

# C# Essentials Introductie

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- Installatie Visual Studio
- Overzicht semester
- VS IDE
- Hello World – WPF
- Hello World – console

# Installatie Visual Studio

- [www.visualstudio.com](http://www.visualstudio.com)
- Download Visual Studio Community edition

## Meet the Visual Studio family



**Visual Studio | Windows**

The best comprehensive IDE for .NET and C++ developers on Windows. Fully packed with a sweet array of tools and features to elevate and enhance every stage of software development.

[Learn more >](#)

[Download Visual Studio](#) ▾

- Community 2022
- Professional 2022
- Enterprise 2022

Free for individual developers, academic uses, and open source



**Visual Studio for Mac | macOS**

A comprehensive IDE for .NET developers that's native to macOS. Includes top-notch support for web, cloud, mobile, and game development.

[Learn more >](#)

[Read more about activating your license](#)

[Download Visual Studio for Mac](#)



**Visual Studio Code | Windows, macOS, Linux**

A standalone source code editor that runs on Windows, macOS, and Linux. The top pick for JavaScript and web developers, with extensions to support just about any programming language.

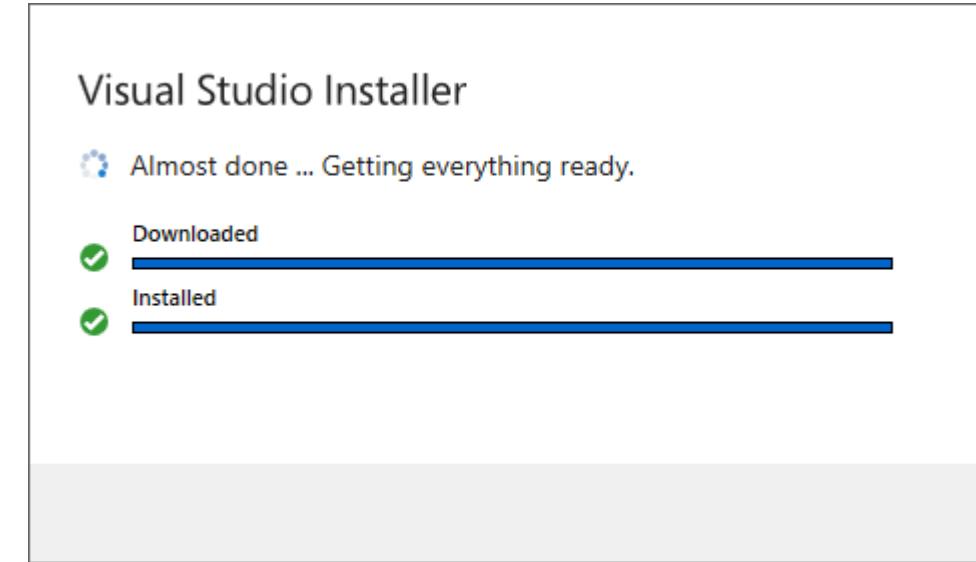
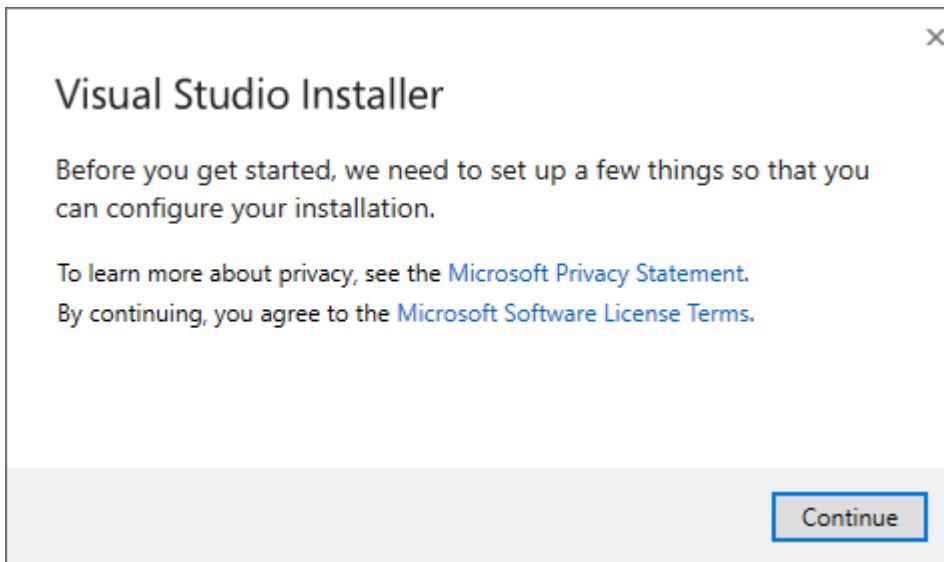
[Learn more >](#)

By using Visual Studio Code you agree to its [license](#) & [privacy statement](#)

[Download Visual Studio Code](#) ▾

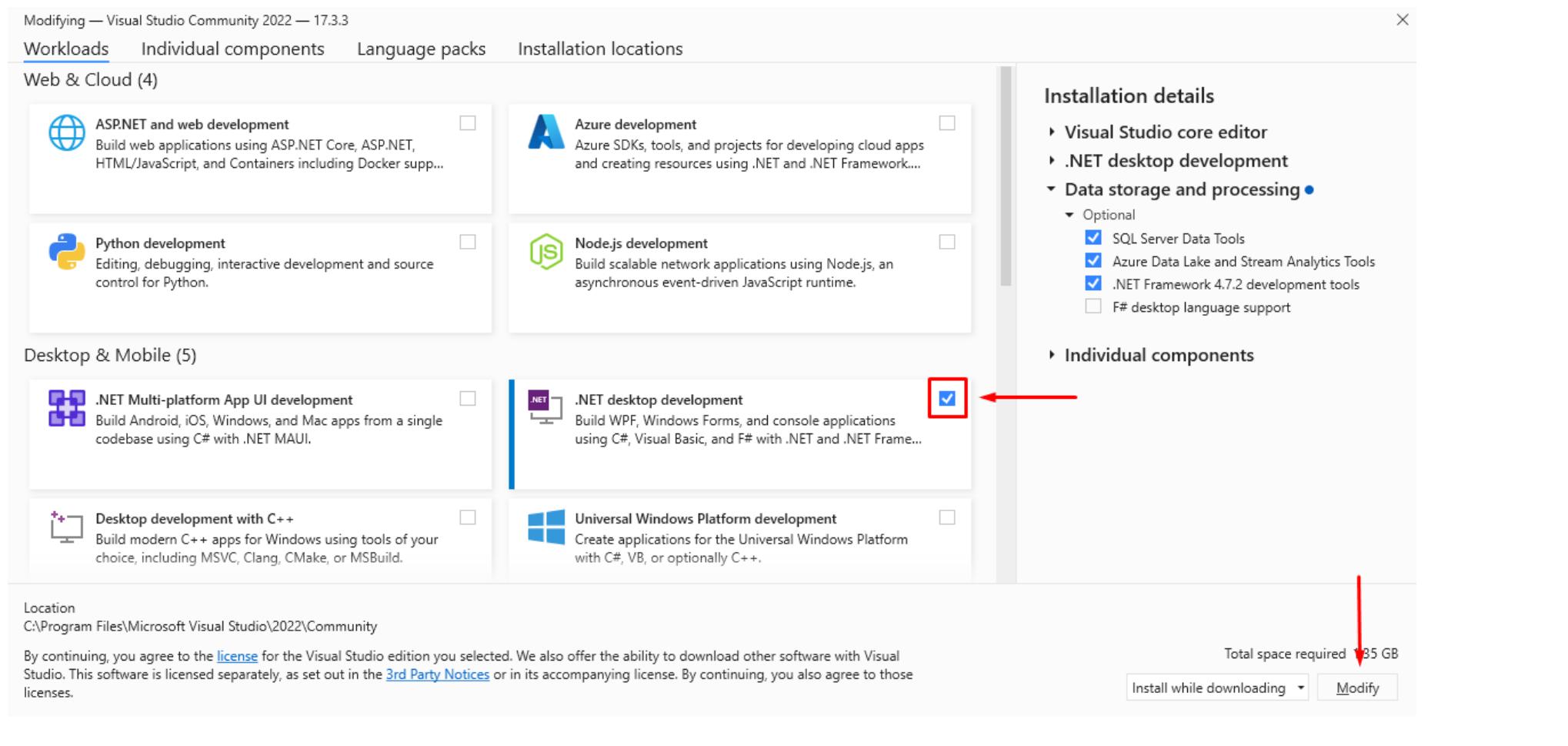
# Installatie Visual Studio

- Klik *Continue*



# Installatie Visual Studio

- Selecteer .NET desktop development



# Installatie Visual Studio

Visual Studio Installer

Installed Available

All installations are up to date.

 Visual Studio Community 2022

Downloading and verifying: 68 MB of 2,68 GB ( 5 MB/sec )

2% 

Installing: package 0 of 0

0% 

Verifying...

[Release notes](#)

[Pause](#)

Developer News

[Adding color to bracket pairs](#)  
When dealing with deeply nested brackets in Visu...  
maandag 5 september 2022

[Live Share: Enterprise Policies are here!](#)  
Securing your Visual Studio Live Share session ha...  
woensdag 17 augustus 2022

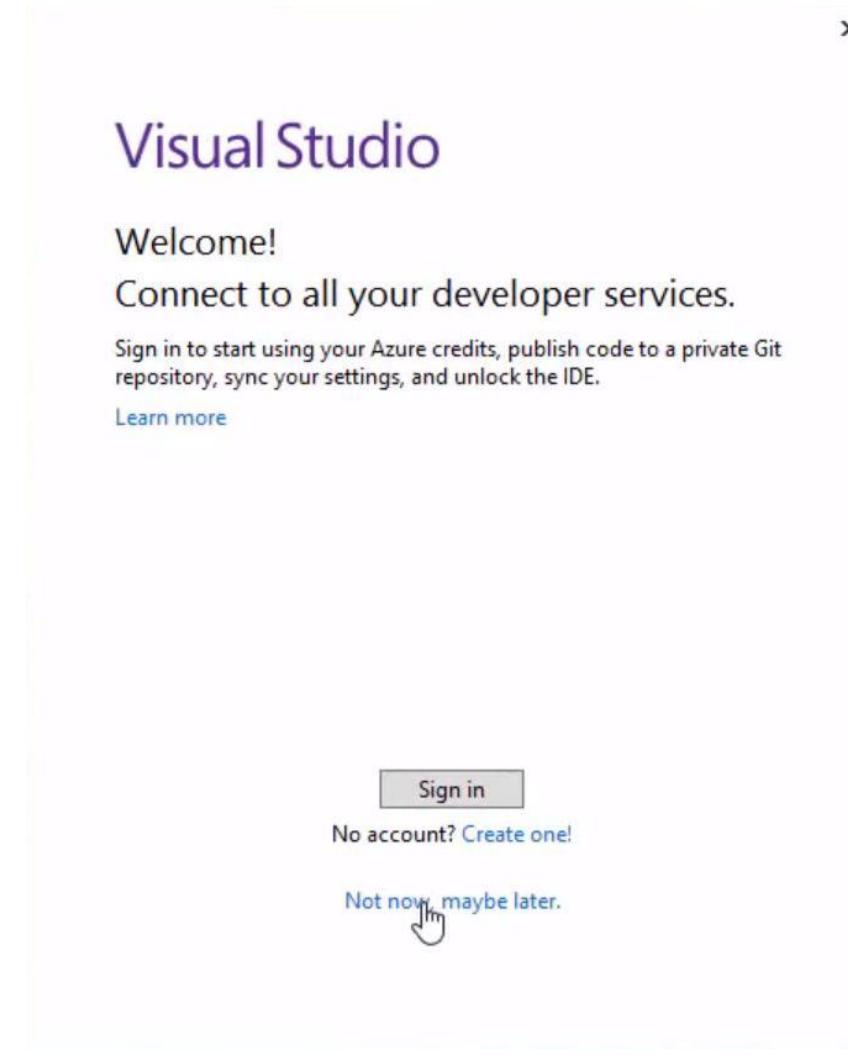
[What's New in Visual Studio 2022 17.4 Preview 1](#)  
We released Visual Studio 2022 17.4 Preview 1 las...  
dinsdag 16 augustus 2022

[View more Microsoft developer news...](#)

Need help? Check out the [Microsoft Developer Community](#) or reach us via [Visual Studio Support](#).

Installer version 3.3.2181.41457

# Installatie Visual Studio



# Installatie Visual Studio

x

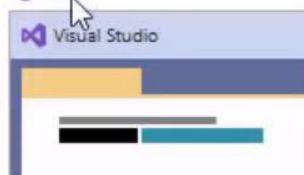
## Visual Studio

Start with a familiar environment

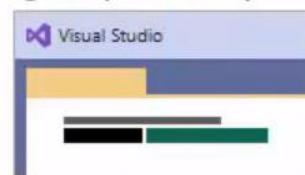
Development Settings: General

Choose your color theme

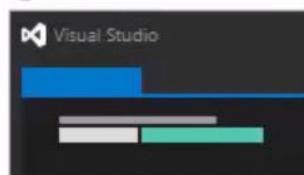
Blue



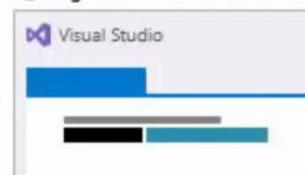
Blue (Extra Contrast)



Dark



Light



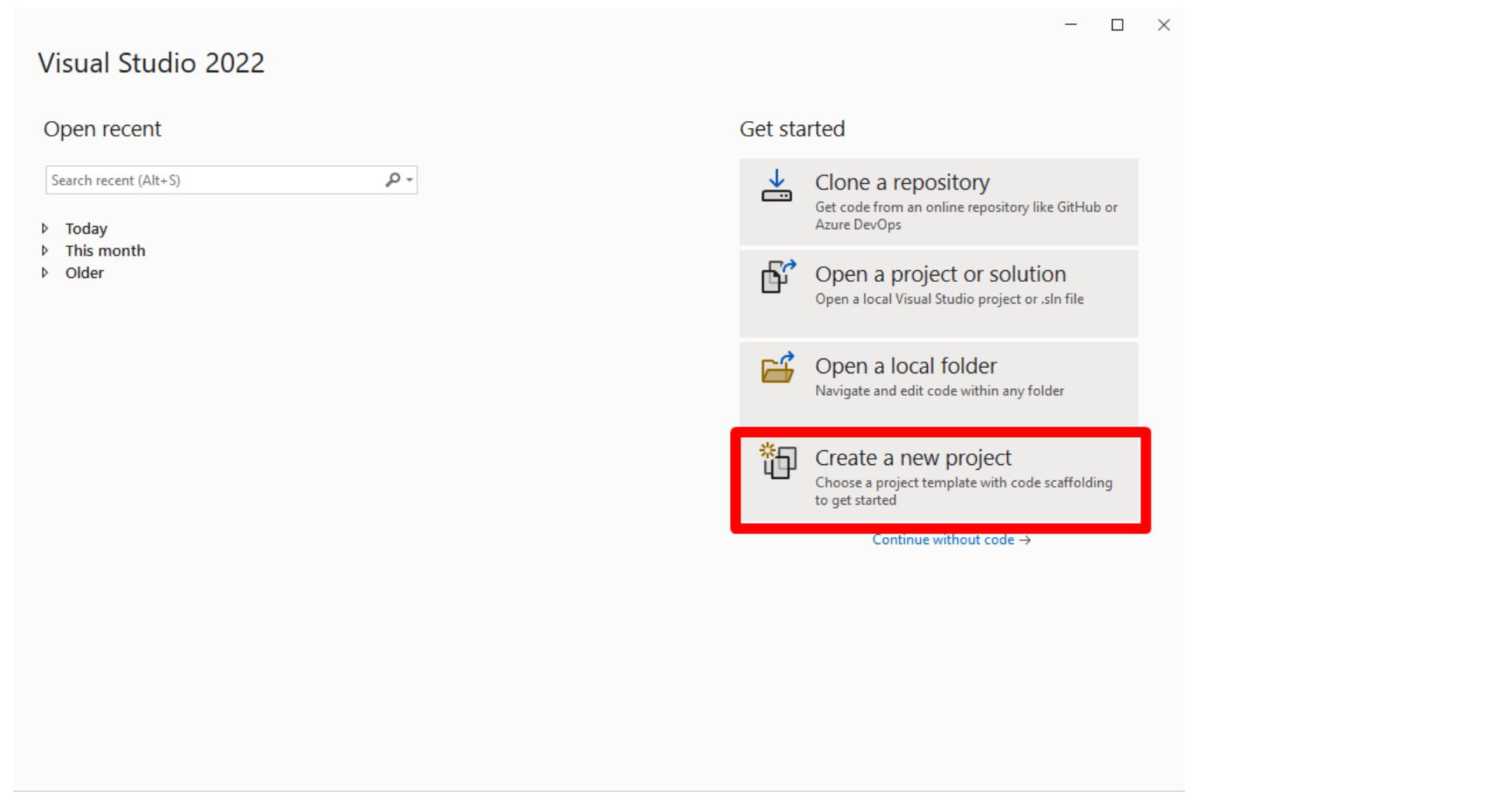
You can always change these settings later.

Start Visual Studio

# Overzicht semester

- Eigen applicaties maken met WPF en console
- Variabelen en berekeningen met operatoren
- Conversies
- Veel gebruikte klassen: String, Tekst, DateTime, ...
- Beslissingen: if en switch
- Herhaling: for, while, do en foreach
- Event procedures en parameters
- Arrays en generieke collecties
- Fouten opsporen: de gestructureerde foutafhandeling
- Programmeerstijl en naamconventies
- Elementaire WPF-besturingselementen

# Visual Studio IDE



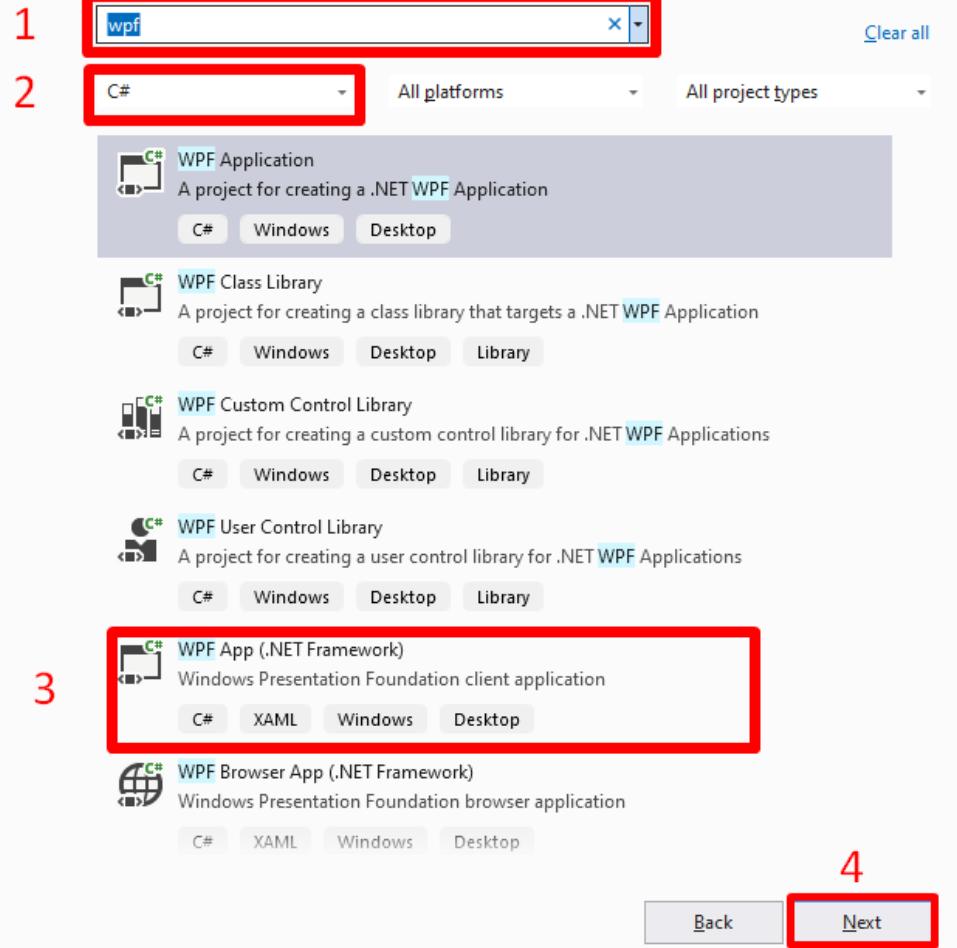
# Visual Studio IDE

Create a new project

Recent project templates

WPF App (.NET Framework)

C#



# Visual Studio IDE

Configure your new project

WPF App (.NET Framework) C# XAML Windows Desktop

Project name

HelloWorld

Location

C:\Users\20003926\source\repos



Solution name ⓘ

HelloWorld

Place solution and project in the same directory

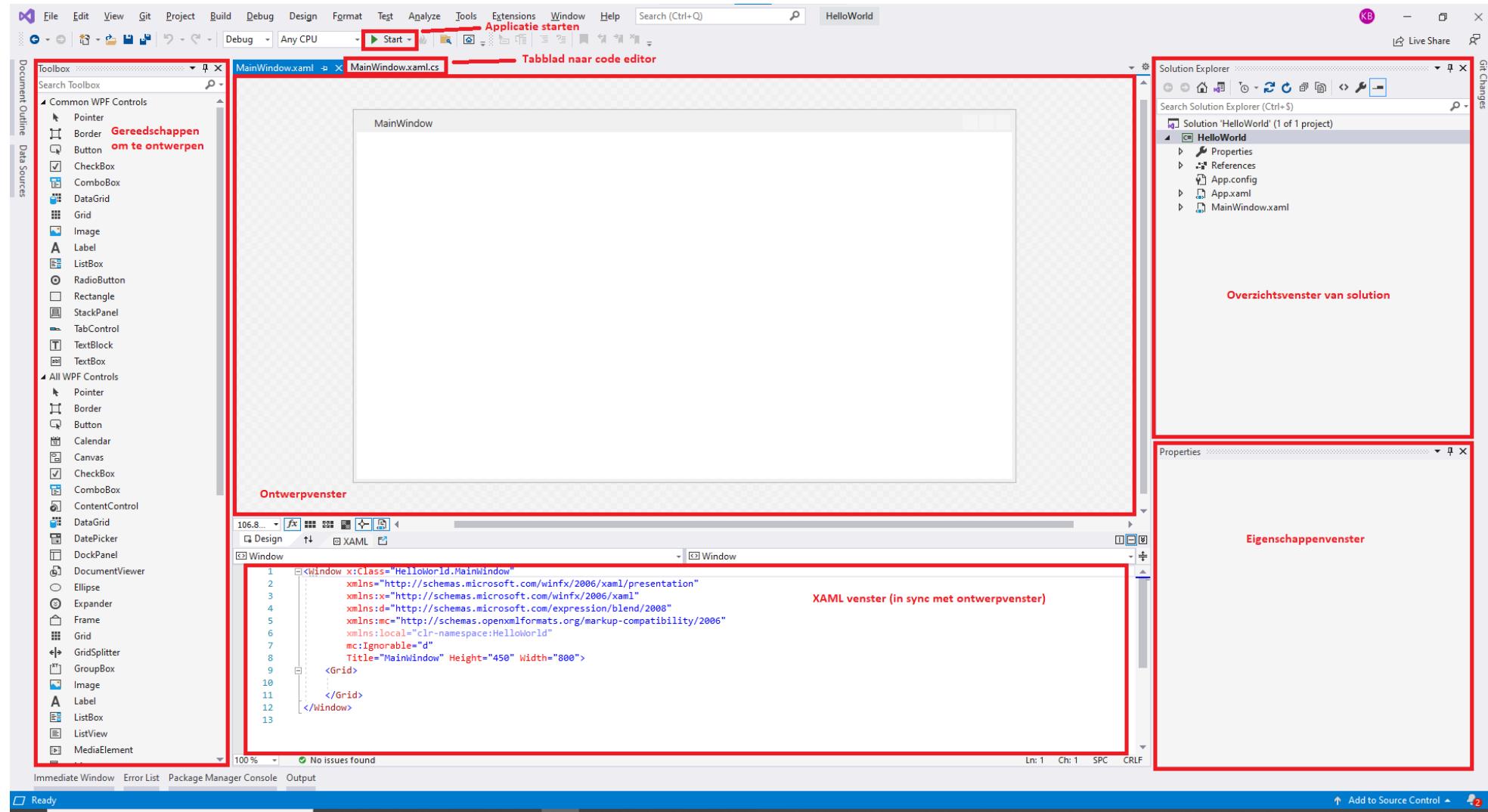
Framework

.NET Framework 4.7.2

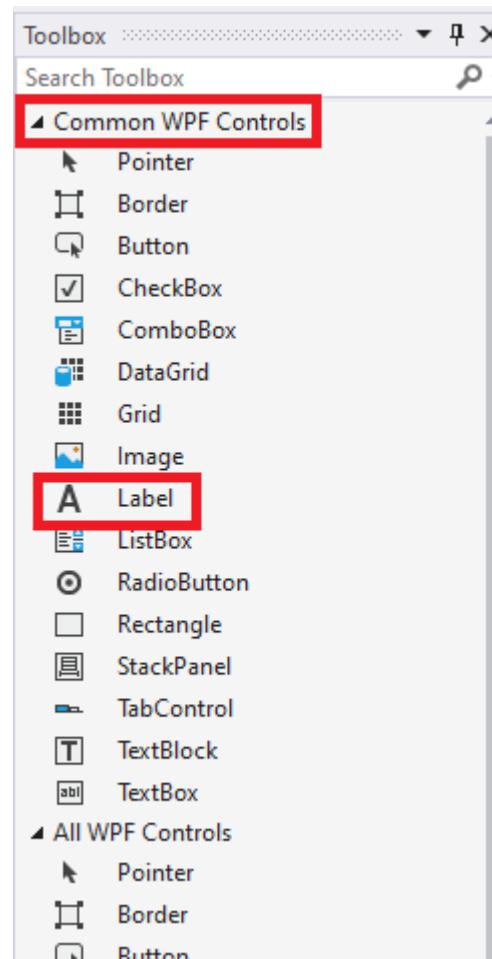
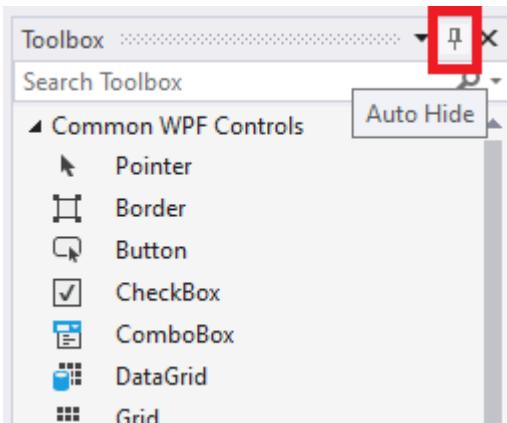
Back

