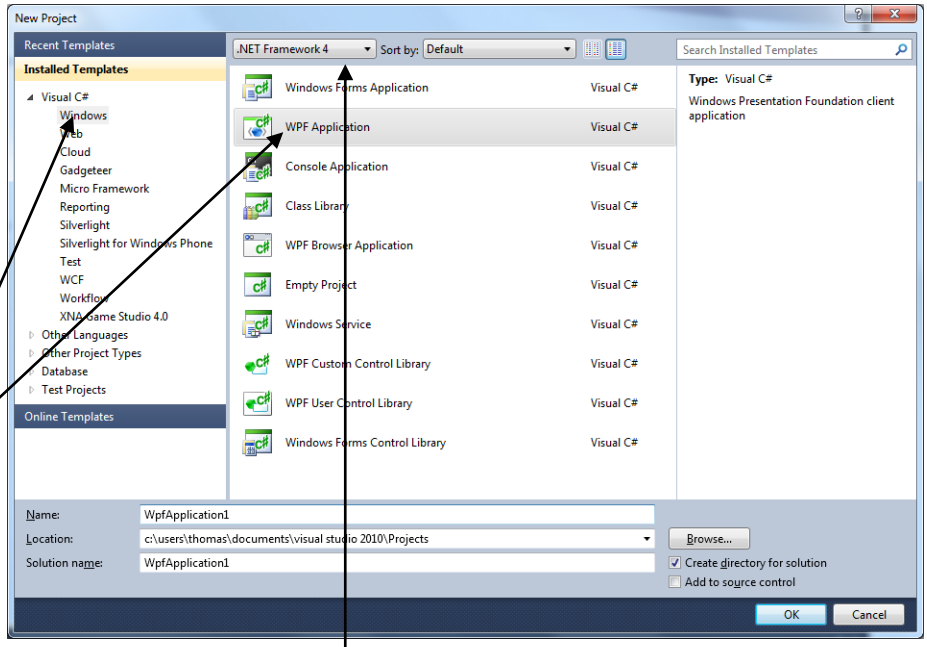
Følgende opgave er en introduktion til Visual Studio og de mest anvendte værktøjer vi benytter os af når vi udvikler programmer til Windows med C# og Windows Presentation Foundation (WPF)

I den følgende opgave skal vi lave en simpel "Hello, world!" applikation.

Undervejs i opgaven vil du blive introduceret til flere af de koncepter som gør Visual studio, WPF og C# til et af de absolut hurtigste udviklings værktøjer til dato.

Start med at oprette et nyt **C# WPF projekt**.

Det er vigtigt at du her husker at vælge **C#** og ikke et af de andre sprog!

