

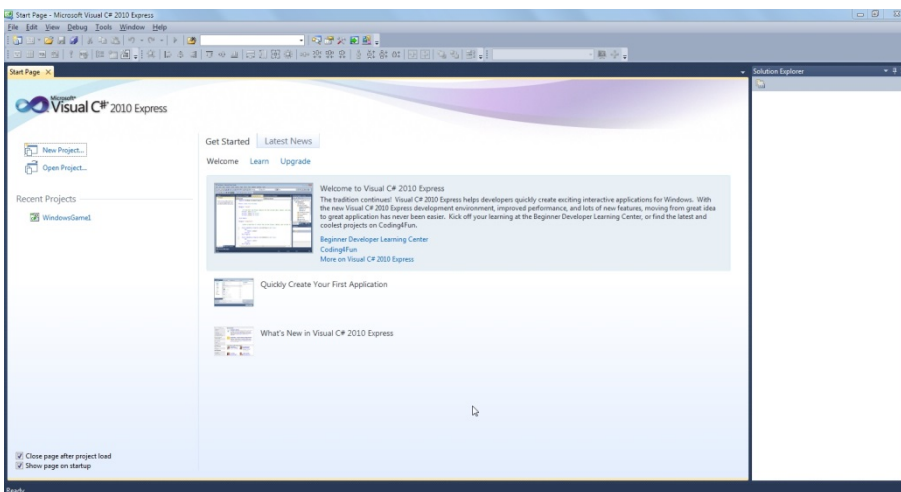
Kapitel 1 I gang med C#

Vi skal starte med at oprette et meget simpelt program, så du kan se hvad der ligger bag et C# projekt. Når du er færdig med dette kapitel vil du have lært:

- Hvordan du opretter et nyt projekt
- Hvad Solution Explorer er
- De mange filer der danner et C# projekt
- Hvordan du gemmer dit arbejde
- Hvordan du kører et program
- Betydningen af udtrykket **Main**

Det første simple program vi skal lave kaldes en **Console Application**. Vi skal ikke arbejde særlig meget mere med denne type af applikation, da dette notesæt koncentrerer sig om Windows applikationer. Lad os starte.

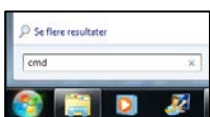
Start Visual C# Express. Når du starter C# (udtales C sharp) ser du et skærmbillede som vist nedenfor:



Når du ser dette program for første gang, kan det godt virke meget komplekst og frygtindgydende. Og ens første indskydelse er det kommer man aldrig til at finde rundt i. Men vær ikke bekymret efter nogle få lektioner vil du begynde at blive fortrolig med programmet, og du vil ikke være nervøs længere!

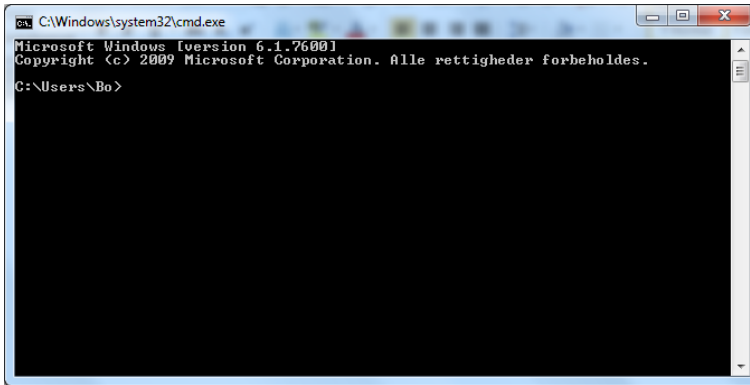
En simpel konsol applikation

En konsol applikation er en der ligner et DOS vindue. Hvis du ikke ved hvad det er, kan du klikke på Start-menuen og skriv CMD i søgefeltet i bunden af startmenuen:



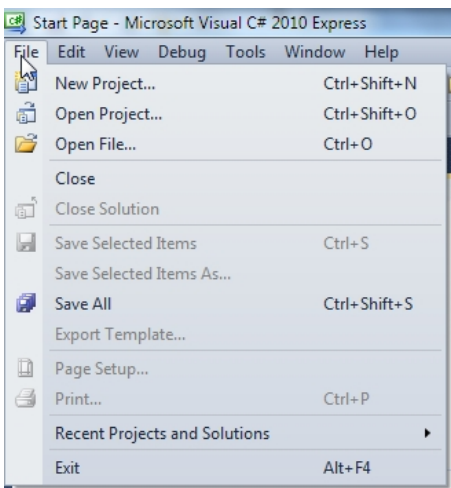
Klik på cmd.exe for at se konsollen.

Du vil nu se en sort skræm som denne:

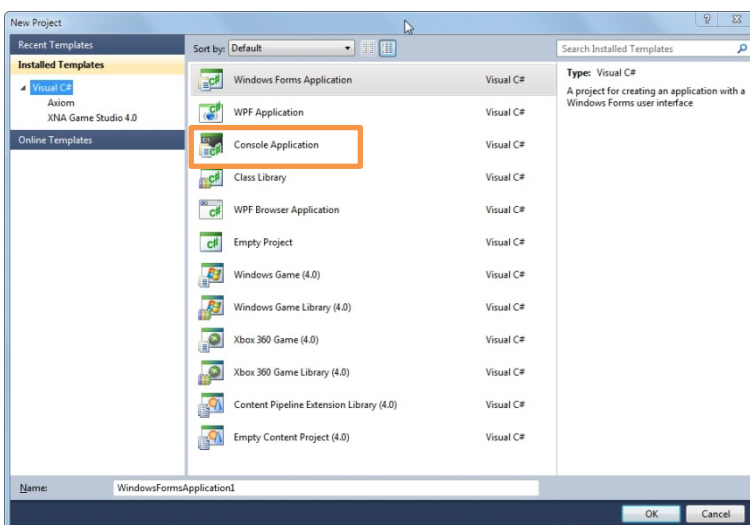


Det er et vindue som dette du vil se, når vi er færdige med vores konsol applikation. Når du laver en Windows formular, er der en masse kode du skal vende dig til. Konsol applikationen er relativ simpel at starte med, og du får set hvilken del af programmet der er vigtigst.

Med Visual C# Express startet klikker du på **File** og vælger **New project**.