Create

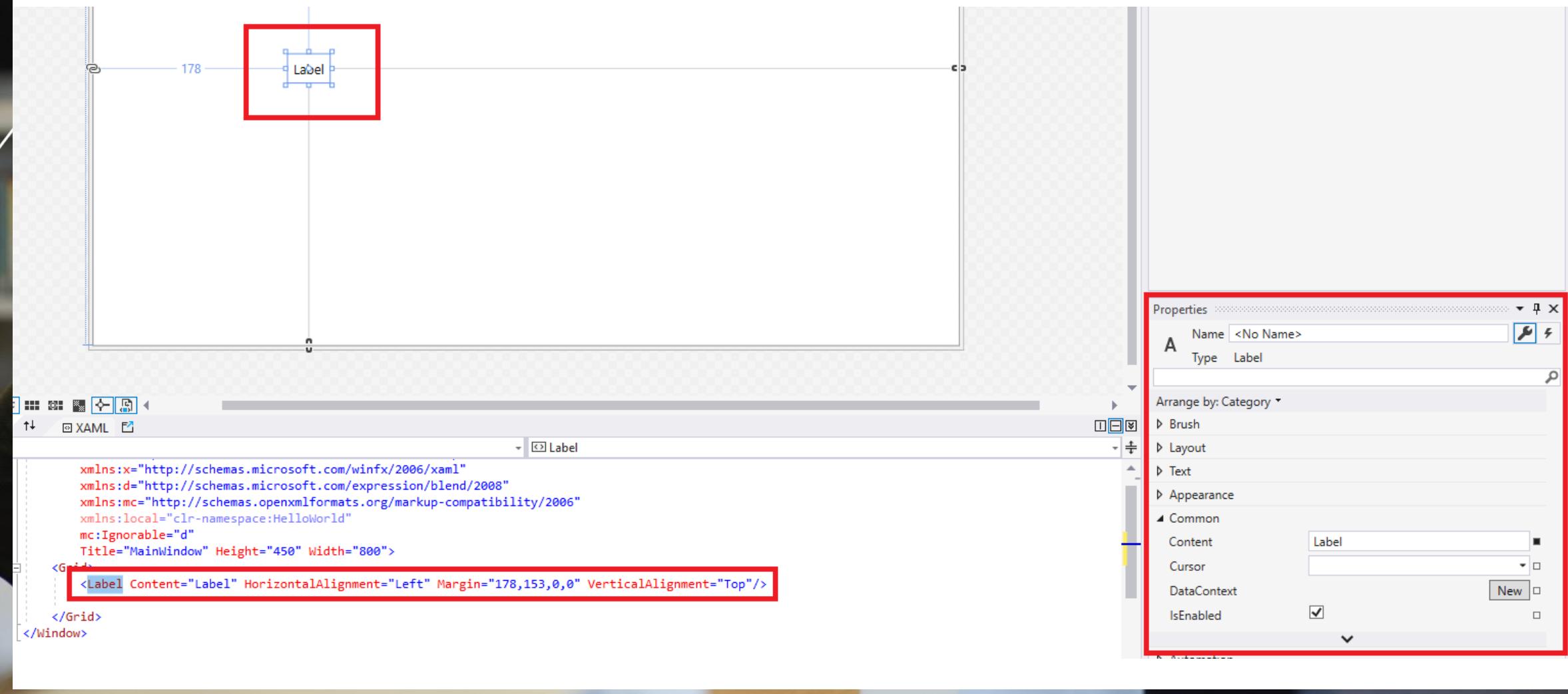
# Visual Studio IDE



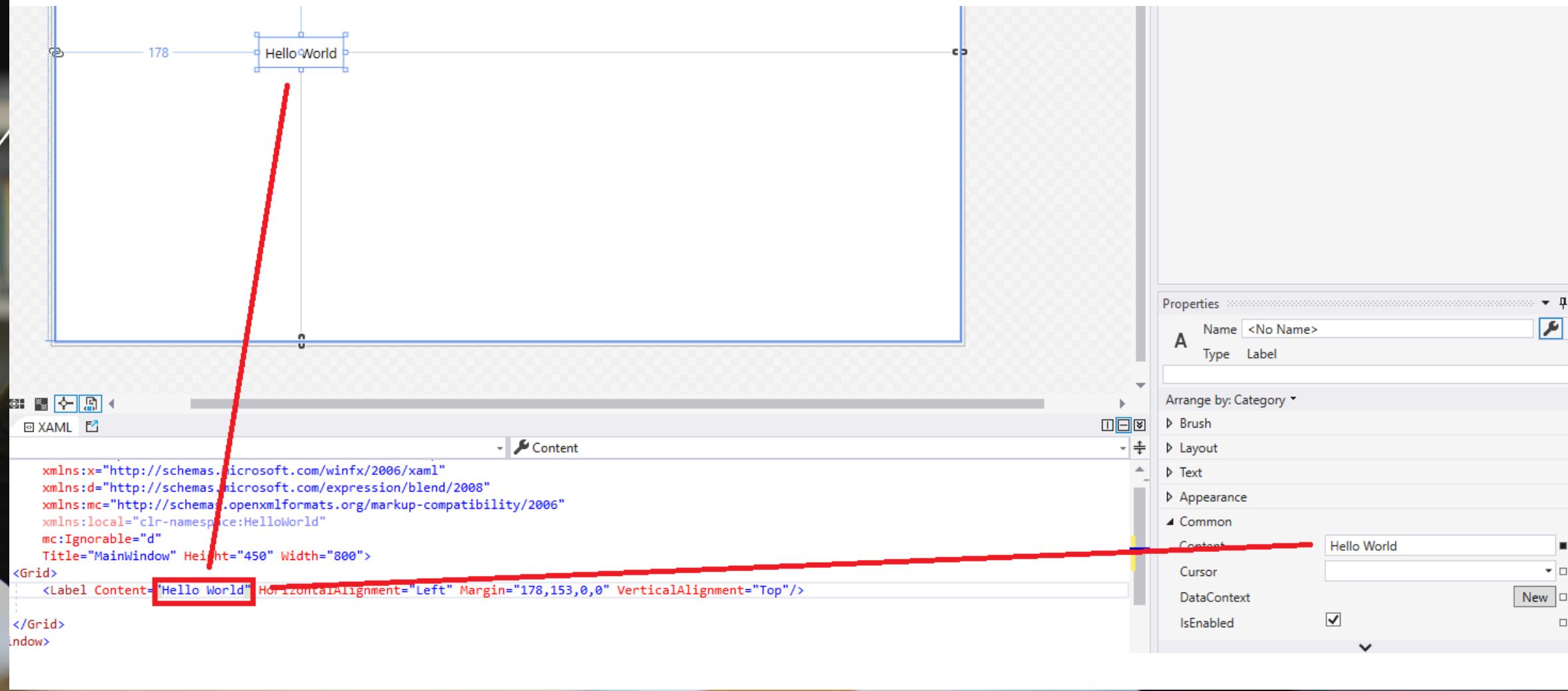
# Visual Studio IDE



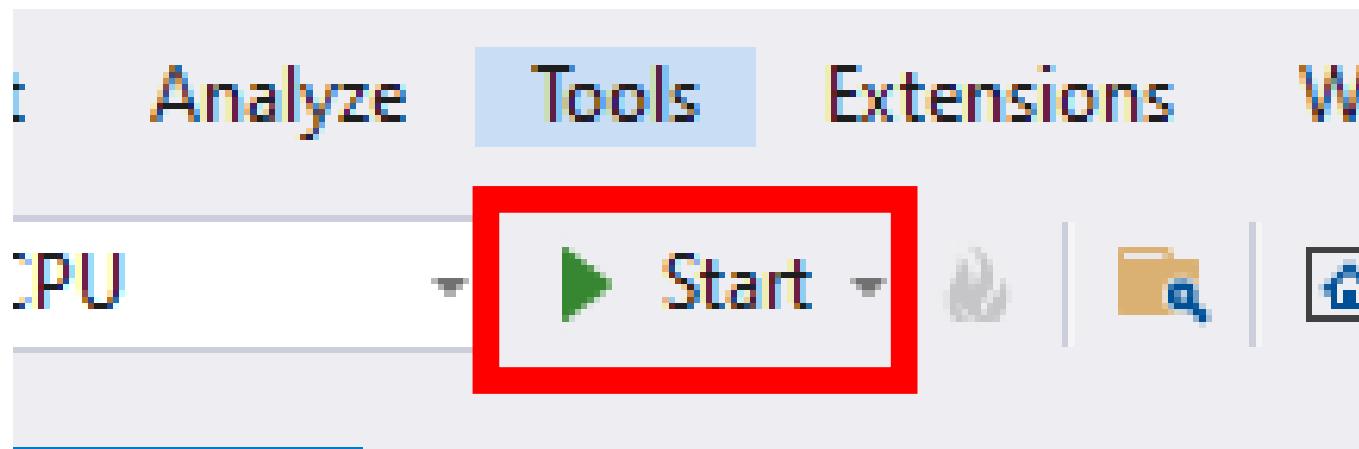
# Visual Studio IDE



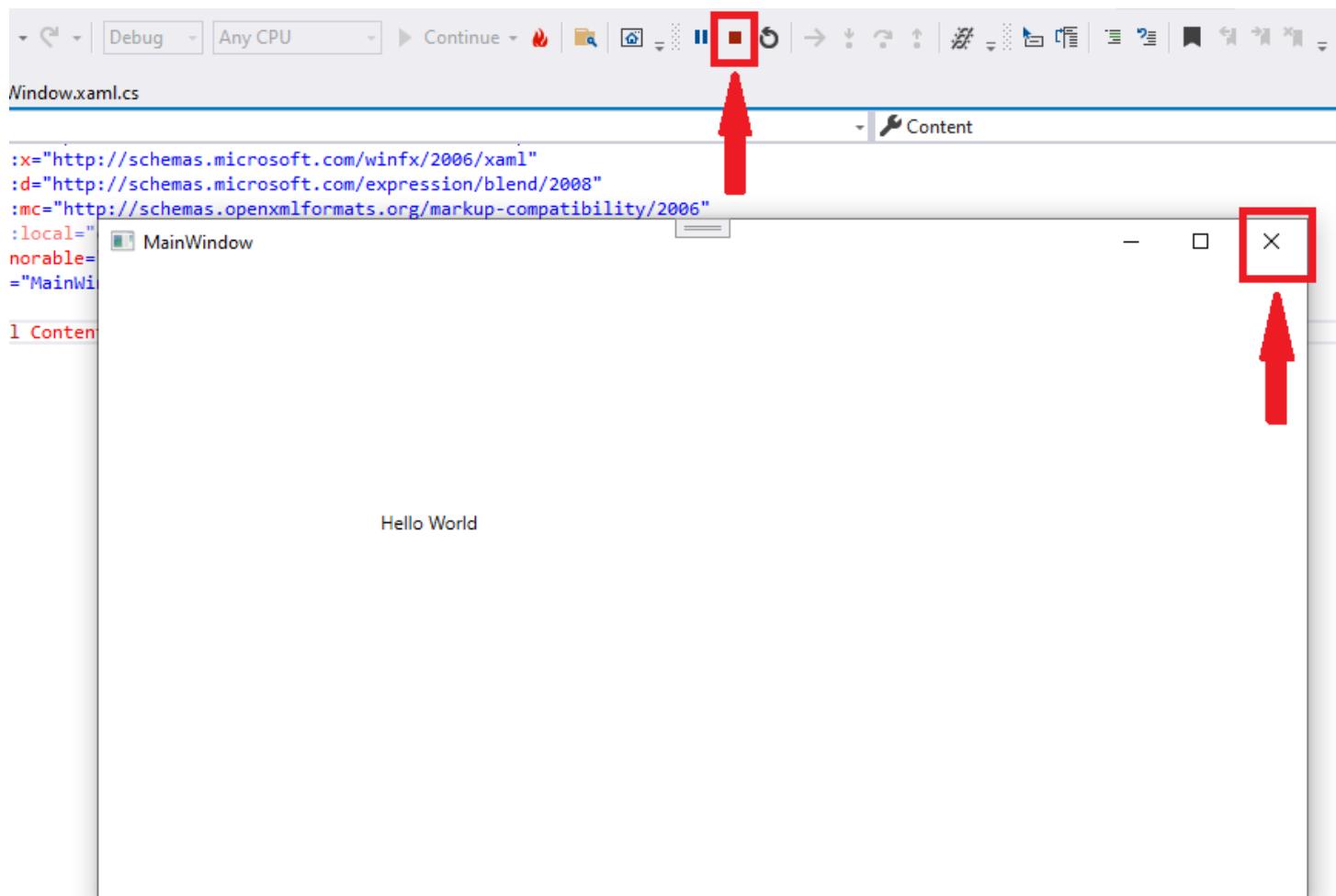
# Visual Studio IDE



# Visual Studio IDE



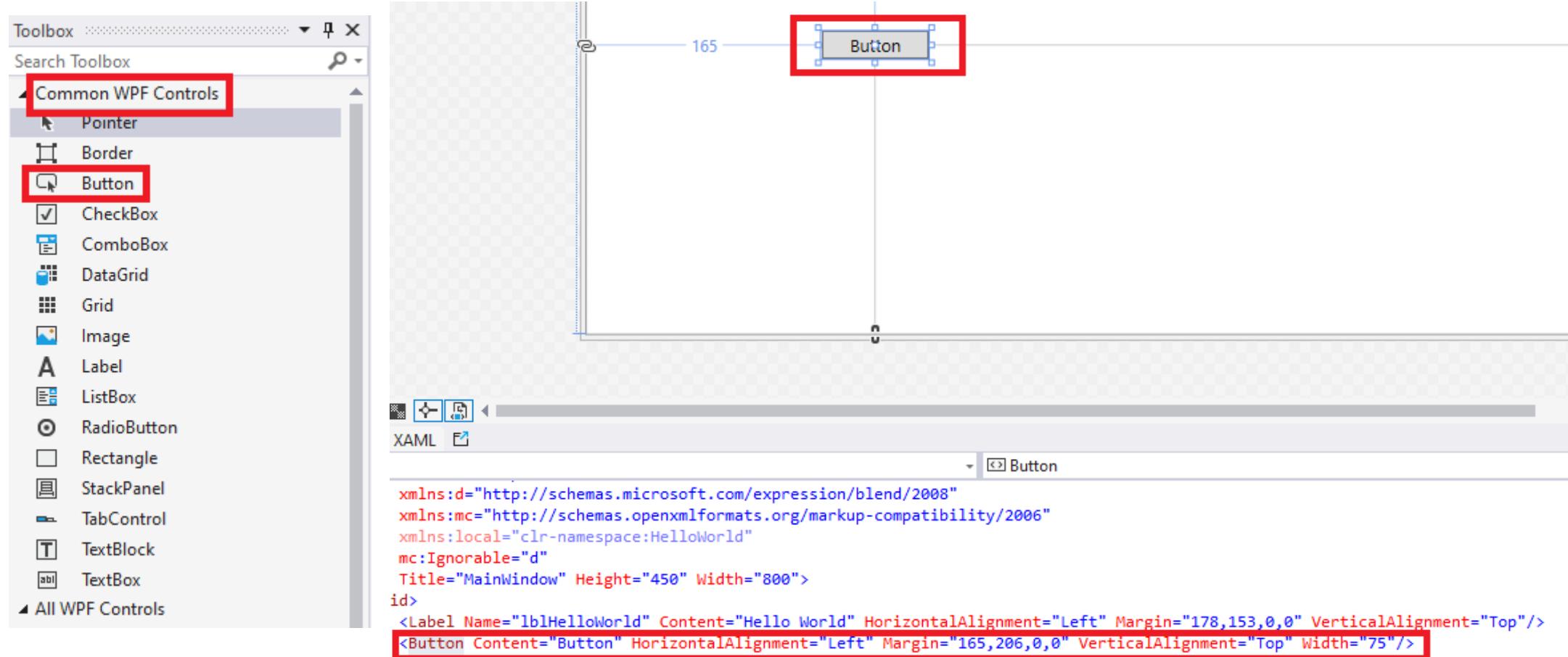
# Visual Studio IDE



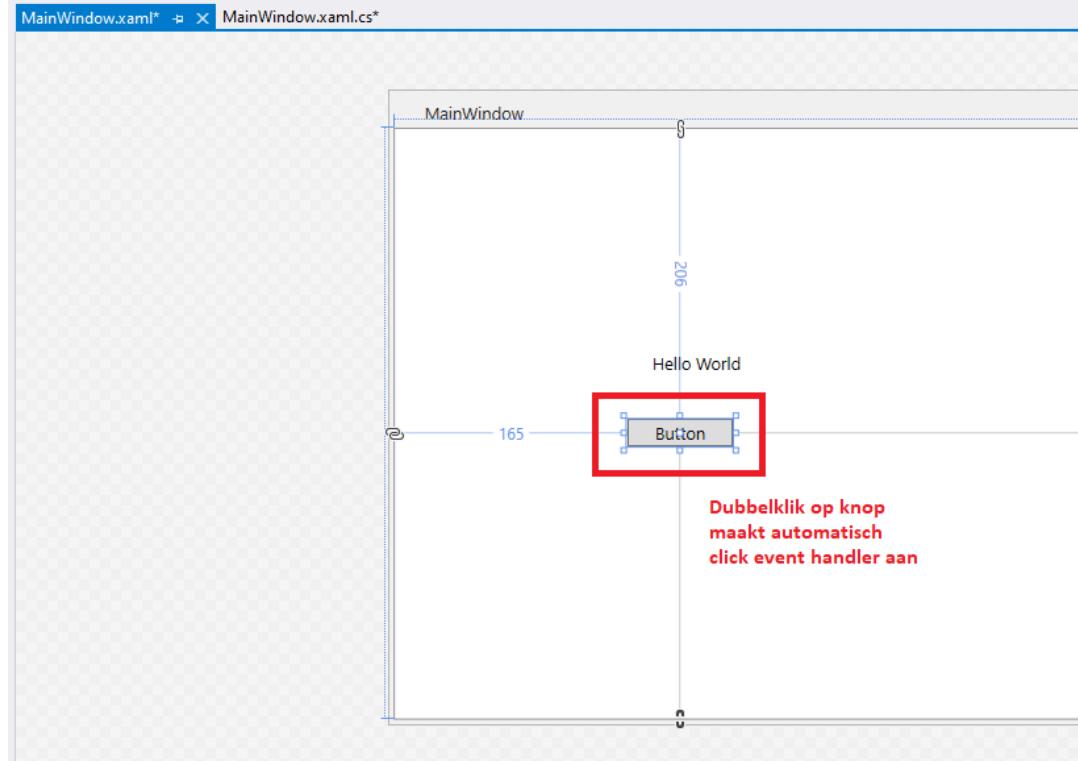
# Hello World

```
Title="MainWindow" Height="450" Width="800">
Grid>
  <Label Name="lblHelloWorld" Content="Hello World" HorizontalAlignment="Left" Margin="178,153,0,0" VerticalAlignment="Top"/>
/Grid>
```

# Hello World



# Hello World

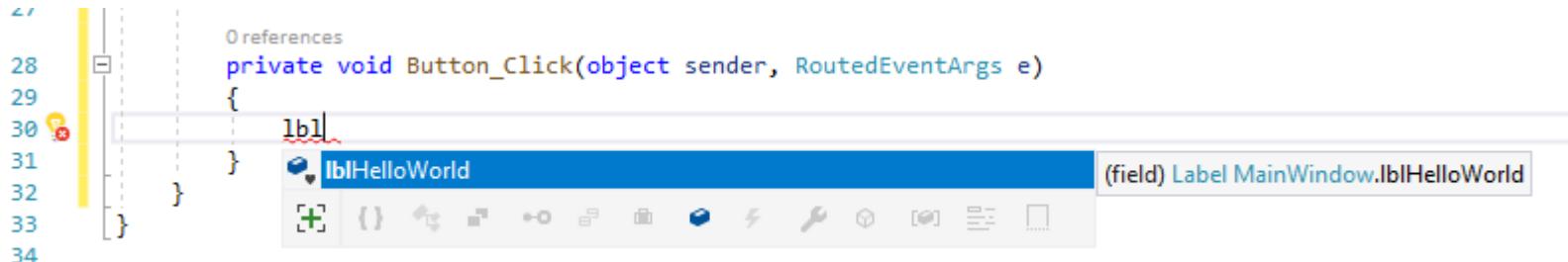


The screenshot shows the Visual Studio code editor with the file 'MainWindow.xaml.cs\*' open. The tab bar at the top shows 'MainWindow.xaml\*' and 'MainWindow.xaml.cs\*' (the latter is highlighted with a red box). The code editor displays the following C# code:

```
6  using System.Windows;
7  using System.Windows.Controls;
8  using System.Windows.Data;
9  using System.Windows.Documents;
10 using System.Windows.Input;
11 using System.Windows.Media;
12 using System.Windows.Media.Imaging;
13 using System.Windows.Navigation;
14 using System.Windows.Shapes;
15
16 namespace HelloWorld
17 {
18     /// <summary>
19     /// Interaction logic for MainWindow.xaml
20     /// </summary>
21     public partial class MainWindow : Window
22     {
23         public MainWindow()
24         {
25             InitializeComponent();
26         }
27
28         private void Button_Click(object sender, RoutedEventArgs e)
29         {
30
31         }
32     }
33 }
34
```

A red box highlights the entire code block, and another red box highlights the 'private void Button\_Click(object sender, RoutedEventArgs e)' method definition.

# Hello World



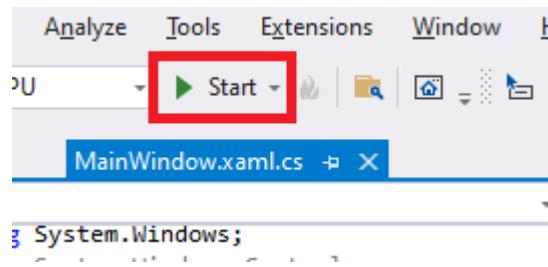
```
27
28    private void Button_Click(object sender, RoutedEventArgs e)
29    {
30        lblHelloWorld.Content = "Welkom in de wereld van C#!";
31    }
32}
33
34
```

The screenshot shows a code editor with two sections of C# code. The top section contains a button click event handler with a breakpoint at line 30. The bottom section shows the code after it has been run, with the content of the label updated to "Welkom in de wereld van C#!".

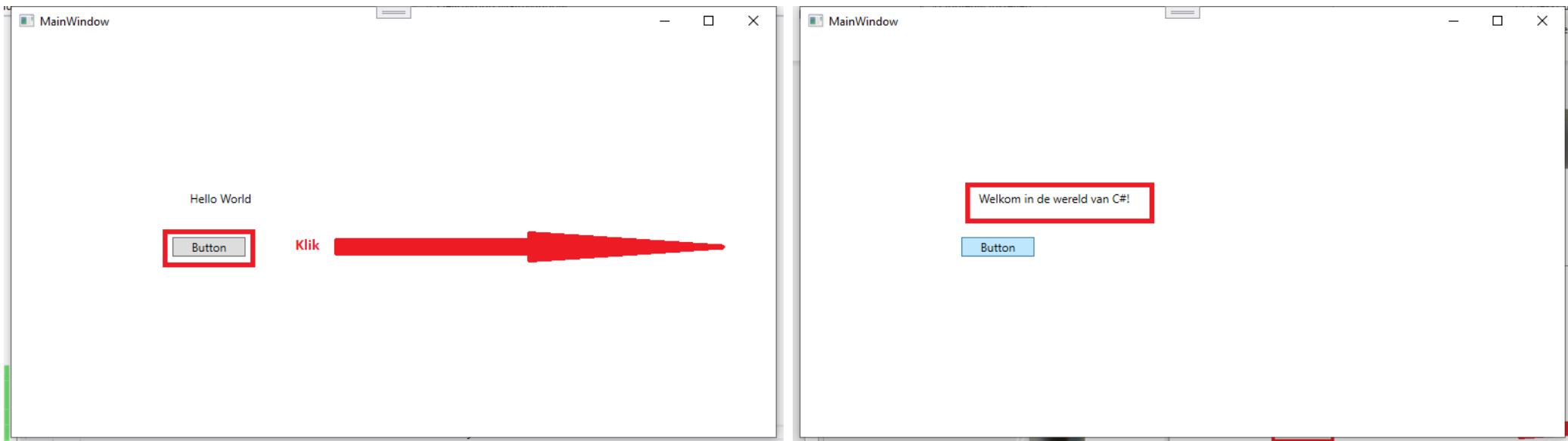


```
28
29    private void Button_Click(object sender, RoutedEventArgs e)
30    {
31        lblHelloWorld.Content = "Welkom in de wereld van C#!";
32    }
33
34
```

The screenshot shows a code editor with the same C# code as the previous one, but the content of the label has been changed to "Welkom in de wereld van C#!" by the button click event.