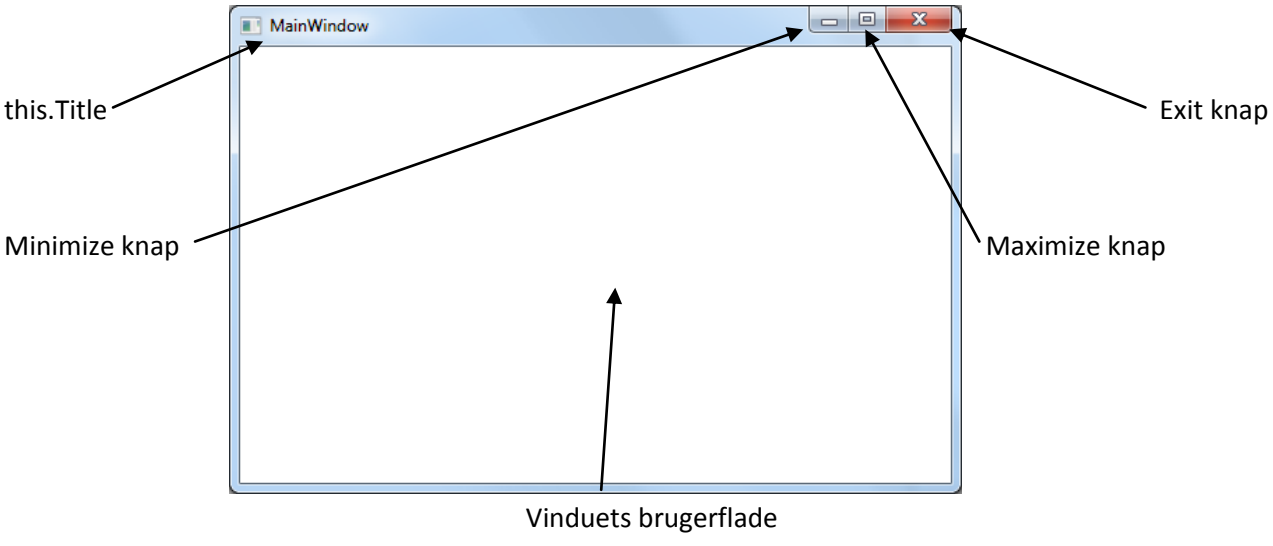


Navngiv Projektet så du efterfølgende kan genkende det. Eks. "HelloWorld" eller "opgave_1" eller lignende.

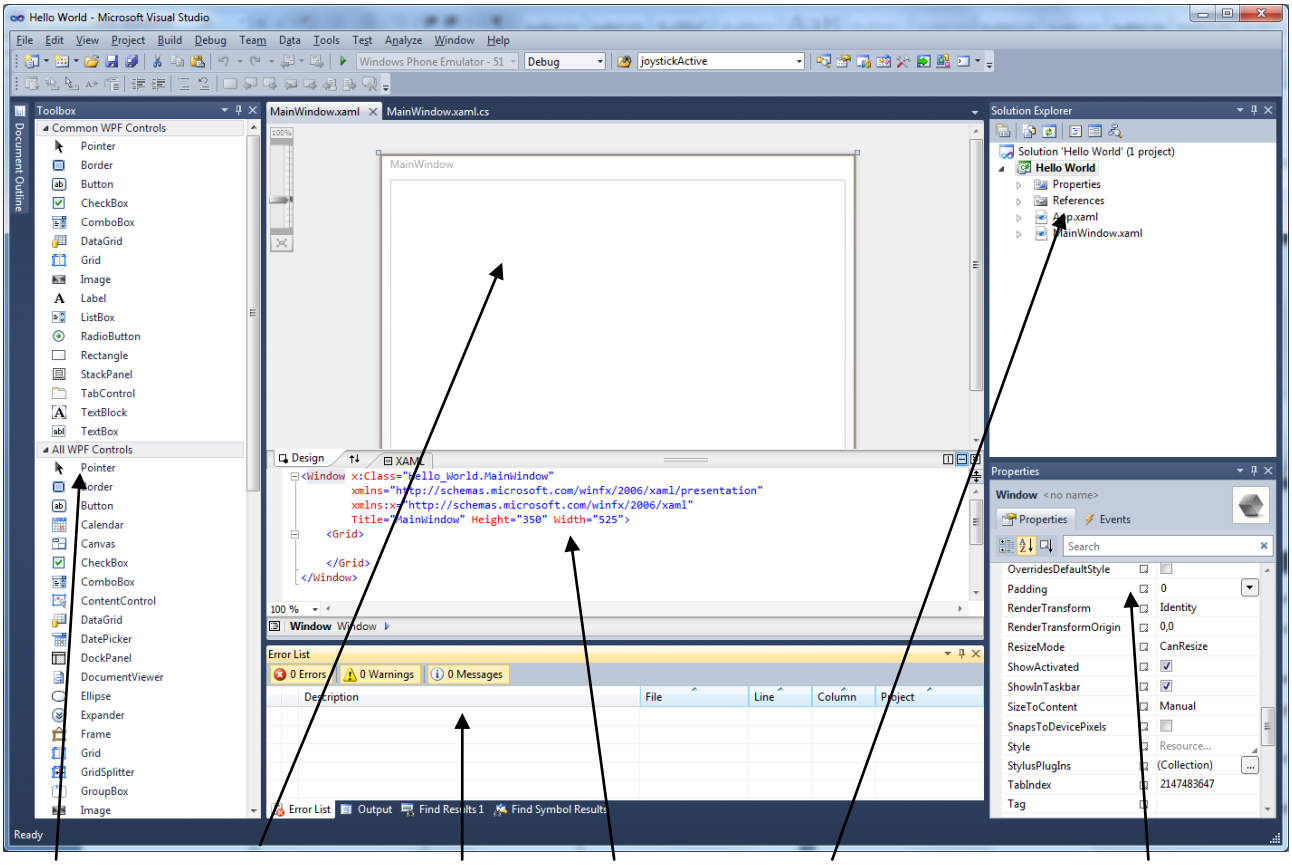
I billedet til højre kan du ligeledes se at jeg har valgt .NET framework 4

Når du er tilfreds med dine valg trykker du OK. Så vil Visual Studio efterfølgende generere dit nye projekt med alt den tilhørende kode.