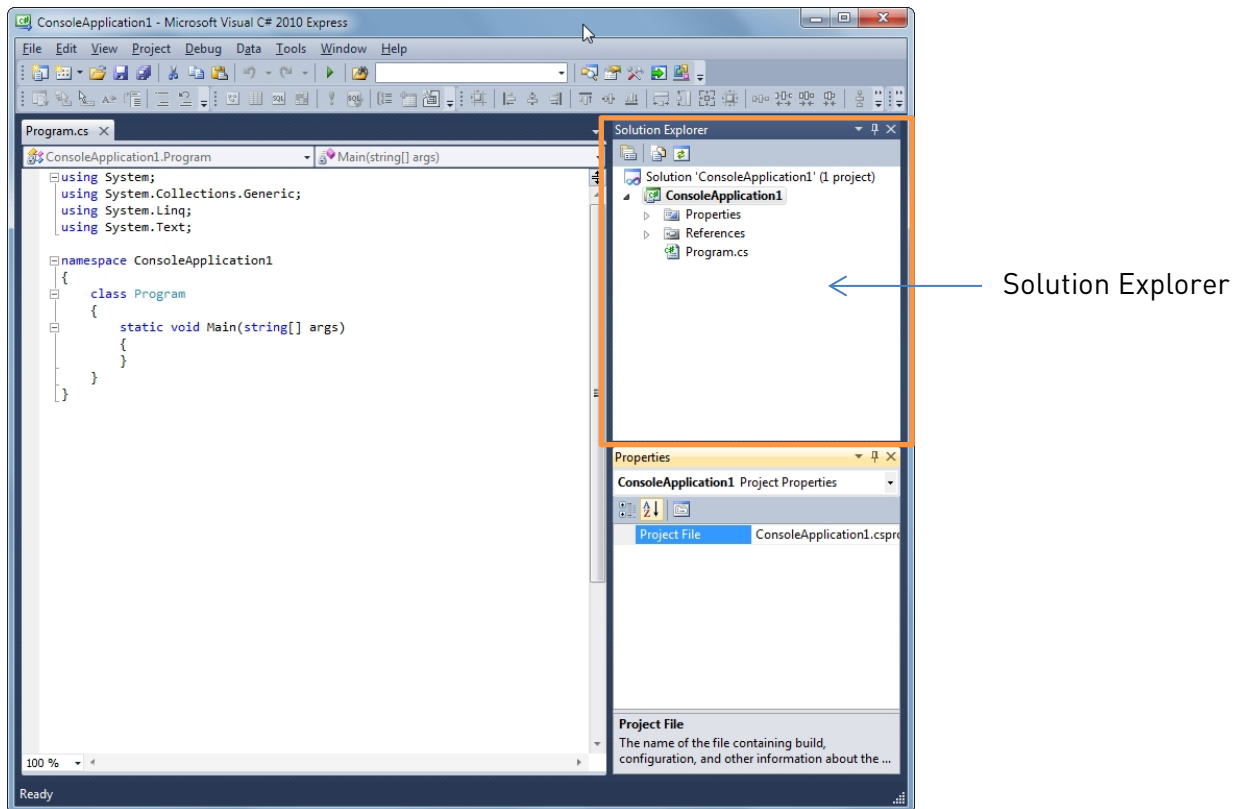


Når du klikker på **New Project** ser du følgende dialogboks:



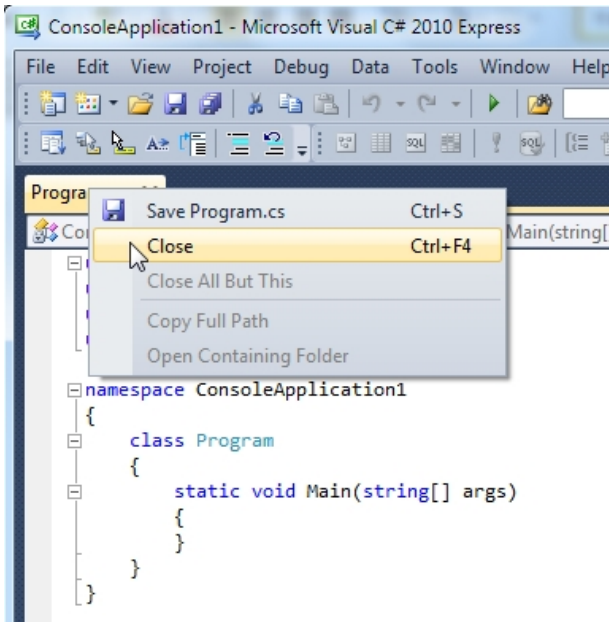
Her vælger du den projekttype du vil arbejde med. Hvis du har Express udgaven af C# er udvalget en kende mindre end den store udgave. I resten af notesættet vil vi arbejde med Windows Applications. Men nu starter vi med at vælge **Console Application**. Klik OK.

Når du klikker OK vil et nyt Console Application projekt starte op til dig. Du vil se lidt koder:



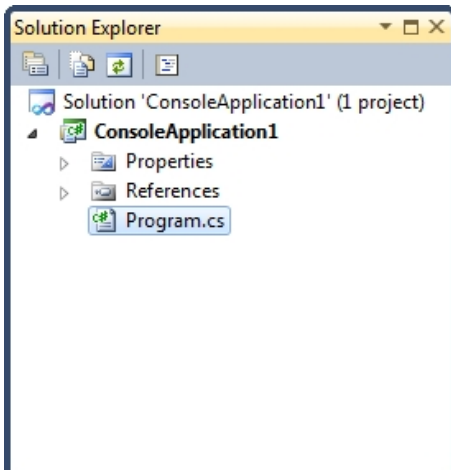
Du vil blandt andet se lidt koder. I den højre side af vinduet ser du området **Solution Explorer**. Her ser du alle de filer der tilhører dit projekt. (Hvis du ikke ser Solution Explorer klikker du på **View** og **Solution Explorer**.)