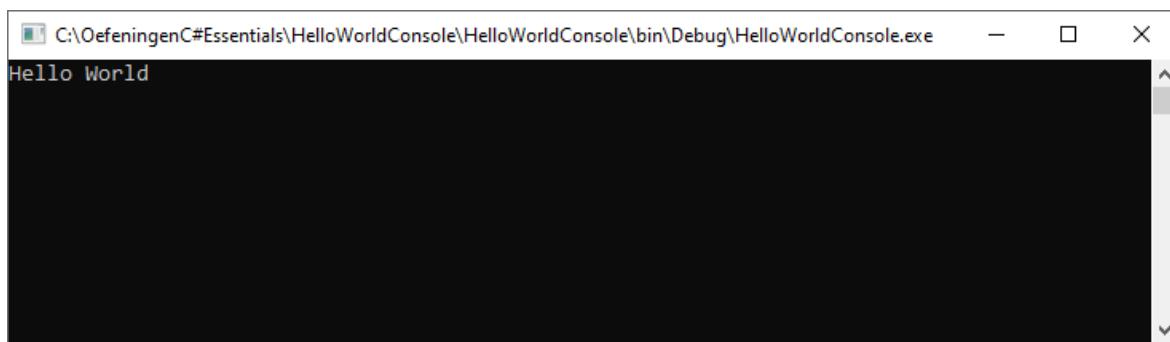


# Hello World

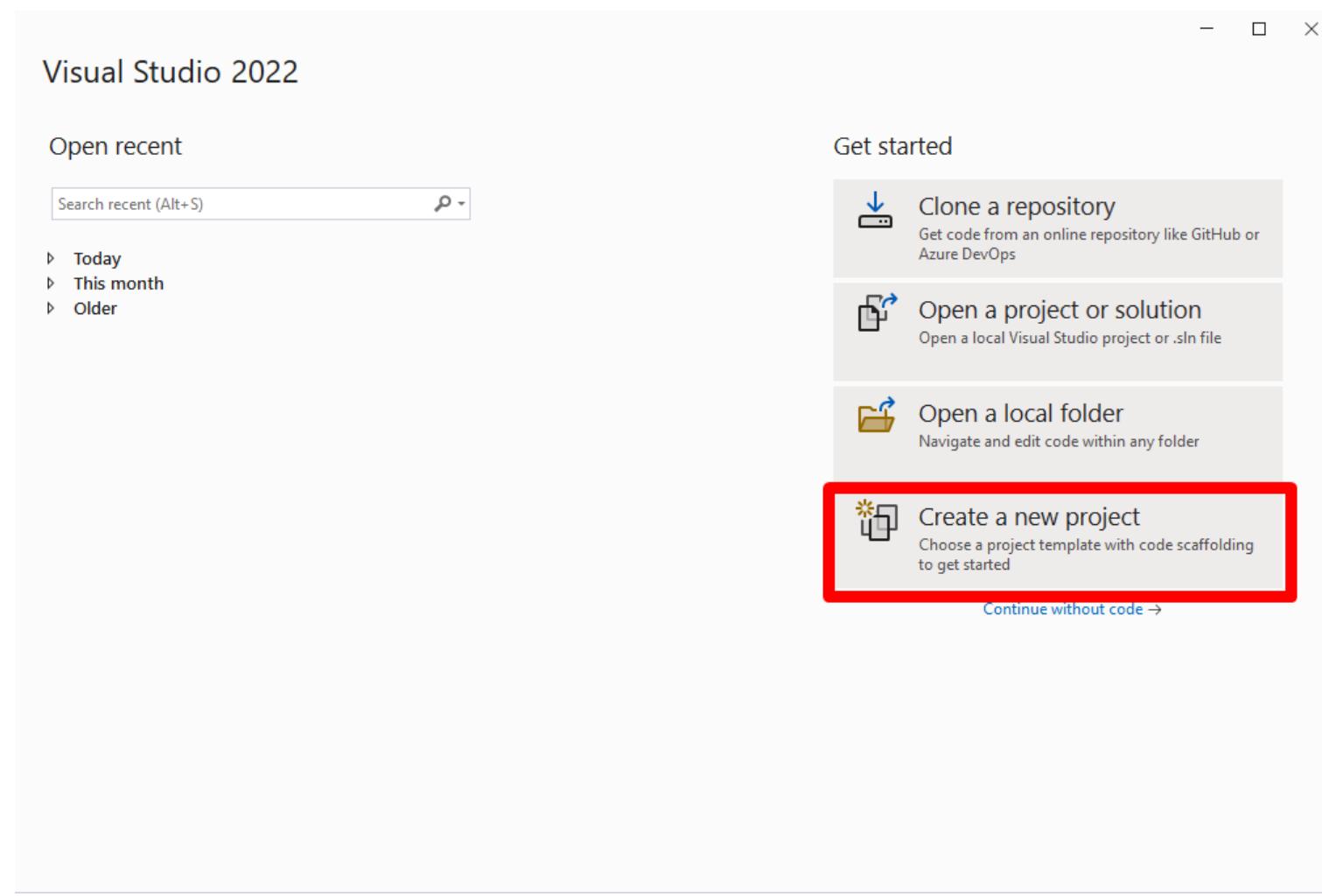


# Console applicatie

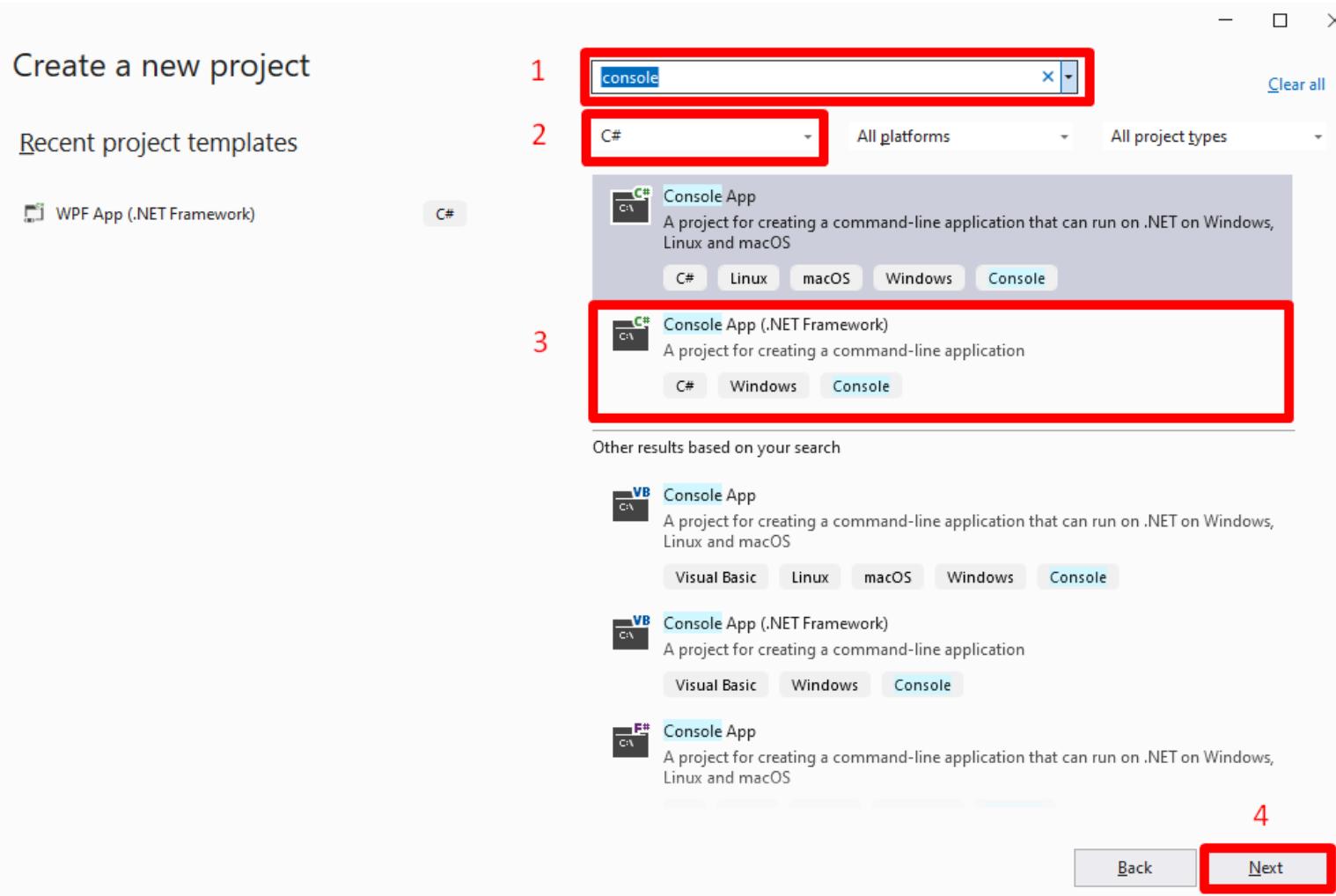
- Toepassing zonder vensters en componenten
  - Geen grafische interface (GUI)
  - Geen labels, tekstvelden, knoppen, ...
  - Puur tekst
- Gebruikt om stukken code *snel* te testen
- Sneller dan WPF applicatie, maar (veel) beperkter in mogelijkheden
- Ook gebruikt om de syntax (C#) te leren



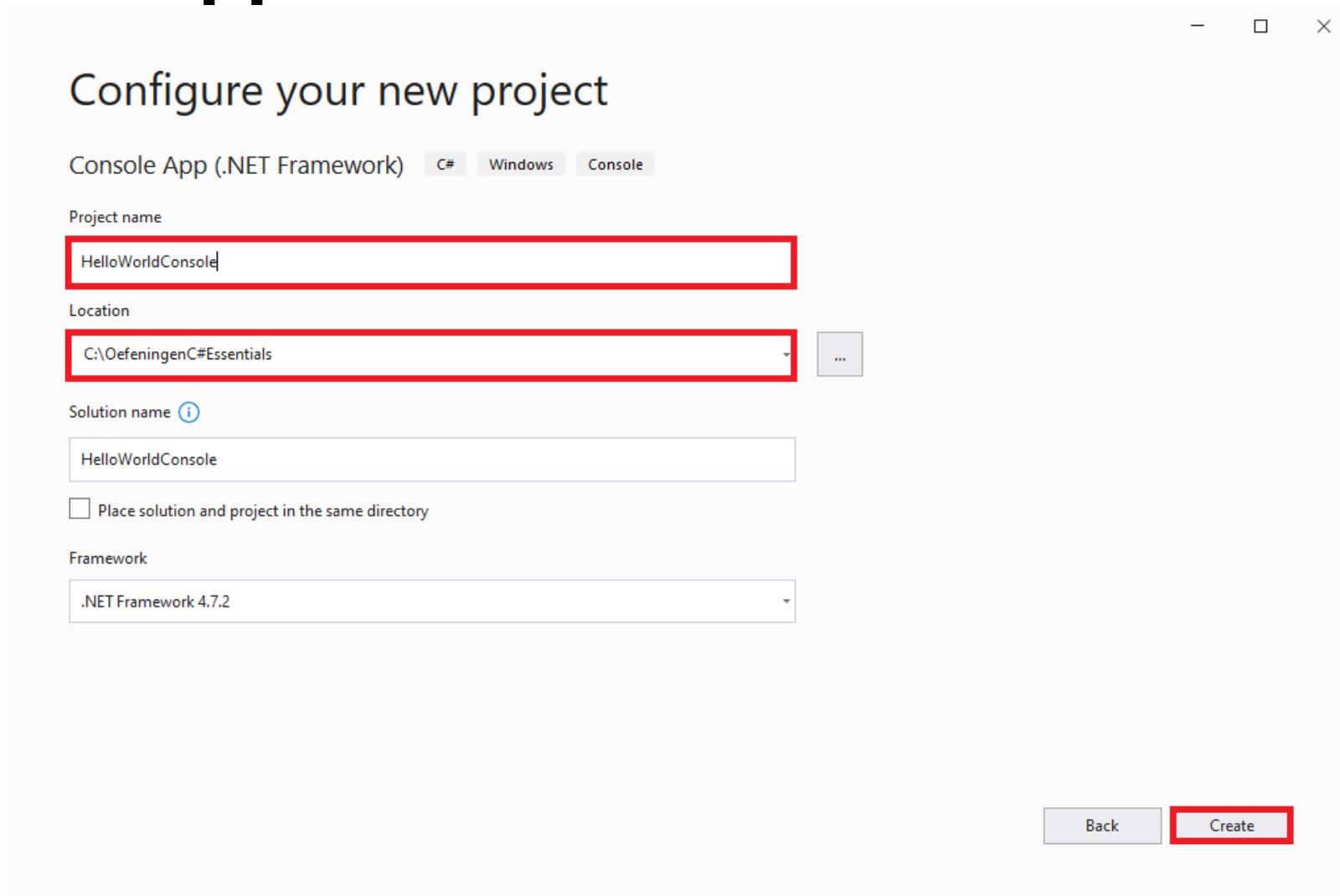
# Visual Studio IDE



# Console applicatie



# Console applicatie



# Console applicatie

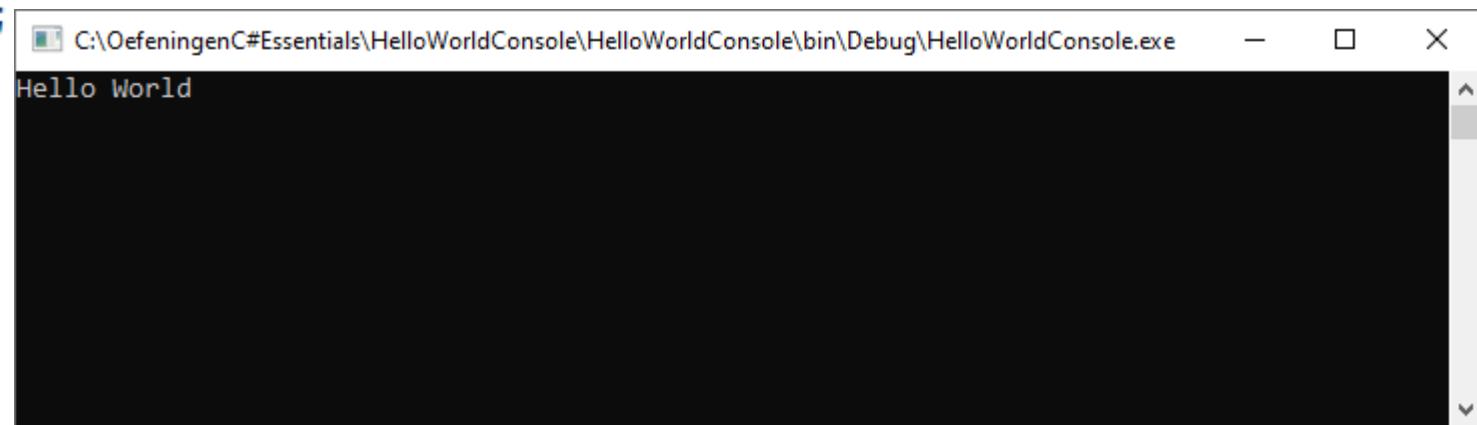
- Main is het startpunt van een console applicatie

```
namespace HelloWorldConsole
{
    internal class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

# Console applicatie

- Console.WriteLine("") om tekst naar de console te sturen
- Console.ReadLine() om tekst van de console te lezen

```
namespace HelloWorldConsole
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
            Console.ReadLine();
        }
    }
}
```



# Console applicatie

- Voorbeeld met naam en getal inlezen en afdrukken

# Console applicatie

```
static void Main(string[] args)
{
    string naam;
    int getal;

    //Inlezen
    Console.Write("Geef een naam: ");
    naam = Console.ReadLine();
    Console.Write("Geef een getal: ");
    getal = int.Parse(Console.ReadLine());

    //Afdruk
    Console.WriteLine("Afdruk");
    Console.WriteLine("-----");
    Console.WriteLine($"Naam: {naam}\t Getal: {getal}");
    Console.WriteLine("Druk op enter om af te sluiten...");
    Console.ReadLine();
}
```

```
Geef een naam: PXL
Geef een getal: 55
Afdruk
-----
Naam: PXL      Getal: 55
Druk op enter om af te sluiten...
```

# Naming conventions

- camelCasing
  - Begint met een kleine letter
  - Elk woord in de samenstelling begint met een hoofdletter
  - Gebruikt voor variabelen en functieparameters
  - bijv.: naamCursist, rekeningBank, studentNaamPerAfdeling
- PascalCasing
  - Ook wel UppercamelCasing genoemd
  - Gebruikt voor functienamen, class definities, e.d.

# Gestructureerd analytisch denken

- Fase 1: Probleemanalyse
- Fase 2: Oplossingsstrategie
- Fase 3: Algoritme
- Fase 4: Programmeren
- Fase 5: Testen
- Fase 6: Documenteren

# Fase 1: Probleemanalyse

- Wat moet er gebeuren?
- Hoe pak ik dit aan?
- Wat is het probleem?
- Over welke gegevens beschik ik?
- Welk resultaat wordt gevraagd?

## *Fase 1 Probleemanalyse*

Gegeven: /

Gevraagd:

- Wijzig tekst in "Welkom programmeurs"
- Berichtenvenster "Hallo netwerkbeheer"
- Wijzig tekst in "Muis zweeft over label"

# Fase 2: Oplossingsstrategie

- De basisstructuren voor het programma worden hier gemaakt.
- Hoe los ik het probleem op?
- Welke verfijnde technieken kan ik hiervoor gebruiken?

## *Fase 2 Oplossingsstrategie*

- Klik op 'Wijzigen': druk tekst "Welkom programmeurs" in label
- Klik op 'Bericht': berichtenvenster "Hallo netwerkbeheer"
- Muis over label: druk tekst "Muis zweeft over label" in label

# Fase 3: Algoritme

- Voorschrift
- Stapsgewijs opgebouwd
- Een algoritme:
  - is duidelijk, ondubbelzinnig en volledig gedefinieerd
  - is vrij zijn van redundantie
  - heeft uitgeschreven invoer- en uitvoerspecificaties
  - is gestructureerd
  - moet eindigen

## *Fase 3 Pseudocode*

```
Begin BtnWijzigen_  
    druk "Welkom programmeurs" (LblHallo)  
Einde
```

```
Begin BtnBericht_Click  
    druk "Hallo netwerkbeheer" (berichtenvenster)  
Einde
```

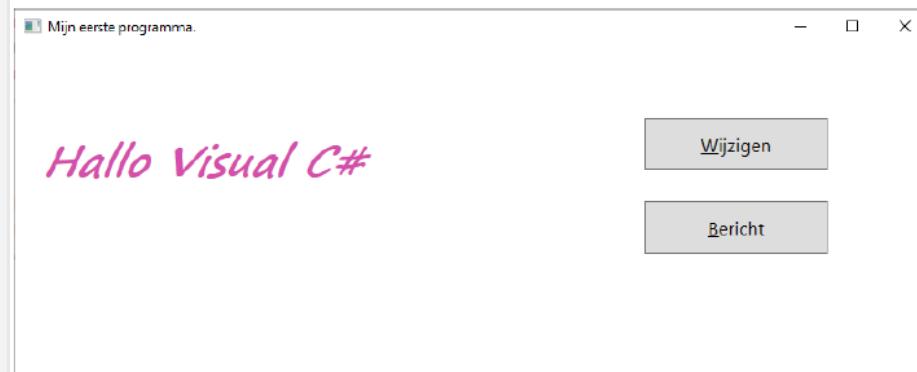
```
Begin LbTekst_MouseEnter  
    druk "Muis zweeft over label" (LblHallo)  
Einde
```

# Fase 4: Programmeren

- Het effectief maken van het programma in bijv. C#

## Fase 4 Programmeren

### 4.1 Ontwerp fase / Visual programming



### 4.2 Programmeer fase / Code programming

```
private void ButtonWijzigen_Click(object sender, RoutedEventArgs e)
{
    LabelHallo.Content = "Welkom programmeurs!";
}

private void LabelHallo_MouseEnter(object sender, MouseEventArgs e)
{
    LabelHallo.Content= "Muis zweeft over label.";
}

private void ButtonBericht_Click(object sender, RoutedEventArgs e)
{
    // Berichtenvenster oproepen.
    MessageBox.Show("Hallo 1PRO C/D", "Berichtenvenster");
}
```

# Fase 5: Testen

- Op basis van testgegevens
- Kan met hulp-tools, zoals unit testing, e.d.

# Fase 6: Documenteren

- Commentaarregels om (stukken) code te verduidelijken
- Nuttig voor toekomstige aanpassingen
- Noodzakelijk bij werken in ontwikkelteams
- Documentatie/handleiding kan hieruit gegenereerd worden

```
private void ButtonBericht_Click(object sender, RoutedEventArgs e)
{
    // Berichtenvenster oproepen.
    MessageBox.Show("Hallo 1PRO C/D", "Berichtenvenster");
}
```

# Oefeningen 1-2

# C# Essentials Variabelen & operatoren

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- Variabelen vs constanten
- Datatypes
- Naming conventions
- Operatoren
- Numerieke conversies
- String conversies (TryParse)

# Variabelen vs constanten

- Variabele
  - Tijdelijke opslag voor gegevens
  - **Verwijst naar een geheugenlocatie**
  - Heeft een bepaald gegevenstype (datatype)
  - Geen waarde = null reference voor bepaalde datatypes
- Constante
  - Beschrijvende naam voor “variabele”
  - Read-only
  - Waarde verandert niet
- Beide hebben een bepaald bereik (scope)
  - Afhankelijk waar ze gedeclareerd zijn
  - Buiten scope = niet (meer) toegankelijk

# Datatypes

- **int**  
geheel getal
- **float**  
7 digits (32 bit)
- **double**  
15-16 digits (64 bit)
- **decimal**  
28-29 significant digits (128 bit)
- **string**  
tekst
- **bool**  
true or false

# Variabelen

- Declaratie

```
static void Main(string[] args)
{
    int eenGeheelGetal;
    bool a;
}
```

- Initialisatie

```
static void Main(string[] args)
{
    int eenGeheelGetalMetWaardeNul = 0;
    bool a = true;
    float eenKommaGetal = 2.5F;
    double eenAnderKommaGetal = 0.5;
    string eenStukTekst = "tekst";
}
```

# Variabelen

- Welke regel bevat een probleem

```
static void Main(string[] args)
{
    // 1
    int leeftijd = 21;
    // 2
    bool isMan = true;
    // 3
    float totaleUitgaveKlant = 2015.61;
    // 4
    string naam = Johan Jansen;
}
```

# Variabelen

- Een variabele heeft een bepaalde scope of bereik
- Bepaald of variabele bereikbaar is

```
static string bereikbaarInClass;
static void Main(string[] args)
{
    string bereikbaarInMain;
    bereikbaarInMain = "PXL";
    bereikbaarInClass = "Main";
    Console.WriteLine(bereikbaarInMethod);
}
private static void TweedeMethode()
{
    string bereikbaarInTweedeMethode;
    bereikbaarInTweedeMethode = "PXL";
    bereikbaarInClass = "TweedeMethode";
    Console.WriteLine(bereikbaarInTweedeMethode);
}
```

# Numerieke variabelen

- Geheel getal => kleinste mogelijk (uint, ...)
- Decimaal getal => automatisch *double*
- Ander datatype => suffix gebruiken

Type	Suffix	Example
uint	U or u	100U
long	L or l	100L
ulong	UL or ul	100UL
float	F or f	123.45F
decimal	M or m	123.45M

# Alfanumerieke variabelen

- Gewone tekenreeks
  - 0 of meer tekens tussen ""
  - Escape sequences (bijv. \t)
- Verbatim tekenreeks
  - @ prefix
  - Geen escape sequences
  - Alles letterlijk

```
string a = "hallo, C #";  
string b = @"hallo, C #";
```

hallo, C #  
hallo, C #

--- gewone tekenreeks  
--- verbatim tekenreeks

```
string c = "hallo \t C #";  
string d = @"hallo \t C #";
```

hallo \t C #  
hallo \t C #

--- gewone tekenreeks  
--- verbatim tekenreeks

```
string e = "Paul is \"lector\" in klas B";  
string f = @"Paul is """lector"" in klas B";
```

Paul is "lector" in klas B  
Paul is "lector" in klas B

--- gewone tekenreeks  
--- verbatim tekenreeks

```
string g = "\\\server\\share\\file.txt";  
string h = @"\\server\\share\\file.txt";
```

\\\server\\share\\file.Txt  
\server\share\file.Txt

--- gewone tekenreeks  
--- verbatim tekenreeks

```
string i = "Visual\r\nStudio\r\n2017";  
string j = @"Visual  
Studio  
2017";
```

Visual  
Studio  
2017

--- gewone tekenreeks  
--- verbatim tekenreeks  
over meerdere regels

# Naming conventions

- camelCasing
  - Begint met een kleine letter
  - Elk woord in de samenstelling begint met een hoofdletter
  - Gebruikt voor variabelen en functieparameters
  - bijv.: naamCursist, rekeningBank, studentNaamPerAfdeling
- PascalCasing
  - Ook wel UppercamelCasing genoemd
  - Gebruikt voor functienamen, class definities, e.d.

# Operatoren

- Rekenkundige (aritmetische)  
Berekeningen met getallen
- String operatoren  
Bewerkingen met tekst
- Toewijzingsoperatoren  
Nauw samenhangend met rekenkundige
- Relationale  
Vergelijkingen tussen twee variabelen
- Logische  
Berekeningen met booleans

# Rekenkundige operatoren

- Binaire operatoren (werken met 2 operators)

Operator	Beschrijving
$+x, -x$	Teken veranderen
$x+y, x-y$	Som en verschil
$x*y, x/y$	Product en quotiënt
$x\%y$	Rest na deling

- Unaire operatoren (werken met 1 operator)

Operator	Beschrijving
$x++$	Verhoog met 1
$x--$	Verminder met 1

# String operatoren

Operator	Beschrijving
"tekst"+"tekst"	Teksten worden aan elkaar geplakt (ook wel concatenatie genoemd)

# Toewijzingsoperatoren

Operator	Alternatief	Beschrijving
$x=10$	$x=10$	Waarde toekennen
$x+=y$	$x=x+y$	Verhogen met waarde
$x-=y$	$x=x-y$	Verminderen met waarde
$x*=y$	$x=x*y$	Vermenigvuldigen met waarde
$x/=y$	$x=x/y$	Delen door waarde
$x\% =y$	$x=x\%y$	Modulus toekennen

# Relationale operatoren

Operator	Beschrijving
$x == y$	is gelijk aan
$x != y$	is verschillend van
$x < y$	is kleiner dan
$x > y$	is groter dan
$x <= y$	is kleiner dan of gelijk aan
$x >= y$	is groter dan of gelijk aan

# Logische operatoren

- ! (not operator)
  - keert de booleaanse expressie om
  - voorbeeld: `bool expressie = !(getal > 0); // wordt getal <= 0`

Voorwaarde	!
true	false
false	true

# Logische operatoren

- `&&`, `&` (and operator)
  - Aan alle voorwaarden moet voldaan zijn om true te zijn
  - `&&` (meeste gebruikt): stopt met voorwaardes checken als er eentje false is
  - `&` controleert alle voorwaardes, zelfs al is er eentje false
  - voorbeeld: `bool expressie = (getal > 0) && (getal < 10) && (getal != 5);`

Voorwaarde 1	Voorwaarde 2	<code>&amp;&amp;</code> of <code>&amp;</code>
true	true	true
false	true	false
true	false	false
false	false	false

# Logische operatoren

- `||`, `|` (or operator)
  - Aan minstens 1 voorwaarden moet voldaan zijn om true te zijn
  - `||` (meeste gebruikt): stopt met voorwaardes checken als er eentje true is
  - `|` controleert alle voorwaardes, zelfs al is er eentje true
  - voorbeeld: `bool expressie = (getal > 0) || (getal < 10) || (getal != 5);`

Voorwaarde 1	Voorwaarde 2	<code>   of  </code>
true	true	true
false	true	true
true	false	true
false	false	false

# Logische operatoren

- $\wedge$  (xor operator)
  - Aan minstens 1 voorwaarden moet voldaan zijn om true te zijn
  - Is hetzelfde als or ( $\mid\mid$  of  $\mid$ ), MAAR als alle voorwaarden true zijn  $\Rightarrow$  toch false als resultaat. Daarom spreken we exclusive or (xor)
  - voorbeeld: `bool expressie = (getal == 5) ^ (getal == 10);`

Voorwaarde 1	Voorwaarde 2	$\wedge$
true	true	false
false	true	true
true	false	true
false	false	false

# Evaluatie prioriteit

Rekenkundige operatoren

- (negatie)

\*, /, +, -, %

++, -- (voorvoegsel)

<, >, <=, >=, ==, !=

!

&, &&

|, ||

^

=, \*=, /=, +=, -=, %=

++, -- (achtervoegsel)

Vergelijkingsoperatoren

Logische operatoren

Toewijzingsoperatoren

# Numerieke conversies

- Omzetten van datatypes

Verbredend	Versmallend
Gegevensverlies mogelijk	Kunnen (compile)errors geven
Worden automatisch uitgevoerd	Casting om gegevensverlies op te vangen
Ook wel impliciete conversies genoemd	Ook wel expliciete conversies genoemd

