιμή 12.50 προέρχεται από την ιδιότητα Amount1 του αντικειμένου Customer:



Αν ο χρήστης φτάσει στην ελάχιστη τιμή (1.00) τότε , λόγω του style trigger, το DoubleUpDown μορφοποιείται ως εξής:



Παράλληλα, απενεργοποιείται και το down arrow button. Αν ο χρήστης, πληκτρολογήσει απ' ευθείας

μία **τιμή εκτός ορίων**, τότε το control εισέρχεται σε **κατάσταση λάθους**:

Επίσης, αν ο χρήστης, πληκτρολογήσει μία **τιμή λανθασμένου τύπου**, τότε το control πάλι εισέρχεται σε **κατάσταση λάθους**:

Συμβάντα

Όνομα	Περιγραφή
MinimumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.
MaximumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.
ValueChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

MinimumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MinimumChanged As RoutedPropertyChangedEventHandler(Of Double)
```

XAML attribute usage:

```
<zeus:DoubleUpDown MinimumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MinimumChanged του UpDown1:

XAML:

```
<zeus:DoubleUpDown x:Name="UpDown1" Minimum="0" Maximum="500" Increment="0.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   MinimumChanged="UpDown1_MinimumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MinimumChanged(sender As  Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Double))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new minimum value is {0}", e.NewValue))

     End If

End Sub
```

MaximumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MaximumChanged As RoutedPropertyChangedEventHandler(Of Double)
```

XAML attribute usage:

```
<zeus:DoubleUpDown MaximumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MaximumChanged του UpDown1:

XAML:

```
<zeus:DoubleUpDown x:Name="UpDown1" Minimum="0" Maximum="500" Increment="0.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   MaximumChanged="UpDown1_MaximumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MaximumChanged(sender As Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Double))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new maximum value is {0}", e.NewValue))

     End If

End Sub
```

ValueChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event ValueChanged As RoutedPropertyChangedEventHandler(Of Double)
```

XAML attribute usage:

```
<zeus:DoubleUpDown ValueChanged ="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν ValueChanged του UpDown1:

XAML:

```
<zeus:DoubleUpDown x:Name="UpDown1" Minimum="0" Maximum="500" Increment="0.50"
                   HorizontalContentAlignment="Right"
                   Value="{Binding Amount1}"  ValueFormat="N2"
                   ValueChanged="UpDown1_ValueChanged" ... />
```

VB:

```
 Private Sub UpDown1_ValueChanged(sender As Object, _
                                  e As RoutedPropertyChangedEventArgs(Of Double))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The current value is {0}", e.NewValue))

     End If

End Sub
```

Styles και Templates

Parts και States

Το default ControlTemplate περιλαμβάνει κάποια **part names** και **visual states**. Μπορείτε να τροποποιήσετε το default ControlTemplate ώστε να δώσετε στο control μία μοναδική εμφάνιση. Το default ControlTemplate μπορείτε να το δείτε εδώ ώστε να πάρετε μία ιδέα και με βάση αυτό να δημιουργήσετε ένα δικό σας. Ένα παράδειγμα custom ControlTemplate μπορείτε να δείτε εδώ.

Ο παρακάτω πίνακας εμφανίζει τα **part names** του **DoubleUpDown** control:

Part	Τύπος	Περιγραφή
PART_Value	TextBox	Το standard TextBox control.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την αύξηση της τιμής.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την μείωση της τιμής.

Ο παρακάτω πίνακας εμφανίζει τα **visual states** του **DoubleUpDown** control:

VisualState	VisualStateGroup	Περιγραφή
Normal	CommonStates	Το default state.
MouseOver	CommonStates	Το state όταν ο δείκτης του ποντικιού είναι πάνω από το control.
Disabled	CommonStates	Το state όταν το control είναι disabled.
Focused	FocusStates	Το state όταν το control έχει το focus.
Unfocused	FocusStates	Το state όταν το control δεν έχει το focus.

Το default ControlTemplate έχει καθορισμένο μόνο το Disabled state. Μπορείτε να δημιουργήσετε ένα custom ControlTemplate για να το αλλάξετε ή/και για να καθορίσετε τα υπόλοιπα.

Το default Style και ControlTemplate

Ο XAML κώδικας για το **default Style** και **ControlTemplate** φαίνεται παρακάτω. Μπορείτε να βασιστείτε σε αυτόν για να δημιουργήσετε μία μικρή ή μεγάλη παραλλαγή του δικού σας custom Style και ControlTemplate:

```
    xmlns:z="clr-namespace:Zeus.WPF.Controls.UpDownControls">

    <Style TargetType="{x:Type z:DoubleUpDown}" >

        <Setter Property="Focusable" Value="True"/>
        <Setter Property="BorderBrush" Value="Black" />
        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="Background"  Value="Transparent" />
        <Setter Property="Width" Value ="120"/>
        <Setter Property="Height" Value="23"/>

        <Setter Property="Template">

            <Setter.Value>

                <ControlTemplate TargetType="{x:Type z:DoubleUpDown}">

                    <ControlTemplate.Resources >
                        <z:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                    </ControlTemplate.Resources>

                    <!-- Root element -->
                    <Border Name="Border"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">

                        <VisualStateManager.VisualStateGroups >

                            <VisualStateGroup Name="CommonStates">

                                <VisualState Name="Normal"/>

                                <VisualState Name="Disabled">

                                    <Storyboard >

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Background.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                     Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Foreground.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                    Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ObjectAnimationUsingKeyFrames
```

```
                                        Storyboard.TargetName="PART_Value"
                                    Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                        <DiscreteObjectKeyFrame KeyTime="0"
                                 Value="{Binding Source={x:Static FontWeights.Normal}}" />
                                    </ObjectAnimationUsingKeyFrames>

                                </Storyboard>

                            </VisualState>


                        </VisualStateGroup>

                        <VisualStateGroup Name="FocusStates">

                            <VisualState Name="Focused" />
                            <VisualState Name="Unfocused"/>

                        </VisualStateGroup>

                    </VisualStateManager.VisualStateGroups>

                    <!-- Content -->
                    <Grid Name="mainGrid" Background="Transparent" >

                        <Grid.RowDefinitions >
                            <RowDefinition Height="*" />
                        </Grid.RowDefinitions>

                        <!-- The width of the second Grid column is about for the
                             RepeatButtons and depends on ButtonsWidth property -->
                        <Grid.ColumnDefinitions >
                            <ColumnDefinition Width="*" />
                            <ColumnDefinition
                                  Width="{Binding RelativeSource={Relative Source
                      Mode=FindAncestor, AncestorType={x:Type z:DoubleUpDown}},
Path=ButtonsWidth, Converter={StaticResource doubleToGridLengthConverter}}" />
                        </Grid.ColumnDefinitions>

                         <TextBox Name="PART_Value"
    HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
        VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                 Background="{TemplateBinding Background }"
                                 Foreground="{TemplateBinding Foreground }"
                                 Padding="2,0,2,0"
                           Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                        </TextBox>

                        <Viewbox Stretch="Fill"   Grid.Column="1">

                            <StackPanel Name="stkRepeatButtons"
                                        VerticalAlignment="Center"
                                        HorizontalAlignment="Left"  Focusable="False">

                                <RepeatButton Name="PART_IncrementUp"
                                              Focusable="False"  >
                                    <RepeatButton.Content>
                                        <Path Fill="Black" Stroke="Green"
                                              Data="M 1,10 10,1 20,10 Z" />
                                    </RepeatButton.Content>
                                </RepeatButton>
```

```
                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                    <RepeatButton.Content>
                                        <Path Fill="Black" Stroke="Green"
                                              Data="M 10,10 1,1 20,1 Z" />
                                    </RepeatButton.Content>
                                </RepeatButton>

                            </StackPanel>

                        </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

Η δεύτερη στήλη του Grid περιέχει τα repeat buttons μέσα σε ένα StackPanel. Το πλάτος της δεύτερης στήλης του Grid προσδιορίζεται με βάση την τιμή της ιδιότητας ButtonsWidth. Για να μετατραπεί η τιμή Double της ιδιότητας ButtonsWidth σε τιμή τύπου GridLength χρησιμοποιούμε μία κλάση μετατροπής τιμής (ValueConverter). Θα τη βρείτε στο Παράρτημα 2.

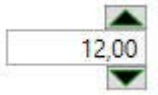
Παράδειγμα custom Style και ControlTemplate

Στο επόμενο παράδειγμα, δημιουργούμε ένα custom Style και ControlTemplate. Συγκεκριμένα:

- Ορίζουμε ότι **όταν λάβει την εστίαση, το background να γίνει κίτρινο**. Αυτό γίνεται, θέτοντας στο **Focused state** το σχετικό Storyboard που περιέχει ένα ColorAnimation που κάνει animate το background color σε Yellow σε χρόνο 0.2 sec.

- Στη δομή του ControlTemplate, **αλλάζουμε θέση στα up-down buttons**. Το up-button βρίσκεται πάνω δεξιά από το TextBox ενώ το down-button βρίσκεται από κάτω δεξιά (3 γραμμές). Για να επιτευχθεί αυτό αλλάζει λίγο η διάταξη του Grid container, που τώρα περιέχει 3 RowDefinintions. Η λειτουργικότητα του control παραμένει η ίδια.

Η εικόνα του custom DoubleUpDown σε κανονική κατάσταση - **Normal state**:



Η εικόνα του custom DoubleUpDown όταν έχει την εστίαση - **Focused state**:

Ο XAML κώδικας για το custom Style και ControlTemplate του παραπάνω DoubleUpDown είναι ο εξής:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownCont
                                                                              rols"

<Style TargetType="{x:Type zeus:DoubleUpDown}" >

    <Setter Property="Focusable" Value="True"/>
    <Setter Property="BorderBrush" Value="Black" />
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="Background"  Value="Transparent" />
    <Setter Property="Width" Value ="120"/>
    <Setter Property="Height" Value="23"/>

    <Setter Property="Template">

        <Setter.Value>

            <ControlTemplate TargetType="{x:Type zeus:DoubleUpDown}">

                <ControlTemplate.Resources >
                    <zeus:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                </ControlTemplate.Resources>
```

```
        <!-- Root element -->
        <Border Name="Border"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="{TemplateBinding BorderThickness}">

            <VisualStateManager.VisualStateGroups >

                <VisualStateGroup Name="CommonStates">

                    <VisualState Name="Normal"/>

                    <VisualState Name="Disabled">

                        <Storyboard>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Background.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
 Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Foreground.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ObjectAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                            Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                <DiscreteObjectKeyFrame KeyTime="0"
                     Value="{Binding Source={x:Static FontWeights.Normal}}" />
                            </ObjectAnimationUsingKeyFrames>

                        </Storyboard>

                    </VisualState>


                </VisualStateGroup>

                <VisualStateGroup Name="FocusStates">

                   <VisualState Name="Focused" >

                        <Storyboard >

                            ColorAnimation Storyboard.TargetName="PART_Value"
                                 Storyboard.TargetProperty="Background.Color"
                                           To="Yellow" Duration="0:0:0.2"/>

                        </Storyboard>

                    </VisualState>

                    <VisualState Name="Unfocused"/>

                </VisualStateGroup>
```

```
                    </VisualStateManager.VisualStateGroups>

                    <!-- Content -->
                   <Grid Name="mainGrid" Background="Transparent"
                                         Margin="{TemplateBinding Padding}">

                            <Grid.RowDefinitions >
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                            </Grid.RowDefinitions>

                            <TextBox Name="PART_Value" Grid.Row="1"
              HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                  VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                   Background="{TemplateBinding Background }"
                                   Foreground="{TemplateBinding Foreground }"
                                   Padding="2,0,2,0"
                               Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                           </TextBox>

                           <Viewbox Stretch="Uniform" Grid.Row="0" Height="12"
                                    HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementUp"
                                              Focusable="False"  >
                                     <RepeatButton.Content>
                                           <Path Fill="Black" Stroke="Green"
                                                 Data="M 1,10 10,1 20,10 Z" />
                                      </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                           <Viewbox  Stretch="Uniform" Grid.Row="2" Height="12"
                                     HorizontalAlignment="Right">

                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                     <RepeatButton.Content>
                                          <Path Fill="Black" Stroke="Green"
                                                Data="M 10,10 1,1 20,1 Z" />
                                     </RepeatButton.Content>
                                </RepeatButton>

                           </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

DecimalUpDown

Ένα **UpDown control** που επιτρέπει την **είσοδο μόνο αριθμητικών δεδομένων, τύπου Decimal**. Η τιμή απαιτείται (required).

Σύνταξη:

VB:

```
<TemplateVisualState(Name:="Normal", GroupName:="CommonStates"),
TemplateVisualState(Name:="MouseOver", GroupName:="CommonStates"),
TemplateVisualState(Name:="Disabled", GroupName:="CommonStates"),
TemplateVisualState(Name:="Focused", GroupName:="FocusStates"),
TemplateVisualState(Name:="Unfocused", GroupName:="FocusStates"),
TemplatePart(Name:="PART_Value", Type:=GetType(TextBox)),
TemplatePart(Name:="PART_IncrementUp", Type:=GetType(RepeatButton)),
TemplatePart(Name:="PART_IncrementDown", Type:=GetType(RepeatButton))>
<DefaultProperty("Value")>
Public Class DecimalUpDown
             Inherits UpDownBase
```

XAML Object Element Usage:

Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

Χρήση:

```
<zeus:DecimalUpDown ... />
```

Παράδειγμα:

Στο παράδειγμα που ακολουθεί, τοποθετούμε ένα **DecimalUpDown** μέσα σε ένα **Grid** ενός παραθύρου. Συνδέουμε την **ιδιότητα Value** με την **ιδιότητα Amount1**, τύπου **Decimal**, ενός **Customer** αντικειμένου (source object). Η **κλάση Customer** ορίζεται στο project και συνεπώς πρέπει να εισάγουμε και το αντίστοιχο namespace με alias p. Επίσης, ορίζουμε ένα **ErrorTemplate** για το **Validation** (όπου μέσω αυτού θα εμφανιστεί το μήνυμα λάθους που ορίζουμε στην **ιδιότητα ErrorMessageOnIncorrectValueType** ή στην **ιδιότητα ErrorMessageOnValueOutOfRange**). Επιπλέον, ορίζουμε ένα **Style**, στοχευμένο στην κλάση UpDownBase, για την ομοιόμορφη εμφάνιση των UpDown controls στο παράθυρο.