Visual Studio vil nu præsentere dig for dit nye projekt som indtil videre kun består af et hel grundlæggende vindue som vist herunder (Du kan se det i funktion, ved at starte programmet med F5 tasten - debug)



Visual Studio vil nu se ud nogenlunde som herunder, hvor billedet viser selve IDE't med de mest almindelige værktøjer fremme:



Toolbox Visual Designer Error List XAML View Solution Explorer Object Properties

Toolbox: Det er her vi finder de mest anvendte "kontroller" som vi frit kan anvende i vores WPF applikation.

Visual Designer: Her designer vi brugerfladen primært ved hjælp af "træk og slip" metoden.

Error List: Viser løbende fejl i forbindelse med vores kodning, samt eventuelle kompilerings fejl.

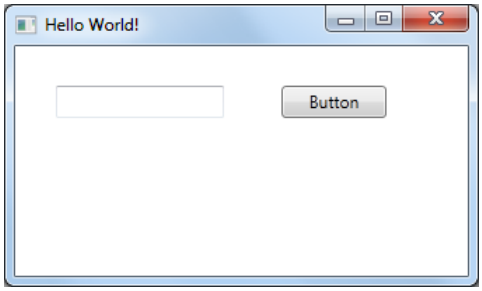
XAML View: Dette er markup koden den visuelle del af brugerfladen er opbygget af.

Solution Explorer: Viser hele vores projekt som en sammensat enhed af filer og generelle egenskaber.

Objekt Properties: Viser det markerede objekts egenskaber. I dette tilfælde er det hele formen der er markeret.

Skulle der mangle et af vinduerne, kan du finde dem igen under menu punktet "View".

Start med at trække en tekst boks og en knap fra værktøjskassen og ind på brugerfladen, så den ser ud nogenlunde som her til højre:



Vi skal nu i gang med at modificerer de to kontroller så de opfører sig korrekt. Dette gør vi direkte i objektets property vindue.

TextBoks:

Start med at markerer tekstboksen og ret derefter de enkelte properties så de matcher indstillingerne herunder:

Name	tb_Navn (dette skrives i toppen af property vinduet. Der hvor der i forvejen står "button1")
Margin	20,20,0,0
Width	180

Button:

Herefter markerer du knappen og ændrer egenskaberne, igen som vist herunder:

Name	btn_SayHello
Content	Say, hello!
Margin	220,20,0,0

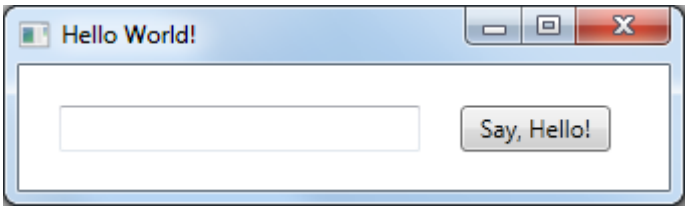
Vinduet:

Til sidst skal vi have modificeret Vinduet så den passer i størrelse:

Title	Hello, world!
Height	100
Width	340

Som du sikkert allerede har bemærket bliver de properties vi har ændret markeret med et sort ikon for at indikere at de ikke længere matcher standard indstillingerne for et "normalt" vindue. Du kan højre klikke på ikonet og vælge "Reset value" for at få de oprindelige indstillinger tilbage.

Hvis du tester programmet med F5 skulle endelige resultat gerne se ud præcis som vist herunder:



Vi har indtil videre teste vores nye Windows program ved at bruge F5. Dette starter programmet op i "debug mode" hvilket er at foretrække, da vi jo er i gang med at udvikle på det. Men du kan også starte programmet op uden om debuggere, hvis du skulle have behov for at se hvordan tingen kører uden debugger... Altså:

Med debugger: Ved at trykke "F5" tasten

Her køres programmet med en debugger tilkoblet så vi kan standse og undersøge programmet om nødvendigt.