```
short shortResult, shortVal = 4;
int integerVal = 67;
long longResult;
float floatVal = 10.5F;
double doubleResult, doubleVal = 99.999;
string stringResult, stringVal = "17";
bool boolVal = true;

doubleResult = floatVal * shortVal; // impliciet: 10.5 * 4 = 42 (berekening in double)
shortResult = (short)floatVal; // expliciete casting: 10
longResult = integerVal + Convert.ToInt64(stringVal); // mix: 67 + 17 = 84
stringResult = Convert.ToString(boolVal) + Convert.ToString(doubleVal); // string: True99.999
```

# Numerieke conversies

- System.Convert klasse
  - Bevat functies om te converteren
  - Enkel voor numerieke conversies
- Gehele getallen
  - short = Convert.ToInt16(val)
  - int = Convert.ToInt32(val)
  - long = Convert.ToInt64(val)
- Kommagetallen
  - float = Convert.ToSingle(val)
  - double = Convert.ToDouble(val)
  - decimal = Convert.ToDecimal(val)

# Formatting van getallen

- `string.format();`

```
int getal1 = 123456;
float getal2 = 123.456F;
Console.WriteLine(string.Format("Geheel getal: {0} en floating point: {1}",getal1, getal2));
```

- `string interpolation`

```
int getal1 = 123456;
float getal2 = 123.456F;
Console.WriteLine($"Geheel getal: {getal1} en floating point: {getal2}");
```

# String conversies

- Parsing wordt gebruikt om een string om te zetten naar een numerieke waarde

```
string tekst = "120";
int i = int.Parse(tekst); // geeft 120
float j = float.Parse(tekst) / 13; // geeft 9,230769
```

Conversie functie	Parse method
Convert.ToDecimal()	decimal.Parse()
Convert.ToDouble()	double.Parse()
Convert.ToInt32()	int.Parse()
Convert.ToInt64()	long.Parse()
Convert.ToSingle()	float.Parse()

# String conversies

- TryParse
  - geeft true als conversie slaagt
  - testen tijdens conversie
  - geen runtime errors

```
string tekst = "5";
int cijfer;
bool resultaat = int.TryParse(tekst, out cijfer); // resultaat = true
//of
bool resultaat = int.TryParse(tekst, out int cijfer); // resultaat = true
```

```
string tekst = "PXL";
int cijfer;
bool resultaat = int.TryParse(tekst, out cijfer); // resultaat = false
//of
bool resultaat = int.TryParse(tekst, out int cijfer); // resultaat = false
```

# String conversies

- Numerieke waarde omzetten naar string
  - Convert.ToString(variabeleNaam);
  - variabeleNaam.ToString();

```
int getal = 10;  
string s1 = Convert.ToString(getal);  
//of  
string s1 = getal.ToString();  
//ToString() heeft betere foutmeldingen en is performanter!
```

# WPF controls

- Window

```
<Window x:Class="_51WpfTryCatchDatum.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:_51WpfTryCatchDatum"
    mc:Ignorable="d"
    Title="Correcte datum controleren...." Height="333.515" Width="536.376" FontSize="18" Loaded="Window_Loaded">
<Window.Background>
    <ImageBrush ImageSource="Afbeeldingdatum.jpg"/>
</Window.Background>
```

Eigenschappen	Name	Naam van formulier
	Title	Tekst in titelbalk
	Width	Breedte (in pixels)
	Height	Hoogte (in pixels)
	Background	Achtergrondkleur
	FontFamily	Lettertype voor alle objecten op het formulier
	WindowState	Venstergrootte (Minimized, Maximized, Normal)
Methods	Close()	Formulier wordt gesloten
Events	Loaded()	Wordt getriggerd wanneer formulier geladen wordt

# WPF controls

- Label

```
<Label Content="Personeelslid" HorizontalAlignment="Left"  
VerticalAlignment="Top" Margin="60,51,0,0" HorizontalContentAlignment="Right"/>
```

Eigenschappen	Name	Naam van label
	Content	Tekst in label
	Background	Achtergrondkleur
	Foreground	Tekstkleur
	FontFamily	Lettertype
	HorizontalAlignment	Plaats van tekst in label
	Visibility	Zichtbaar of niet

# WPF controls

- **Textbox**

```
<TextBox x:Name="TxtResultaat" HorizontalAlignment="Left" Height="200"  
Margin="65,190,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="395"  
IsEnabled="False" Background="#FFEAE5E5" FontFamily="Consolas"/>
```

Eigenschappen	Name	Naam van tekstvak
	Tekst	Inhoud van tekstvak
	FontFamily/FontWeight	Lettertype, grootte, ...
	Background	Achtergrondkleur
	Foreground	Tekstkleur
	.IsEnabled	Beschikbaarheid (kan geklikt worden of niet)
	IsReadOnly	Inhoud kan gewijzigd worden (is altijd aanklikbaar)
	MaxLength	Maximaal aantal karakters
	MaxLine	Maximaal aantal lijnen
	VerticalScrollBarVisibility	Vertikale schuifbalk
	HorizontalScrollBarVisibility	Horizontale schuifbalk
	TextWrapping	Tekstterugloop
	Visibility	Zichtbaarheid

# WPF controls

- Textbox

```
<TextBox x:Name="TxtResultaat" HorizontalAlignment="Left" Height="200"  
Margin="65,190,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="395"  
IsEnabled="False" Background="#FFEAE5E5" FontFamily="Consolas"/>
```

Methods	Focus()	Tekstvak krijgt de focus
	Clear()	Inhoud van tekstvak wordt gewist
Events	Click()	Getriggerd wanneer op tekstvak wordt geklikt
	TextChanged()	Getriggerd wanneer inhoud wordt aangepast
	GotFocus()	Tekstvak krijgt de focus
	LostFocus()	Tekstvak verliest de focus

# WPF controls

- **Button** `<Button Name="btnBerekenen" Content="_Berekenen" HorizontalAlignment="Left" Margin="355,35,0,0" VerticalAlignment="Top" Width="105" Height="35" IsDefault="True" Click="BtnBerekenen_Click"/>`

Eigenschappen	Name	Naam van knop
	Tekst	Tekst op knop
	Width	Breedte in pixels
	Height	Hoogte in pixels
	FontFamily	Lettertype, ...
	FontStyle	Italic, ...
	FontWeight	Bold, ...
	Background	Achtergrondkleur
	Foreground	Tekstkleur
	IsDefault	Standaardknop (reageert op enter)
	.IsEnabled	Beschikbaarheid
	Visibility	Zichtbaarheid

# WPF controls

- **Button** `<Button x:Name="ButtonBerekenen" Content="_Berekenen" HorizontalAlignment="Left" Margin="355,35,0,0" VerticalAlignment="Top" Width="105" Height="35" IsDefault="True" Click="ButtonBerekenen_Click"/>`

Methods	Focus()	Tekstvak krijgt de focus
Events	Click()	Getriggerd wanneer op knop geklikt wordt
	GotFocus()	Knop krijgt de focus
	LostFocus()	Knop verliest de focus
	MouseEnter()	Muis komt over de knop (zonder te klikken)

# WPF controls

- **TextBlock**
  - zoals label gebruik om tekst te tonen
  - verschil met label
    - TextBlock kan geen rand hebben
    - TextBlock kan niet disabled worden
    - TextBlock kan geen afbeelding tonen
    - TextBlock kan geen sneltoets bevatten
    - TextBlock heeft geen ContentTemplate
  - voorkeur om enkel tekst te tonen
  - veel lichter object dan label

# Oefeningen 3-7

# C# Essentials Controle structuren

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- Selectie
- Iteratie

# Selectie

- Selecties worden in programma's gebruikt om keuzes te maken. We onderscheiden:
  - if
  - if else
  - Ternary operator
  - Nesting
  - Switch

# Selectie – if

- if
  - Keuzes maken
  - Als (voorwaarde), dan {code uitvoeren}
  - voorwaarde is een booleaanse expressie (true of false)

```
static void Main(string[] args)
{
    Console.WriteLine("Geef een getal...");
    string tekst = Console.ReadLine();
    int getal = Convert.ToInt32(tekst);
    if (getal == 10)
    {
        Console.WriteLine("Dit getal is gelijk aan 10.");
    }
    getal = 100;
    if (getal == 5)
    {
        // Deze blok wordt niet uitgevoerd! Waarom?
        Console.WriteLine("Dit getal is gelijk aan 5.");
    }
    Console.ReadLine();
}
```

# Selectie – if

- Als er maar 1 statement binnen {} van *if* staat, mag je {} weglaten

```
static void Main(string[] args)
{
    Console.WriteLine("Geef een getal...");
    string tekst = Console.ReadLine();
    int getal = Convert.ToInt32(tekst);
    if (getal == 10)
        Console.WriteLine("Dit getal is gelijk aan 10.");
    getal = 100;
    if (getal == 5) Console.WriteLine("Dit getal is gelijk aan 5.");
    Console.ReadLine();
}
```

# Selectie – if else

- Voer code uit in {} van else indien niet voldaan aan voorwaarde in if

```
static void Main(string[] args)
{
    Console.WriteLine("Geef een getal...");
    string tekst = Console.ReadLine();
    int getal = Convert.ToInt32(tekst);
    if (getal == 10)
    {
        Console.WriteLine("Dit getal is gelijk aan 10.");
    }
    else
    {
        Console.WriteLine("Dit getal is verschillend van 10.");
    }
    Console.ReadLine();
}
```

# Selectie – if else

- Als er maar 1 statement binnen {} van *if* of *else* staat, mag je {} weglaten

```
static void Main(string[] args)
{
    Console.WriteLine("Geef een getal...");
    string tekst = Console.ReadLine();
    int getal = Convert.ToInt32(tekst);
    if (getal == 10)
        Console.WriteLine("Dit getal is gelijk aan 10.");
    else
        Console.WriteLine("Dit getal is verschillend van 10.");
    Console.ReadLine();
}
```

# Selectie – ternary operator

```
static void Main(string[] args)
{
    Console.WriteLine("Geef een getal...");
    string tekst = Console.ReadLine();
    int getal = Convert.ToInt32(tekst);
    string uitvoer = (getal == 10) ? "Dit getal is gelijk aan 10." : "Dit getal is verschillend van 10.";
    Console.WriteLine(uitvoer);
    Console.ReadLine();
}
```

# Selectie – nesting

- Je kan onbeperkt *else if* blokken gebruiken

```
static void Main(string[] args)
{
    double getal = 15.2;
    if (getal < 10.0)
    {
        Console.WriteLine("Dit getal is kleiner dan 10.");
    }
    else if (getal >= 10 && getal < 20)
    {
        Console.WriteLine("Dit getal is groter of gelijk aan 10 en kleiner dan 20.");
    }
    else
    {
        Console.WriteLine("Dit getal groter of gelijk aan 20");
    }
}
```

# Selectie – nesting

- Een *if* blok kan je ook in een andere *if* blok gebruiken.

```
static void Main(string[] args)
{
    double getal = 2.718;
    if (getal < 3)
    {
        if (getal > 2)
            Console.WriteLine("tussen 2 en 3");
        else if (getal > 1)
            Console.WriteLine("tussen 1 en 2");
        else
            Console.WriteLine("<= 1");
    }
}
```

# Selectie – switch

- Test op expliciete gevallen (cases)
- Enkel echte gelijkheden (geen < of >)
- Met *break* spring je uit de switch

# Selectie – switch

```
static void Main(string[] args)
{
    string fruit = Console.ReadLine();
    bool isLekker = false;
    switch (fruit)
    {
        case "peer":
            isLekker = true;
            break;
        case "kers":
            isLekker = true;
            break;
        case "citroen":
        case "pompeimoes":
            isLekker = false;
            break;
        default:
            break;
    }
    string uitvoer = "Fruit is " + (isLekker ? "lekker" : "niet lekker");
    Console.WriteLine(uitvoer);
    Console.ReadLine();
}
```

# Iteratie

- Wat is itereren?
  - Instructies herhalen (opnieuw doen)
  - Verschillende soorten
    - for  
vaste hoeveelheid herhalingen
    - while  
herhalen onder voorwaarde
    - do while  
herhalen onder voorwaarde, maar eerst uitvoeren en dan pas voorwaarde controleren

# Iteratie – for

- Een vaste hoeveelheid herhalingen
  - `for (beginwaarde; voorwaarde; stapgrootte) { //code }`

```
Console.WriteLine("Optellen van 0 t.e.m. 9 in stappen van 1");
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}
Console.WriteLine("Aftellen van 9 t.e.m. 0 in stappen van 1");
for (int i = 9; i >= 0; i--)
{
    Console.WriteLine(i);
}
```

# Iteratie – while

- Herhalen onder voorwaarde
  - while (voorwaarde) { //code }

```
int waarde = 0;
while (waarde < 100)
{
    Console.WriteLine("Waarde {0} is kleiner dan 100!", waarde);
    waarde++;
}
```

# Iteratie – do while

- Herhalen onder voorwaarde
  - Hetzelfde als while, maar de voorwaarde wordt pas op het einde gecontroleerd
  - Minstens 1 herhaling
  - do { //code } while (voorwaarde);
  - Afsluiten met ;

```
int waarde = 1;  
do  
{  
    waarde *= 2;  
    Console.WriteLine(waarde);  
} while (waarde < 100);
```



# Iteratie – nesting

- Lussen in andere lussen steken
- Soms nodig in complexere programma's
- Proberen dit te minimaliseren!

```
Console.WriteLine("Print een 10x10 vierkant:\r\n");
for (int i = 0; i < 10; i++)
{
    for (int j = 0; j < 10; j++)
    {
        Console.Write("* ");
    }
    Console.WriteLine(); // lege regel afdrukken
}
```

# Iteratie – oneindige lus

- Lus die het programma laat vastlopen
- Blijft oneindig doorgaan en stopt niet

```
Console.WriteLine("Tellen van 10 tot oneindig.");
for (int i = 10; i >= 10; i++)
{
    Console.WriteLine(i);
}
```

# Iteratie – oneindige lus

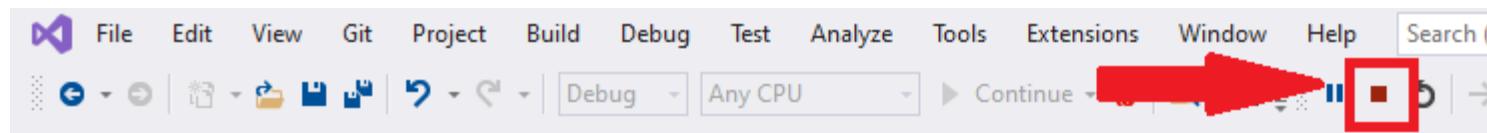
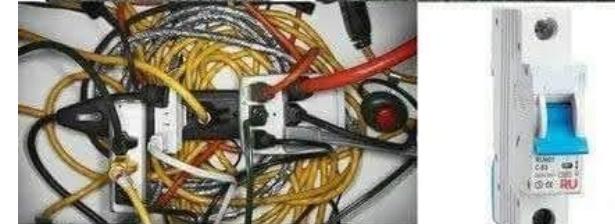
```
int i = 0;
while (true)
{
    Console.WriteLine(i++);
}
Console.WriteLine("Dit raakt nooit afgeprint!");
```

```
static void Main(string[] args)
{
    int i = 0;
    while (true)
    {
        Console.WriteLine(i++);
    }
    Console.WriteLine("Dit raakt nooit afgeprint!");
}
```

 **class System.Console**  
Represents the standard input, output, and error streams for console applications. This class cannot be inherited. To browse the .NET Framework source code for this type, see the Reference Source.  
CS0162: Unreachable code detected  
Show potential fixes (Alt+Enter or Ctrl+;)

# Iteratie – oneindige lus

- Oneindige lus => programma voortijdig afbreken
  - Enkel in console applicatie => CTRL-C drukken
  - In GUI / console applicatie
    - Stop debugging knop
    - Shift-F5 in Visual Studio



# C# Essentials

# Veel gebruikte klassen

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)

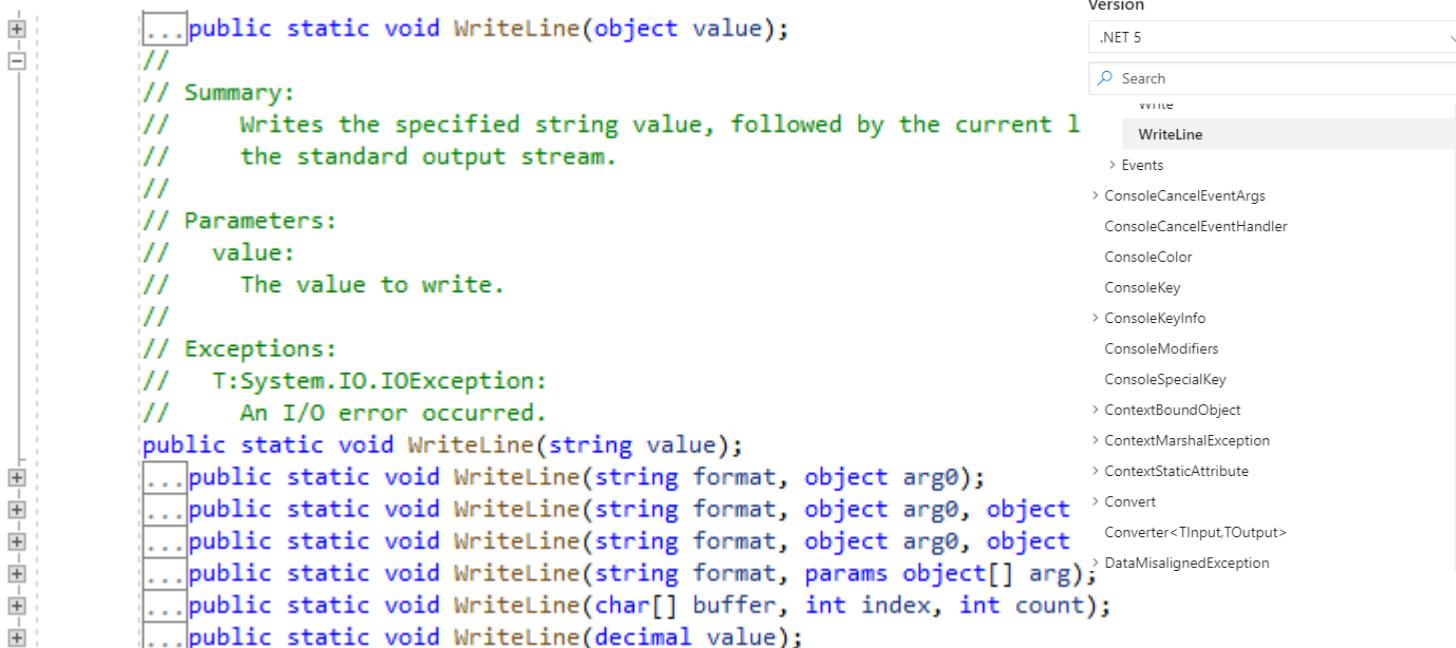




- System.Math
- System.String

# Veelgebruikte klassen

- Zelf documentatie opzoeken tijdens coderen
  - CTRL-klik op de method binnen VS
  - Class opzoeken op <https://docs.microsoft.com>



```
public static void WriteLine(object value);

/// Summary:
///     Writes the specified string value, followed by the current line terminator, to the standard output stream.

/// Parameters:
///     value:
///         The value to write.

/// Exceptions:
///     T:System.IO.IOException:
///         An I/O error occurred.

public static void WriteLine(string value);
... public static void WriteLine(string format, object arg0);
... public static void WriteLine(string format, object arg0, object arg1);
... public static void WriteLine(string format, object arg0, object arg1, object arg2);
... public static void WriteLine(string format, params object[] arg);
... public static void WriteLine(char[] buffer, int index, int count);
... public static void WriteLine(decimal value);
```

Docs / .NET / .NET API browser / System / Console / Methods / WriteLine

Version .NET 5

Search  WriteLine

- > Events
- > ConsoleCancelEventArgs
- > ConsoleCancelEventHandler
- > ConsoleColor
- > ConsoleKey
- > ConsoleKeyInfo
- > ConsoleModifiers
- > ConsoleSpecialKey
- > ContextBoundObject
- > ContextMarshalException
- > ContextStaticAttribute
- > Convert
- > Converter<TInput, TOutput>
- > DataMisalignedException

## Console.WriteLine Method

### Definition

Namespace: System  
Assembly: System.Console.dll

Writes the specified data, followed by the current line terminator, to the standard output stream.

### Overloads

[WriteLine\(String, Object, Object\)](#) Writes the text representation of the specified objects, followed by the standard output stream using the specified format info.

[WriteLine\(String\)](#) Writes the specified string value, followed by the current line terminator, to the standard output stream.

[WriteLine\(Char\[\], Int32, Int32\)](#) Writes the specified subarray of Unicode characters, followed by the standard output stream.

# System.Math (namespace: using System;)

- Constantes
  - Math.PI = 3.14159265358979 (double)
  - Math.E = 2.71828182845905 (double)
- Functies
  - Absolute waarde
    - Math.Abs(-12.34) = 12.34
  - Minimum en maximum
    - Math.Min(17, 35) = 17
    - Math.Max(17,35) = 35
  - Afronden naar dichtstbijzijnde even (!) getal i.g.v. rand gevallen
    - Math.Round(10.5) = 10
    - Math.Round(10.51) = 11 (11 is niet even, maar wel het dichtstbijzijnde getal)
    - Math.Round(11.5) = 12
    - Math.Round(-12.3456,2) = -12,35

# System.Math (namespace: using System;)

- Functies
  - Afronden naar boven
    - Math.Ceiling(-11.5) = -11
    - Math.Ceiling(11.5) = -12
  - Afronden naar beneden
    - Math.Floor(-11.5) = -12
    - Math.Floor(11.5) = 11
  - Machtverheffing
    - Math.Pow(2, 3) =  $2^3 = 2 \cdot 2 \cdot 2 = 8$
  - Logaritme (inverse van machtverheffing)
    - Math.Log(8, 2) = 3 (tot de hoeveelste macht 2 nemen om 8 te krijgen)
  - Vierkantswortel
    - Math.Sqrt(16) = 4

# System.String (namespace: using System;)

- String str = "C Sharp";
- Testen of string leeg is
  - if (str.Length == 0) {}
  - NIET DOEN: if (str == "") {}
- Strings vergelijken
  - if (str.Equals("C Sharp")) {}
  - NIET DOEN: if (str == "C Sharp") {}
- Equals geeft foutmelding als string null is => kan opgevangen worden

# System.String (namespace: using System;)

Functie	Resultaat	
string.concat("pannen", "koek");	"pannenkoek"	String samenvoegen
string.Compare("aa", "AA");	-1	String vergelijken 0 = gelijk -1 = linkse kleiner dan rechtse 1 = linkse groter dan rechtse
string.Compare("aa", "AA", true);	0	true negeert hoofdletters!
string.CompareOrdinal("A", "a");	-32	Vergelijkt ASCII-code (A=65 en a=97)
string.Equals("abcd", "Abcd");	false	Zijn strings gelijk
string.Equals("A", "a", StringComparison.OrdinalIgnoreCase)		Zijn strings gelijk Negeert hoofdletters

# System.String (namespace: using System;)

Method	Resultaat	
"Visual CSharp".Substring(2);	"sual CSharp"	Deel uit string halen vanaf 2 <sup>de</sup> teken tot einde (start vanaf 0!!!)
"Visual CSharp".Substring(2,5);	"sual "	Deel uit string halen vanaf 2 <sup>de</sup> teken, 5 tekens lang (start vanaf 0!!!)
"Visual CSharp".PadLeft(20); "Visual CSharp".PadRight(20);	“ Visual CSharp” “Visual Csharp ”	Vult string aan langs links/rechts met spaties tot 20 chars.
"Visual CSharp".PadLeft(15,'*'); "Visual CSharp".PadRight(15,'*');	“**Visual CSharp” “Visual Csharp**”	Vult string aan langs rechts met * tot 15 chars.
"Visual CSharp".Length;	13	Lengte van string (property)
"Visual CSharp".Contains("CSharp") "Visual CSharp".Contains("C#")	true false	Kijkt of string substring bevat
"Visual CSharp".IndexOf("*"); "Visual CSharp".IndexOf("CSh");	-1 7	Geef positie vanaf 0 waar de substring in de string zit. -1 indien niet gevonden

# System.String (namespace: using System;)

Method	Resultaat	
" Visual CSharp ".Trim();	"Visual CSharp"	Verwijdert whitespaces (spaties, newlines, ...), zowel aan begin als einde
" Visual CSharp ".TrimEnd();	" Visual Csharp"	Enkel aan einde trimmen
" Visual CSharp ".TrimStart();	"Visual CSharp "	Enkel aan begin trimmen
"Visual CSharp".Remove(4);	"Visu"	Verwijder vanaf 4 <sup>de</sup> karakter (0-based!)
"Visual CSharp".Remove(4,3);	"VisuCSharp"	Verwijder 3 karakters vanaf 4 <sup>de</sup> karakter (0-based!)
"Visual C#".StartsWith("Visual");	true	Start de string met een bepaalde substring? (true/false)
"Visual CSharp".Insert(2,"TEST");	"ViTESTsual CSharp"	Voegt string "TEST" toe vanaf het 2 <sup>de</sup> karakter (0-based!)
"Visual CSharp".Replace("a","o");	Visuol Cshorp	Vervangt elke a door o
"Visual CSharp".ToUpper(); "Visual CSharp".ToLower();	VISUAL CSHARP visual csharp	String in hoofdletters String in kleine letters

# C# Essentials Methodes en parameters

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- Anatomie
- By Value/By reference
- Parameters

# Methodes

- Void procedures:
  - Geven geen waarde terug.
  - Vb. Console.WriteLine(), Close() , ...
- Functie procedures:
  - Geven 1 waarde terug van een bepaald datatype.
  - Vb. Round(), int.Parse(), ToString() , ...
- Event procedures:
  - Click(), KeyDown(), MouseEnter() , ...

# Methodes

- Binnen object-georiënteerd programmeren heet een procedure een methode.
- Waarom methodes maken?
  - Om herhaling van statements (code) te voorkomen.
  - De code die in methods staat is herbruikbaar (bijv. libraries).
  - Programma's zijn gemakkelijker te lezen en korter.
  - Programma's zijn gemakkelijker te maken (in groep).
  - Als je methodes public maakt in plaats van private:
    - kan je deze ook in andere files oproepen
    - of in andere projecten gebruiken.
- Wanneer je code dupliceert (copy-paste), probeer dit dan te veralgemenen naar een herbruikbare method.

# Anatomie

- Een methode ziet er als volgt uit:

```
// Void method
private static void EersteMethode()
{
    Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
    return a + 1;
}
```

# Anatomie

- Access modifier (private, public)
- static vs non static
- return type
- naam van de methode
- parameters
- method body

```
// Void method
private static void EersteMethode()
{
    Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
    return a + 1;
}
```

# Access modifier (private, public)

- Toegang bepaalt van waar de methode opgeroepen kan worden

```
// Void method
private static void EersteMethode()
{
    Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
    return a + 1;
}
```

# Static vs non static

- statische methodes kunnen opgeroepen worden zonder object van een klasse
- statische methodes kunnen enkel andere statische methodes oproepen

```
// Void method
private static void EersteMethode()
{
    Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
    return a + 1;
}
```

# Return type

- Bepaalt wat de methode teruggeeft
  - void geeft niets terug

```
// Void method
private static void EersteMethode()
{
    Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
    return a + 1;
}
```

# Naam van de methode

- Elke methode heeft een naam
- Nodig om de methode te kunnen oproepen/gebruiken

```
// Void method
private static void EersteMethode()
{
    Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
    return a + 1;
}
```

# Method parameters

- Parameters zijn argumenten die meegegeven worden aan de methode

```
// Void method
private static void EersteMethode()
{
    Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
    return a + 1;
}
```

# Method body

- De inhoud van de methode

```
// Void method
private static void EersteMethode()
{
    Console.WriteLine(1);
}
// Method met return value
public int TweedeMethode(int a)
{
    return a + 1;
}
```

# By Value

- Enkel de waarde wordt doorgegeven

```
static void Main(string[] args)
{
    int getal = 5;
    TweedeMethode(getal);
    Console.WriteLine($"waarde van integer na methode = {getal}");
}
private static void TweedeMethode(int a)
{
    a = a + 1;
    Console.WriteLine($"waarde van integer binnen methode = {a}");
}
```

```
waarde van integer binnen methode = 6
waarde van integer na methode = 5
Press any key to continue . . .
```

# By Reference

- De geheugenplaats wordt doorgegeven

```
static void Main(string[] args)
{
    int getal = 5;
    TweedeMethode(ref getal);
    Console.WriteLine($"waarde van integer na methode = {getal}");
}
private static void TweedeMethode(ref int a)
{
    a = a + 1;
    Console.WriteLine($"waarde van integer binnen methode = {a}");
}
```

```
waarde van integer binnen methode = 6
waarde van integer na methode = 6
Press any key to continue . . .
```

# Out parameter

- “out” verwijst naar een uitgaande parameter.
- De methode garandeert initialisatie van de “out” parameters

```
static void Main(string[] args)
{
    int getal = 5;
    DerdeMethode(getal, out int c);
    Console.WriteLine(c);
}
private static void DerdeMethode(int a, out int b)
{
    b = a * 10;
    Console.WriteLine("In deze methode wordt a*10 opgeslagen in b.");
}
```

In deze methode wordt a\*10 opgeslagen in b.  
50  
Press any key to continue . . .

# Verschil ref vs out

- ref

- Variabele die je doorgeeft aan de methode is buiten de methode al geïnitialiseerd!