```
<Window x:Class="MainWindow"
         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:p="clr-namespace:UpDownControlsTestProject"
        xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
        mc:Ignorable="d"
        Title="UpDownControls Sample Project"
        Height="350" Width="525" Loaded="Window_Loaded" >

    <Window.Resources>

        <!-- The Validation Error ControlTemplate -->
        <ControlTemplate x:Key="validationErrorTemplate">

            <StackPanel Orientation="Horizontal">

                <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                        Background="#11FF0000"
                        IsHitTestVisible="False" >
                    <AdornedElementPlaceholder x :Name="placeholder"  />
                </Border>

                <Grid  Margin="20,0,0,0"  >
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

        <Style TargetType="{x:Type TextBlock}">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="HorizontalAlignment" Value="Right" />
            <Setter Property="Margin" Value="5"/>
        </Style>

        <!-- The style base for the UpDown controls. -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>
```

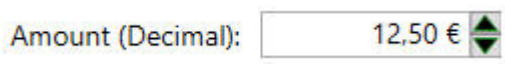
```
                    <Trigger Property="HasError" Value="True">
                        <Setter Property="Foreground" Value="Red"/>
                        <Setter Property="BorderBrush" Value="Red"/>
                        <Setter Property="BorderThickness" Value="2"/>
                    </Trigger>

                </Style.Triggers>

            </Style>

        <!-- A sample Customer resource object for the Bindings -->
        <p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
                                     Amount1="12"/>


    </Window.Resources>

    <Grid  Margin="10,40,10,10"
           DataContext="{Binding Source={StaticResource customer}}">

        ...

        <TextBlock Text="Amount (Decimal):" />
        <zeus:DecimalUpDown Name="UpDown1" Grid.Column="1"
                            Minimum="1" Maximum="500"
                            Increment="0.50" Language="el"
                            HorizontalContentAlignment="Right"
                 ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                 ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                  500. Please, retry."
                          Value="{Binding Amount1}"  ValueFormat="C2"
                         ValueChanged="UpDown1_ValueChanged"
                         Style="{StaticResource UpDownStyle}"/>

       ...

    </Grid>


</Window>
```

VB:

```
Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
UpDown1.ErrorTemplate = errTemplate
```

- Για την **κλάση Customer** δείτε το Παράρτημα 1.

Μία κανονική κατάσταση του **DecimalUpDown (νομισματική μορφή με culture "el")**:



Παρακάτω, βλέπουμε την **ένδειξη λάθους** όταν ο χρήστης εισάγει μία **λανθασμένη (ως προς το εύρος) τιμή**. Αν πάει το δείκτη του ποντικιού πάνω στο **θαυμαστικό** θα εμφανιστεί το μήνυμα λάθους "`The number must be between 1 and 500. Please, retry.`"

Ιδιότητες

Όνομα	Περιγραφή
ButtonsWidth	Καθορίζει το **πλάτος**, σε pixels, **των buttons αυξομείωσης του control**. (*Το ύψος είναι σταθερά ορισμένο στο μέσον του ύψους του control)*. (Κληρονομείται από την UpDownBase).
ErrorMessageOnIncorrectValueType	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι τύπου Decimal. (Κληρονομείται από την UpDownBase).
ErrorMessageOnValueOutOfRange	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι στο αποδεκτό εύρος (Minimum-Maximum). (Κληρονομείται από την UpDownBase).
ErrorTemplate	Καθορίζει ένα **ControlTemplate** που θα εφαρμόζεται **όταν το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
HasError	Αν είναι **True**, τότε **το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
Increment	Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Decimal. Η default τιμή είναι 1.
IsMinimum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο ελάχιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Minimum. (Κληρονομείται από την UpDownBase).
IsMaximum	Αν είναι **True**, τότε η **τρέχουσα τιμή έχει φτάσει στο μέγιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα Maximum. (Κληρονομείται από την UpDownBase).
Minimum	Καθορίζει το **ελάχιστο όριο** στο control, τύπου Decimal. Η default τιμή είναι Decimal.MinValue.

Maximum	Καθορίζει το **μέγιστο όριο** στο control, τύπου Decimal. Η default τιμή είναι Decimal.MaxValue.
Value	Η **τρέχουσα τιμή** στο control, τύπου Decimal. Η default τιμή είναι 0.
ValueFormat	Καθορίζει το **StringFormat** της τιμής του control. (Κληρονομείται από την UpDownBase).

Increment

Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Decimal.

Σύνταξη:

VB:

```
Public Property Increment As Decimal
```

Τύπος: System.Decimal

Προσδιορίζουμε μία θετική τιμή για το βήμα αύξησης της τιμής. Η default τιμή είναι 1.

Dependency Property Information:

Identifier field: `IncrementProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το βήμα αύξησης στο UpDown1 σε 1.50.

XAML:

```
<zeus:DecimalUpDown x:Name="UpDown1" Minimum="1" Maximum="500" Increment="1.50"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="C2"
                    ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Increment = 1.50
```

Minimum

Καθορίζει το **ελάχιστο όριο** στο control, τύπου Decimal.

Σύνταξη:

VB:

```
Public Property Minimum As Decimal
```

Τύπος: System.Decimal

Προσδιορίζουμε μία τιμή, ως το ελάχιστο αποδεκτό όριο. Η default τιμή είναι Decimal.MinValue.

Dependency Property Information:

Identifier field: `MinimumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το ελάχιστο όριο στο UpDown1 σε 10.50.

XAML:

```
<zeus:DecimalUpDown x:Name="UpDown1" Minimum="10.50" Maximum="500" Increment="0.50"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="C2"
                    ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Minimum = 10.50
```

Maximum

Καθορίζει το **μέγιστο όριο** στο control, τύπου Decimal.

Σύνταξη:

VB:

```
Public Property Maximum As Decimal
```

Τύπος: System.Decimal

Προσδιορίζουμε μία τιμή, ως το μέγιστο αποδεκτό όριο. Η default τιμή είναι Decimal.MaxValue.

Dependency Property Information:

Identifier field: `MaximumProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το μέγιστο όριο στο UpDown1 σε 900.

XAML:

```
<zeus:DecimalUpDown x:Name="UpDown1" Minimum="10.50" Maximum="900" Increment="0.50"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="C2" Language="el"
                    ButtonsWidth ="25" ... />
```

VB:

```
UpDown1.Maximum = 900
```

Value

Η **τρέχουσα τιμή** στο control, τύπου Decimal.

Σύνταξη:

VB:

```
Public Property Value As Decimal
```

Τύπος: System.Decimal

Η τρέχουσα τιμή. Η default τιμή είναι 0.

Dependency Property Information:

Identifier field: `ValueProperty`

Παρατηρήσεις:

Σε μία WPF εφαρμογή, η λογική χρήση της ιδιότητας αυτής είναι μέσω μίας Binding έκφρασης. Με αυτόν τον τρόπο, η τρέχουσα τιμή συνδέεται με μία source πηγή (π.χ. μία public ιδιότητα ενός data object). Η τιμή προέρχεται από το source object και οποιαδήποτε μεταβολή της επιστρέφει στο source object (σε ένα two-way mode, που είναι και το default). Συμπερασματικά, **συνιστάται η χρήση της με Binding έφραση**.

Παράδειγμα:

Στο επόμενο παράδειγμα, η τιμή της **ιδιότητας Value** στο **UpDown1** συνδέεται, **μέσω Binding έκφρασης**, με την **ιδιότητα Amount1** του **source object**. Το τελευταίο (ένα **αντικείμενο Customer**) τίθεται ως πηγή δεδομένων στην **ιδιότητα DataContext** σε ένα container, που εδώ είναι το **Grid** panel. Οι τιμές των ιδιοτήτων του αντικειμένου Customer τίθενται στο τμήμα Window.Resources, αλλά θα μπορούσαν να τεθούν και στο VB κώδικα, σε κάποιον event handler.

```
<Window x:Class="MainWindow"
          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
          xmlns:p="clr-namespace:UpDownControlsTestProject"
          xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
          mc:Ignorable="d"
          Title="UpDownControls Sample Project"
          Height="350" Width="525" Loaded="Window_Loaded" >

        <Window.Resources>
```

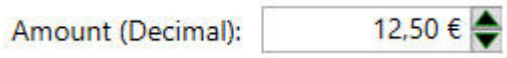
```
<!-- The Validation Error ControlTemplate -->
<ControlTemplate x:Key="validationErrorTemplate">

    <StackPanel Orientation="Horizontal">

        <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                Background="#11FF0000"
                IsHitTestVisible="False" >
            <AdornedElementPlaceholder x :Name="placeholder"  />
        </Border>

        <Grid  Margin="20,0,0,0"  >
            <Ellipse  Width="20" Height="20"  Fill="Red"  />
            <TextBlock Foreground="White" FontSize="14" Text="!"
                       FontWeight="ExtraBold"
                       VerticalAlignment="Center"
                       HorizontalAlignment="Center" />
            <Grid.ToolTip>
                <TextBlock Text="{Binding Path=/ErrorContent}"/>
            </Grid.ToolTip>
        </Grid>

    </StackPanel>

</ControlTemplate>

<Style TargetType="{x:Type TextBlock }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="HorizontalAlignment" Value="Right" />
    <Setter Property="Margin" Value="5"/>
</Style>

<!-- The style base for the UpDown controls. -->
<Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Background" Value="White"/>
    <Setter Property="HorizontalAlignment" Value="Left" />
    <Setter Property="Margin" Value="5,0,0,0"/>

    <Style.Triggers >
        <Trigger Property="IsMinimum" Value="True">
            <Setter Property="Background" Value="Orange" />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="IsMaximum" Value="True">
            <Setter Property="Background" Value="Green"  />
            <Setter Property="Foreground"  Value="White" />
        </Trigger>

        <Trigger Property="HasError" Value="True">
            <Setter Property="Foreground" Value="Red"/>
            <Setter Property="BorderBrush" Value="Red"/>
            <Setter Property="BorderThickness" Value="2"/>
        </Trigger>

    </Style.Triggers>

  </Style>

<!-- A sample Customer resource object for the Bindings -->
<p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
```

```
                                        Amount1="12"/>

        </Window.Resources>

        <Grid  Margin="10,40,10,10"
               DataContext="{Binding Source={StaticResource customer}}">

            ...

            <TextBlock Text="Amount (Decimal):" />
            <zeus:DecimalUpDown Name="UpDown1" Grid.Column="1"
                                Minimum="1" Maximum="500"
                                Increment="0.50" Language="el"
                                HorizontalContentAlignment="Right"
                        ErrorMessageOnIncorrectValueType = "Please, enter a valid value!"
                        ErrorMessageOnValueOutOfRange = "The number must be between 1 and
                                                                    500. Please, retry."
                                  Value="{Binding Amount1}"  ValueFormat="C2"
                                 ValueChanged="UpDown1_ValueChanged"
                                 Style="{StaticResource UpDownStyle}"/>

           ...