Uden debugger: Ved at trykke ctrl + "F5" tasten.

Her køres programmet som i den "virkelige" verden. Altså uden debugger tilkoblet. Og vi fanger ikke eventuelle fejl i programmet.

Vi vil i samtlige efterfølgende opgaver udelukkende benytte kørsel ***med*** debugger. Da det giver os rig lejlighed for at fange og undersøge eventuelle fejl i vores kode.

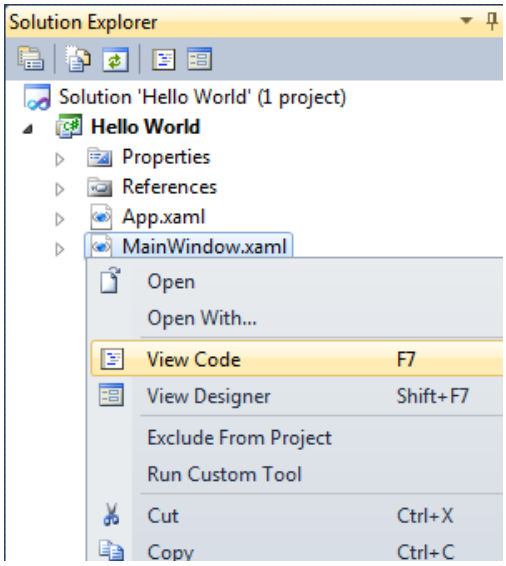
Hvis du eksperimenterer lidt vil du se at minimer, maksimer og luk knapperne allerede virker. Dette skyldes den, i vinduet, indbyggede funktionalitet som vi uden videre har fået forærende.

Du kan yderligere teste programmet ved at skrive i tekst boksen eller trykke på knappen. Der vil dog ikke ske noget særligt, da vi selv skal tilføje den ønskede funktionalitet.

Code Behind

For at få noget funktionalitet i vores program skal vi i gang med at kode. Dette gøres også i IDE'et.

Du kan markere design vinduet og trykker "F7" eller som vist her til højre, ved at højre klikke på MainWindow.xaml filen og vælg View Code.



Code behind filen indeholder størstedelen af den C# kode vi vil beskæftige os med henover de næste par dage. Der er dog tidspunkter hvor det bliver nødvendigt at kigge lidt i et par andre kode filer. Men mere herom senere…

Code Behind fortsat:

Her ser koden fra C# code behind filen. MainWindow.xaml.cs

Bare rolig hvis din kode ikke ligner helt. Det skyldes muligvis at du har dobbelt klikket på en eller flere af kontrollerne på formen. Jeg vil senere vise hvordan du automatisk fjerner den uønskede kode igen.

Vi er som sagt nu inde i selve C# koden. Og det er blandt andet her vi kan tilføje funktionalitet til vores program.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Hello_World
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

Vi kan ligeledes ændre de enkelte kontroller's properties direkte fra koden. Prøv at ændre programmet som vist nedenfor: (tilføj koden markeret med **fed**)

```
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            tb_Navn.Text = "Skriv dit navn!";
        }
    }
```

Vær opmærksom på at kun dele af den samlede kode er vist i eksemplet herover!

Prøv og kør programmet (med F5) . Hvad der er ændret?

Hvis du går tilbage til design viewet (vælg fanebladet MainWindow.xaml og kigger på properties for tekstboksen vil du se at "text" propertyen stadig er tom. Dette skyldes at de værdier vi ser her udelukkende er gældende, lige når programmet starter. Og alle ændringer vi efterfølgende laver via kode bliver derfor ikke registreret i property vinduet.

Tekstboksen:

Inden vi begynder at kode bør vi kigge lidt nærmere på tekstboksen.

