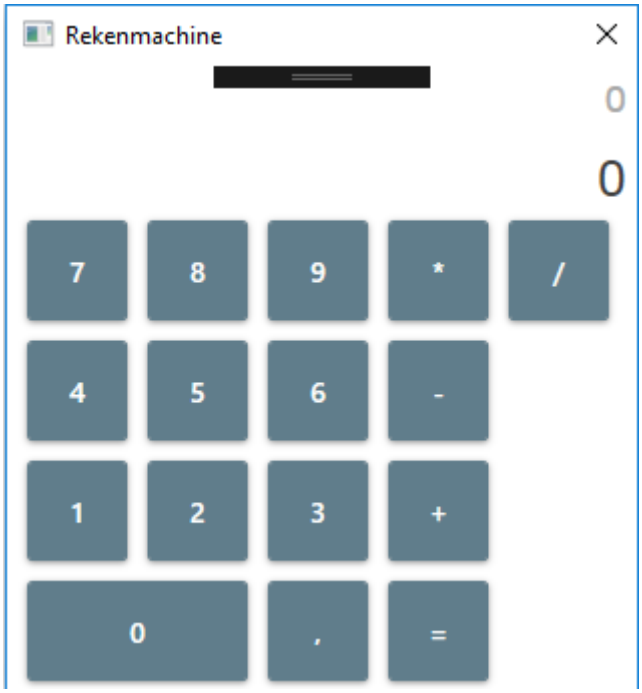
Handleiding rekenmachine in WPF

Inhoud

Doel

In deze handleiding wordt er stap voor stap uitgelegd hoe we dit resultaat kunnen bekomen :



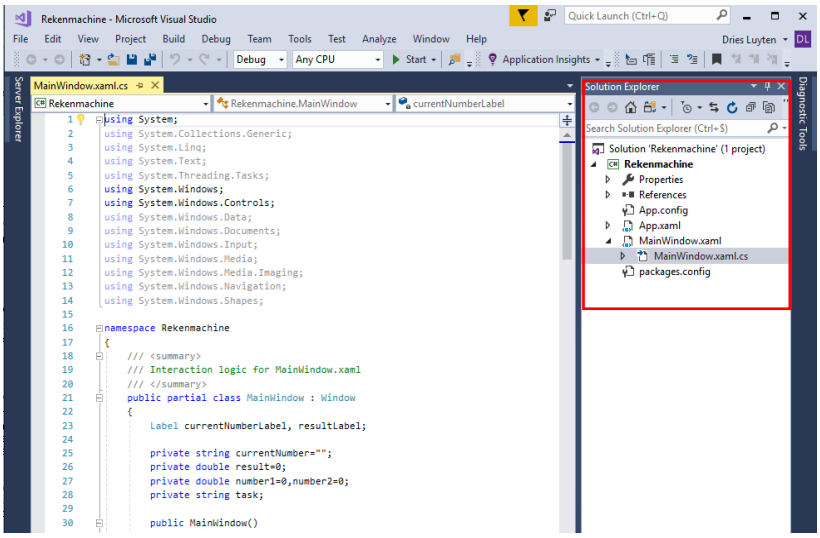
Verloop

Wij hebben een startbestand voor jullie klaargemaakt waar al enkele code in staat, die gaan we eerst en vooral overlopen dat alles duidelijk is tegen we de applicatie gaan vereenvoudigen en uitbreiden. Als dit gebeurd is gaan we beginnen met de bestaande code efficiënter te maken, daarna is er nog ruimte om enkele uitbreidingen toe te passen.