```
int getal = 5;  
TweedeMethode(ref getal);
```

- De methode kan de waarde van de variabele **lezen en aanpassen**.

- out

- Variabele die je doorgeeft aan de methode MOET NIET geïnitialiseerd zijn!

```
DerdeMethode(out int c);
```

- De methode moet zelf een waarde toekennen.

```
public static void DerdeMethode(out int getal)  
{  
    getal = 10; // BELANGRIJK: INITIALISEREN BINNEN DE METHOD!!!  
    getal += 10;  
}
```

# Method overloading

- Methodes doen exact hetzelfde, maar parameters hebben ander datatype

```
private static int PlusMethodInt(int a, int b)
{
    return a + b;
}
private static int PlusMethodDouble(double a, double b)
{
    return a + b;
}
```

# Method overloading

- Beide functies kunnen dezelfde naam gebruiken

```
private static int PlusMethod(int a, int b)
{
    return a + b;
}
private static double PlusMethod(double a, double b)
{
    return a + b;
}
```

- Afhankelijk doorgegeven parameters wordt juiste method opgeroepen

```
int myNum1 = PlusMethod(8, 5);
double myNum2 = PlusMethod(4.3, 6.26);
Console.WriteLine("Int: " + myNum1);
Console.WriteLine("Double: " + myNum2);
```

# Method overloading

- Methodes kunnen dezelfde naam hebben **indien ze andere parameters gebruiken**

```
static void Main(string[] args)
{
    int w = 1, x = 5, y = 10, z = 20;
    Console.WriteLine(Plus(w, x));
    Console.WriteLine(Plus(w, x, y));
    Console.WriteLine(Plus(w, x, y, z));
}
private static int Plus(int a, int b)
{
    return a + b;
}
private static int Plus(int a, int b, int c)
{
    return a + b + c;
}
private static int Plus(int a, int b, int c, int d)
{
    return a + b + c + d;
}
```

# C# Essentials

# Veel gebruikte klassen

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- System.Text.StringBuilder
- System.Random
- System.Windows.MessageBox
- Microsoft.VisualBasic.Interaction

# System.Text.StringBuilder (using System.Text;)

- Stringbuilder is performanter dan string concatenatie!
- Append()
  - Nieuwe tekst toevoegen
- AppendLine()
  - Nieuwe tekst toevoegen en nieuwe regel starten

```
StringBuilder sb = new StringBuilder(); // start van lege string
sb.Append("Nieuwe tekst:\r\n");
sb.AppendLine("Punt:").Append(" 10").AppendLine();
string str = sb.ToString(); // op het einde omvormen naar een string voor gebruik
```

# System.Text.StringBuilder (using System.Text;)

```
StringBuilder sb = new StringBuilder(); // start van lege string
sb.Append("Nieuwe tekst:\r\n");
sb.AppendLine("Punt:").Append(" 10").AppendLine();
Console.WriteLine(sb);
Console.WriteLine();
sb.Insert(0, "StringBuilder:").Insert(14, Environment.NewLine);
sb.Replace(":", "-", 15, sb.Length - 15);
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 2; j++)
    {
        sb.AppendFormat("{0}-{1},", i, j);
    }
    sb.Append("|");
}
Console.WriteLine(sb);
Console.WriteLine();
sb.Remove(0, 14);
Console.WriteLine(sb);
Console.WriteLine();
sb.Clear();
Console.WriteLine(sb);
Console.WriteLine();
```

# System.Random

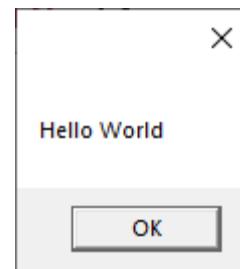
- Willekeurig (random) getal genereren
- Maak eerst instantie aan
- Lege constructor  
telkens verschillende stroom
- Seed meegeven  
in dezelfde stroom

```
Random rnd = new Random();
int getal = rnd.Next(1, 7); //genereer nieuw getal van 1 t.e.m. 6 (!)
double dbl = rnd.NextDouble(); //dbl >= 0.0 en < 1.0
Console.WriteLine(getal);
Console.WriteLine(dbl);
```

# System.Windows.MessageBox

- Berichtvenster om mededeling te tonen
- **NIET in console applicatie, ENKEL in WPF!!!**

```
MessageBox.Show("Hello World");
```



```
MessageBox.Show("Hello World", "Hoofding", MessageBoxButton.YesNo);
```

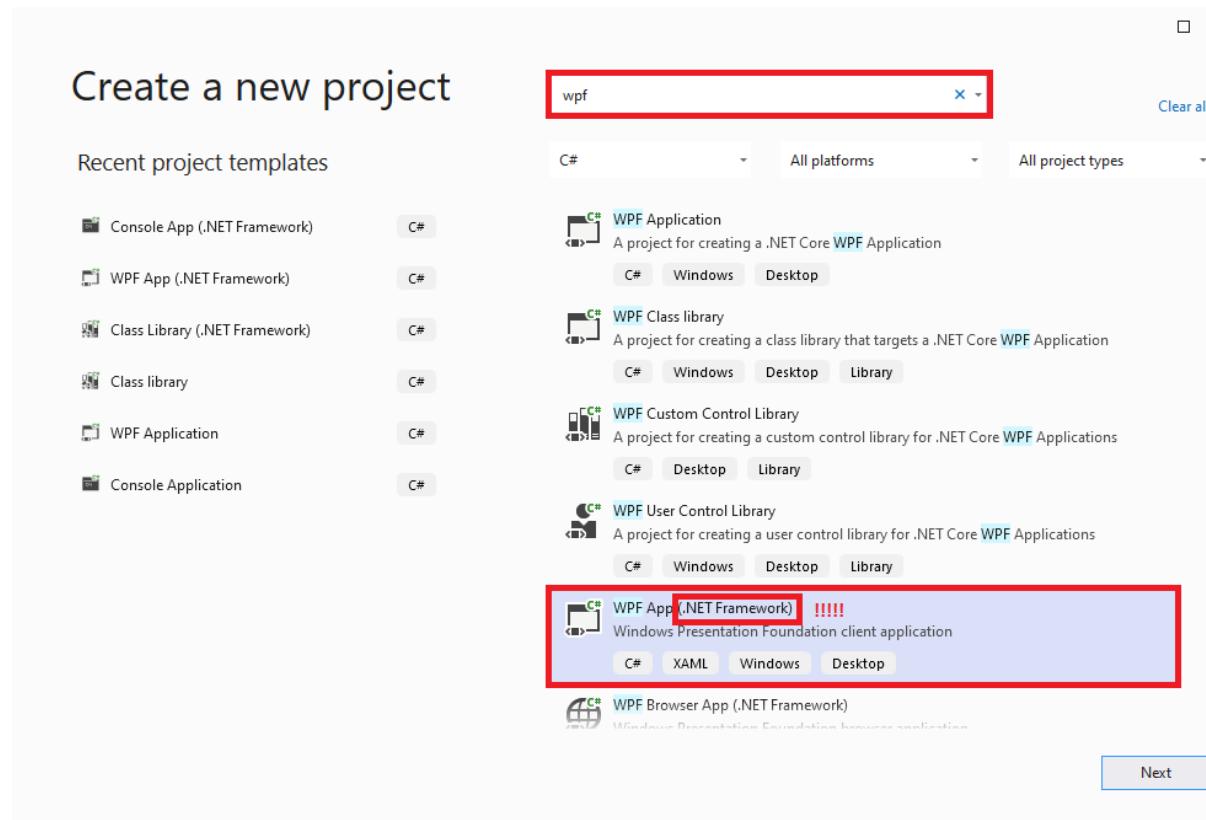
# System.Windows.MessageBox

- Aangeklikt resultaat opvragen

```
MessageBoxResult antwoord = MessageBox.Show("Wil je echt afsluiten?", "Project  
afsluiten", MessageBoxButton.YesNo, mess?MessageBoxImage.Question,  
MessageBoxResult.No);  
  
if (antwoord == MessageBoxResult.Yes)  
{  
    //Project wordt afgesloten  
}  
else  
{  
    //Project wordt niet afgesloten  
}
```

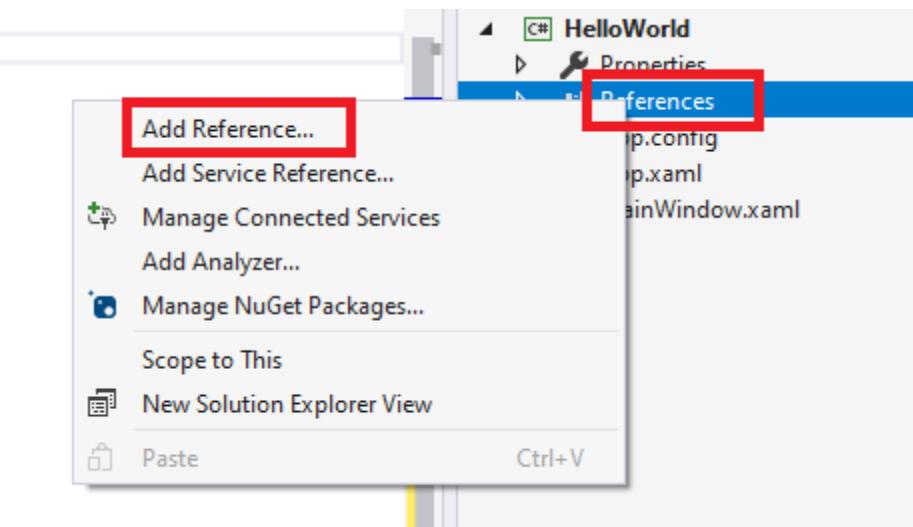
# Microsoft.VisualBasic.Interaction

- Bestaat **niet** in native C#
- Werkt voorlopig nog niet in .NET Core!



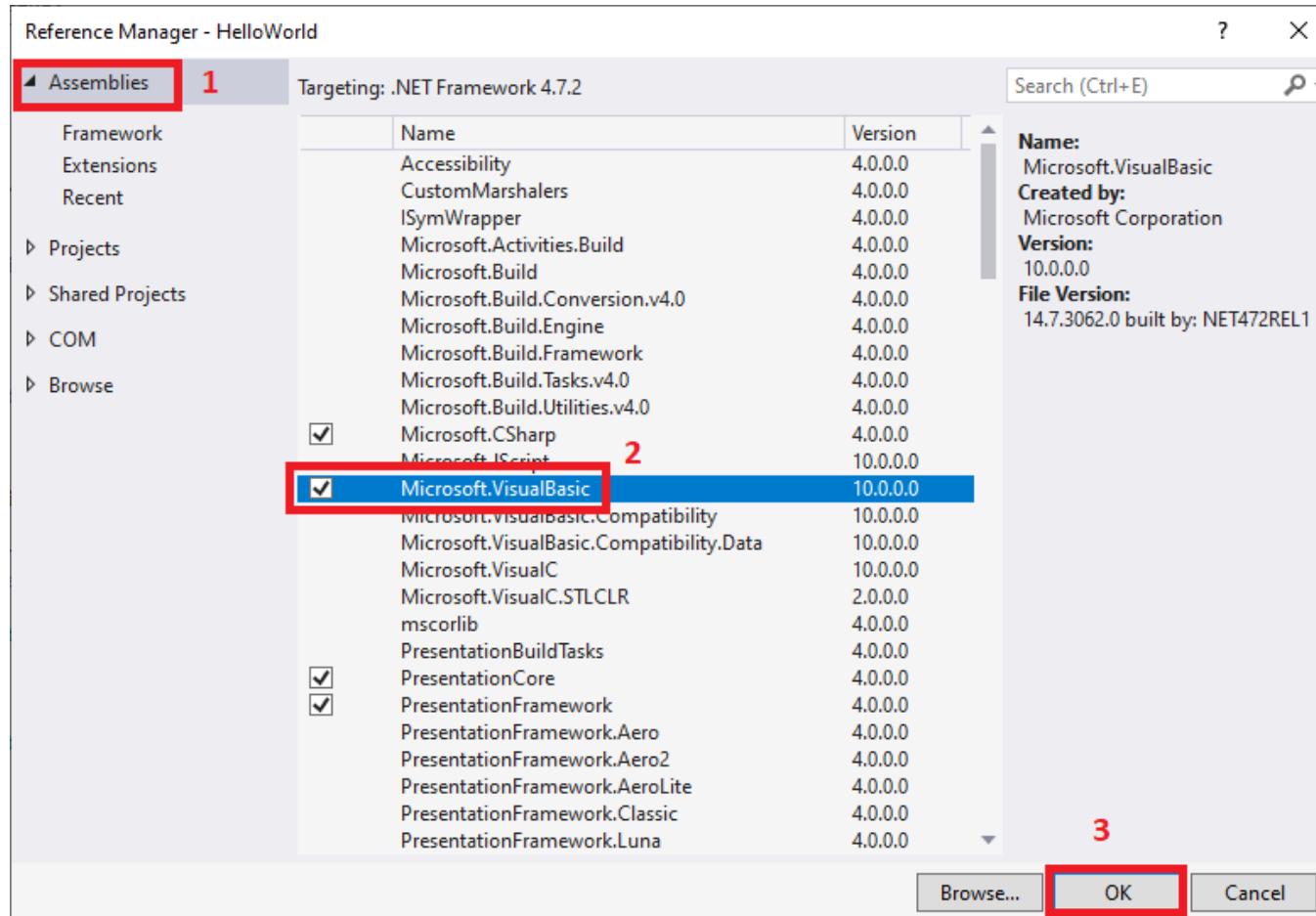
# Microsoft.VisualBasic.Interaction

- Rechtsklik op References => Add Reference...



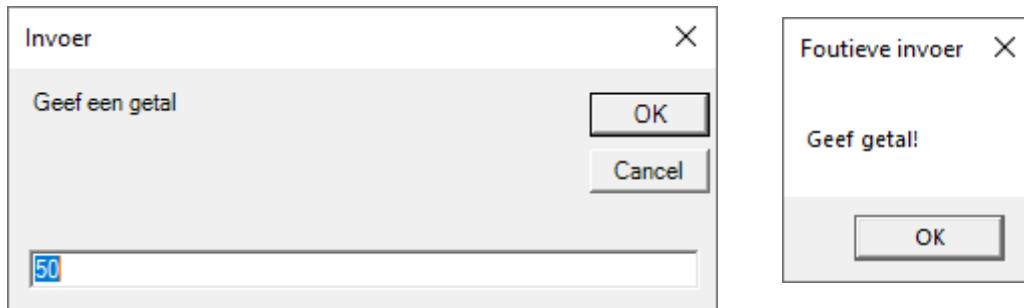
# Microsoft.VisualBasic.Interaction

- Duid bij Assemblies aan: Microsoft.VisualBasic



# Microsoft.VisualBasic.Interaction

```
string antwoord = Interaction.InputBox("Geef een getal", "Invoer", "50", 500);
while (string.IsNullOrEmpty(antwoord))
{
    MessageBox.Show("Geef getal!", "Foutieve invoer");
    antwoord = Interaction.InputBox("Geef een getal", "Invoer", "50", 500);
}
```



# C# Essentials Dispatcher Timer

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- **DateTime**
- **TimeSpan**
- **DispatcherTimer**

# System.DateTime

- DateTime is een datatype voor een datum tussen volgende range
  - 1 januari 0001 en 31 december 9999
  - 00:00:00 en 23:59:59

DateTime.Today	21/10/2020 0:00:00
DateTime.Now	21/10/2020 15:54:10
datum.Date	21/10/2020 15:54:10;5545904
datum.Day	21
datum.DayOfWeek	Wednesday
datum.DayOfYear	297
datum.Hour	15
datum.Minute	54
datum.Month	10
datum.TimeOfDay	15:54:10.5545904
datum.Year	2020

# System.DateTime

- Methods

datum.AddDays(36)  
datum.AddMonths(12)  
datum.AddYears(2)  
datum.Subtract(datum)  
datum.Subtract(datum).Days  
DateTime.Parse (string)  
datum.ToString("yyyy-MM-dd")  
datum.ToShortDateString()  
datum.ToString("HH:mm:ss")  
datum.ToShortTimeString()

voegt dagen toe of trek dagen af  
voegt maanden toe of trek maanden af  
voegt jaren toe of trek jaren af  
geeft verschil in dagen, uren en minuten (geen maanden, jaren)  
geeft verschil in dagen tussen de opgegeven datums  
zet string om naar datum  
geeft lange datum: dddd d mmmm yyyy  
geeft korte datumnotatie: d/mm/yyyy  
geeft lange tijdsnotatie: hh:mm:ss  
geeft korte tijdsnotatie: hh:mm

# System.DateTime

- Voorbeelden

```
DateTime vandaag = DateTime.Today;           // huidige systeemdatum
DateTime datum = new DateTime(2021, 10, 3);    // 3 oktober 2021
DateTime conversieDatum = DateTime.Parse("2020-10-3"); // converteert string naar datum
Console.WriteLine(DateTime.Now);              //3/10/2021 16:32:47
Console.WriteLine(vandaag.AddYears(2));         //3/10/2023 16:32:47
Console.WriteLine(vandaag.TimeOfDay);          //16:32:47.2911001
Console.WriteLine(vandaag.DayOfWeek);          // Saturday
Console.WriteLine((int)vandaag.DayOfWeek);      // 6
Console.WriteLine(vandaag.Subtract(datum).Days); // geeft dagen tussen vandaag en datum
Console.WriteLine(vandaag.AddMonths(12));        //3/10/2022 16:32:47
Console.WriteLine(vandaag.AddDays(-6));         //27/9/2021 16:32:47
Console.WriteLine(vandaag.ToString());          //zaterdag 3 oktober 2021
Console.WriteLine(vandaag.ToString("yyyy-MM-dd")); //3/10/2021
Console.WriteLine(vandaag.ToString("HH:mm:ss")); //16:32:47
Console.WriteLine(vandaag.ToString("mm:ss"));    //16:32
```

# System.TimeSpan

- TimeSpan is een tijdspanne of tijdsinterval

```
//Constructor
TimeSpan interval = new TimeSpan(12, 30, 45); //uren, minuten, seconden

//Definieer twee datums
DateTime date1 = new DateTime(2010, 1, 1, 8, 0, 15);
DateTime date2 = new DateTime(2010, 8, 18, 13, 30, 30);

//Bereken het interval tussen deze twee datums
TimeSpan verschil = date2 - date1;
Console.WriteLine($"{date2} - {date1} = {verschil}");
```

# System.DispatcherTimer

- Wordt gebruikt om op vaste tijdsintervallen taken uit te voeren
  - DispatcherTimer aanmaken
  - Ken taak toe als event
  - Stel interval in
  - Start de DispatcherTimer
- Using System.Windows.Threading

# System.DispatcherTimer

```
public partial class MainWindow : Window
{
    private DispatcherTimer klok = new DispatcherTimer();
    public MainWindow()
    {
        InitializeComponent();

        klok.Tick += new EventHandler(klokAfgelopen); //Event koppelen
        klok.Interval = new TimeSpan(0, 0, 1); //Elke seconde
        klok.Start(); //Timer starten
    }

    private void klokAfgelopen(object sender, EventArgs e)
    {
        LblTijd.Content = $"'{DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss")}'";
    }
}
```

# C# Essentials Event procedures

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





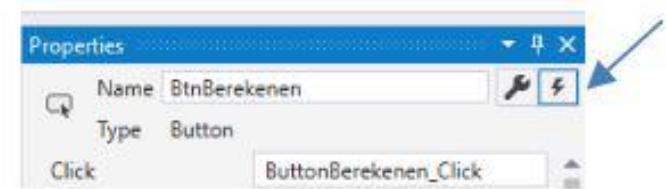
- Event procedures
- Parameters

# Event procedures

- Event aanmaken

Je kan de gebeurtenis (vb. Click) op 3 verschillende manieren aan je object koppelen:

- Dubbel klik op button en je krijgt het standardevent. Vb Click bij Button,TextChanged bij TextBox,...
- Selecteer in het Property-venster de juiste gebeurtenis.
- Typ Click in de XAML-tag van het object (XAML venster) en kies vervolgens op <New Event Handler>



# Event procedures

- Event procedure is een speciale method
- Heeft altijd 2 parameters (sender,e):
  - sender
    - Heeft als type **object**
    - Is een referentie naar het object die het event oproept (triggert)
    - Kan gebruikt worden om eigenschappen van die control aan te passen
    - Kan ook gebruikt worden om te weten “vanwaar we komen”
  - e
    - Geeft eventdata
    - Click event: EventArgs of RoutedEventArgs
    - Key event: KeyEventArgs

# KeyDown event

- KeyDown
  - Wordt getriggerd wanneer je een toets op je toetsenbord **indrukt**
  - KeyEventArgs heeft 2 belangrijke eigenschappen
    - Key geeft het karakter dat ingetypt is
    - Handled geeft aan of de KeyDown wordt afgehandeld

# KeyDown event

```
private void TextBox_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key >= Key.NumPad0 && e.Key <= Key.NumPad9)
    {
        // Test op numeriek toetsenbord.
        TxtResultaat.Text = $"Numeriek {e.Key}";
    }
    else if (e.Key == Key.K)
    {
        TxtResultaat.Text = "Kleine letter k of hoofdletter K";
    }
    else if (Keyboard.IsKeyDown(Key.LeftShift) || Keyboard.IsKeyDown(Key.RightShift))
    {
        TxtResultaat.Text = "Shift";
    }
    else if (e.Key == Key.Return)
    {
        TxtResultaat.Text = "Enter";
    }
    else if (e.Key >= Key.F1 && e.Key <= Key.F12)
    {
        TxtResultaat.Text = $"Functietoets {e.Key}";
    }
    else
    {
        TxtResultaat.Text = $"Key {e.Key} wordt niet ondersteund.";
        e.Handled = true;
    }
}
```

# KeyDown event

- Je kan ook testen op de speciale toetsen ALT, CONTROL en SHIFT d.m.v. Modifiers

```
private void TextBox_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyboardDevice.Modifiers == ModifierKeys.Alt)
    {
        TxtResultaat.Text = "ALT toets";
    }
    if (e.KeyboardDevice.Modifiers == ModifierKeys.Control)
    {
        TxtResultaat.Text = "Control toets";
    }
    if (e.KeyboardDevice.Modifiers == ModifierKeys.Shift)
    {
        TxtResultaat.Text = "Shift toets";
    }
}
```

# KeyDown event

- Toetsaanslagen in combinatie met speciale toetsen ALT, CONTROL en SHIFT d.m.v. Modifiers

```
private void TextBox_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key >= Key.D0 && e.Key <= Key.D9 && e.KeyboardDevice.Modifiers == ModifierKeys.Shift)
    {
        TxtResultaat.Text = $"Numerieke toets {e.Key}";
    }
    if (e.Key == Key.K && e.KeyboardDevice.Modifiers == ModifierKeys.Shift)
    {
        TxtResultaat.Text = "Hoofdletter K";
    }
    if (e.Key == Key.K && e.KeyboardDevice.Modifiers == ModifierKeys.None)
    {
        TxtResultaat.Text = "kleine k";
    }
}
```

# KeyDown event

- Toetsaanslagen met en zonder speciale toetsen ALT, CONTROL en SHIFT d.m.v. Modifiers
- Alt Gr toets

```
private void TextBox_KeyDown(object sender, KeyEventArgs e)
{
    if (e.Key >= Key.D0 && e.Key <= Key.D9 && e.KeyboardDevice.Modifiers == ModifierKeys.Shift)
    {
        TxtResultaat.Text = $"Numerieke toets met shift: {e.Key}";
    }
    if (e.Key >= Key.D0 && e.Key <= Key.D9 && e.KeyboardDevice.Modifiers == ModifierKeys.None)
    {
        TxtResultaat.Text = $"Gewone aanslag: {e.Key}";
    }
    // Alt Gr is een combinatie van Alt + Crtl
    if (e.Key >= Key.D0 && e.Key <= Key.D9 && e.KeyboardDevice.Modifiers == (ModifierKeys.Alt | ModifierKeys.Control))
    {
        TxtResultaat.Text = $"Aanslag met Alt Gr: {e.Key}";
    }
}
```

# KeyUp event

- Wordt getriggerd wanneer je een toets **loslaat**

```
private void TextBox_KeyUp(object sender, KeyEventArgs e)
{
    if (e.Key == Key.T)
    {
        TxtResultaat.Text = $"{e.Key} gelost";
    }
}
```

# Click event

- Wordt getriggerd wanneer je klikt op een object
- De *sender* parameter geeft je het oproepende object

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    Button sndr = (Button)sender;
    TxtResultaat.Text = $"{sndr.Name} is geklikt";
    sndr.Content += " geklikt";
}
```

# C# Essentials WPF controls

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)