Koden i sig selv ser noget kompliceret ud, hvis du ikke er vant til programmering. Det kommer vi til straks. Klik nu på fanen Program.cs i toppen og vælg Close i menuen der vises:



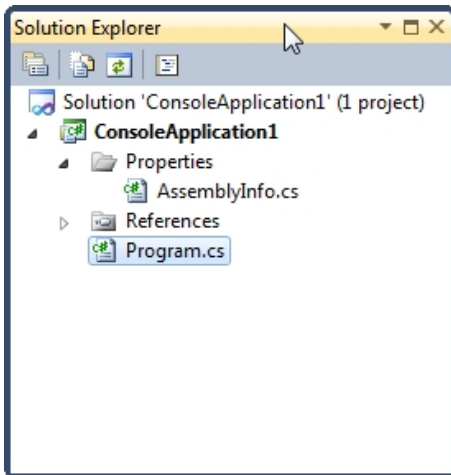
Eller du kan klikke på X i højre side.

Dobbeltklik nu på filen Program.cs i **Solution Explorer**:



Når du dobbeltklikker på Program.cs dukker koden op igen. Det viser at den kode du ser, er programmet, der bliver afviklet når du starter din applikation.

Dobbeltklik nu på trekantsymbolet ved siden af Properties i Solution Explorer. Du ser følgende:

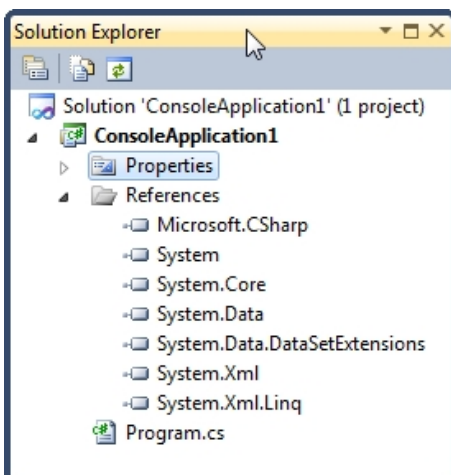


Filen AssemblyInfo.cs indeholder informationer om dit program. Dobbeltklik på filen og kig på koden. Her er vist lidt af koden:

```
[assembly: AssemblyTitle("ConsoleApplication1")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("ConsoleApplication1")]
[assembly: AssemblyCopyright("Copyright © 2014")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture(")]
```

Koden vist med rødt kan du ændre på. Du kan tilføje en titel, beskrivelse, copyright, varemærke etc.

Luk filen AssemblyInfo.cs. Klik på plussymbolet ved siden af References:

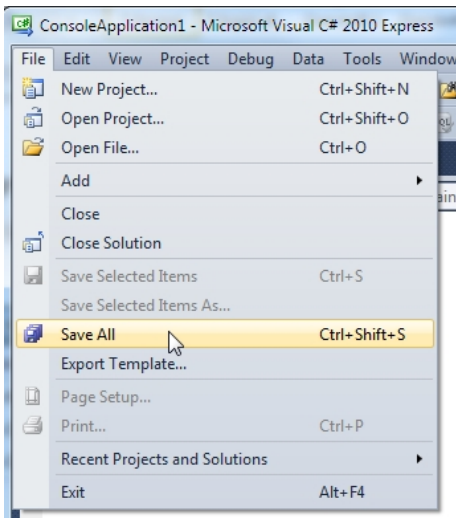


Disse er referencer til indbygget dele i C#. Senere vil du se hvordan man tilføjer sine egne filer til denne sektion.

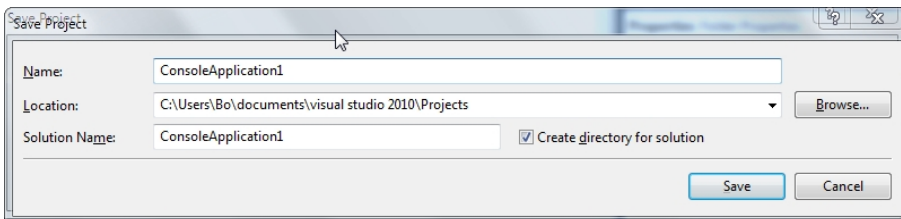
Før vi tilføjer lidt kode til vores projekt skal vi gemme det først. Det gør vi nedenfor.

Gem dit arbejde i C#

Når du gemmer dit arbejde vil C# automatisk oprette nogle få mapper og filer til dig. Klik på **File** og **Save All**:



Når du klikker på Save All ser du følgende dialogboks:



Du kan nu indtaste et navn til dit projekt. Standardnavnet er **ConsoleApplication1**.

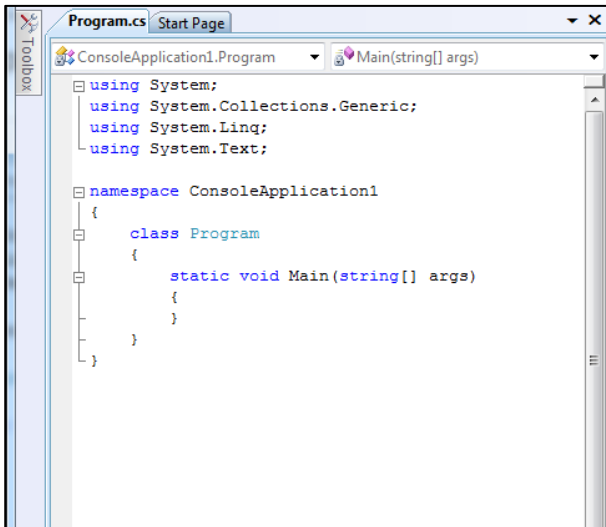
Opret eventuelt en mappe i din dokumentmappe og gem dine projekter heri.

Før du klikker på knappen Save skal du være sikker på at der sat et hak i feltet "Create directory for solution". Klik på **Save**.

Se efter i din dokumentmappe om du kan finde dit projekt.

Din første linje med kode i C#

Vi skal nu skrive lidt kode, der kan skrive noget tekst på skærmen. Her er koden som Visual C# har forberedt til dig da du oprettede din Console Application:



Til at begynde med skal du ignorere de første linjer der starter med **using** (dem tager vi os af senere). De refererer til indbygget koder. Linjen **namespace** inkluderer navnet på din applikation. Et namespace er en måde at gruppere relaterede koder på. Men igen skal du ikke spekulere på udtrykket namespace, da du vil lære om det senere.

Det som er vigtigst nu er ordet **class**. Alle dine koder vil blive skrevet i classes (klasser). Denne kaldes **Program** (du kan kalde dem hvad du vil, så længe C # har ikke har reserveret ordet). Men tænk på en class som en del af koden, som du tildeler et navn.