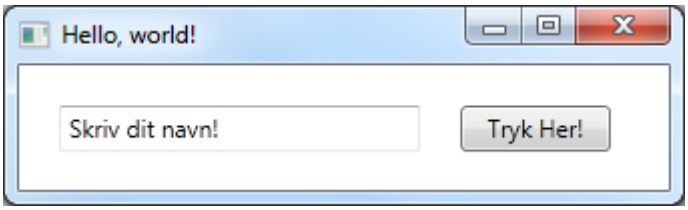
Tekstboksen har vi trukket ud fra vores toolboks. Og toolboksen indeholder som du sikket allerede har set, også en hel del andre controls. Fælles for alle disse controls er at de er en del af et Base Class Library (BCL).

Vi trak ligeledes en Button control ind på brugerfladen. Og hvis du undersøger egenskaberne for knappen, vil du se at mange af de egenskaber som tekstboksen har, går igen på knappen. Dette skyldes at begge controller nedarver visse fælles egenskaber. Hvilket for øvrigt gælder temmelig mange af de controls du kan finde i toolboksen. Og netop nedarving er en af hjørnestenene i objekt orienteret programmering. Og der er desværre skrevet store, tykke og seriøst kedelige bøger omkring nedarvning, så på nuværende tidspunkt vil vi blot konstatere at de to controls har flere fællestræk. Og så gemme nedarvning til lidt senere.

Men princippet med at alle de mange controls har flere fælles egenskaber, gør vores arbejde som programmør meget nemmere. For "text" propertyen i en tekst boks benyttes på akkurat samme måde som "text" propertyen på en TextBlock. Det samme gælder Height, Margin, Fontsize og mange af de andre properties.

Prøv eksempelvis om du via koden kan ændre teksten på knappen så den ligner eksemplet nedenfor:

(teksten på knappen er ændret via kode fra "Say, hello!" til "Tryk ker!"



Hvad skulle der tilføjes?

Event Styring

WPF Applikationer er event styret. Det vil sige at programmet som sådant ikke ”gør” noget med mindre det får besked på det, enten via bruger input eller fra andre programdele (heriblandt også Windows)

Selv om det lyder avanceret er det faktisk meget simpelt. Primært fordi mange af de events, der skal til, for at vores applikation virker, allerede er indbygget i programmet.

Af automatiske events kan bl.a. nævnes at vinduet ”tegner” sig selv hver gang det flyttes, eller når musen eller andre programmer flyttes så de dækker eller afdækker dele af vinduet.

Tre andre ”automatiske” events du allerede kender er minimer, maksimer og luk knapperne. De er ligeledes koblet op så vi ikke skal foretage os yderligere for at de virker.

Hvis vi ønsker at tilføje vores egne events kan det gøres på flere måder:

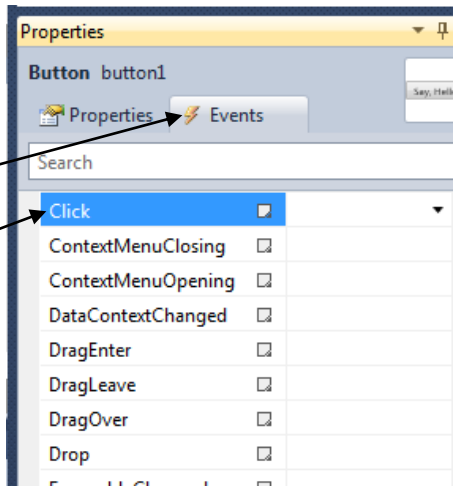
Via kode:

Her koder vi os ud af problemerne, i samme stil som da vi ændrede teksten på knappen. Metoden er ikke specielt avanceret, men vi vil i denne omgang holde os til de to efterfølgende metoder.

Via designeren: (den nemme metode)

Ved at dobbeltklikke på den control vi ønsker at tilføje et event til. Denne metode tilføjer et standart event til vores kode. I knappens tilfælde vil eventet være klik med musen. For tekstboksens vedkommende ville eventet være at der blev lavet ændringer af teksten…

Via designeren: (den rigtige måde)



Ved at markere den control vi ønsker at tilføje et event til (her er det knappen ”**btnSayHello**”), hvorefter vi via property vinduet vælger ”events” menyen. (**det lille gule lyn**). I events menuen vælger vi så det event vi ønsker at tilføje til controllen. Her til højre er det ”**Click” eventet**.

Ved at dobbelt klikke på ”Click” tilføjes den nødvendige kode til vores MainWindow.xaml.cs fil. Og vi bliver automatisk overført til denne kode som vist på næste side.