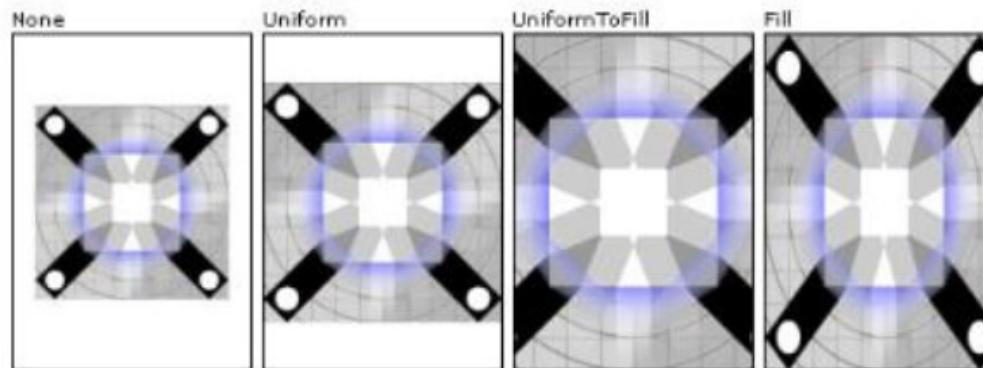


- Image
- CheckBox
- RadioButton

# Image

```
<Image x:Name="ImgDuimen" Source="duimenOmhoog.jpg" Stretch="Fill"/>
```

Eigenschappen	Name	Naam van object
	Width	Breedte in pixels
	Height	Hoogte in pixels
	Source	Afbeelding toekennen
	.IsEnabled	Beschikbaarheid
	Visibility	Zichtbaarheid
	Stretch	Formaat van de afbeelding



# Image

- Om een image op de achtergrond van een knop te zetten moet je in de Designer (XAML) een ImageBrush toekennen aan de Background

```
<Button Name="BtnStart" HorizontalAlignment="Left" Margin="303,157,0,0"  
VerticalAlignment="Top" Width="143" Click="BtnStart_Click" Height="146">  
  <Button.Background>  
    <ImageBrush ImageSource="DuimOmhoog.jpg" Stretch="Uniform"/>  
  </Button.Background>  
</Button>
```

# Image

- Als je over een knop met de muis beweegt, verdwijnt je afbeelding op de knop doordat de content van de knop wordt getoond. Als je in de Designer je Background en je Content met dezelfde afbeelding opgeeft, blijft je afbeelding netjes staan.

```
<Button Name="BtnStart" HorizontalAlignment="Left" Margin="303,157,0,0"  
VerticalAlignment="Top" Width="143" Click="BtnStart_Click" Height="146">  
    <Button.Background>  
        <ImageBrush ImageSource="DuimOmhoog.jpg" Stretch="Uniform"/>  
    </Button.Background>  
    <Button.Content>  
        <Image Source="DuimOmhoog.jpg"/>  
    </Button.Content>  
</Button>
```

# Image

- Via code afbeelding op knop zetten

```
// Afbeelding
Image afb = new Image();
afb.Source = new BitmapImage(new Uri(@“duimOmhoog.jpg”, UriKind.RelativeOrAbsolute));
afb.Stretch = Stretch.Fill;
BtnStart.Content = afb;
```

# CheckBox

- Wordt gebruikt om een ja/nee veld aan te duiden

```
<CheckBox x:Name="ChkWerkzoekend" Content="Werkzoekend"  
HorizontalAlignment="Left" Height="26" Margin="55,150,0,0"  
VerticalAlignment="Top" Width="173" Unchecked="ChkWerkzoekend_Unchecked"  
Checked="ChkWerkzoekend_Checked"/>
```

Eigenschappen	Name	Naam van object
	Width	Breedte in pixels
	Height	Hoogte in pixels
	IsChecked	Is de checkbox aangevinkt of niet
	FlowDirection	Plaats van het vakje (links of rechts)

Events	Checked()	Wordt getriggerd wanneer de checkbox wordt aangevinkt
	UnChecked()	Wordt getriggerd wanneer de checkbox wordt uitgevinkt

# RadioButton

- Alle keuzerondjes op een formulier vormen automatisch een groep
- Van de reeks keuzerondjes wordt 1 keuze geaccepteerd en ze sluiten elkaar wederzijds uit.

```
<RadioButton x:Name="RadJongen" Content="Jongen" IsChecked="True"  
Checked="RadJongen_Checked" Width="243" HorizontalAlignment="Center"  
VerticalAlignment="Center" VerticalContentAlignment="Center"/>
```

Eigenschappen	Name	Naam van object
	Width	Breedte in pixels
	Height	Hoogte in pixels
	.IsChecked	Is de radiobutton aangevinkt of niet
	FlowDirection	Keuzerondje voor of na de tekst plaatsen
Events	Checked()	Wordt getriggerd wanneer de radiobutton wordt aangevinkt
	UnChecked()	Wordt getriggerd wanneer de radiobutton wordt uitgevinkt

# C# Essentials WPF controls

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- Panels

# Panels

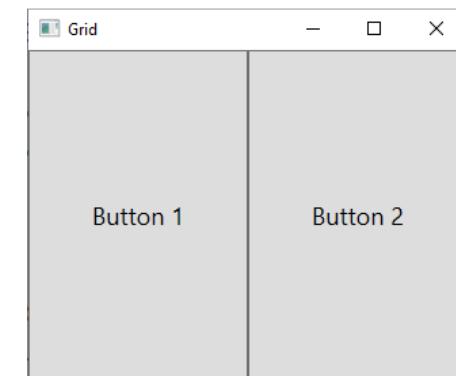
- Eén van de belangrijkste soorten WPF controls
- Fungeren als containers voor andere besturingselementen
- Bepalen de lay-out van window
- Soorten panels:
  - Grid
  - Canvas
  - StackPanel
  - WrapPanel
  - DockPanel
  - Viewbox

# Grid

- Een grid kan meerdere rijen en kolommen bevatten
- Voor de eerste component (vb. Button) moet je niet vertellen welke kolom of rij deze is, maar het mag wel.
  - Grid.Column="0" Grid.Row="0"

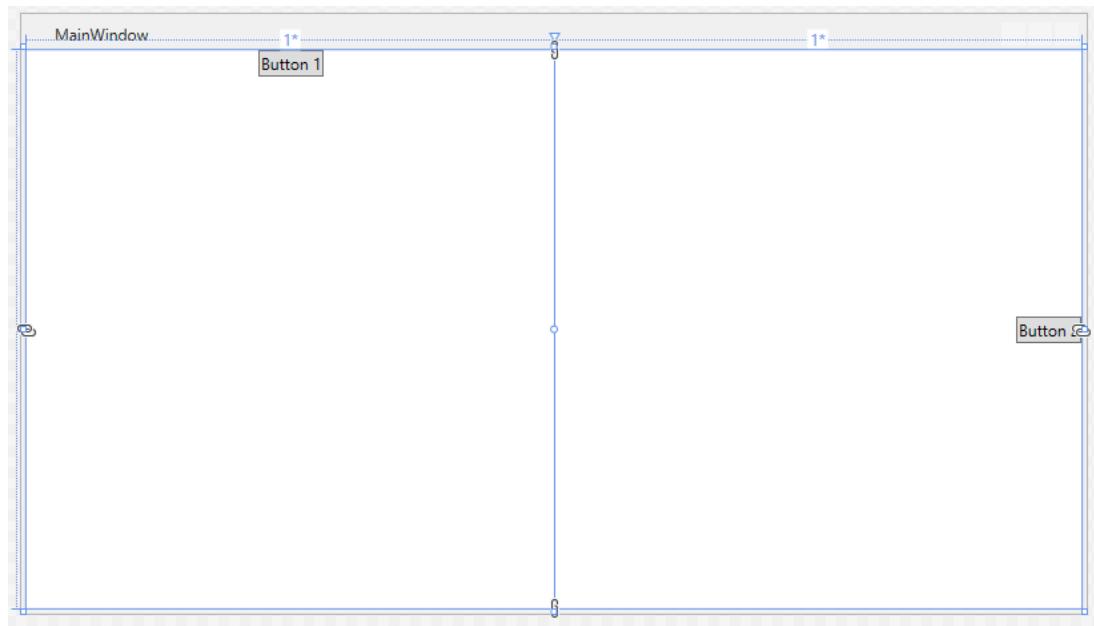
```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Button Content="Button 1"/>
    <Button Grid.Column="1" Content="Button 2"/>
</Grid>
```

Nemen beide evenveel kolommen of breedte in



# Grid

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="*" />
    <ColumnDefinition Width="*" />
  </Grid.ColumnDefinitions>
  <Button VerticalAlignment="Top" HorizontalAlignment="Center">Button 1</Button>
  <Button Grid.Column="1" VerticalAlignment="Center" HorizontalAlignment="Right">
    Button 2
  </Button>
</Grid>
```



# Grid

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="2*" />
    <ColumnDefinition Width="1*" />
    <ColumnDefinition Width="1*" />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="2*" />
    <RowDefinition Height="1*" />
    <RowDefinition Height="1*" />
  </Grid.RowDefinitions>
  <Button>Button 1</Button>
  <Button Grid.Column="1">Button 2</Button>
  <Button Grid.Column="2">Button 3</Button>
  <Button Grid.Row="1">Button 4</Button>
  <Button Grid.Column="1" Grid.Row="1">Button 5</Button>
  <Button Grid.Column="2" Grid.Row="1">Button 6</Button>
  <Button Grid.Row="2">Button 7</Button>
  <Button Grid.Column="1" Grid.Row="2">Button 8</Button>
  <Button Grid.Column="2" Grid.Row="2">Button 9</Button>
</Grid>
```

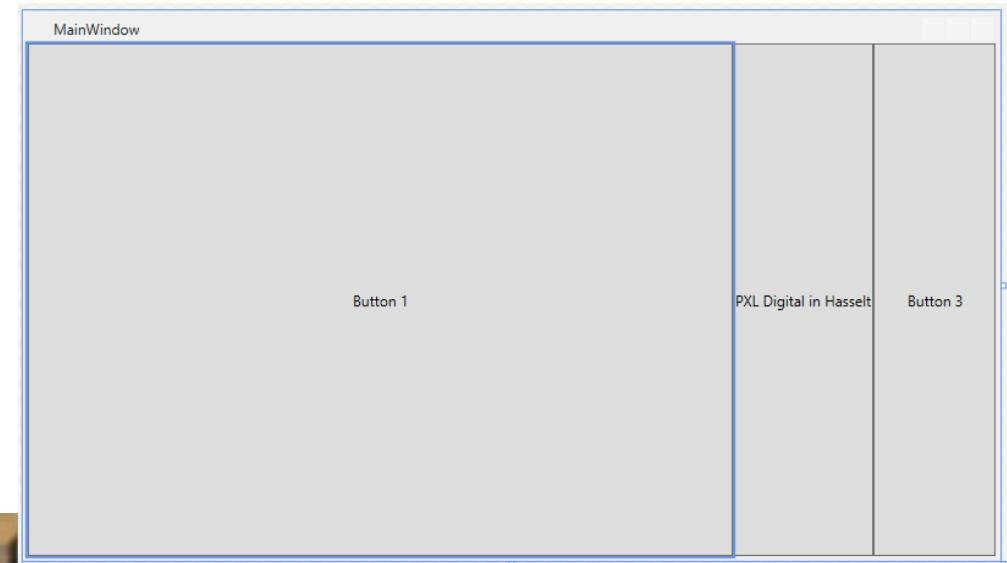
Neemt 2 maal  
zoveel plaats in  
qua kolommen

Neemt 2 maal  
zoveel plaats in  
qua rijen



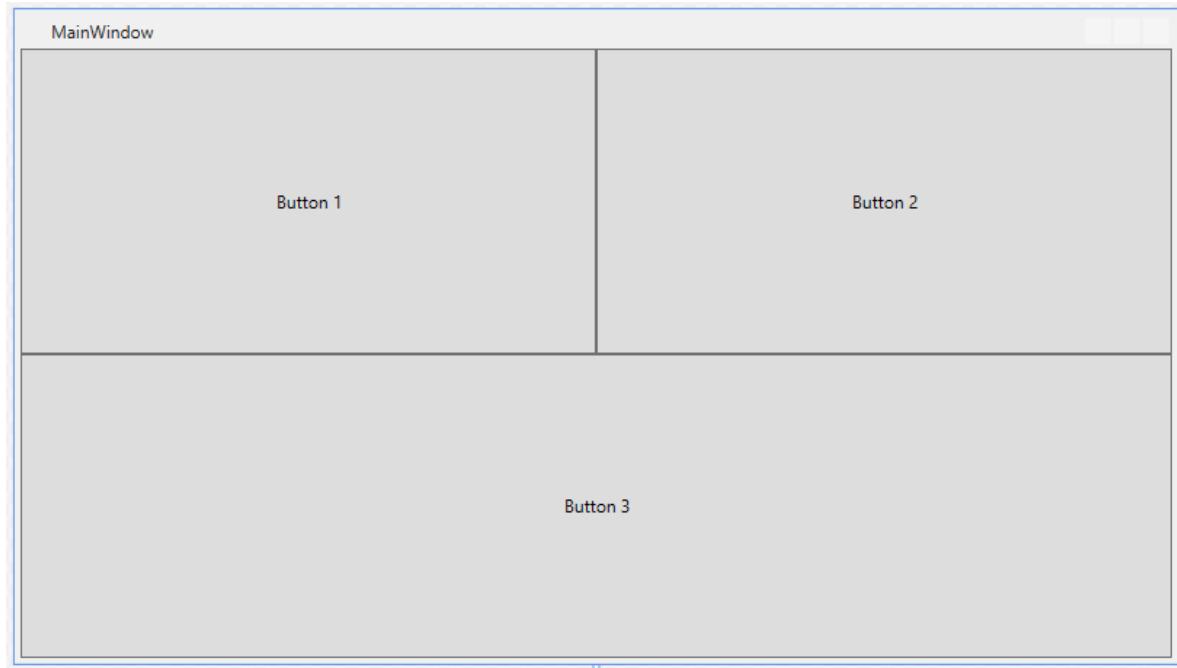
# Grid

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="1*" />
        <ColumnDefinition Width="Auto" />
        <!--Breedte past zich aan de content van button aan-->
        <ColumnDefinition Width="100" />
    </Grid.ColumnDefinitions>
    <Button>Button 1</Button>
    <Button Grid.Column="1">PXL Digital in Hasselt</Button>
    <Button Grid.Column="2">Button 3</Button>
</Grid>
```



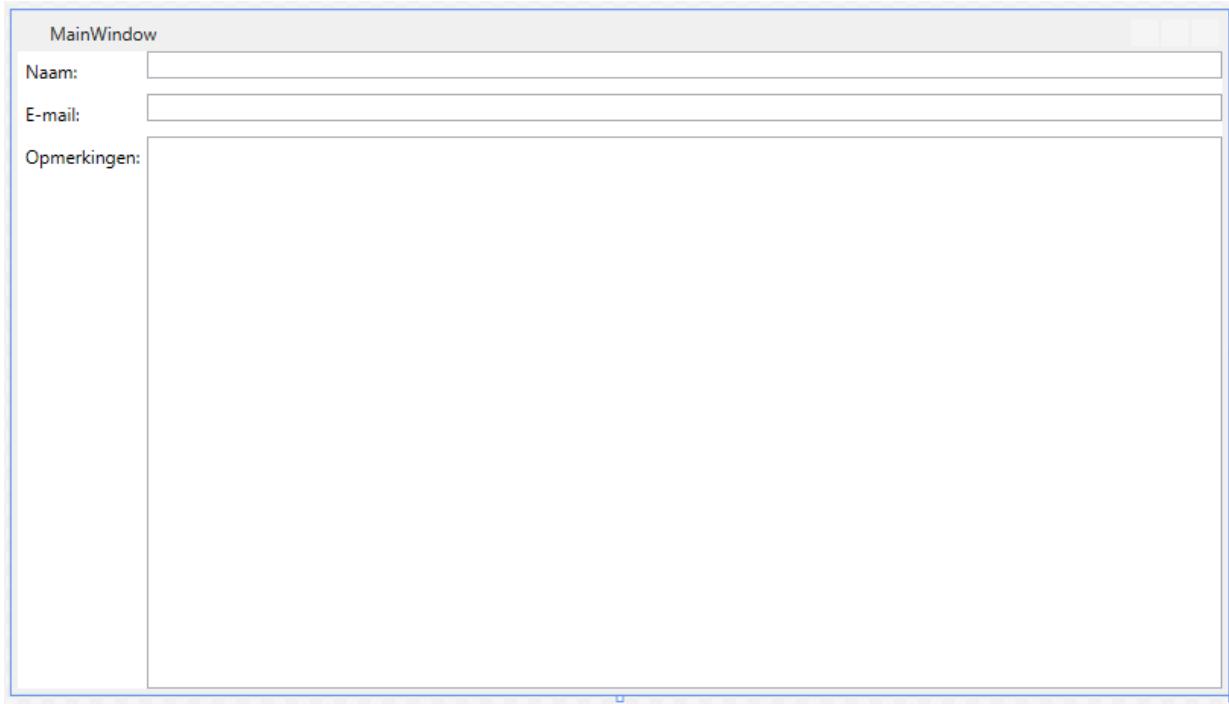
# Grid

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="1*" />
        <ColumnDefinition Width="1*" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <Button>Button 1</Button>
    <Button Grid.Column="1">Button 2</Button>
    <Button Grid.Row="1" Grid.ColumnSpan="2">Button 3</Button>
    <!--Spanwijdte over 2 kolommen-->
</Grid>
```



# Grid

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="*" />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
  </Grid.RowDefinitions>
  <Label>Naam:</Label>
  <TextBox Grid.Column="1" Margin="0,0,0,10" />
  <Label Grid.Row="1">E-mail:</Label>
  <TextBox Grid.Row="1" Grid.Column="1" Margin="0,0,0,10" />
  <Label Grid.Row="2">Opmerkingen:</Label>
  <TextBox Grid.Row="2" Grid.Column="1" AcceptsReturn="True" />
</Grid>
```



# Canvas

- Eenvoudigste paneel.
- Doet niets, groepeert gewoon componenten die er in staan.
  - Relatieve positie t.o.v. canvas via Canvas.Top, Canvas.Left,...
  - Als je canvas verplaatst, verplaats je ook elementen die in canvas zitten

```
<Canvas x:Name="CanTotaal" HorizontalAlignment="Left" Width="400" Height="400" Margin="10,10,0,0" VerticalAlignment="Top">
    <Label Content="Totaal" HorizontalAlignment="Left" Height="35" VerticalAlignment="Top" Width="70"
          HorizontalContentAlignment="Right" Canvas.Left="10" Canvas.Top="10"></Label>
    <TextBox HorizontalAlignment="Left" Height="35" TextWrapping="Wrap" VerticalAlignment="Top" Width="70"
             Canvas.Left="10" Canvas.Top="50"></TextBox>
</Canvas>
```

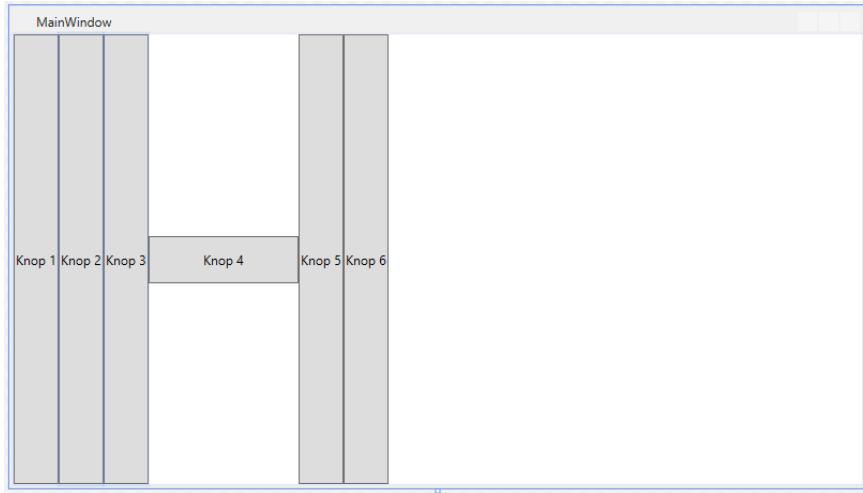
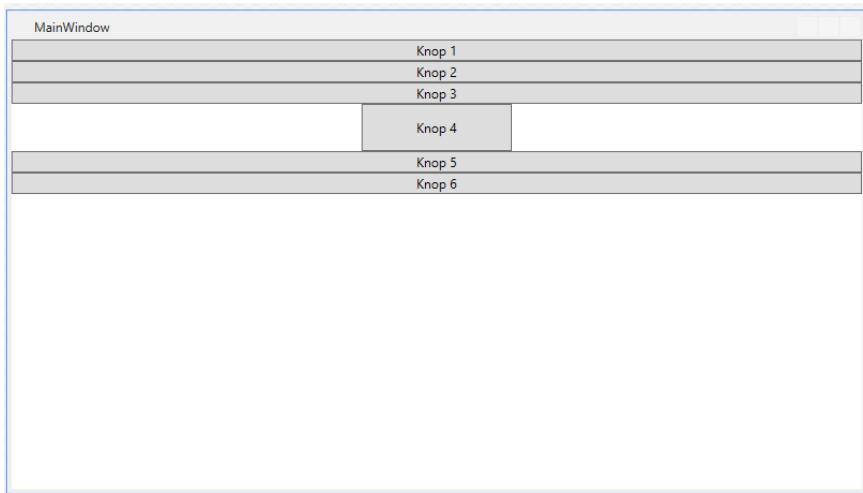
# StackPanel

- Stapelt besturingselementen verticaal of horizontaal:
  - Orientation="Vertical" (standard)
    - Elementen krijgen dezelfde breedte (volgens breedste element).
  - Orientation="Horizontal"
    - Elementen krijgen dezelfde hoogte (volgens hoogste element).
- Als je wil dat besturingselementen volledige breedte of hoogte innemen.

# StackPanel

```
<StackPanel>
    <Button>Knop 1</Button>
    <Button>Knop 2</Button>
    <Button>Knop 3</Button>
    <Button Width="140" Height="44">Knop 4</Button>
    <Button>Knop 5</Button>
    <Button>Knop 6</Button>
</StackPanel>

<StackPanel Orientation="Horizontal">
    <Button>Knop 1</Button>
    <Button>Knop 2</Button>
    <Button>Knop 3</Button>
    <Button Width="140" Height="44">Knop 4</Button>
    <Button>Knop 5</Button>
    <Button>Knop 6</Button>
</StackPanel>
```

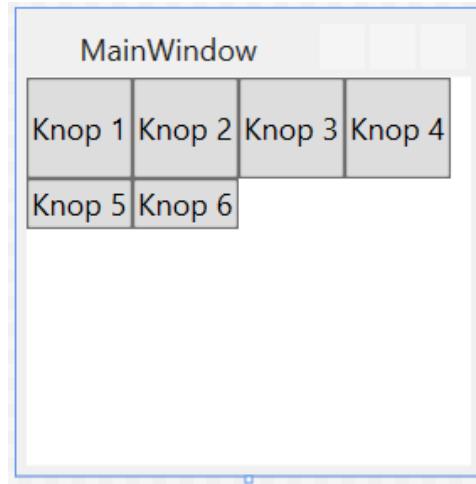


# WrapPanel

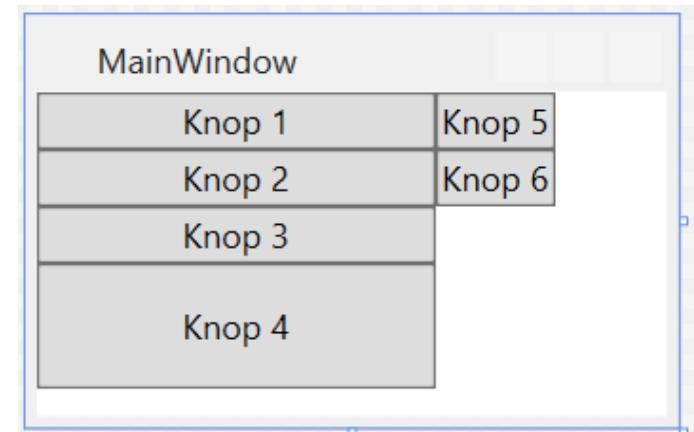
- Plaatst besturingselementen naast of onder elkaar
- Orientation="Horizontal" (standaard)
  - Wanneer rij vol ⇒ nieuwe rij starten
  - Controls krijgen dezelfde hoogte als hoogste element
- Orientation="Vertical"
  - Wanneer kolom vol ⇒ nieuwe kolom starten
  - Controls krijgen dezelfde breedte als breedste element

# WrapPanel

```
<WrapPanel>
    <Button>Knop 1</Button>
    <Button>Knop 2</Button>
    <Button>Knop 3</Button>
    <Button Height="40">Knop 4</Button>
    <Button>Knop 5</Button>
    <Button>Knop 6</Button>
</WrapPanel>
```



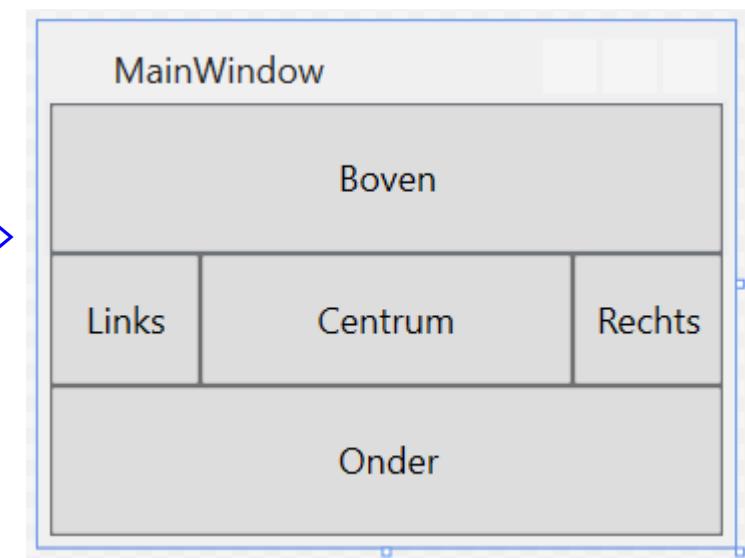
```
<WrapPanel Orientation="Vertical">
    <Button>Knop 1</Button>
    <Button>Knop 2</Button>
    <Button>Knop 3</Button>
    <Button Width="140" Height="44">Knop 4</Button>
    <Button>Knop 5</Button>
    <Button>Knop 6</Button>
</WrapPanel>
```



# DockPanel

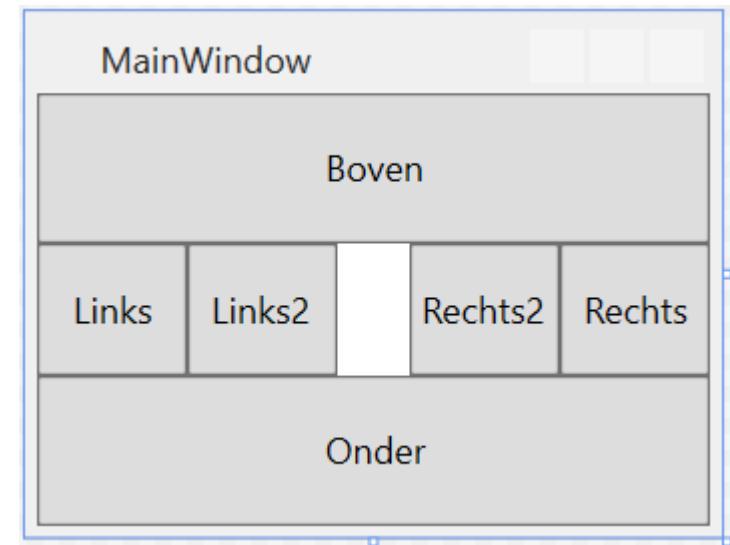
- Element boven, onder, links en rechts van elkaar plaatsen.
- Het eerste element komt standaard links, tenzij anders aangegeven.
- Het laatste element komt volledig gevuld in het midden.
  - Zet LastChildFill op false als je dit niet wil.