Het startbestand: uitleg

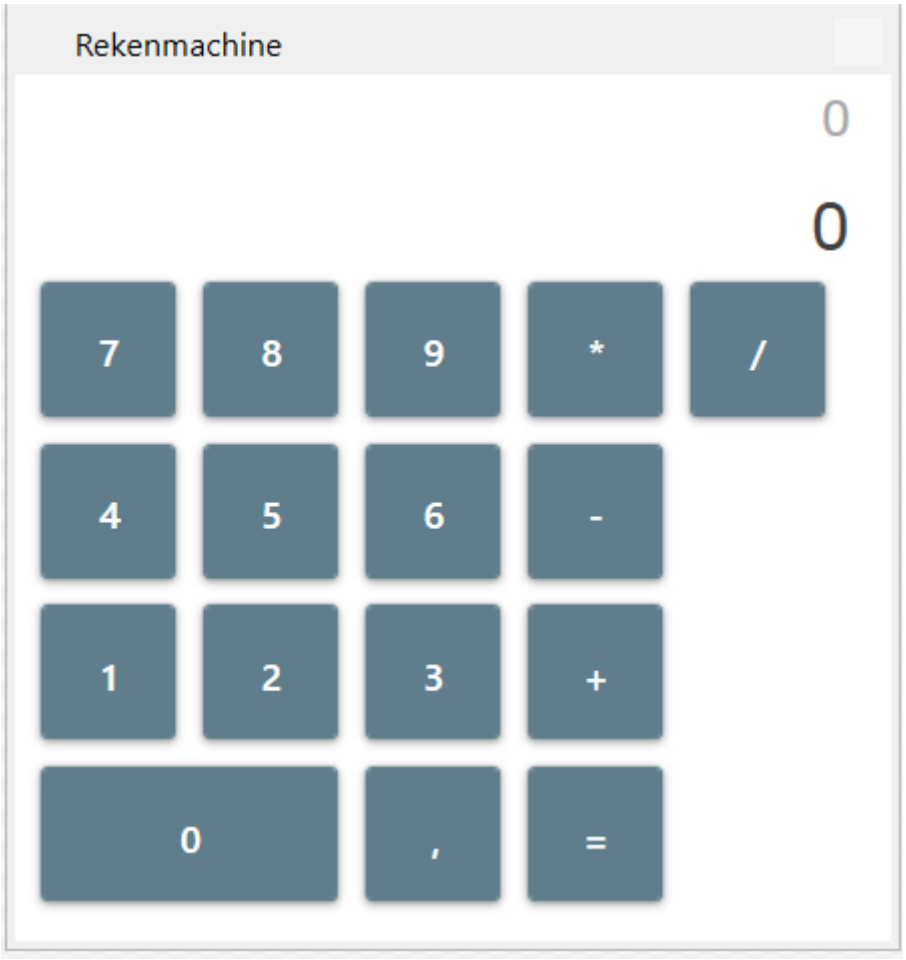
Solution Explorer

We gaan vandaag enkel werken in de MainWindow.xaml en de bijhorende codebehind(MainWindow.xaml.cs) file om het zo simpel mogelijk te houden



MainWindow.xaml

Design:



XAML code van het design:

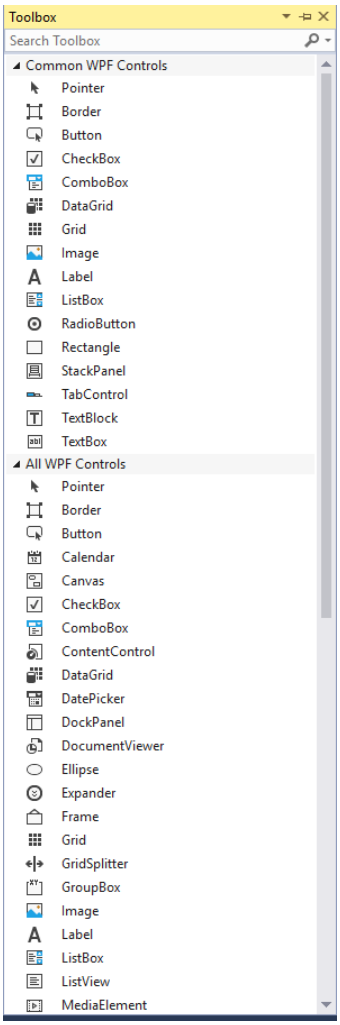
bel VerticalAlignment

```
<Window x:Class="Rekenmachine.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:local="clr-namespace:Rekenmachine"
        mc:Ignorable="d"
        ResizeMode="NoResize"
        Title="Rekenmachine" Height="350" Width="330">
    <Grid>
        <Label HorizontalAlignment="Left" Height="36" Margin="10,0,0,0" Content="0"
               VerticalAlignment="Top" Width="304" FontFamily="Verdana" FontSize="18" x:Name="labelResult" HorizontalContentAlignment="Right" Foreground="DarkGray"/>
        <Label HorizontalAlignment="Left" Height="36" Margin="10,36,0,0" Content="0"
               VerticalAlignment="Top" Width="304" FontFamily="Verdana" FontSize="24" Name="labelCurrentNumber" HorizontalContentAlignment="Right"/>
        <Button Content="0" HorizontalAlignment="Left" Margin="10,257,0,0" VerticalAlignment="Top" Width="110" Height="50" Click="Button_0"/>
        <Button Content="1" HorizontalAlignment="Left" Margin="10,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_1"/>
        <Button Content="2" HorizontalAlignment="Left" Margin="70,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_2"/>
        <Button Content="3" HorizontalAlignment="Left" Margin="130,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_3"/>
        <Button Content="4" HorizontalAlignment="Left" Margin="10,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_4"/>
        <Button Content="7" HorizontalAlignment="Left" Margin="10,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_7"/>
        <Button Content="5" HorizontalAlignment="Left" Margin="70,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_5"/>
        <Button Content="8" HorizontalAlignment="Left" Margin="70,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_8"/>
        <Button Content="9" HorizontalAlignment="Left" Margin="130,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_9"/>
        <Button Content="6" HorizontalAlignment="Left" Margin="130,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_6"/>
        <Button Content="," HorizontalAlignment="Left" Margin="130,257,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_dot"/>
        <Button Content="=" HorizontalAlignment="Left" Margin="190,257,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_result"/>
        <Button Content="+" HorizontalAlignment="Left" Margin="190,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_plus"/>
        <Button Content="-" HorizontalAlignment="Left" Margin="190,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_minus"/>
        <Button Content="*" HorizontalAlignment="Left" Margin="190,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_multiply"/>
        <Button Content="/" HorizontalAlignment="Left" Margin="250,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_divide"/>
    </Grid>
</Window>
```

XAML verduidelijking

Controls plaatsen

De code die hierboven getoond werd kan zelf geschreven worden of men kan deze genereren door controls vanuit de toolbox in het design venster te slepen.



Controls properties

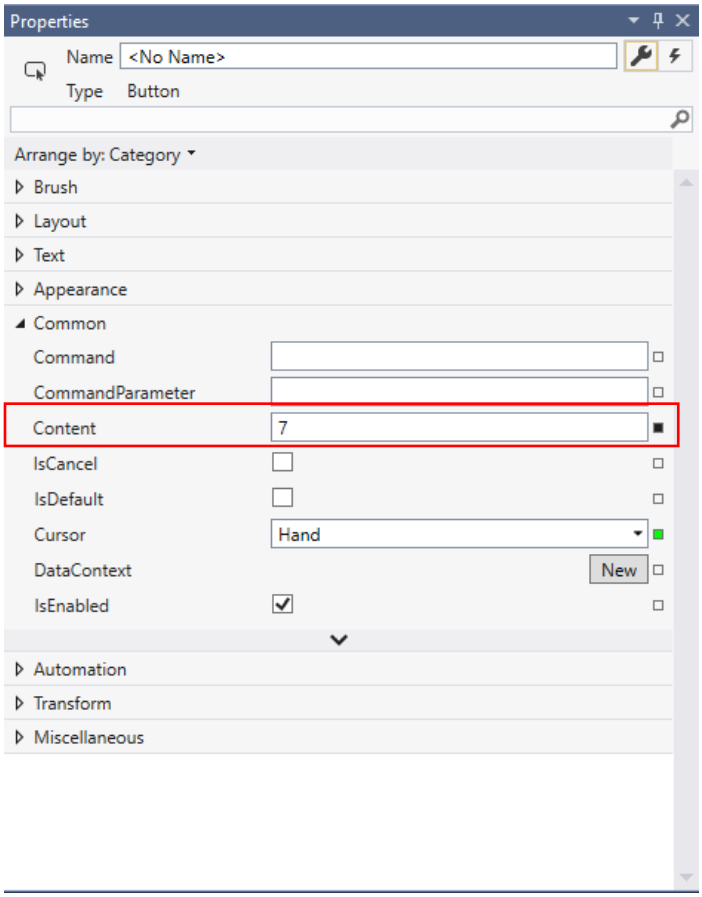
De properties van elke control kunnen ook zelf geschreven worden ofwel gegenereerd worden door het property venster.