Ønsker vi at fjerne et event som vi har tilføjet skal du højreklikke på eventet og vælge ”Reset” Dette fjerner også den tilhørende kode i .cs filen. Med mindre vi har ændret i denne.

Herunder kan du se den kode Visual Studio automatisk tilføjer til vores program. (Markeret med **fed**) Koden bliver herefter kørt hver gang du klikker på knappen.

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
        tb_Navn.Text = "Skriv dit navn!";
    }

    private void button1_Click(object sender, RoutedEventArgs e)
    {

    }
}
```

Denne stump kode kaldes i daglig tale, en method. Og vi vil i senere opgaver komme til at lave vores helt egne methods. Men igen, vil vi lige nu nøjes med at fokuserer på navnet ”btnSayHello_Click” samt de to tuborgklammer.

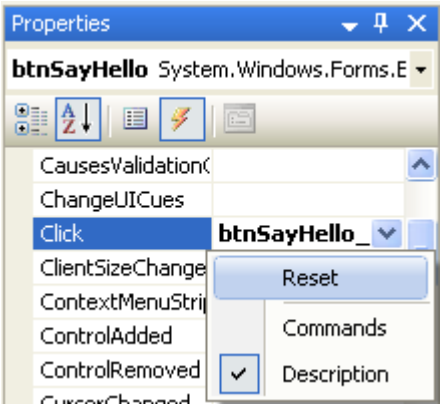
Metode navnet:

> Har, som du sikkert har regnet ud, sit udgangspunkt i to ting. Nemlig knappens navn(den knap eventet er tilkoblet), samt det specifikke event som funktionen håndterer. Dette navn er unikt og identificerer denne lille del af koden. (kodeblok)

Tuborgklammerne:

> Afgrænser funktionen til den før nævnte kodeblok. Det vil sige, når du trykker på knappen, er det *kun* koden inden for de to tuborgklammer der bliver afviklet. Afgrænsningen disse to tuborgklammer definerer, kaldes også ”et scope”. Vi vil kigge nærmere på scopes, når vi skal i gang med selv at lave funktioner.

Ønsker du at fjerne eventet igen, skal du som jeg også nævnte før, tilbage i design viewet, markere knappen og vælge events i Property vinduet. Her højreklikker du på ”Click” eventet og vælger Reset.



Det er vigtigt at du ***ikke*** bare sletter koden direkte i MainWindow.xaml.cs filen. Da dette vil resulterer i defekt kode. (hvilket dog kan rettes manuelt)

Det er også vigtigt at huske på at events er koblet til de objekter der kan ”fyre” dem. Det betyder at hvis du sletter knappen, så virker eventet ikke mere. Men koden vil stadig være i MainWindow.xaml.cs filen.

Vores program har været lidt kedeligt indtil videre, en simpel brugerflade uden noget reelt indhold… Det skal vi have lavet om på.

Tilføj følgende linje kode til din nye btn_SayHello_Click funktion:

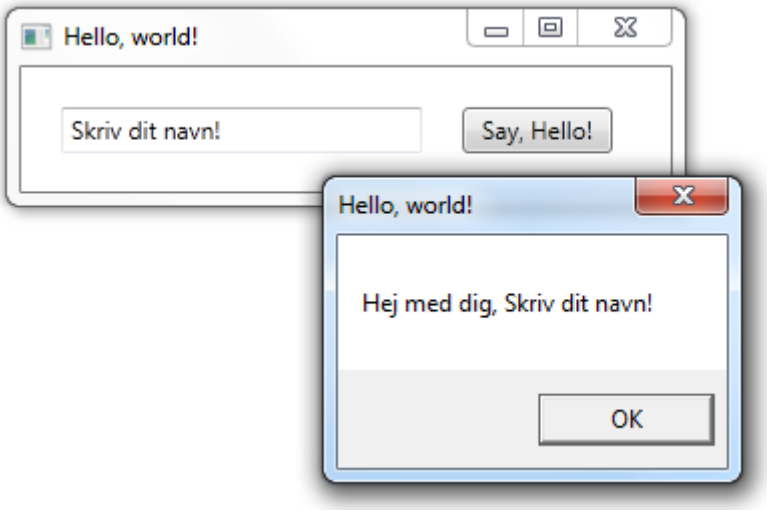
```
MessageBox.Show("Hej med dig, " + tb_Navn.Text, "Hello, world!");
```

Sammensætningen af indeholder er: Besked til brugeren + tekstboksens indhold, Overskrift

`MessageBox` er en så kaldt ”static class”. Og er endnu et af de færdige "værktøjer" Microsoft har tilføjet til deres .NET BCL og som vi uden større problemer kan benytte i vores kode.

Show er en method tilhørende MessageBox class, som generer og viser en besked til brugeren. Denne besked eller messagebox som det jo reelt er, kan modificeres på et utal af måder. Og du vil i en af de senere opgaver lære hvordan du kan spørge og efterfølgende modtage input fra brugeren via en messageboks.