        </Grid>


</Window>
```

VB:

```
Private Sub Window_Loaded(sender As Object, e As RoutedEventArgs)

    Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
    UpDown1.ErrorTemplate = errTemplate

End Sub
```

Βλέπουμε την αρχική κατάσταση του UpDown1. Η τιμή 12.50 προέρχεται από την ιδιότητα Amount1 του αντικειμένου Customer:



Αν ο χρήστης φτάσει στην μέγιστη τιμή (500) τότε, λόγω του style trigger, το DecimalUpDown μορφοποιείται ως εξής:

Παράλληλα, απενεργοποιείται και το up arrow button. Αν ο χρήστης, πληκτρολογήσει απ' ευθείας

μία **τιμή εκτός ορίων**, τότε το control εισέρχεται σε **κατάσταση λάθους**:

Επίσης, αν ο χρήστης, πληκτρολογήσει μία **τιμή λανθασμένου τύπου**, τότε το control πάλι εισέρχεται σε **κατάσταση λάθους**:

Συμβάντα

Όνομα	Περιγραφή
MinimumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.
MaximumChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.
ValueChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

MinimumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Minimum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MinimumChanged As RoutedPropertyChangedEventHandler(Of Decimal)
```

XAML attribute usage:

```
<zeus:DecimalUpDown MinimumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MinimumChanged του UpDown1:

XAML:

```
<zeus:DecimalUpDown x:Name="UpDown1" Minimum="0" Maximum="500" Increment="0.50"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="C2" Language="el"
                    MinimumChanged="UpDown1_MinimumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MinimumChanged(sender As  Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Decimal))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new minimum value is {0}", e.NewValue))

     End If

End Sub
```

MaximumChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Maximum**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event MaximumChanged As RoutedPropertyChangedEventHandler(Of Decimal)
```

XAML attribute usage:

```
<zeus:DecimalUpDown MaximumChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν MaximumChanged του UpDown1:

XAML:

```
<zeus:DecimalUpDown x:Name="UpDown1" Minimum="0" Maximum="500" Increment="0.50"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="C2" Language="el"
                    MaximumChanged="UpDown1_MaximumChanged" ... />
```

VB:

```
 Private Sub UpDown1_MaximumChanged(sender As Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Decimal))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The new maximum value is {0}", e.NewValue))

     End If

End Sub
```

ValueChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα Value**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event ValueChanged As RoutedPropertyChangedEventHandler(Of Decimal)
```

XAML attribute usage:

```
<zeus:DecimalUpDown ValueChanged ="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον eventHandler για το συμβάν ValueChanged του UpDown1:

XAML:

```
<zeus:DecimalUpDown x:Name="UpDown1" Minimum="0" Maximum="500" Increment="0.50"
                    HorizontalContentAlignment="Right"
                    Value="{Binding Amount1}"  ValueFormat="C2" Language="el"
                    ValueChanged="UpDown1_ValueChanged" ... />
```

VB:

```
 Private Sub UpDown1_ValueChanged(sender As Object, _
                                  e As RoutedPropertyChangedEventArgs(Of Decimal))

     If e IsNot Nothing Then

         MessageBox.Show(String.Format("The current value is {0}", e.NewValue))

     End If

End Sub
```

Styles και Templates

- Parts και States
- Το default Style και ControlTemplate
- Παράδειγμα custom Style και ControlTemplate

Parts και States

Το default ControlTemplate περιλαμβάνει κάποια **part names** και **visual states**. Μπορείτε να τροποποιήσετε το default ControlTemplate ώστε να δώσετε στο control μία μοναδική εμφάνιση. Το default ControlTemplate μπορείτε να το δείτε εδώ ώστε να πάρετε μία ιδέα και με βάση αυτό να δημιουργήσετε ένα δικό σας. Ένα παράδειγμα custom ControlTemplate μπορείτε να δείτε εδώ.

Ο παρακάτω πίνακας εμφανίζει τα **part names** του **DecimalUpDown** control:

Part	Τύπος	Περιγραφή
PART_Value	TextBox	Το standard TextBox control.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την αύξηση της τιμής.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την μείωση της τιμής.

Ο παρακάτω πίνακας εμφανίζει τα **visual states** του **DecimalUpDown** control:

VisualState	VisualStateGroup	Περιγραφή
Normal	CommonStates	Το default state.
MouseOver	CommonStates	Το state όταν ο δείκτης του ποντικιού είναι πάνω από το control.
Disabled	CommonStates	Το state όταν το control είναι disabled.
Focused	FocusStates	Το state όταν το control έχει το focus.
Unfocused	FocusStates	Το state όταν το control δεν έχει το focus.

Το default ControlTemplate έχει καθορισμένο μόνο το Disabled state. Μπορείτε να δημιουργήσετε ένα custom ControlTemplate για να το αλλάξετε ή/και για να καθορίσετε τα υπόλοιπα.

Το default Style και ControlTemplate

Ο XAML κώδικας για το **default Style** και **ControlTemplate** φαίνεται παρακάτω. Μπορείτε να βασιστείτε σε αυτόν για να δημιουργήσετε μία μικρή ή μεγάλη παραλλαγή του δικού σας custom Style και ControlTemplate:

```
    xmlns:z="clr-namespace:Zeus.WPF.Controls.UpDownControls">

    <Style TargetType="{x:Type z:DecimalUpDown}" >

        <Setter Property="Focusable" Value="True"/>
        <Setter Property="BorderBrush" Value="Black" />
        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="Background"  Value="Transparent" />
        <Setter Property="Width" Value ="120"/>
        <Setter Property="Height" Value="23"/>

        <Setter Property="Template">

            <Setter.Value>

                <ControlTemplate TargetType="{x:Type z:DecimalUpDown}">

                    <ControlTemplate.Resources >
                        <z:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                    </ControlTemplate.Resources>

                    <!-- Root element -->
                    <Border Name="Border"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">

                        <VisualStateManager.VisualStateGroups >

                            <VisualStateGroup Name="CommonStates">

                                <VisualState Name="Normal"/>

                                <VisualState Name="Disabled">

                                    <Storyboard >

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Background.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                     Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_Value"
                                            Storyboard.TargetProperty ="Foreground.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                    Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ObjectAnimationUsingKeyFrames
```

```
                                        Storyboard.TargetName="PART_Value"
                                    Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                        <DiscreteObjectKeyFrame KeyTime="0"
                                 Value="{Binding Source={x:Static FontWeights.Normal}}" />
                                        </ObjectAnimationUsingKeyFrames>

                                    </Storyboard>

                                </VisualState>


                            </VisualStateGroup>

                            <VisualStateGroup Name="FocusStates">

                                <VisualState Name="Focused" />
                                <VisualState Name="Unfocused"/>

                            </VisualStateGroup>

                        </VisualStateManager.VisualStateGroups>

                        <!-- Content -->
                        <Grid Name="mainGrid" Background="Transparent" >

                            <Grid.RowDefinitions >
                                <RowDefinition Height="*" />
                            </Grid.RowDefinitions>

                            <!-- The width of the second Grid column is about for the
                                 RepeatButtons and depends on ButtonsWidth property -->
                            <Grid.ColumnDefinitions >
                                <ColumnDefinition Width="*" />
                                <ColumnDefinition
                                      Width="{Binding RelativeSource={Relative Source
                              Mode=FindAncestor, AncestorType={x:Type z:DecimalUpDown}},
Path=ButtonsWidth, Converter={StaticResource doubleToGridLengthConverter}}" />
                            </Grid.ColumnDefinitions>

                             <TextBox Name="PART_Value"
                HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                    VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                                Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                            </TextBox>

                            <Viewbox Stretch="Fill"   Grid.Column="1">

                                <StackPanel Name="stkRepeatButtons"
                                            VerticalAlignment="Center"
                                            HorizontalAlignment="Left"  Focusable="False">

                                    <RepeatButton Name="PART_IncrementUp"
                                                  Focusable="False"  >
                                        <RepeatButton.Content>
                                            <Path Fill="Black" Stroke="Green"
                                                  Data="M 1,10 10,1 20,10 Z" />
                                        </RepeatButton.Content>
                                    </RepeatButton>
```

```
                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                    <RepeatButton.Content>
                                        <Path Fill="Black" Stroke="Green"
                                              Data="M 10,10 1,1 20,1 Z" />
                                    </RepeatButton.Content>
                                </RepeatButton>

                            </StackPanel>

                        </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

Η δεύτερη στήλη του Grid περιέχει τα repeat buttons μέσα σε ένα StackPanel. Το πλάτος της δεύτερης στήλης του Grid προσδιορίζεται με βάση την τιμή της ιδιότητας ButtonsWidth. Για να μετατραπεί η τιμή Double της ιδιότητας ButtonsWidth σε τιμή τύπου GridLength χρησιμοποιούμε μία κλάση μετατροπής τιμής (ValueConverter). Θα τη βρείτε στο Παράρτημα 2.

Παράδειγμα custom Style και ControlTemplate

Στο επόμενο παράδειγμα, δημιουργούμε ένα custom Style και ControlTemplate. Συγκεκριμένα:

- Ορίζουμε ότι **όταν λάβει την εστίαση, το background να γίνει κίτρινο**. Αυτό γίνεται, θέτοντας στο **Focused state** το σχετικό Storyboard που περιέχει ένα ColorAnimation που κάνει animate το background color σε Yellow σε χρόνο 0.2 sec.

- Στη δομή του ControlTemplate, **αλλάζουμε θέση στα up-down buttons**. Το up-button βρίσκεται πάνω δεξιά από το TextBox ενώ το down-button βρίσκεται από κάτω δεξιά (3 γραμμές). Για να επιτευχθεί αυτό αλλάζει λίγο η διάταξη του Grid container, που τώρα περιέχει 3 RowDefinintions. Η λειτουργικότητα του control παραμένει η ίδια.

Η εικόνα του custom DecimalUpDown σε κανονική κατάσταση - **Normal state**:

Η εικόνα του custom DecimalUpDown όταν έχει την εστίαση - **Focused state**:

Ο XAML κώδικας για το custom Style και ControlTemplate του παραπάνω DecimalUpDown είναι ο εξής:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownCont
                                                                        rols"

<Style TargetType="{x:Type zeus:DecimalUpDown}" >

    <Setter Property="Focusable" Value="True"/>
    <Setter Property="BorderBrush" Value="Black" />
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="Background"  Value="Transparent" />
    <Setter Property="Width" Value ="120"/>
    <Setter Property="Height" Value="23"/>

    <Setter Property="Template">

        <Setter.Value>

            <ControlTemplate TargetType="{x:Type zeus:DecimalUpDown}">

                <ControlTemplate.Resources >
                    <zeus:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
```

```
        </ControlTemplate.Resources>

        <!-- Root element -->
        <Border Name="Border"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="{TemplateBinding BorderThickness}">

            <VisualStateManager.VisualStateGroups >

                <VisualStateGroup Name="CommonStates">

                    <VisualState Name="Normal"/>

                    <VisualState Name="Disabled">

                        <Storyboard>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Background.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
 Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                                Storyboard.TargetProperty ="Foreground.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ObjectAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_Value"
                            Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                <DiscreteObjectKeyFrame KeyTime="0"
                     Value="{Binding Source={x:Static FontWeights.Normal}}" />
                            </ObjectAnimationUsingKeyFrames>

                        </Storyboard>

                    </VisualState>


                </VisualStateGroup>

                <VisualStateGroup Name="FocusStates">

                   <VisualState Name="Focused" >

                        <Storyboard >

                            ColorAnimation Storyboard.TargetName="PART_Value"
                                 Storyboard.TargetProperty="Background.Color"
                                           To="Yellow" Duration="0:0:0.2"/>

                        </Storyboard>

                    </VisualState>

                    <VisualState Name="Unfocused"/>

                </VisualStateGroup>
```

```
                    </VisualStateManager.VisualStateGroups>

                    <!-- Content -->
                   <Grid Name="mainGrid" Background="Transparent"
                                         Margin="{TemplateBinding Padding}">

                            <Grid.RowDefinitions >
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                            </Grid.RowDefinitions>

                            <TextBox Name="PART_Value" Grid.Row="1"
              HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment }"
                  VerticalContentAlignment="{TemplateBinding VerticalContentAlignment }"
                                    Background="{TemplateBinding Background }"
                                    Foreground="{TemplateBinding Foreground }"
                                    Padding="2,0,2,0"
                                 Validation.ErrorTemplate="{TemplateBinding ErrorTemplate}">
                        </TextBox>

                        <Viewbox Stretch="Uniform" Grid.Row="0" Height="12"
                                 HorizontalAlignment="Right">

                            <RepeatButton Name="PART_IncrementUp"
                                          Focusable="False"  >
                                <RepeatButton.Content>
                                      <Path Fill="Black" Stroke="Green"
                                            Data="M 1,10 10,1 20,10 Z" />
                                 </RepeatButton.Content>
                            </RepeatButton>

                        </Viewbox>

                        <Viewbox  Stretch="Uniform" Grid.Row="2" Height="12"
                                  HorizontalAlignment="Right">

                            <RepeatButton Name="PART_IncrementDown"
                                         Focusable="False" >
                                <RepeatButton.Content>
                                     <Path Fill="Black" Stroke="Green"
                                           Data="M 10,10 1,1 20,1 Z" />
                                </RepeatButton.Content>
                            </RepeatButton>

                        </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

Ημερομηνίες

Στα επόμενα, παρουσιάζονται τα **UpDownControls** για **είσοδο ημερομηνιακών δεδομένων.**

- DatePickerUpDown

DatePickerUpDown

Ένα **UpDown control** που επιτρέπει την **είσοδο ημερομηνιακών δεδομένων, τύπου Nullable(of Date)** ή **Nullable(of DateTime)**. Η τιμή δεν απαιτείται (can be null).