(geen properties venster? View -> Properties Window (F4))

Bijvoorbeeld:

```
<Button Content="7"
```

Komt overeen met:



MainWindow.xaml: verduidelijking

Labels: properties

De belangrijkste properties bij deze labels zijn de ‘Name’ & ‘Content’ properties, de rest zijn gewoon opmaak eigenschappen.

```
<Label HorizontalAlignment="Left" Height="36" Margin="10,0,0,0" Content="0"
      VerticalAlignment="Top" Width="304" FontFamily="Verdana" FontSize="18" x:Name="labelResult" HorizontalContentAlignment="Right" Foreground="DarkGray"/>
<Label HorizontalAlignment="Left" Height="36" Margin="10,36,0,0" Content="0"
      VerticalAlignment="Top" Width="304" FontFamily="Verdana" FontSize="24" Name="labelCurrentNumber" HorizontalContentAlignment="Right"/>
```

- De ‘Name’ property wordt in de codebehind gebruikt als men deze control wil oproepen.

```
x:Name="labelResult"

Name="labelCurrentNumber"
```

```
public MainWindow()
{
    InitializeComponent();
    currentNumberLabel = (Label)this.FindName("labelCurrentNumber");
    resultLabel = (Label)this.FindName("labelResult");
}
```

- De ‘Content’ property wordt in de codebehind aangepast met het resultaat, bij initialisatie is deze standaard “0” omdat we hem dat toewijzen, anders is het een lege label.

```
<Label HorizontalAlignment="Left" Height="36" Margin="10,0,0,0" Content="0"
```

Buttons: properties

Hier zijn de belangrijkste properties ‘Click’ & ook ‘Content’, ook hier is de rest gewoon opmaak.

```
<Button Content="0" HorizontalAlignment="Left" Margin="10,257,0,0" VerticalAlignment="Top" Width="110" Height="50" Click="Button_0"/>
<Button Content="1" HorizontalAlignment="Left" Margin="10,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_1"/>
<Button Content="2" HorizontalAlignment="Left" Margin="70,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_2"/>
<Button Content="3" HorizontalAlignment="Left" Margin="130,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_3"/>
<Button Content="4" HorizontalAlignment="Left" Margin="10,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_4"/>
<Button Content="7" HorizontalAlignment="Left" Margin="10,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_7"/>
<Button Content="5" HorizontalAlignment="Left" Margin="70,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_5"/>
<Button Content="8" HorizontalAlignment="Left" Margin="70,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_8"/>
<Button Content="9" HorizontalAlignment="Left" Margin="130,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_9"/>
<Button Content="6" HorizontalAlignment="Left" Margin="130,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_6"/>
<Button Content="," HorizontalAlignment="Left" Margin="130,257,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_dot"/>
<Button Content="=" HorizontalAlignment="Left" Margin="190,257,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_result"/>
<Button Content="+" HorizontalAlignment="Left" Margin="190,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_plus"/>
<Button Content="-" HorizontalAlignment="Left" Margin="190,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_minus"/>
<Button Content="*" HorizontalAlignment="Left" Margin="190,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_multiply"/>
<Button Content="/" HorizontalAlignment="Left" Margin="250,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="Button_divide"/>
```