```
<DockPanel>
    <Button DockPanel.Dock="Top" Height="50" Content="Boven"/>
    <Button DockPanel.Dock="Bottom" Height="50" Content="Onder"/>
    <Button DockPanel.Dock="Left" Width="50" Content="Links"/>
    <Button DockPanel.Dock="Right" Width="50" Content="Rechts"/>
    <Button Content="Centrum"/>
</DockPanel>
```



# DockPanel

```
<DockPanel LastChildFill="false">
    <Button DockPanel.Dock="Top" Height="50" Content="Boven"/>
    <Button DockPanel.Dock="Bottom" Height="50" Content="Onder"/>
    <Button DockPanel.Dock="Left" Width="50" Content="Links"/>
    <Button DockPanel.Dock="Right" Width="50" Content="Rechts"/>
    <Button DockPanel.Dock="Right" Width="50" Content="Rechts2"/>
    <Button Width="50" Content="Links2"/>
</DockPanel>
```

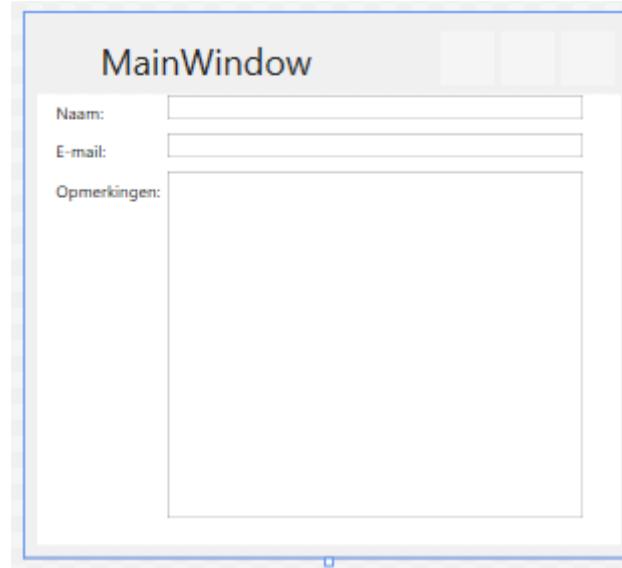


# ViewBox

- Past de inwendige elementen hun grootte aan naargelang wat er beschikbaar is.
- Formaat van inhoud aanpassen en schalen (transformatie).
- Vaak gebruikt voor 2D-afbeeldingen.

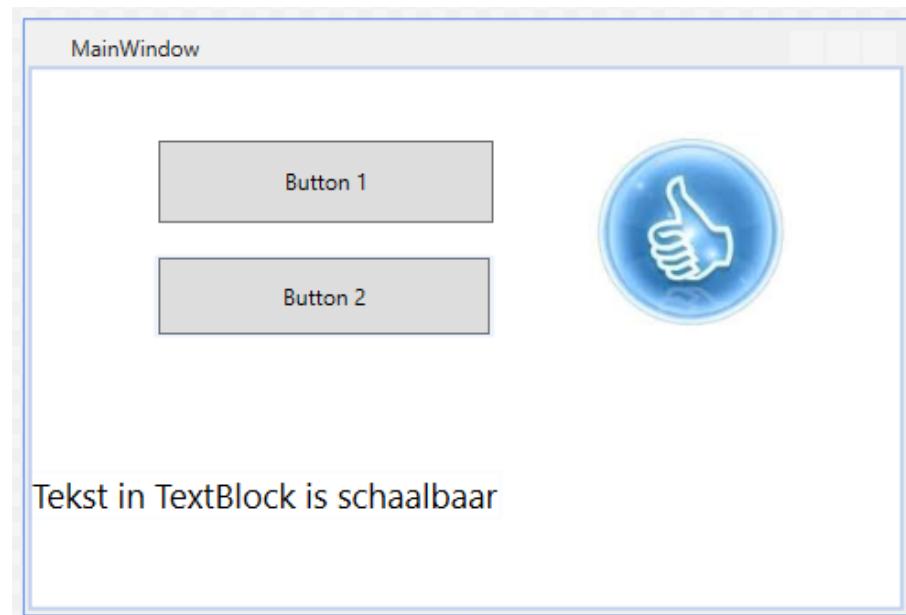
# ViewBox

```
<Viewbox>
  <Grid>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="Auto" />
      <ColumnDefinition Width="*" />
      <ColumnDefinition Width="20" />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto" />
      <RowDefinition Height="Auto" />
      <RowDefinition Height="*" />
      <RowDefinition Height="20" />
    </Grid.RowDefinitions>
    <Label>Naam:</Label>
    <TextBox Grid.Column="1" Margin="0,0,0,10" MinWidth="300"/>
    <Label Grid.Row="1">E-mail:</Label>
    <TextBox Grid.Row="1" Grid.Column="1" Margin="0,0,0,10" />
    <Label Grid.Row="2">Opmerkingen:</Label>
    <TextBox Grid.Row="2" Grid.Column="1" AcceptsReturn="True" MinHeight="250" />
  </Grid>
</Viewbox>
```



# ViewBox

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="98*"/>
        <RowDefinition Height="97*"/>
        <RowDefinition Height="138*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="316*"/>
        <ColumnDefinition Width="271*"/>
    </Grid.ColumnDefinitions>
    <Button Content="Button 1" HorizontalAlignment="Left" Height="45"
        Margin="69,39,0,0" VerticalAlignment="Top" Width="183"/>
    <Button Content="Button 2" HorizontalAlignment="Left" Height="42"
        Margin="69,17,0,0" VerticalAlignment="Top" Width="181" Grid.Row="1"/>
    <Viewbox Margin="16,34,25,27" Grid.Column="1" Grid.RowSpan="2">
        <Image Source="duimOmhoog.jpg"/>
    </Viewbox>
    <Viewbox VerticalAlignment="Center" Grid.Row="2" Grid.RowSpan="2">
        <TextBlock Text="Tekst in TextBlock is schaalbaar" />
    </Viewbox>
</Grid>
```



# C# Essentials WPF controls

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)

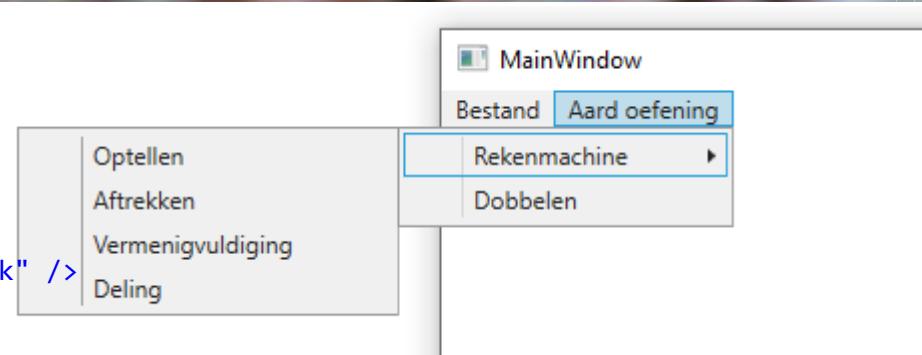




- Menu
- Slider
- Brushes

# Menu

```
<Menu x:Name="Hoofdmenu" HorizontalAlignment="Left" VerticalAlignment="Top" >
    <MenuItem Header="Bestand" >
        <MenuItem x:Name="MnuAfsluiten" Header="Afsluiten" Click="MnuAfsluiten_Click" />
    </MenuItem>
    <MenuItem x:Name="MnuAard" Header="Aard oefening" >
        <MenuItem x:Name="MnuRekenen" Header="Rekenmachine">
            <MenuItem x:Name="MnuOptellen" Header="Optellen" Click="Menu_Click" />
            <MenuItem x:Name="MnuAftrekken" Header="Aftrekken" Click="Menu_Click" IsCheckable="True"/>
            <MenuItem x:Name="MnuVermenigvuldiging" Header="Vermenigvuldiging" Click="Menu_Click" />
            <MenuItem x:Name="MnuDeling" Header="Deling" Click="Menu_Click" />
        </MenuItem>
        <MenuItem x:Name="MnuDobbelen" Header="Dobbelen" IsCheckable="True">
            <MenuItem x:Name="MnuStartDobbelen" Header="Start dobbelspel" Click="Menu_Click" />
        </MenuItem>
    </MenuItem>
</Menu>
```



Eigenschappen	IsCheckable	Vinkje blijft staan (geen submenu!)
	ToolTip	Extra info (tooltip) bij menu item
	.IsEnabled	Bruikbaar of niet
	VerticalAlignment	Verticale positie van menu in windows/grid
	HorizontalAlignment	Horizontale positie van menu in windows/grid
Events	Click	Treedt op als menu item wordt geklikt

# Slider

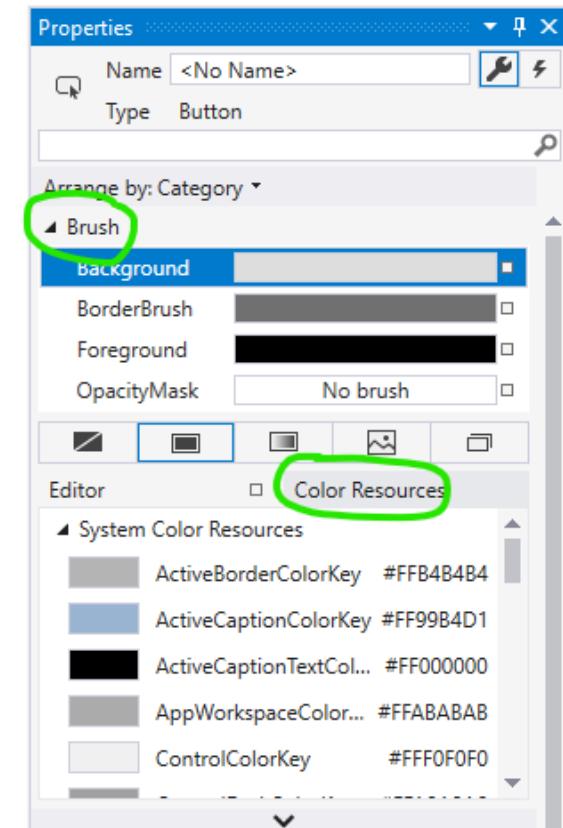
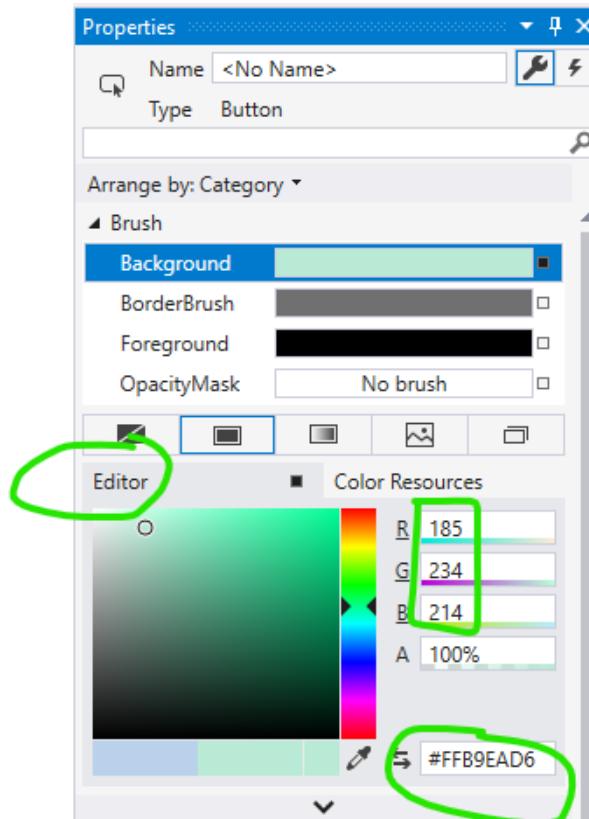


```
<Slider x:Name="Schuifregelaar" Height="35" Width="395"  
Margin="85,115,0,0" TickPlacement="BottomRight"  
AutoToolTipPlacement="TopLeft" Maximum="100000" TickFrequency="10000"  
ValueChanged="Schuifregelaar_ValueChanged" IsSnapToTickEnabled="False"  
SmallChange="100" LargeChange="5000"/>
```

Eigenschappen	SmallChange	Kleine wijziging (bij gebruik pijltjes)
	LargeChange	Grote wijziging (bij klik links en rechts van slider)
	Minimum	Minimum waarde
	Maximum	Maximum waarde
	Orientation	Horizontale of verticale positie
	TickFrequency	Per eenheid het aantal streepjes op balk
	TickPlacement	Plaatsing van de streepjes
	AutoToolTipPlacement	Plaats waar de waarde wordt getoond
	IsSnapToTickEnabled	Enkel waarde op tick gebruiken
Events	ValueChanged	Treedt op als de waarde wijzigt

# Brushes

```
<Button Content="Button" HorizontalAlignment="Left" Width="75" Click="Button_Click">  
    <Button.Background>  
        <SolidColorBrush Color="{DynamicResource {x:Static SystemColors.GradientActiveCaptionColorKey}}"/>  
    </Button.Background>  
</Button>
```



# Brushes

- Kleuren in code veranderen via Brushes
  - Brushes.KleurNaam

```
BtnTest.Background = Brushes.DarkBlue;
```

- Zelf hexadecimale kleurencodes (ARGB) gebruiken ipv. ingebouwde namen

```
// Kleurenpallet met hexadecimale waarde gebruiken.  
var bc = new BrushConverter();  
BtnTest.Background = (Brush)bc.ConvertFrom("#FFE8E6E6");
```

# C# Essentials Foutopsporing

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- Foutopsporing
- Foutafhandeling
- Exceptions
- Debugging

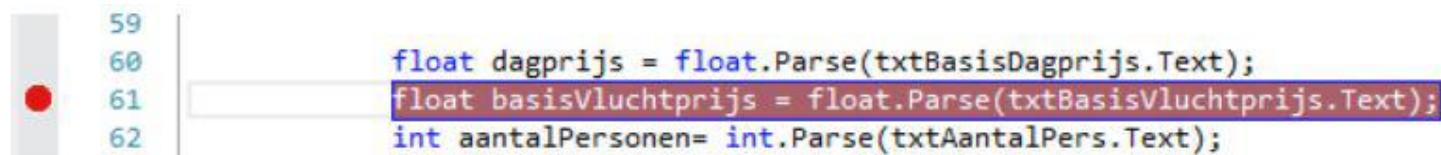
# Foutopsporing

Verschillende soorten fouten

- **Syntaxfouten:**
  - Fouten tegen de taalregels (grammatica) van programmeertaal.
  - Zie je aan rode gekleurde golfjes tijdens programmeren.
- **Logische fouten/bugs:**
  - Gebruik van foute logica in je programma. Vb. foute indexwaarde
  - Kan je nagaan door te debuggen met breakpoints.
- **Runtime-fouten:**
  - Crashen van je programma terwijl het runt.
  - Opvangen via gestructureerde foutafhandeling (try-catch).

# Debugging: fouten opsporen

- Breakpoints
  - Onderbrekingspunten aanduiden op een bepaalde regel in je code
  - Om code te pauzeren op die regel
  - Kijken welke waarden de variabelen hebben op dat moment
- Breakpoints instellen
  - Klik in Visual Studio op de grijze blak vóór het regelnummer
  - OF: klik op een regel code, ga naar Debug > Toggle Breakpoint of druk F9

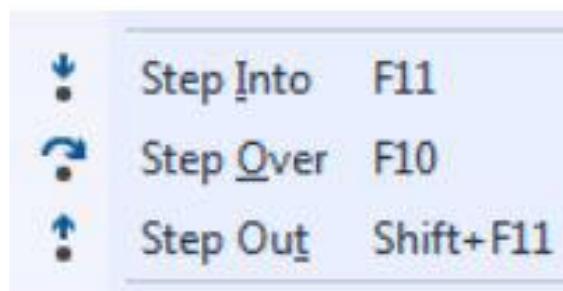


A screenshot of the Visual Studio code editor. On the left, there's a vertical grey bar with a red circular breakpoint icon at the top of line 61. To the right of the bar, the line numbers 59, 60, 61, and 62 are visible. The code itself consists of three lines:  
59: float dagprijs = float.Parse(txtBasisDagprijs.Text);  
60: float basisVluchtprijs = float.Parse(txtBasisVluchtprijs.Text);  
61: int aantalPersonen= int.Parse(txtAantalPers.Text);  
62:

# Debugging: code doorlopen

Programma doorlopen na breakpoint in te stellen

- Bovenaan Visual Studio is een balkje verschenen
- Step Into (F11)  
spring in de method waar je nu zit
- Step Over (F10)  
ga 1 regel verder in code
- Step Out (Shift+F11)  
spring uit de method waar je nu zit
- Run To Cursor (Ctrl+F10)  
stuk code uitvoeren tot waar cursor is



# Debugging: waardes controleren

- Watch venster (wanneer je aan het debuggen bent ga naar: **Debug > Windows > Watch > Watch 1**)
- Is het meest gebruikte debug venster!
- Typ eerst variabele namen in om deze te gaan inspecteren
- Waardes en eigenschappen van lokale variabelen bekijken
- Rood geeft de laatste wijziging aan

Name	Value	Type
vluchtprijs	640.0	float
dagprijs	60.0	float
aantalDagen	7	int
verblijfsprijs	1176.0	float
aantalPersonen	4	int
teBetalen	0.0	float

# Debugging: waardes controleren

- **Autos** venster (**Debug > Windows > Autos**)
- Toont de waardes en eigenschappen van het huidig en vorige statement
- Rood geeft de laatste wijziging aan in de huidige method

Name	Value	Type
korting	90.8	float
reisprijs	1816.0	float
teBetalen	0.0	float
this	{reiskost2.frmReiskost2, Text: }	reiskost2.frmReiskost2
txtKortingspercentage	{System.Windows.Forms.TextBox, Text: 5}	System.Windows.Forms.TextBox

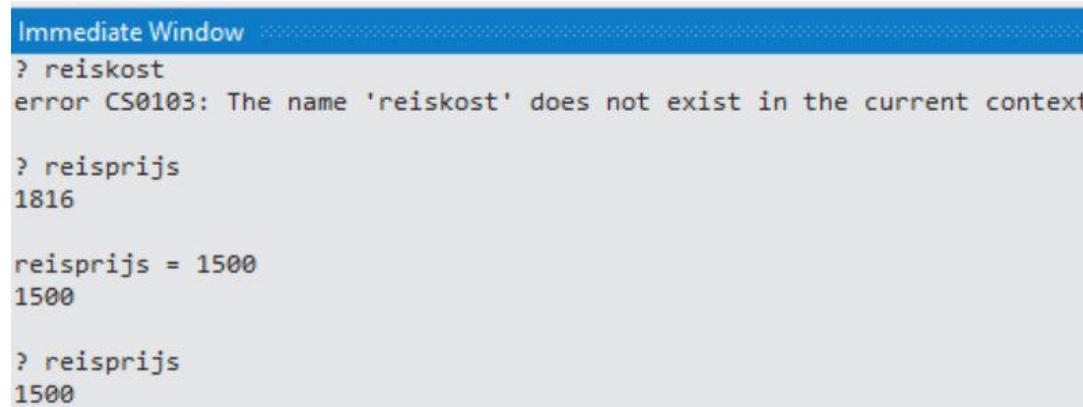
# Debugging: waardes controleren

- **Locals** venster (**Debug > Windows > Locals**)
- Toont alle lokale variabelen binnen de huidige method tijdens uitvoering van programma

Name	Value	Type
▶ <input checked="" type="checkbox"/> this	{WpfReiskost2.MainWindow}	WpfReiskost2.MainWindow
▶ <input checked="" type="checkbox"/> sender	{System.Windows.Controls.Button: Berekenen}	object {System.Windows.Controls.Button}
▶ <input checked="" type="checkbox"/> e	{System.Windows.RoutedEventArgs}	System.Windows.RoutedEventArgs
▶ <input checked="" type="checkbox"/> vluchtprijs	640	float
▶ <input checked="" type="checkbox"/> verblijfsprijs	1176	float
▶ <input checked="" type="checkbox"/> reisprijs	1816	float
▶ <input checked="" type="checkbox"/> teBetalen	1725.2	float
▶ <input checked="" type="checkbox"/> testBasis	true	bool
▶ <input checked="" type="checkbox"/> dagprijs	60	float
▶ <input checked="" type="checkbox"/> testVlucht	true	bool
▶ <input checked="" type="checkbox"/> basisVluchtprijs	200	float
▶ <input checked="" type="checkbox"/> testPersonen	true	bool
▶ <input checked="" type="checkbox"/> aantalPersonen	4	int
▶ <input checked="" type="checkbox"/> testDagen	true	bool
▶ <input checked="" type="checkbox"/> aantalDagen	7	int
▶ <input checked="" type="checkbox"/> testKorting	true	bool
▶ <input checked="" type="checkbox"/> korting	90.8	float

# Debugging: waardes controleren

- **Immediate Window (Debug > Windows > Immediate Window)**
- Waarden van variabelen of expressies tijdens het runnen aanpassen



```
Immediate Window
? reiskost
error CS0103: The name 'reiskost' does not exist in the current context

? reisprijs
1816

reisprijs = 1500
1500

? reisprijs
1500
```

# Debugging: waardes controleren

- Hover effect om waarde te tonen

```
0 references
static void Main(string[] args)
{
    string text = "test";
    text = text + text;
    Console.WriteLine(text);
```

# Debugging: fouten afhandelen

- Runtime-fouten
  - Crashen van je programma terwijl het runt
    - Software of hardware probleem
  - Kunnen gegevens doen verloren gaan in bestanden
  - Kunnen fouten in bestanden veroorzaken ⇒ beschadigd bestand
- Runtime-fouten afhandelen
  - Fouten opvangen via gestructureerde foutafhandeling (try-catch)
  - Runtime-fouten behandel je als exceptions

# Debugging: fouten afhandelen

- Algemene vorm van foutafhandeling

```
try
{
    'Beschermd code
    Statements die een runtimefout veroorzaken
}
catch
{
    'ErrorHandler of exceptionhandler
    Statements die uitgevoerd worden bij een runtimefout
}
finally
{
    Optionele statements die altijd uitgevoerd worden
}
```

# Debugging: fouten afhandelen

- Voorbeeld van foutafhandeling

```
try
{
    string nr = "vijf";
    int intnr = int.Parse(nr); // runtime fout: Input string was not in a correct format.
}
catch (Exception ex)
{
    // fout afhandelen en programma niet laten crashen
    Console.WriteLine("Dit is een fout!"); // een eigen melding
    Console.WriteLine(ex.Message); // toon de echte foutmelding die het systeem geeft
}
finally
{
    Console.WriteLine("Deze code wordt ALTIJD uitgevoerd na try indien geen fout of na catch
indien wel een fout");
}
```

# Debugging: fouten afhandelen

- try ... catch ... finally
- Try blok
  - Hierin komt je gewone code waarin mogelijks een fout zit.
  - Je probeert (try) die code uit.
  - Als er een fout optreedt belanden we in een catch blok.
- Catch blok
  - Hierin komt code die optreedt wanneer een fout voorkomt.
  - ER KUNNEN MEERDERE CATCH BLOKKEN ZIJN!
- Finally blok
  - Deze code wordt altijd uitgevoerd.
  - Meestal voor opruimcode (bestanden sluiten, databaseconnectie sluiten)

# Debugging: fouten afhandelen

- Voorbeeld meerdere catch blokken

```
int deeltal, deler, quotient;
try
{
    deeltal = Convert.ToInt32(tbGetal1.Text);
    deler = Convert.ToInt32(tbGetal2.Text);
    quotient = deeltal / deler;
    string quotientText = quotient.ToString();
}
catch (DivideByZeroException) // deling door 0
{
    LblInfo.Content = "Delen door 0 mag niet!";
}
catch (FormatException) // een getal is k of 1.6 bijvoorbeeld => een conversiefout
{
    LblInfo.Content = "Je moet 2 gehele getallen ingeven!";
}
catch (Exception ex) // andere fout
{
    LblInfo.Content = ex.Message; // druk foutmelding af gegenereerd door het systeem
}
```

# Debugging: fouten afhandelen

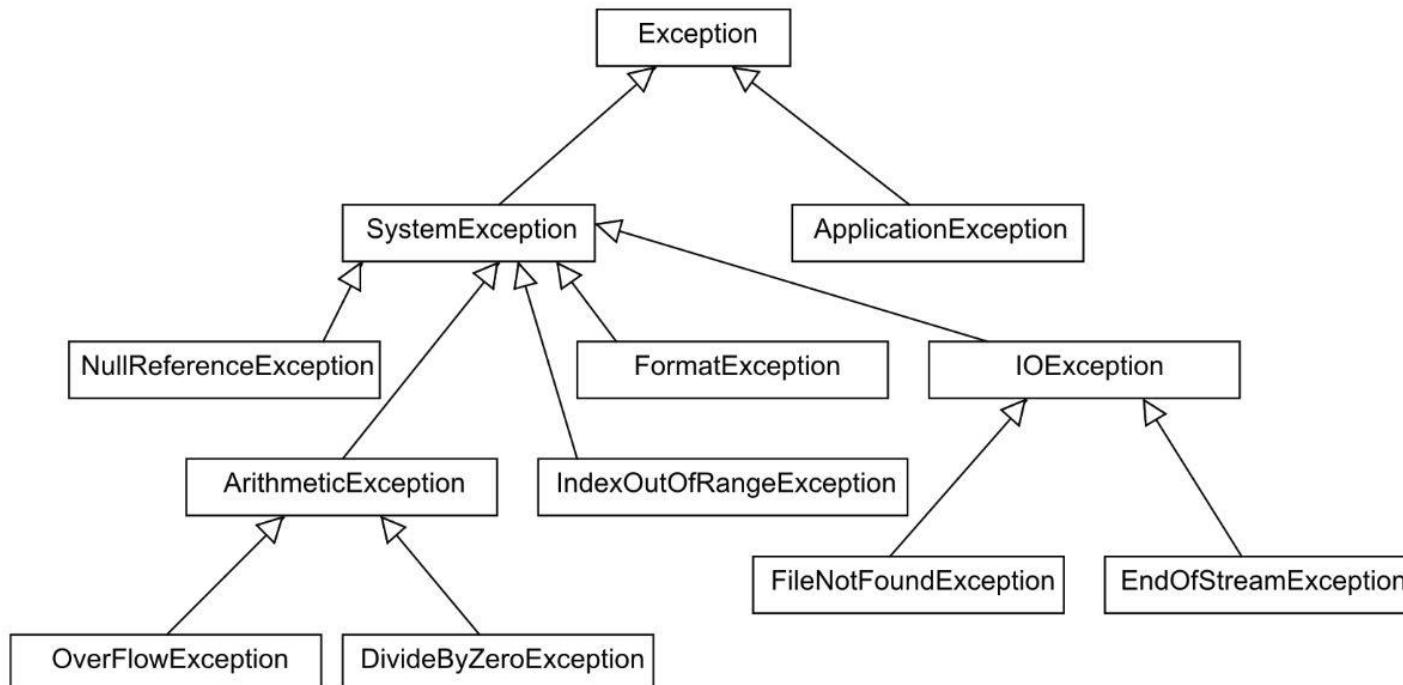
Meerdere catch blokken

- Catch blokken één voor één testen.
- Catch blok met overeenkomstig type exception wordt uitgevoerd!
- Als een exception niet gelijk is aan 1 van deze types, wordt een catch-blok zonder type exception doorlopen.

```
catch // andere fout
{
    //.....
}
```

# Debugging: fouten afhandelen

- Algemener exceptions staan bovenaan hierarchie, specifieker beneden
- Eerst catch-blokken met specifieke fout en daarna pas algemener
- Anders wordt de fout als een algemene exception opgevangen!!!