Σύνταξη:

VB:

```
<TemplateVisualState(Name:="Normal", GroupName:="CommonStates"),
TemplateVisualState(Name:="MouseOver", GroupName:="CommonStates"),
TemplateVisualState(Name:="Disabled", GroupName:="CommonStates"),
TemplateVisualState(Name:="Focused", GroupName:="FocusStates"),
TemplateVisualState(Name:="Unfocused", GroupName:="FocusStates"),
TemplatePart(Name:="PART_DatePicker", Type:=GetType(DatePicker)),
TemplatePart(Name:="PART_IncrementUp", Type:=GetType(RepeatButton)),
TemplatePart(Name:="PART_IncrementDown", Type:=GetType(RepeatButton))>
<DefaultProperty("SelectedDate")>
Public Class DatePickerUpDown
             Inherits UpDownBase
```

XAML Object Element Usage:

Εισαγωγή namespace:

```
xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownContr
ols"
```

Χρήση:

```
<zeus:DatePickerUpDown ... />
```

Παράδειγμα:

Στο παράδειγμα που ακολουθεί, τοποθετούμε ένα **DatePickerUpDown** μέσα σε ένα **Grid** ενός παραθύρου. Συνδέουμε την **ιδιότητα SelectedDate** με την **ιδιότητα DateInsertionNullable** , τύπου **Nullable(Of Date)**, ενός **Customer** αντικειμένου (source object). Η **κλάση Customer** ορίζεται στο project και συνεπώς πρέπει να εισάγουμε και το αντίστοιχο namespace με alias p. Επίσης, ορίζουμε ένα **ErrorTemplate** για το **Validation** (όπου μέσω αυτού θα εμφανιστεί το μήνυμα λάθους που ορίζουμε στην **ιδιότητα ErrorMessageOnValueOutOfRange**). Επιπλέον, ορίζουμε ένα **Style**, στοχευμένο στην κλάση UpDownBase, για την ομοιόμορφη εμφάνιση των UpDown controls στο παράθυρο.

```
<Window x:Class="MainWindow"
```

```
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:p="clr-namespace:UpDownControlsTestProject"
        xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
        mc:Ignorable="d"
        Title="UpDownControls Sample Project"
        Height="350" Width="525" Loaded="Window_Loaded" >

    <Window.Resources>

        <!-- The Validation Error ControlTemplate -->
        <ControlTemplate x:Key="validationErrorTemplate">

            <StackPanel Orientation="Horizontal">

                <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                        Background="#11FF0000"
                        IsHitTestVisible="False" >
                    <AdornedElementPlaceholder x :Name="placeholder"  />
                </Border>

                <Grid  Margin="20,0,0,0"  >
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

        <Style TargetType="{x:Type TextBlock}">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="HorizontalAlignment" Value="Right" />
            <Setter Property="Margin" Value="5"/>
        </Style>

        <!-- The style base for the UpDown controls. -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>
```

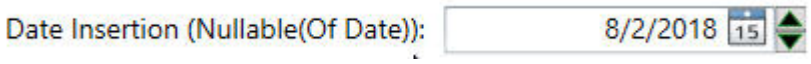
```
                    <Trigger Property="HasError" Value="True">
                        <Setter Property="Foreground" Value="Red"/>
                        <Setter Property="BorderBrush" Value="Red"/>
                        <Setter Property="BorderThickness" Value="2"/>
                    </Trigger>

                </Style.Triggers>

            </Style>

            <!-- A sample Customer resource object for the Bindings -->
            <p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
                                         Amount1="12.50"/>


        </Window.Resources>

        <Grid  Margin="10,40,10,10"
               DataContext="{Binding Source={StaticResource customer}}">

            ...

            <TextBlock Text="Date Insertion (Nullable(Of Date)):" />
            <zeus:DatePickerUpDown Name="dateInsertionUpDown" Width="180"
                                   Increment="1" IncrementType="Days" Language="el"
                                   HorizontalContentAlignment="Right"
                                   ErrorMessageOnValueOutOfRange = "The date is out of
                                                                    range. Please, retry!"
                                 DisplayDateStart="2/1/2018" DisplayDateEnd="2/20/2018"
                                 SelectedDate="{Binding DateInsertionNullable}"
                                 SelectedDateFormat="Short"
                                 DatePickerWatermark="Επιλέξτε Ημερ/νία"
                                 Style="{StaticResource UpDownStyle}"
                               SelectedDateChanged="dateInsertion_SelectedDateChanged" />

           ...

        </Grid>


    </Window>
```

VB:

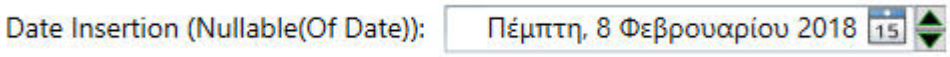
```
Dim errTemplate As ControlTemplate = _
                CType(Me.FindResource("validationErrorTemplate"), ControlTemplate)
UpDown1.ErrorTemplate = errTemplate
```

- Για την **κλάση Customer** δείτε το Παράρτημα 1.

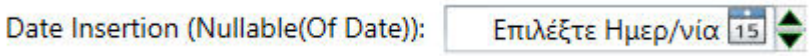
Μία κανονική κατάσταση του **DatePickerUpDown (culture "el", short format)**:



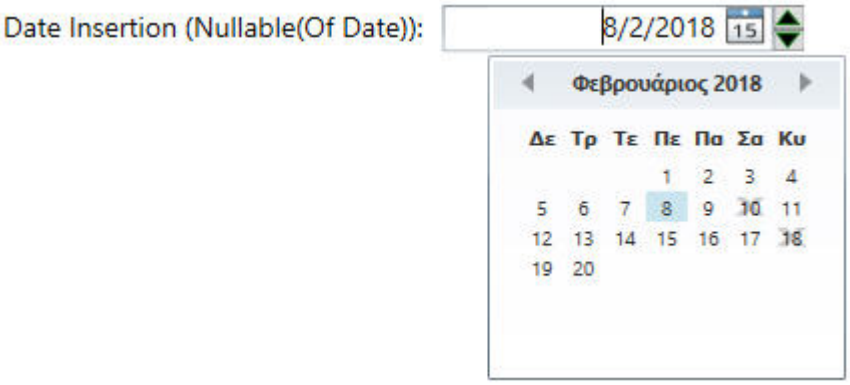
Μία κανονική κατάσταση του **DatePickerUpDown (culture "el", long format)**:



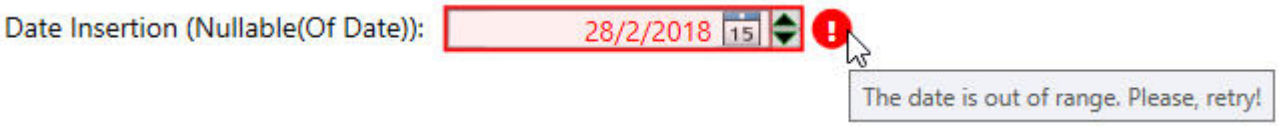
Όταν δεν υπάρχει επιλεγμένη ημερομηνία, εμφανίζεται το empty watermark:



Με ανεπτυγμένο το calendar. Οι ημερομηνίες $10/2/2018$ και $18/2/2018$ έχουν δηλωθεί ως blackout dates:



Παρακάτω, βλέπουμε την **ένδειξη λάθους** όταν ο χρήστης εισάγει μία **λανθασμένη (ως προς το εύρος) τιμή**. Αν πάει το δείκτη του ποντικιού πάνω στο **θαυμαστικό** θα εμφανιστεί το μήνυμα λάθους "`The date is out of range. Please, retry!`"



Απαριθμήσεις

Όνομα	Περιγραφή
IncrementTypeEnum	Περιλαμβάνει τις **τιμές για τον τύπο της βηματικής αυξομείωσης της ημερομηνίας στο control**. Επηρεάζει το πώς "μεταφράζεται" η τιμής της ιδιότητας Increment.

IncrementTypeEnum

Καθορίζει τον **τύπο της βηματικής αυξομείωσης της ημερομηνίας στο control**.

Σύνταξη:

VB:

```
Public Enum IncrementTypeEnum
```

Μέλη:

Τιμή	Περιγραφή
Days	Η αυξομείωση αφορά **ημέρες**.
Months	Η αυξομείωση αφορά **μήνες**.
Years	Η αυξομείωση αφορά **έτη**.

Ιδιότητες

Όνομα	Περιγραφή
ButtonsWidth	Καθορίζει το **πλάτος**, σε pixels, **των buttons αυξομείωσης του control**. (*Το ύψος είναι σταθερά ορισμένο στο μέσον του ύψους του control)*. (Κληρονομείται από την UpDownBase).
CalendarStyle	Καθορίζει το **Style** που θα χρησιμοποιηθεί για την εμφάνιση του **ενσωματωμένου calendar**.
DatePickerWatermark	Καθορίζει το **κείμενο** που θα εμφανίζεται **όταν δεν υπάρχει επιλεγμένη ημερομηνία** (SelectedDate=Nothing ή Text="").
DisplayDate	Καθορίζει την **ημερομηνία που θα εμφανίζεται στο ενσωματωμένο calendar όταν δεν έχει επιλεγεί συγκεκριμένη ημερομηνία.** Είναι τύπου Nullable(Of DateTime). Η default τιμή είναι Nothing.
DisplayDateStart	Καθορίζει την **ημερομηνία αρχής που θα εμφανίζεται στο calendar**, τύπου Nullable(Of DateTime). Λειτουργεί ως το **ελάχιστο αποδεκτό όριο**. Η default τιμή είναι Nothing.
DisplayDateEnd	Καθορίζει την **ημερομηνία τέλους που θα εμφανίζεται στο calendar**, τύπου Nullable(Of DateTime). Λειτουργεί ως το **μέγιστο αποδεκτό όριο**. Η default τιμή είναι Nothing.
ErrorMessageOnValueOutOfRange	Καθορίζει το **μήνυμα λάθους** που θα εμφανίζεται όταν ο χρήστης εισάγει τιμή που δεν είναι στο αποδεκτό εύρος (DisplayDateStart-DisplayDateEnd). (Κληρονομείται από την UpDownBase).
ErrorTemplate	Καθορίζει ένα **ControlTemplate** που θα εφαρμόζεται **όταν το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
FirstDayOfWeek	Καθορίζει την **ημέρα που θα θεωρείται ως πρώτη της εβδομάδας**, τύπου System.DayOfWeek. Αυτή η ιδιότητα επηρεάζει την εμφάνιση του ενσωματωμένου calendar. Η default τιμή είναι System.DayOfWeek.Monday.

HasError	Αν είναι **True**, τότε **το control είναι σε κατάσταση λάθους**. (Κληρονομείται από την UpDownBase).
Increment	Καθορίζει το **βήμα αύξησης της τιμής** στο control, τύπου Integer. Η default τιμή είναι 1. Ο τύπος του βήματος μπορεί να είναι ημέρα/μήνας/έτος και εξαρτάται από την τιμή της ιδιότητας IncrementType.
IncrementType	Καθορίζει τον **τύπο του βήματος αύξησης της τιμής** του control, τύπου IncrementTypeEnum. Η default τιμή είναι IncrementTypeEnum.Days.
IsDropDownOpen	Καθορίζει **αν το ενσωματωμένο calendar θα είναι ανοικτό ή κλειστό**, τύπου Boolean. Η default τιμή είναι False.
IsMinimum	Αν είναι **True**, τότε η **τρέχουσα ημερομηνία έχει φτάσει στο ελάχιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα DisplayDateStart. (Κληρονομείται από την UpDownBase).
IsMaximum	Αν είναι **True**, τότε η **τρέχουσα ημερομηνία έχει φτάσει στο μέγιστο όριο,** όπως αυτό έχει προσδιοριστεί στην ιδιότητα DisplayDateEnd. (Κληρονομείται από την UpDownBase).
IsTodayHighlighted	Καθορίζει **αν η σημερινή ημερομηνία θα είναι φωτισμένη στο ενσωματωμένο calendar**, τύπου Boolean. Αυτή η ιδιότητα επηρεάζει την εμφάνιση του ενσωματωμένου calendar. Η default τιμή είναι True.
SelectedDate	Καθορίζει την **επιλεγμένη ημερομηνία,** τύπου Nullable(Of DateTime). Η default τιμή είναι Nothing.
SelectedDateFormat	Καθορίζει την **μορφή της επιλεγμένης ημερομηνίας** στο control, τύπου DatePickerFormat. Η default τιμή είναι DatePickerFormat.Short.
Text	**Επιστρέφει το κείμενο της επιλεγμένης ημερομηνίας**. Η default τιμή είναι ένα κενό string.

CalendarStyle

Καθορίζει το **Style** για το **ενσωματωμένο calendar**.

Σύνταξη:

VB:

```
Public Property CalendarStyle As Style
```

Τύπος: System.Windows.Style

Προσδιορίζουμε ένα αντικείμενο Style που θα χρησιμοποιηθεί για το rendering του ενσωματωμένου calendar.

Dependency Property Information:

Identifier field: `CalendarStyleProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε ένα resource Style με όνομα redCalendarStyle και το εφαρμόζουμε στην ιδιότητα CalendarStyle.