- De ‘Click’ property is de onclick event die opgeroepen wordt in de codebehind, in andere woorden: men klikt op de button en die doet iets, wat die moet doen wordt gespecifieerd onder een void functie met als naam de waarde in ‘Click’

```
<Button Content="0" HorizontalAlignment="Left" Margin="10,257,0,0" VerticalAlignment="Top" Width="110" Height="50" Click="Button_0"/>
```

```
private void Button_0(object sender, RoutedEventArgs e)
{
    currentNumber += "0";
    UpdateLabel(currentNumber);
}
```

- De ‘Content’ property werkt hetzelfde als bij een label

MainWindow.xaml.cs (codebehind)

Variables

```
Label currentNumberLabel, resultLabel;

private string currentNumber="";
private double result=0;
private double number1=0,number2=0;
private string task;

public MainWindow()
{
    InitializeComponent();
    currentNumberLabel = (Label)this.FindName("labelCurrentNumber");
    resultLabel = (Label)this.FindName("labelResult");
}
```

De labels die we gemaakt hebben gaan we nu in een variabelen stoppen waar we mee kunnen werken:

```
Label currentNumberLabel, resultLabel;

private string currentNumber="";
private double result=0;
private double number1=0,number2=0;
private string task;

public MainWindow()
{
    InitializeComponent();
    currentNumberLabel = (Label)this.FindName("labelCurrentNumber");
    resultLabel = (Label)this.FindName("labelResult");
}
```

Functies: getallen ingeven

Voor elke button met een getal is er tijdelijk een functie voorzien, dit gaan we later nog efficiënter oplossen. Denk al maar eens na over een betere oplossing. Nu ziet de code er als volgt uit:

```
private void Button_0(object sender, RoutedEventArgs e)
{
    currentNumber += "0";
    UpdateLabel(currentNumber);
}

private void Button_1(object sender, RoutedEventArgs e)
{
    currentNumber += "1";
    UpdateLabel(currentNumber);
}

private void Button_2(object sender, RoutedEventArgs e)
{
    currentNumber += "2";
    UpdateLabel(currentNumber);
}

private void Button_3(object sender, RoutedEventArgs e)
{
    currentNumber += "3";
    UpdateLabel(currentNumber);
}
```

Enz….

Hier wordt telkens (per click dus) de waarde hardcoded aan de variabele currentNumber geplakt. En vervolgens wordt het label geüpdatet met het huidige nummer.

Een komma plakken we ook gewoon aan de waarde van currentNumber:

```
private void Button_dot(object sender, RoutedEventArgs e)
{
    currentNumber += ",";
    UpdateLabel(currentNumber);
}
```

Merk op dat we ook elke keer de functie UpdateLabel(currentNumber) uitvoeren, die er dus voor zorgt dat ons label up-to-date blijft, die ziet er als volgt uit:

```
private void UpdateLabel(string text)
{
    currentNumberLabel.Content = text.ToString();
}
```

Hier spreken we de 'Content' property dus aan, vandaar het belang in de xaml code, en geven deze de waarde van de parameter. In onze gevallen was deze parameter telkens 'currentNumber'.