I **class Program** ser du følgende kode:

```
static void Main(string[] args)
{
}
```

Denne stump kode kaldes en **Method**. Navnet på denne method er **Main**. Når du afvikler programmet ser C# efter en Method der hedder Main. C# benytter Main Method som startpunkt for dit program. Den afvikler herefter den kode, der findes mellem de to krøllede parenteser. De blå ord der står ovenfor er alle specialord – nøgleord. Dem lærer du mere om senere.

Placer din markør efter den første krøllede parentes. Tryk derefter på Enter tasten:

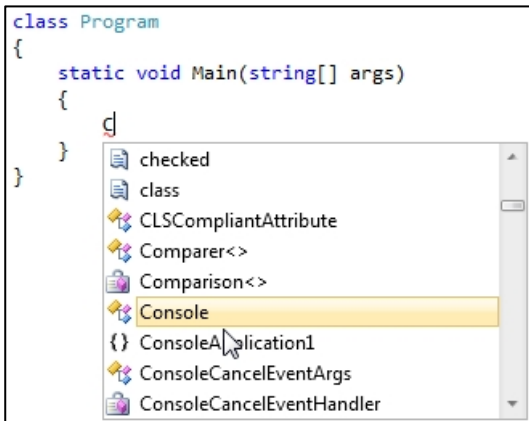
```
class Program
{
    static void Main(string[] args)
    {
        |
    }
}
```

Markøren rykker automatisk ind og du er klar til at skrive kode. Bemærk hvor de krøllede parenteser er placeret i koden ovenfor. Du har et par der hører til **class Program**, og et par til Main method. Hvis du mangler en vil du få en fejlmeddelelse.

Den linje med kode vi skal skrive kommer her (men vent med at skrive den!):

```
Console.WriteLine("Hello C Sharp!");
```

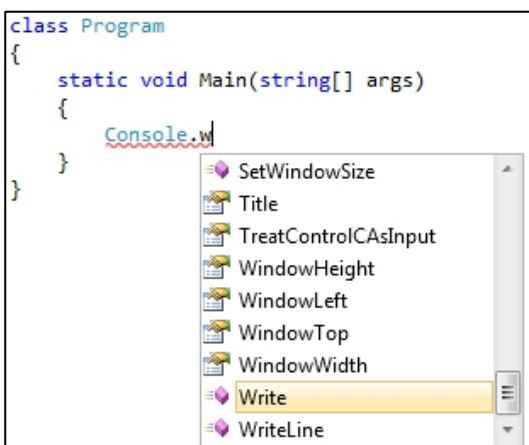
Først skriver du bogstavet "C". Du vil nu se en popup menu. Denne popup menu kaldes en **IntelliSense** menu. Den prøver at gætte på hvad du vil skrive, og tillader dig hurtigt at indsætte emnet fra listen. Det burde se ud som vist nedenfor når du har tastet et stort "C":



Ikonet ved siden af ordet **Console** på listen betyder at det er en Class. Tryk på Enter tasten. Ordet vil nu blive tilføjet din kode:


```
class Program
{
    static void Main(string[] args)
    {
        Console
    }
}
```

Tast nu et punktum (.) lige efter ordet Console. IntelliSense menuen dukker op igen:

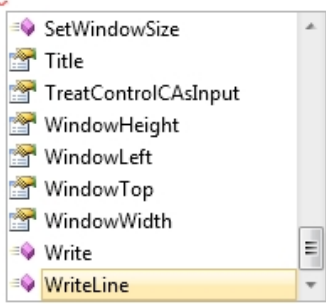


Du kan bruge piletasterne på tastaturet til at bevæge dig op og ned i listen. Hvis du skriver **Write** og derefter bogstavet **L** for **Line** vil IntelliSense automatisk flytte sig ned og vælge det for dig:


```
class Program
{
    static void Main(string[] args)
    {
        Console.Writel
    }
}
```



```
class Program
{
    static void Main(string[] args)
    {
        Console.
    }
}
```

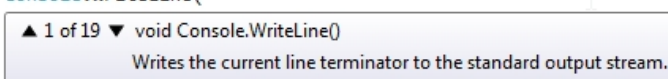


Tryk på Enter tasten for at tilføje ordet **WriteLine** til din kode:

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine
    }
}
```

Skriv nu en blød parentes. Straks du har skrevet den bløde parentes, vil du se dette:

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(
    }
}
```



WriteLine er en anden metode (En metode er bare noget kode som udfører et bestemt job). Den gule boks fortæller dig at der er 19 forskellige versioner af denne metode. Du kan klikke på de små pile og bevæge dig op og ned i listen. Du indtaster dog følgende:

"Hello C Sharp!"

Glem ikke anførselstegn i starten og slutningen af teksten. Du fortæller C# at du ønsker at arbejde med tekst. Din kode vil nu se ud på følgende måde:

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello C Sharp!");
    }
}
```

Tast nu en parentes:

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello C Sharp!");
    }
}
```

Bemærk den røde understregning i slutningen af linjen. Det er programmets måde at fortælle dig at du mangler noget.

Det du mangler er et semikolon. Alle færdiggjorte linjer i C# skal afsluttes med et semikolon. Hvis du mangler et får du en fejlmeddelelse. Tast et semikolon i slutningen ved den røde understregning. Din kode skulle nu gerne se ud som følgende:

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello C Sharp!");
    }
}
```

Bemærk de forskellige farver. Visual C# farver koderne i forskellige farver. Den røde farve mellem anførselstegn betyder at du vil skrive noget tekst. Den grønne farve betyder at det er en Class. Den blå farve er ord der er reserveret af C#.

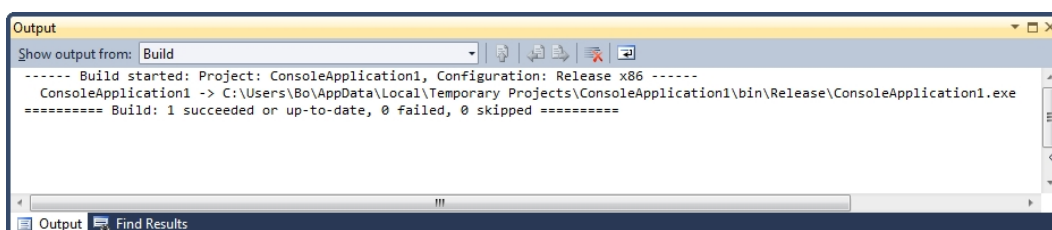
Hvis du ønsker at ændre på disse farver kan du gøre det via **Tools** og **Options**. Under **Environment** klikker du på **Fonts and Colors**.

Nu er det på tider vi prøver at kompilere og afvikler programmet!

Hvordan afvikler du et program i C#

Du kan teste dit program på mange forskellige måder. Først skal det dog kompileres. Det er her hvor alt tjekkes og undersøges for fejl. Prøv dette:

1. Fra **View** vælger du **Output**. Du ser nu et vindue i bunden af C#.
2. Fra **Build** vælger du **Build Solution**.
3. Du ser nu følgende rapport:



Den sidste linje:

Build: 1 succeeded or up-to-date, 0 failed, 0 skipped

Den fortæller dig at alt er i orden.

Prøv nu følgende:


1. Slet semikolonet fra slutningen af din linje med kode.
2. Klik på **Build** og **Build Solution** igen.
3. Undersøg output vinduet.

Nu ser du følgende to linjer i slutningen af din rapport:

Compile complete -- 1 errors, 0 warnings

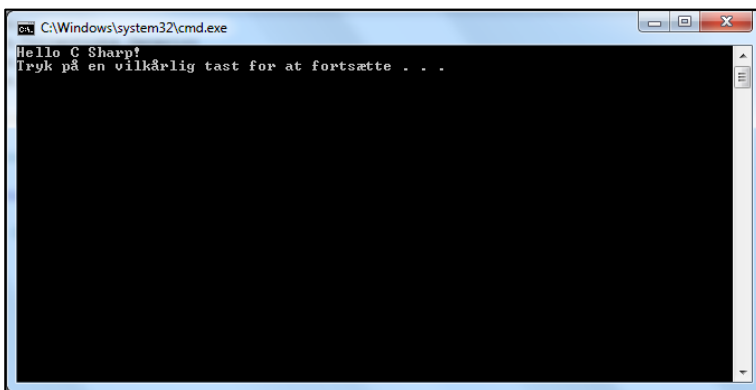
Build: 0 succeeded or up-to-date, 1 failed, 0 skipped

Det fortæller dig at den ikke kunne compilere en løsning til dig. Der var nemlig en fejl.

Sæt semikolonet på plads igen. Klik nu på knappen **Debug**  i toppen af C# vinduet. Du kan også vælge menuen **Debug** og **Start Debug**.

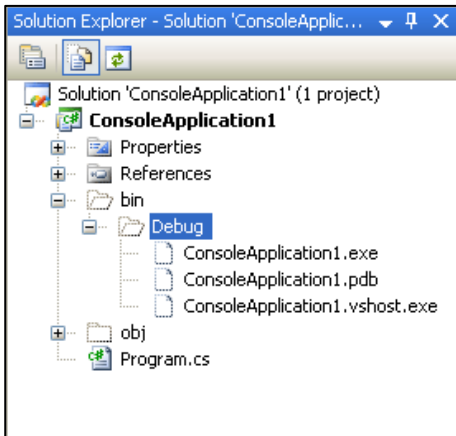
Du ser nu et sort DOS vindue dukke op kortvarigt for så at forsvinde igen. Dit program blev afviklet med succes!

For at se hvad der rent faktisk sker, klikker du på **Debug** og **Start Without Debugging**. Du ser nu følgende:



```
C:\Windows\system32\cmd.exe
Hello C Sharp!
Tryk på en vilkårlig tast for at fortsætte . . .
```

Det er så dit program! Tag et kig på Solution Explorer i højre side. Når du har kompileret dit projekt vil du se yderligere filer under **Debug**:

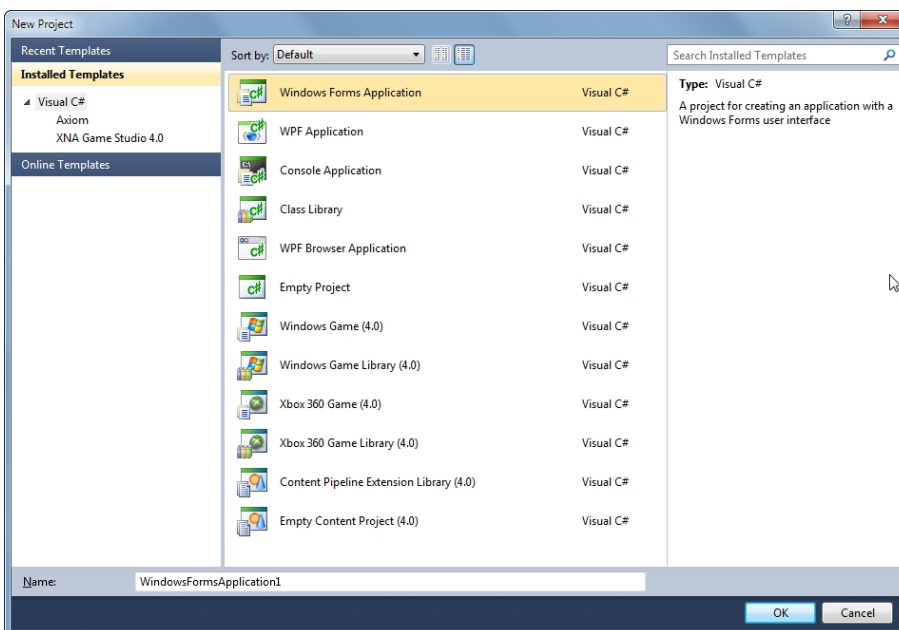


Din første C# Windows formular

Fra nu af vil vi koncentreres os om Windows Applications, i stedet for Console Applications. Med Windows Applications får vi mulighed for at lave noget vi kalder formular. Formularen er som udgangspunkt tom. Du tilføjer derefter forskellige kontrolelementer til din formular. Det kan være ting som for eksempel knapper, tekstbokse, afkrydsningsbokse, menuer osv.

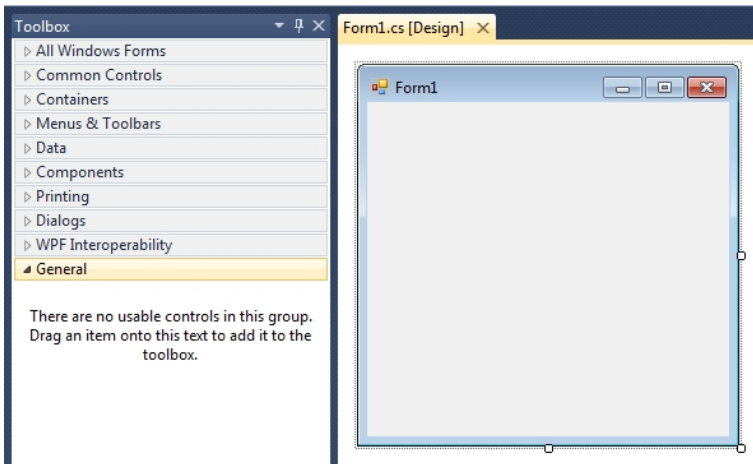
Hvis du stadig har konsol applikation åben fra tidligere klikker du på **File** og **Close Solution**.

Når du skal lave dit første projekt med en Windows formular klikker du på **File** og **New Project**. Du ser nu dialogboksen **New Project**:



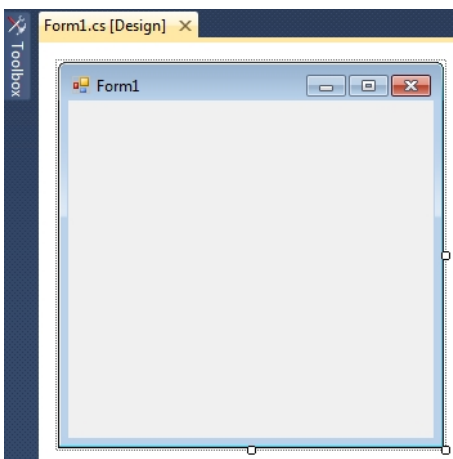
Marker **Windows Forms Application**. Behold navnet **WindowsFormsApplication1** og klik OK.

Når du klikker OK dukker et nyt Windows Applikation projekt frem:



Den store forskel fra Console Application du lavede i forgående øvelse er den tomme formular i hovedvinduet. Bemærk også Toolboxen i venstre side. Vi skal tilføje kontrolelementer fra Toolboxen til din tomme formular Form1, du kan se på billedet ovenfor.

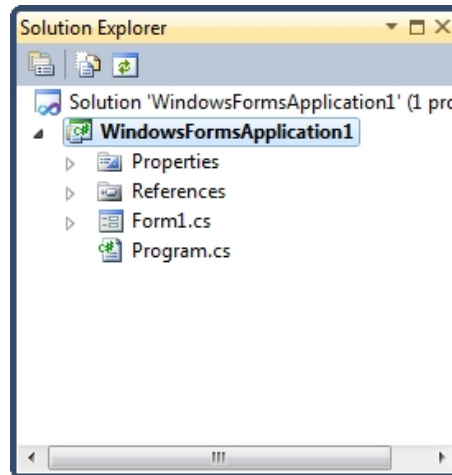
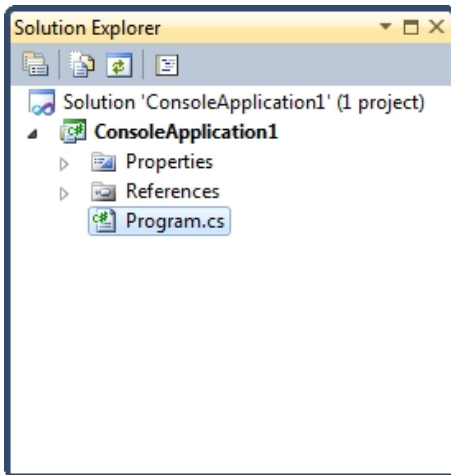
Hvis du ikke kan se Toolbox, men kun en fane som vist nedenfor:



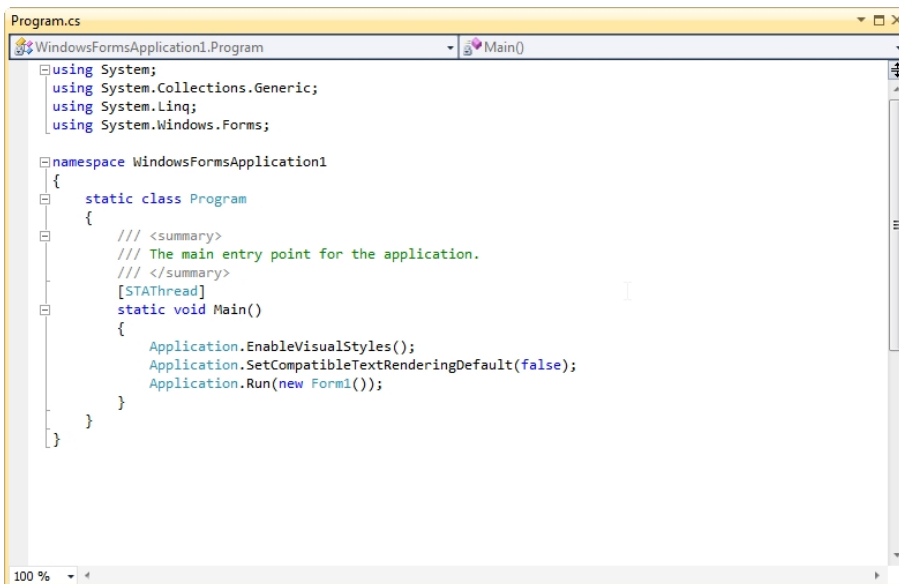
Hvis din skærm ser ud som ovenfor skal du bevæge musen over fanen Toolbox. Den vil udvide sig til den første udgave. Hvis du ønsker at have den fremme permanent klikker du på den lille knapnål:



Bemærk Solution Explorer på din højre side. (Hvis du ikke kan se Solution Explorer skal du klikke View.) Hvis du sammenligner den med den Solution Explorer du havde da du lavede din Console Application, vil du opdage at den eneste forskel er Form.



Vi har stadig Properties, Reference og filen Program.cs. Prøv at dobbeltklik på Program.cs filen for at åbne den. Nu vil du se noget kode, der forhåbentligt er bekendt:



Og her er koden fra Console Application:

```

using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}

```

De har begge de samme Using linjer, et namespace, en class der hedder Program og en **Main** Method.

Main Method er startpunktet for dit program. Koden mellem de krøllede parenteser vil blive eksekveret når programme starter. Den sidste linje i koden for **WindowsApplication1** i koden vist ovenfor er den linje der kører Form1 når applikationen starter.

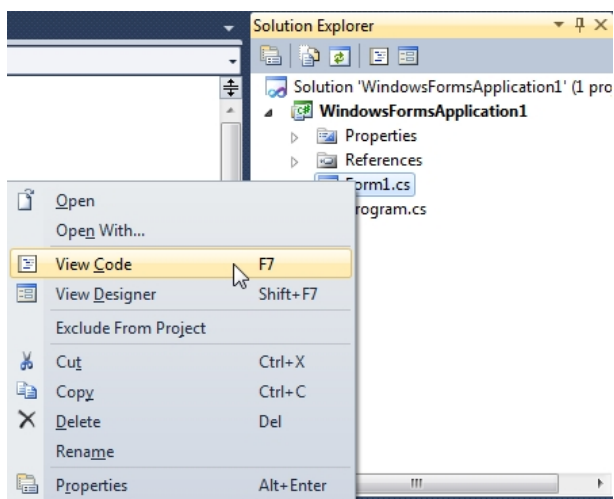
Du kan foretage forskellige ting her. Hvis du for eksempel havde et program, der forbinder sig til en server. Hvis det finder en forbindelse henter den en række informationer fra en database. I Main Method kan du undersøge om forbindelsen til serveren er OK. Hvis den ikke er den vises en anden formular, hvis den er OK vises den første formular.

Nu skal du ikke være nervøs for al den kode. Det du skal huske på nu er at den method der kaldes **Main** starter dit program. Og **Program.cs** i Solution Explorer i den højre side er hvor koden til Main høre hjemme.

Vi vil dog ikke skrive noget kode i filen Program.cs, så vi kan godt lukke filen. Se i toppen af vinduet. Her vil du se en række faner:

Højre klik på fanen Program.cs og vælg **Close**. Du returnere nu til din formular (Du har eventuelt også en fane der hedder Start. Den kan du godt lukke hvis du vil).

For at se det vindue hvor du vil komme til at skrive det meste af din kode skal du højre klikke på **Form1.cs** i Solution Explorer:

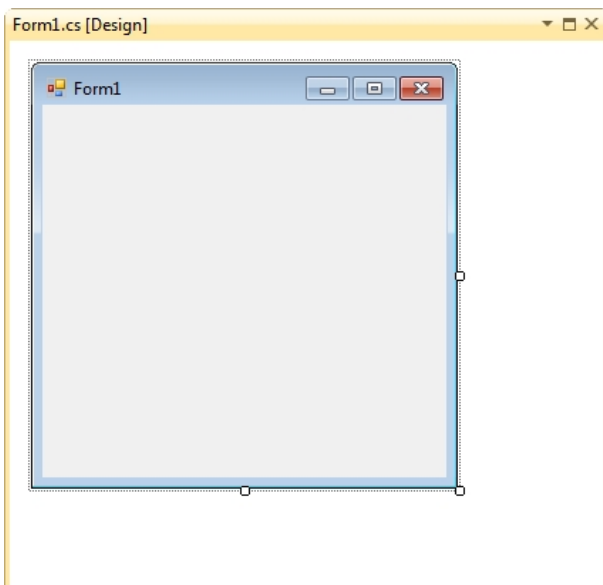


I menuen ser du mulighederne **View Code** og **View Designer**. Designer er formularen du kan se nu. Klik på **View Code** for at se det følgende vindue (du kan også klikke F7):

```
Form1.cs
WindowsFormsApplication1.Form1
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

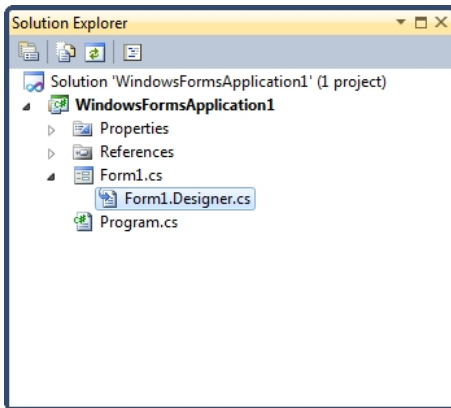
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

Dette er koden der hører til formularen. Dette er selve formularen:

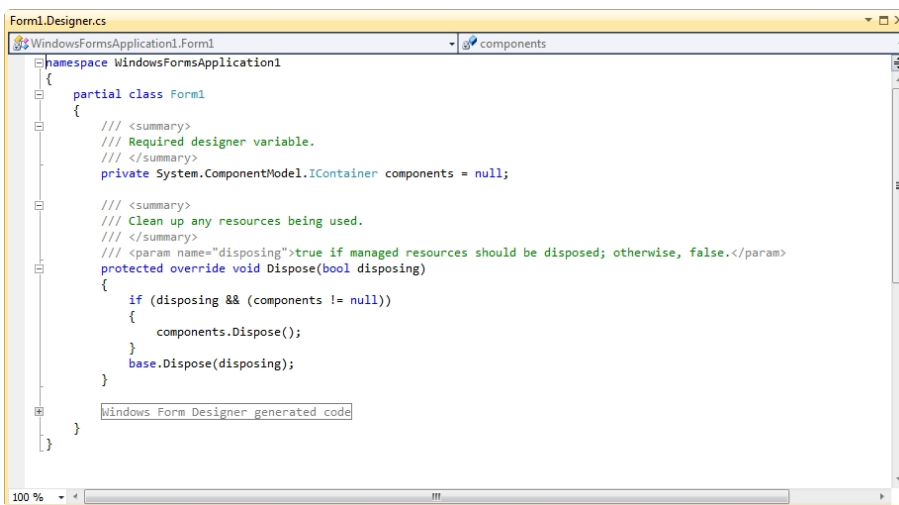


I koden kan du se flere **using** direktiver end før. Dem skal du dog ikke bekymre dig om lige nu. De betyder bare at vi bruger (using) noget kode, der allerede er skrevet.