XAML

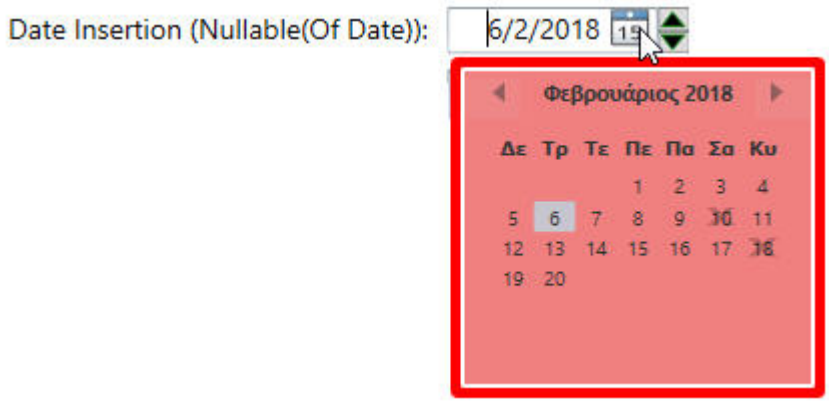
```
 <Window.Resources>

      ...

    <!-- The Calendar style for the DatePickerUpDown -->
    <Style x:Key="redCalendarStyle" TargetType="Calendar" >
          <Setter Property="Background" Value="LightCoral"/>
          <Setter Property="BorderBrush" Value="Red"/>
          <Setter Property="BorderThickness" Value="5"/>
    </Style>

</Window.Resources>

...

<zeus:DatePickerUpDown Name="dateInsertionUpDown"  Grid.Row="2" Grid.Column="1"
                        Margin="5,4,0,4"
                       Increment="1" IncrementType="Days"  Language="el"
                       ErrorMessageOnValueOutOfRange = "The date is out of range.
                                                        Please, retry!"
                       DisplayDateStart="2/1/2018" DisplayDateEnd="2/20/2018"
                       SelectedDate="{Binding DateInsertionNullable}"
                       SelectedDateFormat="Short"
                       Style="{StaticResource UpDownStyle }"
                        CalendarStyle="{StaticResource redCalendarStyle}"
```

```
                        DatePickerWatermark="Επιλέξτε Ημερ/νία"
                        HorizontalContentAlignment="Right"
                        SelectedDateChanged="dateInsertion_SelectedDateChanged"  />
```

VB:

```
Dim redCalendarStyle As New Style With {.TargetType = GetType(Calendar)}

With redCalendarStyle.Setters
    .Add(New Setter With {.Property = Control.BackgroundProperty, _
                          .Value = Brushes.LightCoral})
    .Add(New Setter With {.Property = Control.BorderBrushProperty, _
                          .Value = Brushes.Red})
    .Add(New Setter With {.Property = Control.BorderThicknessProperty, _
                          .Value = New Thickness(5)})
End With

dateInsertionUpDown.CalendarStyle = redCalendarStyle
```

Βλέπουμε το αποτέλεσμα:



DatePickerWatermark

Καθορίζει το **κείμενο** που θα εμφανίζεται **όταν δεν υπάρχει επιλεγμένη ημερομηνία** (SelectedDate=Nothing ή Text="").

Σύνταξη:

VB:

```
Public Property DatePickerWatermark As String
```

Τύπος: System.String

Dependency Property Information:

Identifier field: DatePickerWatermarkProperty

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το watermark "Επιλέξτε Ημερ/νία" όταν δεν υπάρχει επιλεγμένη ημερομηνία. Έχουμε ορίσει ένα style για το DatePickerUpDown που βασίζεται σε αυτό του UpDownStyle και καλύπτει την περίπτωση που η ιδιότητα Text είναι κενή. Τότε ενεργοποιείται το style trigger και δίνει ένα ελαφρύ γκρι χρώμα στο watermark και μικρότερο μέγεθος γραμματοσειράς.

```
<Window x:Class="MainWindow"
          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
          xmlns:p="clr-namespace:UpDownControlsTestProject"
          xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUp
                                                                    DownControls"
          mc:Ignorable="d"
          Title="UpDownControls Sample Project"
          Height="350" Width="525" Loaded="Window_Loaded" >

       <Window.Resources>

           <!-- The Validation Error ControlTemplate -->
           <ControlTemplate x:Key="validationErrorTemplate">

               <StackPanel Orientation="Horizontal">

                   <Border BorderBrush="#FFCB2E2E" BorderThickness="1"
                           Background="#11FF0000"
                           IsHitTestVisible="False" >
                       <AdornedElementPlaceholder x :Name="placeholder"  />
                   </Border>

                   <Grid  Margin="20,0,0,0"  >
```

```
                    <Ellipse  Width="20" Height="20"  Fill="Red"  />
                    <TextBlock Foreground="White" FontSize="14" Text="!"
                               FontWeight="ExtraBold"
                               VerticalAlignment="Center"
                               HorizontalAlignment="Center" />
                    <Grid.ToolTip>
                        <TextBlock Text="{Binding Path=/ErrorContent}"/>
                    </Grid.ToolTip>
                </Grid>

            </StackPanel>

        </ControlTemplate>

        <Style TargetType="{x:Type TextBlock}">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="HorizontalAlignment" Value="Right" />
            <Setter Property="Margin" Value="5"/>
        </Style>

        <!-- The style base for the UpDown controls. -->
        <Style  x:Key="UpDownStyle" TargetType="{x:Type zeus:UpDownBase }">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Background" Value="White"/>
            <Setter Property="HorizontalAlignment" Value="Left" />
            <Setter Property="Margin" Value="5,0,0,0"/>

            <Style.Triggers >
                <Trigger Property="IsMinimum" Value="True">
                    <Setter Property="Background" Value="Orange" />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="IsMaximum" Value="True">
                    <Setter Property="Background" Value="Green"  />
                    <Setter Property="Foreground"  Value="White" />
                </Trigger>

                <Trigger Property="HasError" Value="True">
                    <Setter Property="Foreground" Value="Red"/>
                    <Setter Property="BorderBrush" Value="Red"/>
                    <Setter Property="BorderThickness" Value="2"/>
                </Trigger>

            </Style.Triggers>

          </Style>

        <!-- A sample Customer resource object for the Bindings -->
        <p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
                                     Amount1="12.50"/>


    </Window.Resources>

    <Grid  Margin="10,40,10,10"
           DataContext="{Binding Source={StaticResource customer}}">

        ...

        <TextBlock Text="Date Insertion (Nullable(Of Date)):" />
        <zeus:DatePickerUpDown Name="dateInsertionUpDown" Width="180"
```

```
                                    Increment="1" IncrementType="Days" Language="el"
                                    HorizontalContentAlignment="Right"
                                    ErrorMessageOnValueOutOfRange = "The date is out of
                                                                    range. Please, retry!"
                                DisplayDateStart="2/1/2018" DisplayDateEnd="2/20/2018"
                                SelectedDate="{Binding DateInsertionNullable}"
                                SelectedDateFormat="Short"
                                DatePickerWatermark="Επιλέξτε Ημερ/νία"
                                Style="{StaticResource UpDownStyle}"
                              SelectedDateChanged="dateInsertion_SelectedDateChanged" />

            ...

        </Grid>

    </Window>
```

VB:

```
dateInsertionUpDown.DatePickerWatermark = "Επιλέξτε Ημερ/νία"
```

Βλέπουμε το αποτέλεσμα:

DisplayDate

Καθορίζει την **ημερομηνία που θα εμφανίζεται στο ενσωματωμένο calendar όταν δεν έχει επιλεγεί συγκεκριμένη ημερομηνία** (SelectedDate=Nothing ή Text="").

Σύνταξη:

VB:

```
Public Property DisplayDate As Nullable(Of DateTime)
```

Τύπος: System.Nullable(Of DateTime)

Dependency Property Information:

Identifier field: `DisplayDateProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε την ημερομηνία 13/4/2018 να εμφανίζεται στο calendar στην περίπτωση που δεν έχει επιλεγεί συγκεκριμένη ημερομηνία.

XAML

```
<zeus:DatePickerUpDown Name="dateInsertionUpDown" Width="18
                       Increment="1" IncrementType="Days" Language="el"
                       HorizontalContentAlignment="Right"
                      ErrorMessageOnValueOutOfRange = "The date is out of
                                                        range. Please, retry!"
                      DisplayDateStart="2/1/2018" DisplayDateEnd="6/30/2018"
                      DisplayDate="4/13/2018"
                      SelectedDate="{Binding DateInsertionNullable}"
                      SelectedDateFormat="Short"
                      DatePickerWatermark="Επιλέξτε Ημερ/νία"
                      Style="{StaticResource UpDownStyle}"
                      SelectedDateChanged="dateInsertion_SelectedDateChanged" />
```

VB:

```
dateInsertionUpDown.DisplayDate = #2/13/2018#
```

DisplayDateStart

Καθορίζει την **ημερομηνία αρχής που θα εμφανίζεται στο calendar**. Λειτουργεί ως το **ελάχιστο αποδεκτό όριο**.

Σύνταξη:

VB:

```
Public Property DisplayDateStart As Nullable(Of DateTime)
```

Τύπος: System.Nullable(Of DateTime)

Dependency Property Information:

Identifier field: `DisplayDateStartProperty`

Θέτοντας μία ημερομηνία, αυτή λειτουργεί ως το κάτω αποδεκτό όριο. Η default τιμή είναι Nothing (χωρίς ελάχιστο όριο).

Παρατηρήσεις:

Στο calendar οι ημερομηνίες ξεκινούν από αυτήν που θα θέσουμε εδώ. Παρόλα αυτά, ο χρήστης μπορεί να πληκτρολογήσει απ' ευθείας μία ημερομηνία που είναι κάτω από το αποδεκτό όριο. Τότε το control μπαίνει σε κατάσταση λάθους (HasError=True) και η ημερομηνία δεν γίνεται αποδεκτή.

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε την ημερομηνία αρχής την 1/2/2018 και ημερομηνία τέλους την 20/2/2018.

XAML

```
<zeus:DatePickerUpDown Name="dateInsertionUpDown" Width="18
                       Increment="1" IncrementType="Days" Language="el"
                       HorizontalContentAlignment="Right"
                      ErrorMessageOnValueOutOfRange = "The date is out of
                                                       range. Please, retry!"
                      DisplayDateStart="2/1/2018" DisplayDateEnd="2/20/2018"
                      SelectedDate="{Binding DateInsertionNullable}"
                      SelectedDateFormat="Short"
                      DatePickerWatermark="Επιλέξτε Ημερ/νία"
                      Style="{StaticResource UpDownStyle}"
                      SelectedDateChanged="dateInsertion_SelectedDateChanged" />
```

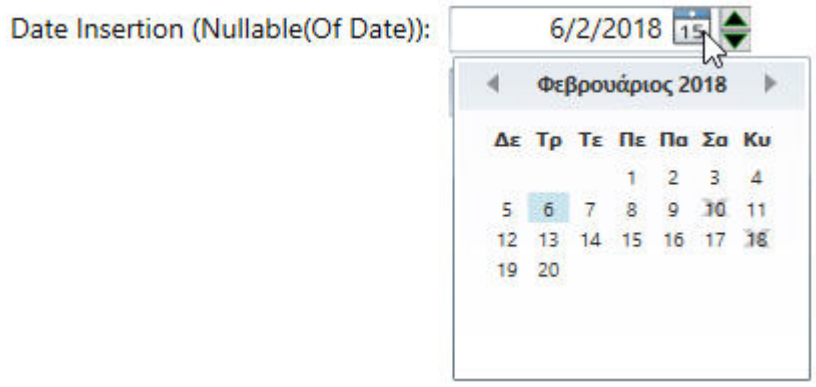
VB:

```
dateInsertionUpDown.DisplayDateStart = #2/1/2018#
dateInsertionUpDown.DisplayDateEnd = #2/20/2018#
```

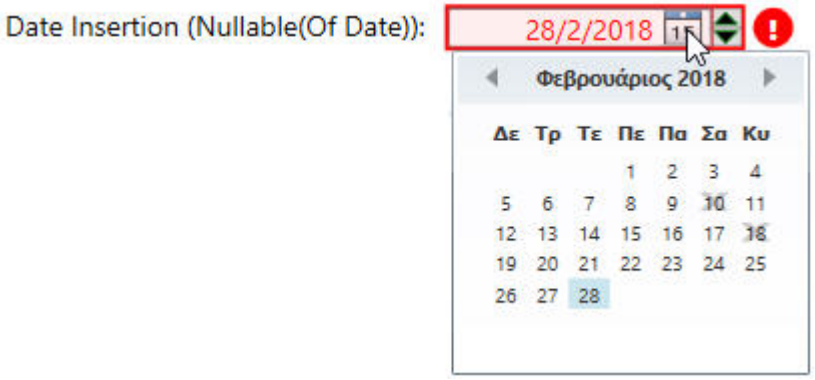
ή

```
dateInsertionUpDown.DisplayDateStart = New DateTime(2018, 2, 1)
dateInsertionUpDown.DisplayDateEnd = New DateTime(2018, 2, 20)
```

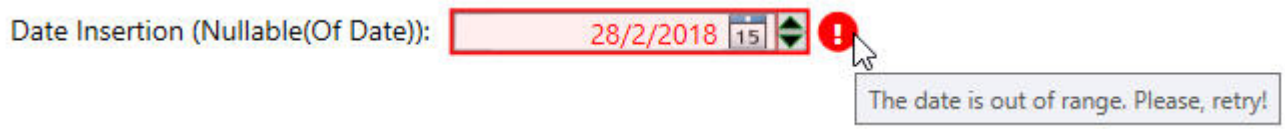
Βλέπουμε το αποτέλεσμα, όπου το calendar έχει περιοριστεί στο εύρος 1/2/2018 έως 20/2/2018:



Αν ο χρήστης πληκτρολογήσει μία **μη έγκυρη ημερομηνία** το calendar θα επεκταθεί αλλά δεν θα κάνει αποδεκτή καμία ημερομηνία που είναι εκτός ορίων:



Και το **μήνυμα λάθους**:



Όταν ο χρήστης επιλέξει ή πληκτρολογήσει μία έγκυρη ημερομηνία τότε το calendar επανέρχεται στο αποδεκτό εύρος.

DisplayDateEnd

Καθορίζει την **ημερομηνία τέλους που θα εμφανίζεται στο calendar**. Λειτουργεί ως το **μέγιστο αποδεκτό όριο**.

Σύνταξη:

VB:

```
Public Property DisplayDateEnd As Nullable(Of DateTime)
```

Τύπος: System.Nullable(Of DateTime)

Dependency Property Information:

Identifier field: `DisplayDateEndProperty`

Θέτοντας μία ημερομηνία, αυτή λειτουργεί ως το άνω αποδεκτό όριο. Η default τιμή είναι Nothing (χωρίς μέγιστο όριο).