*Functies: operatoren (+, -, /, *)*

```
private void Button_multiply(object sender, RoutedEventArgs e)
{
    task = "*";
    ChangeNumber();
}

private void Button_divide(object sender, RoutedEventArgs e)
{
    task = "/";
    ChangeNumber();
}

private void Button_minus(object sender, RoutedEventArgs e)
{
    task = "-";
    ChangeNumber();
}

private void Button_plus(object sender, RoutedEventArgs e)
{
    task = "+";
    ChangeNumber();
}
```

Telkens als we een operator aanklikken wordt ook weer de Click functie opgeroepen. Hierboven ziet men dat elke keer als er op een operator klikt de task gelijk wordt gesteld aan de operator. Merk op dat dit '=' is en niet '+=' we willen namelijk telkens de waarde vervangen en niet er bij aan plakken.

Na dat de task zijn waarde heeft gekregen roepen we ook telkens de ChangeNumber() functie op, die ziet er als volgt uit:

```
private void ChangeNumber()
{
        number1 = Convert.ToDouble(currentNumber);
        currentNumber = "";

        labelResult.Content = number1.ToString();
        labelCurrentNumber.Content = "0";
}
```

Wat hier de bedoeling is, is dat de huidige nummer nu in een variabele(number1) gestopt wordt en de variabel currentNumber leeg wordt gemaakt zodat men een nieuw getal kan ingeven. De resultaat label krijgt nu de waarde van de variabele(number1) en de input label wordt terug op “0” gezet.

Functies: resultaat een calculatie

```
private void Button_result(object sender, RoutedEventArgs e)
{
    Calculate();
}
```

De result(=) knop roept maar 1 functie op namelijk Calculate(), deze ziet er als volgt uit:

```
private void Calculate()
{

    number2 = Convert.ToDouble(currentNumber);
    currentNumber = "0";
    currentNumberLabel.Content = currentNumber;

    switch (task)
    {
        case "-":
            result = number1 - number2;
            break;
        case "+":
            result = number1 + number2;
            break;
        case "/":
            result = number1 / number2;
            break;
        case "*":
            result = number1*number2;
            break;
        case "=":

            break;
    }

    labelCurrentNumber.Content = result.ToString();
    labelResult.Content = "0";

}
```

Hier wordt het tweede getal ook opgeslagen in een variabele, en de input label wordt weer op 0 gezet.

De switch(task) checkt de waarde van de variabele ‘task’, per operator is er dus een verschillende berekening.

De uitkomst van deze berekening wordt opgeslagen in de variabele ‘result’, deze wordt uiteindelijk gebruikt door de input label gelijk te stellen aan de uitkomst.

Nu zijn we klaar om de code efficiënter te maken en uit te breiden

Het startbestand: Uitbreiding

Uitbreiding: buttons efficiënter

Buttons: getallen

Herinner je je nog dat de knoppen niet zo efficiënt gecodeerd waren? Dat gaan we nu oplossen door in plaats van voor elke knop een functie te maken, 1 functie voor elke knop voorzien. Die functie ziet er als volgt uit.

```
private void NumberButtonClick(object sender, RoutedEventArgs e)
{
    var number = (e.Source as Button).Content.ToString();

    currentNumber += number;
    UpdateLabel(currentNumber);
}
```

De bovenstaande functie is de betere versie van dit:

```
private void Button_0(object sender, RoutedEventArgs e)
{
    currentNumber += "0";
    UpdateLabel(currentNumber);
}

private void Button_1(object sender, RoutedEventArgs e)
{
    currentNumber += "1";
    UpdateLabel(currentNumber);
}

private void Button_2(object sender, RoutedEventArgs e)
{
    currentNumber += "2";
    UpdateLabel(currentNumber);
}

private void Button_3(object sender, RoutedEventArgs e)
{
    currentNumber += "3";
    UpdateLabel(currentNumber);
}
```

Enz…

Wat er nu juist het verschil is, is dat de nummer die aan currentNumber wordt geplakt **niet** meer hard gecodeerd(handmatig ingegeven) moet worden, deze wordt nu in een variabele gestopt die de waarde krijgt van de ‘Content’ property van de geklikte knop:

```
var number = (e.Source as Button).Content.ToString();
```

Verder doen we nog hetzelfde als ervoor maar nu stellen we currentNumber gelijk aan variabele ‘number’. Ook de functie(UpdateLabel(currentNumber)) wordt nu maar op 1 plek gebruikt.