Prøv at debugge programmer (F5) og se hvordan det virker – Resultatet skulle gerne være som herunder:



Vi har i denne opgave været igennem alle de trin der er nødvendige for at udvikle en simpel Windows applikation.

1. Oprette og navngive et nyt C# Windows Presentation Foundation projekt
2. Designe bruger grænseflade med tilhørende events.
3. Kode funktionalitet via code behind filen/filerne (MainWindow.xaml.cs filen)
4. Kompilerer programmet til en færdig eksekverbar fil (en .NET assemebly)

Så nu er du faktisk klar til en reel programmerings opgave, men inden vi starter skal vi lige kigge lidt på resultatet af alt vores hårde arbejde.

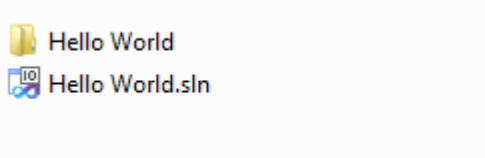
Visual studio har lavet et par tilføjelser til din dokument mappe.

Start med at browse dig ind i dokument mappen og find mappen med navnet ”Visual Studio 2010” i denne mapper skal du videre ned i ”Projects” mappen.

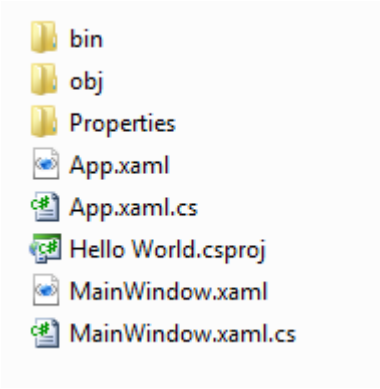
Herinde finder du de projekter du har lavet igennem tiden. Da dette er det første projekt er der sikkert kun mappen med dit Hello World projekt.

HelloWorld mappen indeholder, som navnet antyder, dit Hello, world! projekt. (din projekt mappe hedder selvfølgelig det samme som du har kaldt dit projekt, hvilket ikke nødvendigvis er ”hello World”)

Kigger vi i mappen vil vi se følgende indhold:



Hello World.sln er vores ”Solution” fil. Det vil sige, den der indeholder alt det vi kan se i Solution exploreren inde i Visual Studio.



Går vi endnu et niveau ned, i ”HelloWorld” mappen, ser vi følgende indhold:

Dette er hele vores HelloWorld projekt.

MainWindow.xaml filen indeholder Visual Studio generet kode til opsætning og visning af vores vindue. Vi vil i de senere opgaver kigge nærmere på indholdet af denne fil.

App.xaml/App.xaml.cs filen indeholder vores programs ”start” kode. Denne fil er ligeledes generet af Visual Studio. Og vi vil i denne omgang ikke beskæftige os yderligere med indholdet af disse filer.

MainWindow.xaml.cs filen kender du, det er vores codebehind fil, som vi har været inde og ændre i via Visual Studio.

Kigger du videre ned i ”bin” mappen vil du se to mapper ”Debug” og ”Release”. (Hvis du udelukkende har kørt dit program i debug mode, vil du ikke have noget indhold i Release mappe.)

Fælles for de to mapper er at de indeholder vores program i eksekverbar assemblyes. Oversat til dansk betyder dette bare at filerne kan afvikles på alle maskiner med .NET framework 4. installeret. og de kan derfor i mange tilfælde kopieres frit imellem brugere og Pc’erer hvor de uden videre vil virke.