Παρατηρήσεις:

Στο calendar οι ημερομηνίες τελειώνουν σε αυτήν που θα θέσουμε εδώ. Παρόλα αυτά, ο χρήστης μπορεί να πληκτρολογήσει απ' ευθείας μία ημερομηνία που είναι πάνω από το αποδεκτό όριο. Τότε το control μπαίνει σε κατάσταση λάθους (HasError=True) και η ημερομηνία δεν γίνεται αποδεκτή.

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε την ημερομηνία αρχής την 1/2/2018 και ημερομηνία τέλους την 20/2/2018.

XAML

```
<zeus:DatePickerUpDown Name="dateInsertionUpDown" Width="18
                       Increment="1" IncrementType="Days" Language="el"
                       HorizontalContentAlignment="Right"
                      ErrorMessageOnValueOutOfRange = "The date is out of
                                                       range. Please, retry!"
                      DisplayDateStart="2/1/2018" DisplayDateEnd="2/20/2018"
                      SelectedDate="{Binding DateInsertionNullable}"
                      SelectedDateFormat="Short"
                      DatePickerWatermark="Επιλέξτε Ημερ/νία"
                      Style="{StaticResource UpDownStyle}"
                      SelectedDateChanged="dateInsertion_SelectedDateChanged" />
```

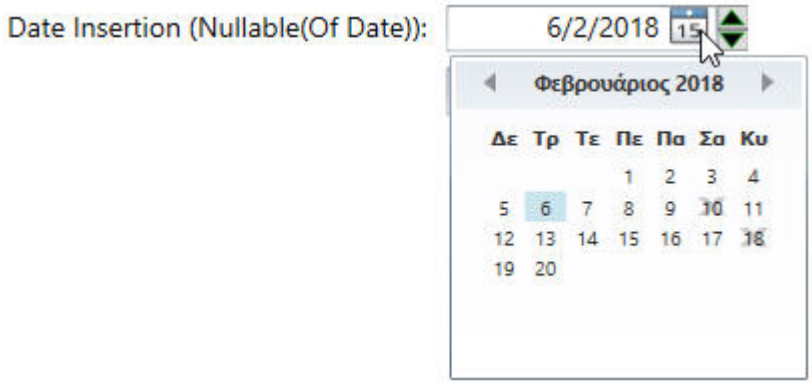
VB:

```
dateInsertionUpDown.DisplayDateStart = #2/1/2018#
dateInsertionUpDown.DisplayDateEnd = #2/20/2018#
```

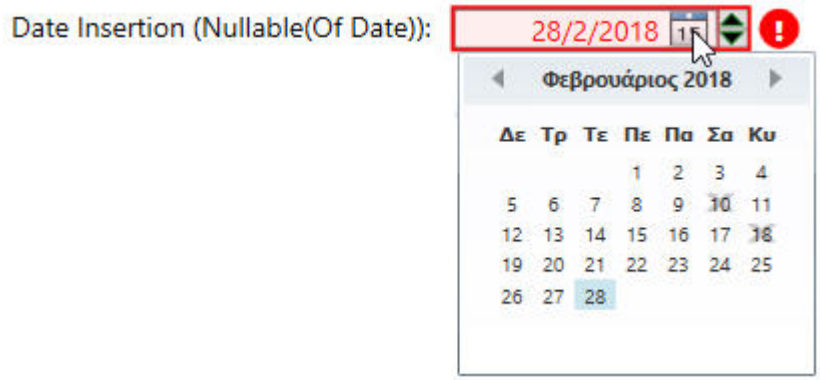
ή

```
dateInsertionUpDown.DisplayDateStart = New DateTime(2018, 2, 1)
dateInsertionUpDown.DisplayDateEnd = New DateTime(2018, 2, 20)
```

Βλέπουμε το αποτέλεσμα, όπου το calendar έχει περιοριστεί στο εύρος 1/2/2018 έως 20/2/2018:



Αν ο χρήστης πληκτρολογήσει μία **μη έγκυρη ημερομηνία** το calendar θα επεκταθεί αλλά δεν θα κάνει αποδεκτή καμία ημερομηνία που είναι εκτός ορίων:



Και το **μήνυμα λάθους**:

Όταν ο χρήστης επιλέξει ή πληκτρολογήσει μία έγκυρη ημερομηνία τότε το calendar επανέρχεται στο αποδεκτό εύρος.

FirstDayOfWeek

Καθορίζει την **ημέρα που θα θεωρείται ως πρώτη της εβδομάδας**, Αυτή η ιδιότητα επηρεάζει την εμφάνιση του ενσωματωμένου calendar.

Σύνταξη:

VB:

```
Public Property FirstDayOfWeek  As DayOfWeek
```

Τύπος: System.DayOfWeek

Dependency Property Information:

Identifier field: `FirstDayOfWeekProperty`

Η default τιμή είναι DayOfWeek.Monday.

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε την πρώτη ημέρα της εβδομάδας την Κυριακή.

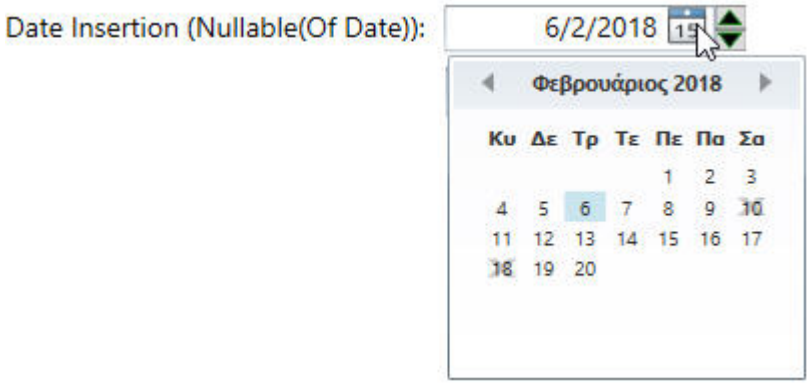
XAML

```
<zeus:DatePickerUpDown Name="dateInsertionUpDown" Width="18
                       Increment="1" IncrementType="Days" Language="el"
                       HorizontalContentAlignment="Right"
                       ErrorMessageOnValueOutOfRange = "The date is out of
                                                        range. Please, retry!"
                       DisplayDateStart="2/1/2018" DisplayDateEnd="2/20/2018"
                       SelectedDate="{Binding DateInsertionNullable}"
                       SelectedDateFormat="Short"
                       DatePickerWatermark="Επιλέξτε Ημερ/νία"
                       FirstDayOfWeek="Sunday"
                       Style="{StaticResource UpDownStyle}"
                       SelectedDateChanged="dateInsertion_SelectedDateChanged" />
```

VB:

```
dateInsertionUpDown.FirstDayOfWeek = DayOfWeek.Sunday
```

Βλέπουμε το αποτέλεσμα:


Date Insertion (Nullable(Of Date)):
6/2/2018
Φεβρουάριος 2018
Κυ Δε Τρ Τε Πε Πα Σα
1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20

Increment

Καθορίζει το **βήμα αύξησης της τιμής** στο control. Ο τύπος του βήματος μπορεί να είναι ημέρα/μήνας/έτος και εξαρτάται από την τιμή της ιδιότητας IncrementType.

Σύνταξη:

VB:

```
Public Property Increment As Integer
```

Τύπος: System.Integer

Προσδιορίζουμε μία θετική τιμή για το βήμα αύξησης της ημερομηνίας. Η default τιμή είναι 1.

Dependency Property Information:

Identifier field: `IncrementProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το βήμα αύξησης στο DatePickerUpDown σε 2. Επειδή η τιμή της ιδιότητας IncrementType είναι "Days", η αυξομείωση είναι ανά 2 ημέρες.

XAML:

```
<zeus:DatePickerUpDown Name="dateInsertionUpDown"
                       Increment="2" IncrementType="Days" ...  />
```

VB:

```
dateInsertionUpDown.Increment = 2
```

IncrementType

Καθορίζει τον **τύπο του βήματος αύξησης της τιμής** του control.

Σύνταξη:

VB:

```
Public Property IncrementType As IncrementTypeEnum
```

Τύπος: DatePickerUpDown.IncrementTypeEnum

Προσδιορίζουμε μία τιμή από την απαρίθμηση IncrementTypeEnum για τον τύπο αύξησης της ημερομηνίας (μέρα/μήνας/έτος) . Η default τιμή είναι IncrementTypeEnum.Days.

Dependency Property Information:

Identifier field: `IncrementTypeProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε τον τύπο βήματος αύξησης στο DatePickerUpDown σε "Months". Αυτό σημαίνει ότι όποτε ο χρήστης κάνει κλικ σε κάποιο arrow button η μεταβολή είναι σε μήνες. Εδώ, η Increment = 1, άρα η αυξομείωση είναι ανά ένα μήνα.

XAML:

```
<zeus:DatePickerUpDown Name="dateInsertionUpDown"
                       Increment="1" IncrementType="Months" ...  />
```

VB:

```
dateInsertionUpDown.IncrementType = DatePickerUpDown.IncrementTypeEnum.Months
```

IsDropDownOpen

Καθορίζει **αν το ενσωματωμένο calendar θα είναι ανοικτό ή κλειστό.**

Σύνταξη:

VB:

```
Public Property IsDropDownOpen As Boolean
```

Τύπος: System.Boolean

Η default τιμή είναι False.

Dependency Property Information:

Identifier field: `IsDropDownOpenProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε το ενσωματωμένο calendar να είναι ανοικτό.

XAML:

```
<zeus:DatePickerUpDown Name="dateInsertionUpDown"
                       Increment="1" IncrementType="Days"
                       IsDropDownOpen="True"  ...  />
```

VB:

```
dateInsertionUpDown.IsDropDownOpen = True
```

IsTodayHighlighted

Καθορίζει **αν η σημερινή ημερομηνία θα είναι φωτισμένη στο ενσωματωμένο calendar.**

Σύνταξη:

VB:

```
Public Property IsTodayHighlighted As Boolean
```

Τύπος: System.Boolean

Η default τιμή είναι True.

Dependency Property Information:

Identifier field: `IsTodayHighlightedProperty`

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε η σημερινή ημερομηνία να μην είναι φωτισμένη στο ενσωματωμένο calendar.

XAML:

```
<zeus:DatePickerUpDown Name="dateInsertionUpDown"
                       Increment="1" IncrementType="Days"
                       IsTodayHighlighted="False"  ...  />
```

VB:

```
dateInsertionUpDown.IsTodayHighlighted = False
```

SelectedDate

Καθορίζει την **επιλεγμένη ημερομηνία**.

Σύνταξη:

VB:

```
Public Property SelectedDate As Nullable(Of DateTime)
```

Τύπος: System.Nullable(Of DateTime)

Dependency Property Information:

Identifier field: `SelectedDateProperty`

Παρατηρήσεις:

Η ιδιότητα αυτή λαμβάνει από τον χρήστη την επιλεγείσα ημερομηνία. Γι' αυτό το λόγο, πρέπει να συνδέεται μέσω Binding με την ιδιότητα μίας πηγής δεδομένων (source object).

Παράδειγμα:

Στο παράδειγμα που ακολουθεί συνδέουμε την **ιδιότητα SelectedDate** με την **ιδιότητα DateInsertionNullable**, τύπου **Nullable(Of Date)**, ενός **Customer** αντικειμένου (source object). Η μορφή της ημερομηνίας καθορίζεται στην ιδιότητα SelectedDateFormat, που εδώ είναι Long.

XAML

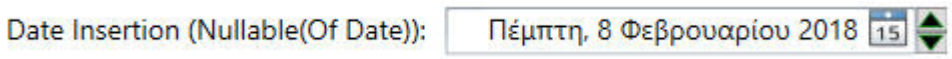
```
<Window.Resources>

    ...

    <!-- A sample Customer resource object for the Bindings -->
    <p:Customer x:Key="customer" LastName="Mouratidis" FirstName="Christos"
                                 Amount1="12.50"/>

</Window.Resources>

<zeus:DatePickerUpDown Name="dateInsertionUpDown"
                       Increment="1" IncrementType="Days"   Language="el"
                       ErrorMessageOnValueOutOfRange = "The date is out of range.
                                                        Please, retry!"
                       DisplayDateStart="2/1/2018" DisplayDateEnd="2/20/2018"
                       SelectedDate="{Binding DateInsertionNullable}"
                       SelectedDateFormat="Long"   ...  />
```

VB:

Η αλλαγή θα γίνει στο source object (με όνομα customer), οπότε μέσω του binding θα περάσει στην ιδιότητα SelectedDate :

```
CType(Me.FindResource("customer"), Customer).DateInsertionNullable = #2/8/2018#
```

Βλέπουμε το αποτέλεσμα όταν ο χρήστης επιλέγει την 8/2/2018:



SelectedDateFormat

Καθορίζει την **μορφή της επιλεγμένης ημερομηνίας** στο control.

Σύνταξη:

VB:

```
Public Property SelectedDateFormat As DatePickerFormat
```

Τύπος: System.Windows.Controls.DatePickerFormat

Dependency Property Information:

Identifier field: `SelectedDateFormatProperty`

Η default τιμή είναι DatePickerFormat.Short.

Παρατηρήσεις:

Οι δυνατές τιμές είναι DatePickerFormat.Short (default) και DatePickerFormat.Long.

Παράδειγμα:

Στο επόμενο παράδειγμα, καθορίζουμε την μορφή της επιλεγμένης ημερομηνίας σε DatePickerFormat.Long.