# Debugging: fouten afhandelen

- Geneste try catch blokken

```
int deeltal, deler, quotient;
try
{
    deeltal = Convert.ToInt32(tbGetal1.Text);
    deler = Convert.ToInt32(tbGetal2.Text);
    for (int i = 3; i >= 0; i--)
    {
        try
        {
            deler /= i;
        }
        catch (DivideByZeroException)
        {
            LblInfo.Content = "Delen door 0 mag niet! Binnenste afhandeling.";
            deler = Convert.ToInt32(tbGetal2.Text);
        }
    }
    quotient = deeltal / deler;
    string quotientText = quotient.ToString();
}
catch (DivideByZeroException) // deling door 0
{
    LblInfo.Content = "Delen door 0 mag niet!";
}
catch (FormatException) // een getal is k of 1.6 bijvoorbeeld => een conversiefout
{
    LblInfo.Content = "Je moet 2 gehele getallen ingeven!";
}
catch (Exception ex) // andere fout
{
    LblInfo.Content = ex.Message; // druk foutmelding af gegenereerd door het systeem
}
```

# Debugging: exceptions throwen

- Zelf exceptions gooien (Zelf een runtime-fout genereren)

```
private static int WoordNaarNummer(string woord)
{
    int resultaat = 0;
    if (woord.Equals("tien"))
        resultaat = 10;
    else if (woord.Equals("honderd"))
        resultaat = 100;
    else
        throw new FormatException("Verkeerde invoer: " + woord); // eigen melding meegeven
    return resultaat;
}
```

- Zelf weer opvangen met try catch

```
try
{
    Console.WriteLine(Convert.ToString(WoordNaarNummer("hXnderd")));
}
catch (FormatException ex)
{
    Console.WriteLine(ex.Message);
}
```

# Debugging: exceptions throwen

Defensief programmeren

- Beter exceptions voorkomen dan dat we ze opvangen.
- Gebruik een if-test om te valideren of er een fout kan komen of niet.
- Gebruik if-test als er een grote kans is op fout.
- Maar: Gebruik try catch voor zeldzame fouten (< 25% kans).

```
int getal = 10;
int deler = 0;
if (deler == 0)
{
    Console.WriteLine("Delen door 0 mag niet!");
}
else
{
    Console.WriteLine($"Resultaat: {getal / deler}.");
}
```

# C# Essentials Arrays

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- Arrays

# Arrays

- Probleem met onderstaande code
  - Wat met 10 getallen?
  - 100, 1000??

```
private static void LelijkeMethod()
{
    float gemiddelde = 0.0f;
    int getal0 = 100;
    int getal1 = 50;
    int getal2 = 20;
    int getal3 = 60;
    int getal4 = 90;
    int getal5 = 80;
    gemiddelde = (getal0 + getal1 + getal2 + getal3 + getal4 + getal5) / 6.0f;
    Console.WriteLine($"Gemiddelde: {gemiddelde}");
}
```

# Arrays

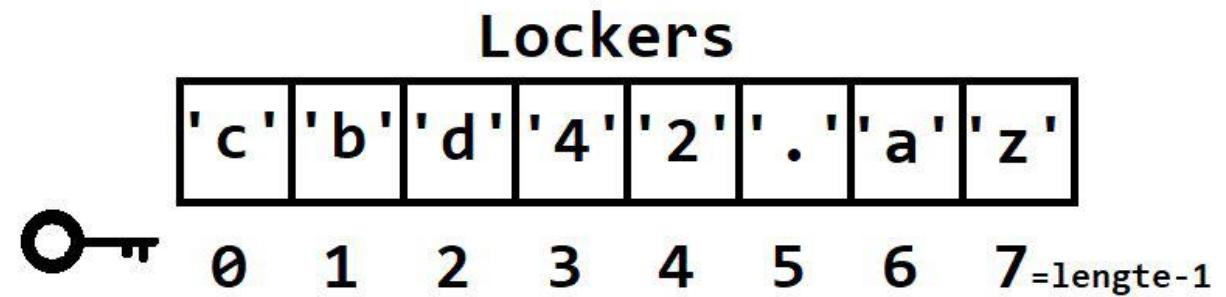
- Oplossing is arrays!

```
private static void LelijkeMethod()
{
    float gemiddelde = 0.0f;
    int getal0 = 100;
    int getal1 = 50;
    int getal2 = 20;
    int getal3 = 60;
    int getal4 = 90;
    int getal5 = 80;
    gemiddelde = (getal0 + getal1 + getal2 + getal3 + getal4 + getal5) / 6.0f;
    Console.WriteLine($"Gemiddelde: {gemiddelde}");
}
```

```
private static void MooieMethod()
{
    float gemiddelde = 0.0f;
    int[] getallen = new int[6] { 100, 50, 20, 60, 90, 80 };
    for (int i = 0; i < getallen.Length; i++)
    {
        gemiddelde += (float)getallen[i];
    }
    gemiddelde /= getallen.Length;
    Console.WriteLine("Gemiddelde: {0}", gemiddelde);
}
```

# Arrays

- Rij of reeks van gegevens van hetzelfde type
- Vaste lengte
- Zijn eigenlijk variabelen met een index



# Arrays

- Rij of reeks van gegevens van hetzelfde type met vaste lengte

```
int[] getallen = new int[6] { 100, 50, 20, 60, 90, 80 };
```

- Zijn eigenlijk variabelen met een index
  - getallen[3]; // 60
  - getallen[0]; // 100
  - getallen[getallen.Length - 1]; // 80
  - getallen[-1]; //Error
    - Waarom?
  - getallen[getallen.Length]; //Error
    - Waarom?
- Gebruik index om variabele van array te lezen
  - gemiddelde += getallen[i];
- Of aanpassen
  - getallen[i] = 100;

# Arrays

- Declaratie

```
double[] getallen = new double[7]; // eerste index: 0, laatste index: 6  
string[] namen = new string[9];
```

- Initialisatie

```
string[] namen = new string[5] {"Peter", "Sven", "Piet", "Tom", "Anna"};  
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Anna"};  
string[] namen = {"Peter", "Sven", "Piet", "Tom", "Anna"};
```

- Achteraf veranderen

```
namen[0] = "Jack";
```

# Arrays

- Loopen

```
for (int i = 0; i < getallen.Length; i++)
{
    Console.WriteLine(getallen[i]);
}
foreach (double getal in getallen)
{
    Console.WriteLine(getal);
}
```

- For
  - als je (deel van) array element per element wil doorlopen
- Foreach
  - als je **alle** array elementen wil doorlopen
  - enkel lezen, niet wijzigen

# Arrays

- Test oefening
- Zoek kleinste en grootste getal uit een reeks van getallen
- Gebruik: arrays, loops (for of foreach)
- Declareer en initializeer op verschillende manieren

# Arrays

- Array returnen uit functiemethod
  - Functiemethod gebruiken

```
int[] getallen = new int[10];
getallen = MaakArray();
```

- De functiemethod

```
private int[] MaakArray()
{
    int[] nieuweGetallen = new int[10];
    for (int i = 0; i < 10; i++)
    {
        nieuweGetallen[i] = i;
    }
    return nieuweGetallen;
}
```

# Arrays

- Array doorgeven als parameter
  - Functiemethod gebruiken

```
int resultaat = Som(getallen);
```

- De functiemethod

```
private int Som(int[] getallen)
{
    int som = 0;
    foreach (int getal in getallen)
    {
        som += getal;
    }
    return som;
}
```

- Analoog voor void method
- Gebruik ref om array aan te passen binnen method

# Arrays

- Wanneer array gebruiken
  - Rij nodig van aantal gegevens
  - Hetzelfde type
  - Aantal elementen ligt vast
  - Loopen over gegevens en inlezen of overschrijven
    - Van begin tot eind
    - Van eind tot begin
    - Bepaald segment
    - Stapgrootte instellen: ( $i+=2$  bijvoorbeeld in plaats van  $i++$ )

# Arrays

- Test oefening
- Zoek kleinste en grootste getal uit een reeks van getallen
- Pas de vorige oefening aan:
  - Array als parameter
  - Return array van kleinste en grootste getal

# C# Essentials Hulpfuncties arrays

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- Hulpfuncties arrays

# Hulpfuncties arrays

- `IndexOf()`: Index van eerste voorkomen

```
string[] namen = new string[] { "Peter", "Tom", "Piet", "Tom", "Benny", "Anna" };
int index = Array.IndexOf(namen, "Tom");
Console.WriteLine(index); // 1
// Index Piet?
// Index Jef? => -1 (als element niet gevonden wordt)
// Index laatste voorkomen van Tom?
```

# Hulpfuncties arrays

- `LastIndexOf()`: Index van laatste voorkomen

```
string[] namen = new string[] { "Peter", "Tom", "Piet", "Tom", "Benny", "Anna" };
int index = Array.LastIndexOf(namen, "Tom");
Console.WriteLine(index); // 3
```

# Hulpfuncties arrays

- Copy(): Array kopiëren

```
string[] namen = {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};
string[] kopie = new string[namen.Length];
Array.Copy(namen, kopie, namen.Length); // Laatste parameter bepaald hoeveel
items er gekopieerd worden
foreach (string naam in kopie)
{
    Console.WriteLine(naam);
}
// Peter
// Sven
// Piet
// Tom
// Benny
// Anna
```

# Hulpfuncties arrays

- Reverse(): Array omkeren

```
string[] namen = {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};
Array.Reverse(namen);
foreach (string naam in namen)
{
    Console.WriteLine(naam);
}
// Anna
// Benny
// Tom
// Piet
// Sven
// Peter
```

# Hulpfuncties arrays

- Find(): Zoek element

```
string[] consoles = {"PS2", "XBox", "Dreamcast", "N64", "Gamecube", "PS5"};  
  
// Zoekt eerste element dat begint met N.  
string console1 = Array.Find(consoles, element => element.StartsWith("N"));  
Console.WriteLine(console1); // N64  
  
// Zoekt eerste element dat 9 karakters heeft.  
string console2 = Array.Find(consoles, element => (element.Length == 9));  
Console.WriteLine(console2); // Dreamcast
```

- Predicates (*element => (element.Length == 9)*) met lambda expression (*lambda operator =>*)

# Hulpfuncties arrays

- `FindAll()`: Zoek alle elementen

```
string[] consoles = {"PS2", "XBox", "Dreamcast", "N64", "Gamecube", "PS5"};
// Zoekt alle elementen die beginnen met een P.
string[] pConsoles = Array.FindAll(consoles, element =>
element.StartsWith("P"));
foreach (string console in pConsoles)
{
    Console.WriteLine(console);
}
// PS2
// PS5
```

# Hulpfuncties arrays

- Exists(): Bestaat het element (true/false)

```
string[] consoles = {"PS2", "XBox", "Dreamcast", "N64", "Gamecube", "PS5"};
bool bestaat = Array.Exists(consoles, element => element.Equals("Phantom"));
Console.WriteLine(bestaat);
// false
// Hoe kijken of er een console is met 3 karakters?
bool bestaat = Array.Exists(consoles, element => element.Length.Equals(3));
```

# Hulpfuncties arrays

- Sort(): Sorteren van klein naar groot

```
string[] namen = {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};
Array.Sort(namen);
foreach (string naam in namen)
{
    Console.WriteLine(naam);
}
// Anna
// Benny
// Peter
// Piet
// Sven
// Tom
```

# Hulpfuncties arrays

- Hoe sorteren van groot naar klein?

```
string[] namen = {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};
//??
Array.Sort(namen);
Array.Reverse(namen);
foreach (string naam in namen)
{
    Console.WriteLine(naam);
}
// Tom
// Sven
// Piet
// Peter
// Benny
// Anna
```

# Hulpfuncties arrays

- Sort(): Bepaald gedeelte van array sorteren

```
string[] namen = { "Peter", "Sven", "Piet", "Tom", "Benny", "Anna" };
int startIndex = 1;
int length = 4;
Array.Sort(namen, startIndex, length);
foreach (string naam in namen)
{
    Console.WriteLine(naam);
}
// {"Sven", "Piet", "Tom", "Benny"} =>
// Peter
// Benny
// Piet
// Sven
// Tom
// Anna
```

# Hulpfuncties arrays

- BinarySearch(): SNEL element zoeken in een **GESORTEERDE** array

```
string[] namen = {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};
Array.Sort(namen);
// Anna Benny Peter Piet Sven Tom
int index = Array.BinarySearch(namen, "Sven");
Console.WriteLine(index);
// 4

// Waarom op voorhand gesorteerd?
```

- Binair zoeken: telkens kijken in de juiste helft

```
// Anna Benny Peter Piet Sven Tom
// Anna Benny Peter [Piet Sven Tom]
// Anna Benny Peter [Piet Sven] Tom
// Anna Benny Peter Piet Sven Tom
```

# Hulpfuncties arrays

- Clear(): Array leegmaken

```
// 1D Array leegmaken
float[] getallen = { 1.1f, 1.2f, 1.3f, 1.4f, 1.5f, 1.6f, 1.7f };
int startIndex = 2;
int length = 4;
Array.Clear(getallen, startIndex, length);
// { 1.1f, 1.2f, 0.0f, 0.0f, 0.0f, 1.7f }

// Hoe volledig leegmaken?
int startIndex = 0;
int length = getallen.Length;
Array.Clear(getallen, startIndex, length);
```

# Hulpfuncties arrays

- **Resize()**: Grootte van array veranderen (wordt weinig gebruikt)

```
// C# alloceert nieuwe array en kopieert inhoud over

int[] getallen = { 3, 4, 5, 6 };
// Groter maken
Array.Resize(ref getallen, 10); // 3 4 5 6 0 0 0 0 0 0
// Kleiner maken
Array.Resize(ref getallen, 3); // 3 4 5
```

# C# Essentials 2D arrays

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- 2D arrays

# 2D arrays

- **1D array**
  - is 1 rij
  - [ 'a' ][ 'b' ][ 'c' ][ 'd' ][ 'e' ][ 'f' ]
- **2D array**
  - tabel van rijen en kolommen.
  - [ 'a' ][ 'b' ][ 'c' ]
  - [ 'd' ][ 'e' ][ 'f' ]
- Bepaal indices?
  - 1D: a (0de element), b (1), c (2), d (3), e (4), f (5)
  - 2D: a (0de rij, 0de kolom), b (0, 1), c (0, 2), d (1, 0), e (1, 1), f (1, 2)

# 2D arrays

- Declaratie

```
int[,] getallen = new int[4, 2]; // 4 rijen en 2 kolommen  
int[,] getallekes = new int[2, 4]; // 2 rijen en 4 kolommen
```

- Initialisatie

```
int[,] getallen = new int[4, 2] { { 8, 6 }, { 9, 1 }, { 7, 8 }, { 5, 6 } };  
int[,] getallen = new int[,] { { 8, 6 }, { 9, 1 }, { 7, 8 }, { 5, 6 } };  
int[,] getallen = { { 8, 6 }, { 9, 1 }, { 7, 8 }, { 5, 6 } };
```

- Waarom gaat deze niet?

```
int[,] getallen = { { 8, 6, 4 }, { 9, 1 }, { 7, 8, 2, 1 }, { 5, 6 }, { 1 } };
```

// Aantal kolommen moet telkens gelijk blijven.

# 2D arrays

- Loopen

```
int aantalRijen = getallen.GetLength(0); // 0 geeft aan: rijen
int aantalKolommen = getallen.GetLength(1); // 1 geeft aan: kolommen
int totaalAantalElementen = getallen.Length;
for (int r = 0; r < aantalRijen; r++)
{
    for (int k = 0; k < aantalKolommen; k++)
    {
        getallen[r, k] = r + k;
    }
}
```

# Hulpfuncties arrays

- Herinner Clear() om een array leeg te maken

```
// 1D Array leegmaken
float[] getallen = { 1.1f, 1.2f, 1.3f, 1.4f, 1.5f, 1.6f, 1.7f };
int startIndex = 2;
int length = 4;
Array.Clear(getallen, startIndex, length);
// { 1.1f, 1.2f, 0.0f, 0.0f, 0.0f, 1.7f }

// Hoe volledig leegmaken?
int startIndex = 0;
int length = getallen.Length;
Array.Clear(getallen, startIndex, length);
```

# Hulpfuncties arrays

- Clear(): Array leegmaken

```
// 2D Array leegmaken
float[,] getallen = { { 1.1f, 1.2f }, { 1.3f, 1.4f }, { 1.5f, 1.6f }, { 1.7f,
1.8f } };
int startIndex = 3;
int length = 4;
Array.Clear(getallen, startIndex, length);
foreach (float getal in getallen)
{
    Console.WriteLine(getal);
}
// Wat is het resultaat?
{ {1.1f, 1.2f}, {1.3f, 0.0f}, {0.0f, 0.0f}, {0.0f, 1.8f} }
```

# C# Essentials Generic lists

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- Generic lists

# Generic lists

- Nadelen van arrays
  - Lengte ligt vast
  - Waarde toevoegen of verwijderen:
    - Kopie maken van array naar nieuwe array met nieuwe grootte
    - Elke waarde opschuiven
- Oplossing
  - Gebruik List<T> uit System.Collections.Generic
    - Kan je extra elementen aan toevoegen.
  - Gebruikt inwendig een array. Wanneer volle capaciteit ⇒ groeien
  - Is generiek
    - Je kan het type T kiezen
    - Als je zelf een generieke class of method maakt => geen aparte versie voor elk datatype nodig!
- Heeft methods en properties beschikbaar

# Generic lists

- Declaratie en initialisatie

```
// Declaratie
List<string> studenten;
// Initialisatie
studenten = new List<string>();

// In één regel
List<int> leeftijd = new List<int>();

// Initialiseren met array
int[] nummers = new int[3] { 1, 2, 3 };
List<int> nummersList = new List<int>(nummers);
```

# Generic lists

Add(T element)	Voegt element toe aan einde van List
AddRange(collection)	List/array toevoegen aan andere List
Clear()	Wist alle elementen van List
Count	Telt het aantal element in List ( <b>property</b> )
ToArray()	Kopieert List naar nieuwe 1D array
GetRange(int startIndex, int aantal)	Maakt kopie van stukje van de lijst
IndexOf(T element) IndexOf(T element, int startIndex) IndexOf(T element, int startIndex, int aantal)	Zoekt de index van een element uit List
Insert(int index, T element)	Voegt element toe op bepaalde index, de elementen erna schuiven 1 plaats op
Remove(T item)	Verwijdert element met bepaalde waarde
RemoveAt(int index)	Verwijdert element op bepaalde index
Reverse() Reverse(int startIndex, int aantal)	draait de volgorde van de elementen om
Sort()	Sorteert de elementen in de List

# Generic lists

- Operator
  - [ ] : om elementen op te zoeken of aan te passen (zoals in array)

```
int[] nummers = new int[3] { 1, 2, 3 };  
List<int> nummersList = new List<int>(nummers);  
int getal = nummersList[2];
```

# Generic lists

- Elementen toevoegen aan een list

```
List<int> waarden = new List<int>{ 10, 20, 30, 40, 50, 60, 70, 80 };  
// Waarden toevoegen aan List  
waarden.Add(90);  
waarden.Add(100);
```

# Generic lists

- AddRange() om array (of zelfs andere List) toe te voegen aan de List

```
int[] getallen = new int[]{ 10, 20, 30, 40, 50, 60, 70, 80 };
List<int> waarden = new List<int>();
// Voeg een array toe aan de List
waarden.AddRange(getallen);
```

# Generic lists

- Elementen inserten in List

```
int[] getallen = new int[]{ 10, 20, 30, 40, 50, 60, 70, 80 };
List<int> waarden = new List<int>();
// Voeg een array toe aan de List
waarden.AddRange(getallen);
// Voeg waarde 1000 toe op index 3 van de List (indices starten vanaf 0!!!)
// De waarden erna schuiven dan 1 plaats op
waarden.Insert(3, 1000);
```

- Element verwijderen uit List

```
// Verwijder element op index 3 van de List
waarden.RemoveAt(3);
// OF verwijder element met waarde 1000 (die als eerste voorkomt!) uit de List
waarden.Remove(1000);
```

# Generic lists

- List omzetten naar array

```
int[] getallen = new int[]{ 10, 20, 30, 40, 50, 60, 70, 80 };
List<int> waarden = new List<int>();
// Voeg een array toe aan de List
waarden.AddRange(getallen);
// List omzetten naar array
int[] waardenArray = waarden.ToArray();
```

- Stukje van list omzetten naar array

```
// Neem enkel het stukje van de list:
// - vanaf index 2
// - 3 elementen als aantal (lengte)
int[] waardenArray = waarden.GetRange(2, 3).ToArray(); // 30, 40, 50
```

# Generic lists

- List sorteren

```
string[] voornamen = { "Wouter", "Paul", "Andreas", "Niels", "Kathleen",
"Paul", "Silvia", "Patricia"};
List<string> waarden = new List<string>(voornamen);

// Waarden sorteren van klein naar groot
waarden.Sort();

// Waarden sorteren van groot naar klein
waarden.Sort();
waarden.Reverse();
```

# Generic lists

- Zoeken naar index in list

```
string[] voornamen = { "Wouter", "Paul", "Andreas", "Niels", "Kathleen", "Paul",
"Silvia", "Patricia"};
List<string> waarden = new List<string>(voornamen);

waarden.IndexOf("Lise"); // => -1: niet gevonden
waarden.IndexOf("Paul", 3); // vanaf index 3 zoeken => 5 is het eerste voorkomen
waarden.IndexOf("Paul", 2, 3); // vanaf index 2 zoeken, aantal elementen: 3 => -1
waarden.IndexOf("Paul", 2, 4); // vanaf index 2 zoeken, aantal elementen: 4 => 5
```

# C# Essentials Dictionaries

Koen Bloemen



**DE HOGESCHOOL  
MET HET NETWERK**

Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)





- Dictionaries

# Dictionaries

- Dictionary<K,V>
- Slaat elementen op als sleutelpaar (key-value pair)
  - Sleutel (key) van type K
  - Waarde (value) van type V
- Met de sleutel kan je waarden opzoeken, aanpassen en verwijderen
  - Is gelijkaardig aan de index van een array
  - Kan je vergelijken met een soort index die eender welke waarde of type kan hebben
- Iedere waarde heeft een sleutel toegekend
  - Waarde kan eender welk type hebben
- Sneller als je vaak gegevens moet opzoeken (sneller dan (binair) zoeken in List)
- Trager als je door gegevens moet loopen

# Dictionaries

- Declaratie en initialisatie

```
Dictionary<int, int> eenDictionaryMetGetallen;

//In dict2 zijn de keys van type string en de values van type int array
Dictionary<string, int[,]> dict2 = new Dictionary<string, int[,]>();

// Je kan een dictionary ook onmiddellijk initialiseren
Dictionary<int, string> dict= new Dictionary<int, string>()
{
    { 1, "hello"},
    { 3, "world"},
    { 9001, "over 9000"}
};
```

# Dictionaries

- Toevoegen en wijzigen van values

```
Dictionary<int, string> dict= new Dictionary<int, string>()
{
    { 1, "hello" },
    { 3, "world" },
    { 9001, "over 9000" }
};

Dictionary<string, int> dictWithIntValues= new Dictionary<string, int>();

// Toevoegen
dict.Add(2, "hello");
dict.Add(2, "gaat niet werken"); // werkt niet => key bestaat al
dict[3] = "vier waarde"; // werkt ook met index-notatie
dictWithIntValues["a"] = 1;

// Aanpassen
dict[1] = "overschreven";
```

# Dictionaries

- Verzameling opvragen van alle keys en values

```
Dictionary<int, string> dict= new Dictionary<int, string>()
{
    { 1, "hello" },
    { 3, "world" },
    { 9001, "over 9000" }
};

// OPGELET: door een dictionary loopen is trager dan loopen door een list!!!
foreach(var key in dict.Keys)
{
    Console.WriteLine(key);
}

foreach(var value in dict.Values)
{
    Console.WriteLine(value);
}
```

# Dictionaries

- Element opzoeken

```
Dictionary<string, int> punten = new Dictionary<string, int>()
{
    { "Tom", 12},
    { "Ozgun", 17},
    { "Elise", 16}
};

// Element opzoeken in dictionary
int puntenTom = punten["Tom"];

// Element PROBEREN opzoeken in dictionary => zelfde werking als TryParse
int puntenAnna;
bool isGevonden = punten.TryGetValue("Anna", out puntenAnna);
if (isGevonden)
    Console.WriteLine($"Anna heeft {puntenAnna}/20");
```

# Dictionaries

- Element verwijderen

```
Dictionary<string, int> punten = new Dictionary<string, int>()
{
    { "Tom", 12},
    { "Ozgun", 17},
    { "Elise", 16}
};
// Element verwijderen uit dictionary
punten.Remove("Tom");

// Resultaat van verwijdering opvragen
bool isVerwijderd = punten.Remove("Anna");
```

# Dictionaries

- Controleren of element (key) bestaat

```
Dictionary<string, int> punten = new Dictionary<string, int>()
{
    { "Tom", 12},
    { "Ozgun", 17},
    { "Elise", 16}
};
bool bestaat = punten.ContainsKey("Tom");
```

# Dictionaries

- Loopen over dictionary (TE VERMIJDEN = TRAAG)

```
Dictionary<string, int> punten = new Dictionary<string, int>()
{
    { "Tom", 12},
    { "Ozgun", 17},
    { "Elise", 16}
};
// Loopen over dictionary en keys en values opvragen van elk item
foreach (var item in punten)
{
    Console.WriteLine($"{item.Key}: {item.Value}/20");
}
```

# Dictionaries

- Dictionary omzetten naar List (TE VERMIJDEN!)

```
Dictionary<string, int> punten = new Dictionary<string, int>()
{
    { "Tom", 12},
    { "Ozgun", 17},
    { "Elise", 16}
};
//Omzetten naar List
List<KeyValuePair<string, int>> list = punten.ToList();
// Daarna afdrukken
foreach (KeyValuePair<string, int> pair in list)
{
    Console.WriteLine($"{pair.Key}: {pair.Value}");
}
```