Buttons: Getallen: XAML

De ‘Content’ property moet blijven staan, anders hebben we geen waardes. De ‘Click’ property moet worden **overal**(ook de komma) aangepast naar “NumberButtonClick” omdat men nu bij elke button click dezelfde functie aanspreekt.

```
<Button Content="0" HorizontalAlignment="Left" Margin="10,257,0,0" VerticalAlignment="Top" Width="110" Height="50" Click="NumberButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="1" HorizontalAlignment="Left" Margin="10,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="NumberButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="2" HorizontalAlignment="Left" Margin="70,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="NumberButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="3" HorizontalAlignment="Left" Margin="130,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="NumberButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="4" HorizontalAlignment="Left" Margin="10,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="NumberButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="7" HorizontalAlignment="Left" Margin="10,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="NumberButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="5" HorizontalAlignment="Left" Margin="70,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="NumberButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="8" HorizontalAlignment="Left" Margin="70,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="NumberButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="9" HorizontalAlignment="Left" Margin="130,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="NumberButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="6" HorizontalAlignment="Left" Margin="130,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="NumberButtonClick" FontFamily="Unispace" Foreground="White"/>
```

Buttons: Operatoren

Voor de operatoren doen we praktisch hetzelfde als hierboven voor dezelfde redenen. De functie ziet er nu als volgt uit:

```
private void TaskButtonClick(object sender, RoutedEventArgs e)
{
    var task_ = (e.Source as Button).Content.ToString();

    task = task_;
    ChangeNumber();
}
```

De bovenstaande functie is de betere versie van dit:

```
private void Button_dot(object sender, RoutedEventArgs e)
{
    currentNumber += ",";
    UpdateLabel(currentNumber);
}

private void Button_multiply(object sender, RoutedEventArgs e)
{
    task = "*";
    ChangeNumber();
}

private void Button_divide(object sender, RoutedEventArgs e)
{
    task = "/";
    ChangeNumber();
}

private void Button_minus(object sender, RoutedEventArgs e)
{
    task = "-";
    ChangeNumber();
}

private void Button_plus(object sender, RoutedEventArgs e)
{
    task = "+";
    ChangeNumber();
}
```

Buttons: Operatoren: XAML

Ook hier alle ‘Content’ properties laten staan, en ook overal ‘Click’ property aanpassen naar “TaskButtonClick”.

```
<Button Content="+" HorizontalAlignment="Left" Margin="190,197,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="TaskButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="-" HorizontalAlignment="Left" Margin="190,137,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="TaskButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="*" HorizontalAlignment="Left" Margin="190,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="TaskButtonClick" FontFamily="Unispace" Foreground="White"/>
<Button Content="/" HorizontalAlignment="Left" Margin="250,77,0,0" VerticalAlignment="Top" Width="50" Height="50" Click="TaskButtonClick" FontFamily="Unispace" Foreground="White"/>
```

Aan de slag!

Dit was de begeleiding, probeer nu zelfstandig verder verder te werken (zie volgende pagina). Vraag gerust om hulp als het nodig is.

Zelfstandige uitbreiding

Probeer nu zelfstandig een knop toe te voegen dat alles terug leeg maakt. De ‘C’ op een rekenmachine dus. Voeg de knop maar toe via de toolbox, stel de click en content property zeker in en werk de clickfunctie verder uit in de codebehind(de klikfunctie kan men makkelijk toevoegen door dubbel op de aangemaakte knop in het design venster te klikken).

Als dit gelukt is ben je vrij om zelf nog layout aan te passen of misschien nog extra functionaliteiten toe te voegen zoals machten of vierkantswortels.