XAML

```
<zeus:DatePickerUpDown Name="dateInsertionUpDown"
                       Increment="1" IncrementType="Days"   Language="el"
                       ErrorMessageOnValueOutOfRange = "The date is out of range.
                                                        Please, retry!"
                       DisplayDateStart="2/1/2018" DisplayDateEnd="2/20/2018"
                       SelectedDate="{Binding DateInsertionNullable}"
                       SelectedDateFormat="Long"   ...  />
```

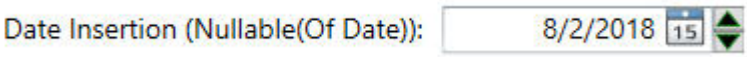
VB:

```
dateInsertionUpDown.SelectedDateFormat = DatePickerFormat.Long
```

Βλέπουμε το αποτέλεσμα:

VB:

```
dateInsertionUpDown.SelectedDateFormat = DatePickerFormat.Short
```



Text

Επιστρέφει το κείμενο της επιλεγμένης ημερομηνίας. Αν δεν υπάρχει επιλεγμένη ημερομηνία επιστρέφει ένα κενό string.

Σύνταξη:

VB:

```
Public Property Text As String
```

Τύπος: **System.String**

Dependency Property Information:

Identifier field: `TextProperty`

Παρατηρήσεις:

Αν δεν υπάρχει επιλεγμένη ημερομηνία τότε επιστρέφει ένα κενό string, κάτι που μπορούμε να χρησιμοποιήσουμε σε ένα style trigger για να μορφοποήσουμε το watermark (σχετικό παράδειγμα υπάρχει στην ιδιότητα DatePickerWatermark).

Συμβάντα

Όνομα	Περιγραφή
SelectedDateChanged	Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα SelectedDate**.

SelectedDateChanged

Ενεργοποιείται **όταν αλλάζει τιμή η ιδιότητα SelectedDate**.

Σύνταξη:

VB (ορισμός):

```
Public Custom Event SelectedDateChanged
                    As RoutedPropertyChangedEventHandler(Of Nullable(Of DateTime))
```

XAML attribute usage:

```
<zeus:DatePickerUpDown SelectedDateChanged="eventHanlder" ... />
```

Παράδειγμα:

Στο επόμενο παράδειγμα, έχουμε τοποθετήσει κάτω από το DatePickerUpDown ένα Slider. Το τελευταίο παίρνει τιμές από το 1 μέχρι το 20 (όσες είναι και οι αποδεκτές ημέρες στο DatePickerUpDown). Θέλουμε σε κάθε αλλαγή ημερομηνίας να μετατοπίζεται ο δείκτης του Slider στο κατάλληλο σημείο (Tick). Για παράδειγμα, αν ο χρήστης επιλέξει την 6/2/2018 τότε ο δείκτης του Slider πρέπει να πάει στην 6η θέση. Αν και αυτές τις συνδέσεις στο WPF συνηθίζουμε να τις πετυχαίνουμε με Bindings, εδώ θα καταδείξουμε τον κλασσικό τρόπο μέσω του eventHandler για το συμβάν SelectedDateChanged:

XAML

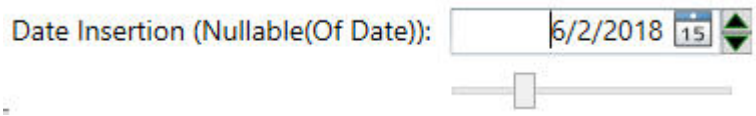
```
<zeus:DatePickerUpDown Name="dateInsertionUpDown"  Grid.Row="2" Grid.Column="1"
                       Margin="5,4,0,4" Width="150"
                       Increment="1" IncrementType="Days"   Language="el"
                       ErrorMessageOnValueOutOfRange = "The date is out of range.
                                                        Please, retry!"
                       DisplayDateStart="2/1/2018" DisplayDateEnd="2/20/2018"
                       SelectedDate="{Binding DateInsertionNullable}"
                       SelectedDateFormat="Long"
                       SelectedDateChanged="dateInsertion_SelectedDateChanged"
                       ...  />

<Slider Name="sldDateDistance" Grid.Row="3" Grid.Column="1"
        TickFrequency="1" Minimum="1" Maximum ="20"
        HorizontalAlignment="Left"  Width="150" Margin="0,5,0,0"/>
```

VB:

```
Private Sub dateInsertion_SelectedDateChanged(sender As Object, _
                                    e As RoutedPropertyChangedEventArgs(Of Date?))

    Dim dtUpDown As DatePickerUpDown = TryCast(sender, DatePickerUpDown)
```

```
        If dtUpDown IsNot Nothing Then

            If e.NewValue IsNot Nothing Then

                sldDateDistance.Value = _
                  (CDate(e.NewValue).Day - dtUpDown.DisplayDateStart.Value.Day) + 1

            End If

        End If

    End Sub
```

Βλέπουμε το αποτέλεσμα όταν ο χρήστης επιλέξει την 6/2/2018:



Μέθοδοι

Όνομα	Περιγραφή
AddBlackoutDates (List(Of CalendarDateRange), Optional Boolean)	**Προσθέτει μία λίστα ημερομηνιών ως μη επιλέξιμες στο ενσωματωμένο calendar του control.** Προαιρετικά, μπορούμε να καθορίσουμε να προστεθούν αυτόματα και όλες οι παρελθούσες ημερομηνίες. Ως τέτοιες θεωρούνται αυτές πρίν την σημερινή (today).
RemoveBlackoutDates (List(Of CalendarDateRange))	**Αφαιρεί μία λίστα ημερομηνιών από τις μη επιλέξιμες του ενσωματωμένου calendar του control.** Οι ημερομηνίες αυτές πλέον μπορούν να επιλεγούν ξανά από το calendar.
ClearBlackoutDays()	**Καθαρίζει τις μη επιλέξιμες ημερομηνίες του ενσωματωμένου calendar του control.** Πλέον, όλες οι αποδεκτές ημερομηνίες μπορούν να επιλεγούν ξανά από το calendar.

AddBlackoutDates

Προσθέτει μία λίστα ημερομηνιών ως μη επιλέξιμες στο ενσωματωμένο calendar του control. Προαιρετικά, μπορούμε να καθορίσουμε να προστεθούν αυτόματα και όλες οι παρελθούσες ημερομηνίες. Ως τέτοιες θεωρούνται αυτές πρίν την σημερινή (today).

Σύνταξη:

VB :

```
Public Sub AddBlackoutDates(
                listBlackoutDates As List(Of CalendarDateRange),
                Optional addDaysInPast As Boolean = False
                )
```

Παράμετροι:

listBlackoutDates
Τύπος: List(Of CalendarDateRange)
Μία λίστα των ημερομηνιών που θα είναι μη επιλέξιμες στο ενσωματωμένο calendar.

addDaysInPast (optional)
Τύπος: Boolean
Αν θέσουμε True τότε θα προστεθούν ως μη επιλέξιμες και όλες οι ημερομηνίες πριν τη σημερινή (today).

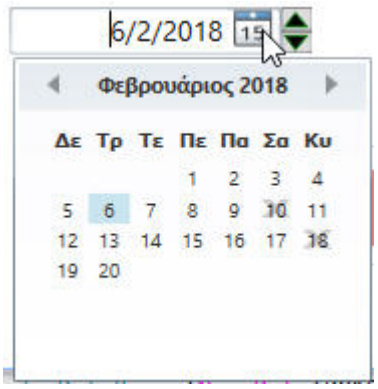
Παράδειγμα:

Στο επόμενο παράδειγμα, προσθέτουμε τις ημερομηνίες 10/2/2018 και 18/2/2018 στη λίστα των μη επιλέξιμων:

VB:

```
Dim lstBlackoutDays As New List(Of CalendarDateRange)

With lstBlackoutDays

       .AddRange({New CalendarDateRange(New Date(2018, 2, 10)),
                  New CalendarDateRange(New Date(2018, 2, 18))})

End With

dateInsertionUpDown.AddBlackoutDates(lstBlackoutDays)
```

Βλέπουμε το αποτέλεσμα:


6/2/2018
Φεβρουάριος 2018
Δε Τρ Τε Πε Πα Σα Κυ
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20

RemoveBlackoutDates

Αφαιρεί μία λίστα ημερομηνιών από τις μη επιλέξιμες του ενσωματωμένου calendar του control. Οι ημερομηνίες αυτές πλέον μπορούν να επιλεγούν ξανά από το calendar.

Σύνταξη:

VB :

```
Public Sub RemoveBlackoutDates(
                listBlackoutDates As List(Of CalendarDateRange)
                )
```

Παράμετροι:

listBlackoutDates
Τύπος: List(Of CalendarDateRange)
Μία λίστα των ημερομηνιών που θα αφαιρεθούν από τις μη επιλέξιμες στο ενσωματωμένο calendar.

Παράδειγμα:

Στο επόμενο παράδειγμα, αφαιρούμε την ημερομηνία 10/2/2018 από τις μη επιλέξιμες:

VB:

```
Dim lstBlackoutDays As New List(Of CalendarDateRange)

With lstBlackoutDays

        .AddRange({New CalendarDateRange(New Date(2018, 2, 10))})

End With

dateInsertionUpDown.RemoveBlackoutDates(lstBlackoutDays)
```

ClearBlackoutDays

Καθαρίζει τις μη επιλέξιμες ημερομηνίες του ενσωματωμένου calendar του control. Πλέον, όλες οι αποδεκτές ημερομηνίες μπορούν να επιλεγούν ξανά από το calendar.

Σύνταξη:

VB :

```
Public Sub ClearBlackoutDates()
```

Παράδειγμα:

Στο επόμενο παράδειγμα, καθαρίζουμε τη συλλογή των μη επιλέξιμων ημερομηνιών στο ενσωματωμένο calendar:

VB:

```
dateInsertionUpDown.ClearBlackoutDates()
```

Styles και Templates

Parts και States

Το default ControlTemplate περιλαμβάνει κάποια **part names** και **visual states**. Μπορείτε να τροποποιήσετε το default ControlTemplate ώστε να δώσετε στο control μία μοναδική εμφάνιση. Το default ControlTemplate μπορείτε να το δείτε εδώ ώστε να πάρετε μία ιδέα και με βάση αυτό να δημιουργήσετε ένα δικό σας. Ένα παράδειγμα custom ControlTemplate μπορείτε να δείτε εδώ.

Ο παρακάτω πίνακας εμφανίζει τα **part names** του **DatePickeUpDown** control:

Part	Τύπος	Περιγραφή
PART_ DatePicker	DatePicker	Το standard DatePicker control.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την αύξηση της τιμής.
PART_IncrementUp	RepeatButton	Το RepeatButton που επιτρέπει την μείωση της τιμής.

Ο παρακάτω πίνακας εμφανίζει τα **visual states** του **DatePickeUpDown** control:

VisualState	VisualStateGroup	Περιγραφή
Normal	CommonStates	Το default state.
MouseOver	CommonStates	Το state όταν ο δείκτης του ποντικιού είναι πάνω από το control.
Disabled	CommonStates	Το state όταν το control είναι disabled.
Focused	FocusStates	Το state όταν το control έχει το focus.
Unfocused	FocusStates	Το state όταν το control δεν έχει το focus.

Το default ControlTemplate έχει καθορισμένο μόνο το Disabled state. Μπορείτε να δημιουργήσετε ένα custom ControlTemplate για να το αλλάξετε ή/και για να καθορίσετε τα υπόλοιπα.

Το default Style και ControlTemplate

Ο XAML κώδικας για το **default Style** και **ControlTemplate** φαίνεται παρακάτω. Μπορείτε να βασιστείτε σε αυτόν για να δημιουργήσετε μία μικρή ή μεγάλη παραλλαγή του δικού σας custom Style και ControlTemplate:

```
    xmlns:z="clr-namespace:Zeus.WPF.Controls.UpDownControls">

    <Style TargetType="{x:Type z:DatePickerUpDown}" >

        <Setter Property="Focusable" Value="True"/>
        <Setter Property="BorderBrush" Value="Black" />
        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="Background"  Value="Transparent" />
        <Setter Property="Width" Value ="120"/>
        <Setter Property="Height" Value="23"/>

        <Setter Property="Template">

            <Setter.Value>

                <ControlTemplate TargetType="{x:Type z:DatePickerUpDown}">

                    <ControlTemplate.Resources >
                        <z:DoubleToGridLengthConverter
                                        x:Key="doubleToGridLengthConverter"/>
                    </ControlTemplate.Resources>

                    <!-- Root element -->
                    <Border Name="Border"
                            BorderBrush="{TemplateBinding BorderBrush}"
                            BorderThickness="{TemplateBinding BorderThickness}">

                        <VisualStateManager.VisualStateGroups >

                            <VisualStateGroup Name="CommonStates">

                                <VisualState Name="Normal"/>

                                <VisualState Name="Disabled">

                                    <Storyboard >

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_DatePicker"
                                            Storyboard.TargetProperty ="Background.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                     Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ColorAnimationUsingKeyFrames
                                                Storyboard.TargetName="PART_DatePicker"
                                            Storyboard.TargetProperty ="Foreground.Color" >
                                            <DiscreteColorKeyFrame KeyTime="0"
                    Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                                        </ColorAnimationUsingKeyFrames>

                                        <ObjectAnimationUsingKeyFrames
```

```
                                        Storyboard.TargetName="PART_DatePicker"
                                    Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                        <DiscreteObjectKeyFrame KeyTime="0"
                                 Value="{Binding Source={x:Static FontWeights.Normal}}" />
                                        </ObjectAnimationUsingKeyFrames>

                                    </Storyboard>

                                </VisualState>


                            </VisualStateGroup>

                            <VisualStateGroup Name="FocusStates">

                                <VisualState Name="Focused" />
                                <VisualState Name="Unfocused"/>

                            </VisualStateGroup>

                        </VisualStateManager.VisualStateGroups>

                        <!-- Content -->
                        <Grid Name="mainGrid" Background="Transparent" >

                            <Grid.RowDefinitions >
                                <RowDefinition Height="*" />
                            </Grid.RowDefinitions>

                            <!-- The width of the second Grid column is about for the
                                 RepeatButtons and depends on ButtonsWidth property -->
                            <Grid.ColumnDefinitions >
                                <ColumnDefinition Width="*" />
                                <ColumnDefinition
                                      Width="{Binding RelativeSource={Relative Source
                                 Mode=FindAncestor, AncestorType={x:Type z:DatePickerUpDown}},
        Path=ButtonsWidth, Converter={StaticResource doubleToGridLengthConverter}}" />
                            </Grid.ColumnDefinitions>

                            <DatePicker Name="PART_DatePicker"
                                        Padding="{TemplateBinding Padding}"
                    HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment}"
                        VerticalContentAlignment="{TemplateBinding VerticalContentAlignment}"
                                        Foreground="{TemplateBinding Foreground }"
                                        Background="{TemplateBinding Background }"
                                        FontSize="{TemplateBinding FontSize }"
                                        FontFamily="{TemplateBinding FontFamily }"
                                        FontStretch="{TemplateBinding FontStretch }"
                                        FontWeight="{TemplateBinding FontWeight }"
                                        Focusable="True" />

                            <Viewbox Stretch="Fill"   Grid.Column="1">

                                <StackPanel Name="stkRepeatButtons"
                                            VerticalAlignment="Center"
                                            HorizontalAlignment="Left"  Focusable="False">

                                    <RepeatButton Name="PART_IncrementUp"
                                                  Focusable="False"  >
                                        <RepeatButton.Content>
                                            <Path Fill="Black" Stroke="Green"
                                                  Data="M 1,10 10,1 20,10 Z" />
```

```
                                    </RepeatButton.Content>
                                </RepeatButton>

                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                    <RepeatButton.Content>
                                        <Path Fill="Black" Stroke="Green"
                                              Data="M 10,10 1,1 20,1 Z" />
                                    </RepeatButton.Content>
                                </RepeatButton>

                            </StackPanel>

                        </Viewbox>

                    </Grid>

                </Border>

            </ControlTemplate>

        </Setter.Value>

    </Setter>

</Style>
```

Η δεύτερη στήλη του Grid περιέχει τα repeat buttons μέσα σε ένα StackPanel. Το πλάτος της δεύτερης στήλης του Grid προσδιορίζεται με βάση την τιμή της ιδιότητας ButtonsWidth. Για να μετατραπεί η τιμή Double της ιδιότητας ButtonsWidth σε τιμή τύπου GridLength χρησιμοποιούμε μία κλάση μετατροπής τιμής (ValueConverter). Θα τη βρείτε στο Παράρτημα 2.

Παράδειγμα custom Style και ControlTemplate

Στο επόμενο παράδειγμα, δημιουργούμε ένα custom Style και ControlTemplate. Συγκεκριμένα:

- Ορίζουμε ότι **όταν λάβει την εστίαση, το background να γίνει κίτρινο**. Αυτό γίνεται, θέτοντας στο **Focused state** το σχετικό Storyboard που περιέχει ένα ColorAnimation που κάνει animate το background color σε Yellow σε χρόνο 0.2 sec.

- Στη δομή του ControlTemplate, **αλλάζουμε θέση στα up-down buttons**. Το up-button βρίσκεται πάνω από το DatePicker ενώ το down-button βρίσκεται από κάτω (3 γραμμές). Για να επιτευχθεί αυτό αλλάζει λίγο η διάταξη του Grid container, που τώρα περιέχει 3 RowDefinintions. Η λειτουργικότητα του control παραμένει η ίδια.

Η εικόνα του custom DatePickerUpDown σε κανονική κατάσταση - **Normal state**:

Η εικόνα του custom DatePickerUpDown όταν έχει την εστίαση - **Focused state**:

Ο XAML κώδικας για το custom Style και ControlTemplate του παραπάνω DatePickerUpDown είναι ο εξής:

```
    xmlns:zeus="clr-namespace:Zeus.WPF.Controls.UpDownControls;assembly=ZeusUpDownCont
                                                                                rols"

  <Style TargetType="{x:Type zeus:DatePickerUpDown}" >

       <Setter Property="Focusable" Value="True"/>
       <Setter Property="BorderBrush" Value="Black" />
       <Setter Property="BorderThickness" Value="0"/>
       <Setter Property="Background"  Value="Transparent" />
       <Setter Property="Width" Value ="120"/>
       <Setter Property="Height" Value="23"/>

       <Setter Property="Template">

           <Setter.Value>

               <ControlTemplate TargetType="{x:Type zeus:DatePickerUpDown}">

                   <ControlTemplate.Resources >
                       <zeus:DoubleToGridLengthConverter
                                           x:Key="doubleToGridLengthConverter"/>
```

```
        </ControlTemplate.Resources>

        <!-- Root element -->
        <Border Name="Border"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="{TemplateBinding BorderThickness}">

            <VisualStateManager.VisualStateGroups >

                <VisualStateGroup Name="CommonStates">

                    <VisualState Name="Normal"/>

                    <VisualState Name="Disabled">

                        <Storyboard>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_DatePicker"
                                Storyboard.TargetProperty ="Background.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
 Value="{StaticResource {x:Static SystemColors.InactiveBorderColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ColorAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_DatePicker"
                                Storyboard.TargetProperty ="Foreground.Color" >
                                <DiscreteColorKeyFrame KeyTime="0"
Value="{StaticResource {x:Static SystemColors.InactiveCaptionColorKey }}" />
                            </ColorAnimationUsingKeyFrames>

                            <ObjectAnimationUsingKeyFrames
                                    Storyboard.TargetName="PART_DatePicker"
                                Storyboard.TargetProperty ="(TextBox.FontWeight)" >
                                <DiscreteObjectKeyFrame KeyTime="0"
                             Value="{Binding Source={x:Static FontWeights.Normal}}" />
                            </ObjectAnimationUsingKeyFrames>

                        </Storyboard>

                    </VisualState>


                </VisualStateGroup>

                <VisualStateGroup Name="FocusStates">

                   <VisualState Name="Focused" >

                        <Storyboard >

                            <ColorAnimation
                                   Storyboard.TargetName="PART_DatePicker"
                                 Storyboard.TargetProperty="Background.Color"
                                       To="Yellow" Duration="0:0:0.2"/>

                            </Storyboard>

                        </VisualState>

                    <VisualState Name="Unfocused"/>
```

```
                        </VisualStateGroup>

                    </VisualStateManager.VisualStateGroups>

                    <!-- Content -->
                   <Grid Name="mainGrid" Background="Transparent"
                                         Margin="{TemplateBinding Padding}">

                            <Grid.RowDefinitions >
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                            </Grid.RowDefinitions>

                           <DatePicker Name="PART_DatePicker"
                                    Padding="{TemplateBinding Padding}"
                HorizontalContentAlignment="{TemplateBinding HorizontalContentAlignment}"
                    VerticalContentAlignment="{TemplateBinding VerticalContentAlignment}"
                                    Foreground="{TemplateBinding Foreground }"
                                    Background="{TemplateBinding Background }"
                                    FontSize="{TemplateBinding FontSize }"
                                    FontFamily="{TemplateBinding FontFamily }"
                                    FontStretch="{TemplateBinding FontStretch }"
                                    FontWeight="{TemplateBinding FontWeight }"
                                    Focusable="True" />

                            <Viewbox Stretch="Uniform" Grid.Row="0" Height="12">

                                <RepeatButton Name="PART_IncrementUp"
                                              Focusable="False"  >
                                     <RepeatButton.Content>
                                           <Path Fill="Black" Stroke="Green"
                                                 Data="M 1,10 10,1 20,10 Z" />
                                      </RepeatButton.Content>
                                </RepeatButton>

                            </Viewbox>

                            <Viewbox  Stretch="Uniform" Grid.Row="2" Height="12">

                                <RepeatButton Name="PART_IncrementDown"
                                              Focusable="False" >
                                     <RepeatButton.Content>
                                          <Path Fill="Black" Stroke="Green"
                                                Data="M 10,10 1,1 20,1 Z" />
                                     </RepeatButton.Content>
                                </RepeatButton>

                            </Viewbox>

                        </Grid>

                    </Border>

                </ControlTemplate>

            </Setter.Value>

        </Setter>

    </Style>
```

Παράρτημα 1: Η κλάση Customer

Στα παραδείγματα της Βοήθειας κάνουμε εκτενή χρήση της **κλάσης Customer.** Κατά περίπτωση, **ιδιότητες** αυτής της κλάσης (πεδία) γίνονται binding με αντίστοιχες ιδιότητες των UpDownControls. Για παράδειγμα, η ιδιότητα Value ενός DecimalUpDown γίνεται binding με την ιδιότητα Amount του αντικειμένου Customer ενώ η ιδιότητα SelectedDate ενός DatePickerUpDown γίνεται binding με την ιδιότητα DateInsertionNullable του αντικειμένου Customer.

Σημείωση: Εδώ, η ιδιότητα Amount είναι τύπου Decimal. Σε παραδείγματα του Help όπου απαιτείται να είναι Byte ή Integer κλπ. μπορείτε να θεωρήσετε ότι είναι του αυτού τύπου. Δεν αλλάζει κάτι άλλο.

Η κλάση αυτή έχει οριστεί ως εξής:

```
Imports System.ComponentModel

'*************************************************
'***********  The class for the Data **************
'*************************************************

Public Class Customer
    Implements INotifyPropertyChanged

    Public Event PropertyChanged As PropertyChangedEventHandler
                          Implements INotifyPropertyChanged.PropertyChanged

    Public Sub OnPropertyChanged(ByVal sender As Object, ByVal e As PropertyChangedEventArgs)

        If e IsNot Nothing Then RaiseEvent PropertyChanged(Me, e)

    End Sub

    'Private backing fields.
    Private _lastName As String
    Private _firstName As String
    Private _amount As Decimal
    Private _dateInsertionNullable As Nullable( Of Date)

    'CLR properties - data fields.
    Public Property LastName As String
        Get
            Return _lastName
        End Get
        Set(ByVal value As String)
            _lastName = value
            OnPropertyChanged(Me, New PropertyChangedEventArgs("LastName"))
        End Set
    End Property

    Public Property FirstName As String
        Get
            Return _firstName
        End Get
        Set(ByVal value As String)
            _firstName = value
            OnPropertyChanged(Me, New PropertyChangedEventArgs("FirstName"))
```

```
        End Set
    End Property

    Public Property Amount As Decimal
        Get
            Return _amount
        End Get
        Set(ByVal value As Decimal)
            _amount = value
            OnPropertyChanged(Me, New PropertyChangedEventArgs("Amount"))
        End Set
    End Property

    Public Property DateInsertionNullable As Nullable(Of Date)
        Get
            Return _dateInsertionNullable
        End Get
        Set(ByVal value As Nullable(Of Date))
            _dateInsertionNullable = value
            OnPropertyChanged(Me, New PropertyChangedEventArgs("DateInsertionNullable"))
        End Set
    End Property


    'Constructor.
    Public Sub New()

        _lastName = String.Empty
        _firstName = String.Empty
        _amount = 0
        _dateInsertionNullable = Nothing

    End Sub

End Class
```

Παράρτημα 2: Η κλάση μετατροπής τιμής DoubleToGridLengthConverter

Στα παραδείγματα της Βοήθειας κάνουμε εκτενή χρήση της **κλάσης DoubleToGridLengthConverter,** η οποία μετατρέπει μία τιμή Double σε μία τιμή GridLength. Τη χρησιμοποιούμε στο default ControlTemplate των UpDownControls για να συνδέσουμε (binding) το πλάτος μίας στήλης του Grid (που μετριέται σε GridLength) με την ButtonsWidth (που μετριέται σε Double).

```
<ValueConversion(GetType(Double), GetType(GridLength))>
Public Class DoubleToGridLengthConverter
                        Implements IValueConverter

    Public Function Convert(value As Object, targetType As Type, _
                            parameter As Object,  culture As Globalization.CultureInfo)
                    As Object Implements IValueConverter.Convert

        Const CONST_MinButtonsWidth As Double = 10

        Try

            Dim val As Double = CDbl(value)
            Dim grdLength As New  GridLength(val, GridUnitType.Pixel)

            Return grdLength

        Catch ex As Exception

            Return New GridLength(CONST_MinButtonsWidth, GridUnitType.Pixel)

        End Try


    End Function


    Public Function ConvertBack(value As Object, targetType As Type, _
                                parameter As Object, _
                                culture As Globalization.CultureInfo)
                    As Object Implements IValueConverter.ConvertBack

        Try
            Dim val As GridLength = value
            targetType = GetType(Double)
            Return val.Value

        Catch
            Throw New Exception("Cannot Convert Back to Double")
        End Try

   End Function

End Class
```