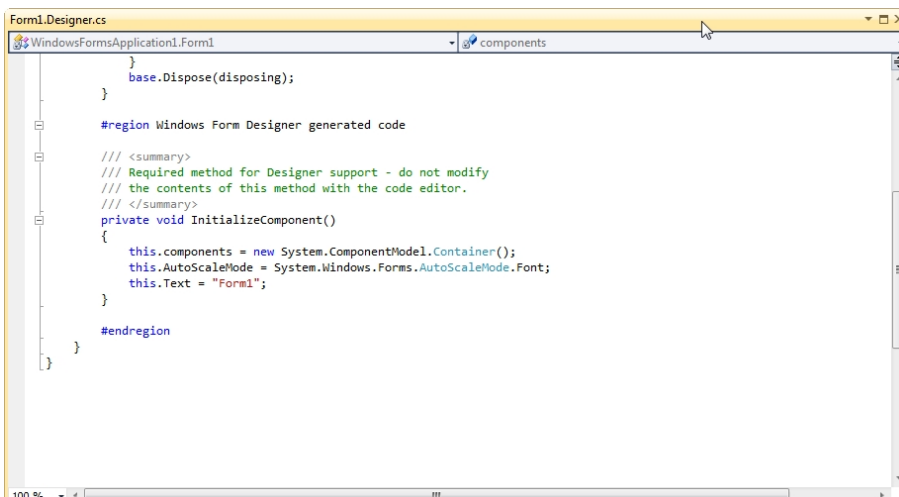
I koden ser vi også at der står partial class Form1. Det betyder at en del af koden er skjult. For at se resten af koden (som vi dog ikke har brug for at ændre), skal du klikke på plus symbolet ved siden af Form1.cs i Solution Explorer:



Dobbeltklik nu på **Form1.Designer.cs**. Du vil se følgende kode:



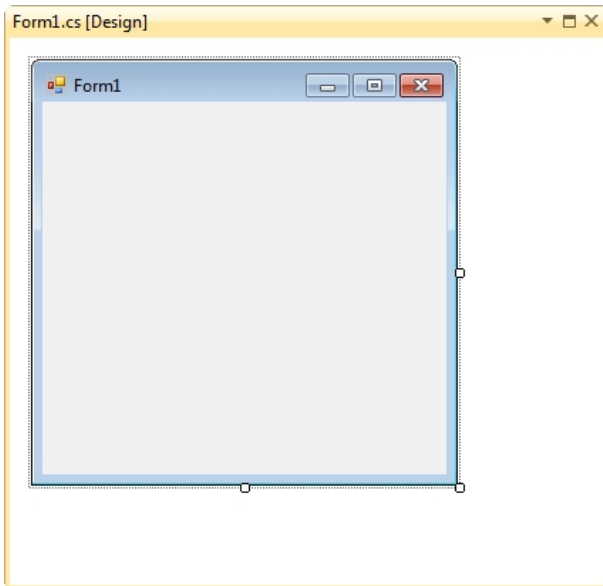
Igen ser du partial class Form1, som er den resterende del af koden. Klik på plus symbolet ved siden af **Windows Form Designer generated code**. Du vil nu se følgende:



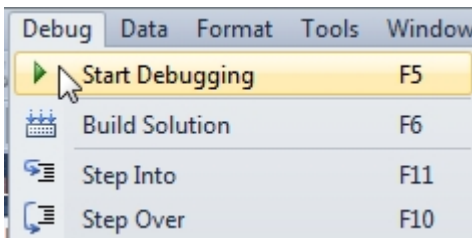
InitializeComponent er kode (en Method) der automatisk genereres når du laver et nyt Windows Application Project. Efterhåndenden som du tilføjer knapper og tekstboks til din formular, vil du se mere kode i denne sektion.

Du skal dog ikke foretage dig noget i dette vindue endnu og du kan derfor lukke fanen **Form1.Designer.cs**.

Klik tilbage på fanen **Form1.cs** som du ser øverst oppe. Hvis fanen ikke vises kan du højre klikke på Form1.cs i Solution Explorer i højre side. Fra menuen kan du vælge **View Designer**. Her er det du gerne skulle se:



Det er i Designer visningen vi vil tilføje ting som knapper og tekstbokse til vores formular. Men du kan godt afvikle programmet som det er. Fra **Debug** vælger du **Start Debugging** (eller du kan taste på F5.):



Når du vælger Start Debugging vil Visual C# compilere programmet først, og derefter afvikle det. Hvis det ikke kan afvikle programmet vil du få en fejlmeddelelse.

Men du burde kunne se din formular blive afviklet oven på Visual Studio. Det har sit eget røde X og dets eget minimer og maksimer knap. Klik på den røde X knap for at lukke programmet, og vende tilbage til Visual C# Express.

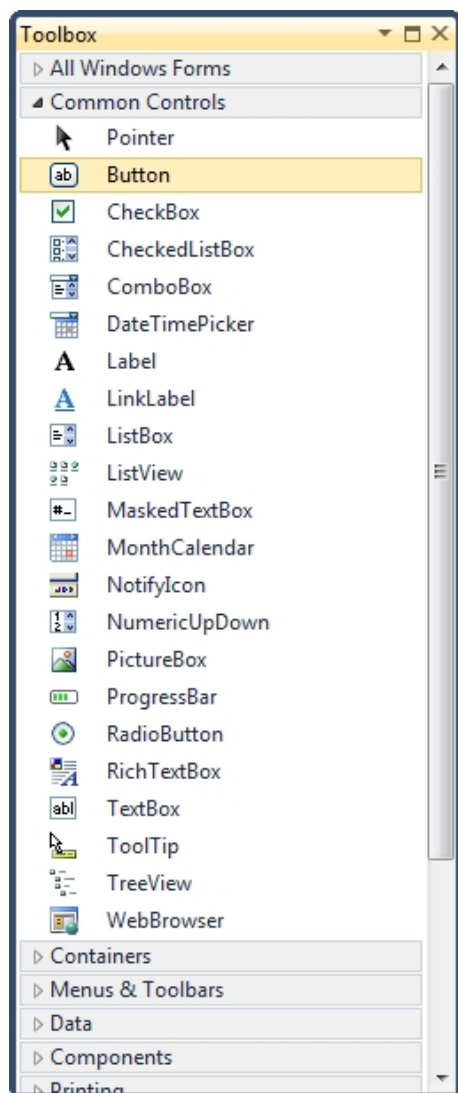
Fra nu af vil vi bruge udtrykket *Kør dit program*, som betyder at du enten taster F5 eller klikker på **Debug** og **Star Debugging**.

Nå, nu er det tid til at tilføje ting til din formular og skrive en smule kode!

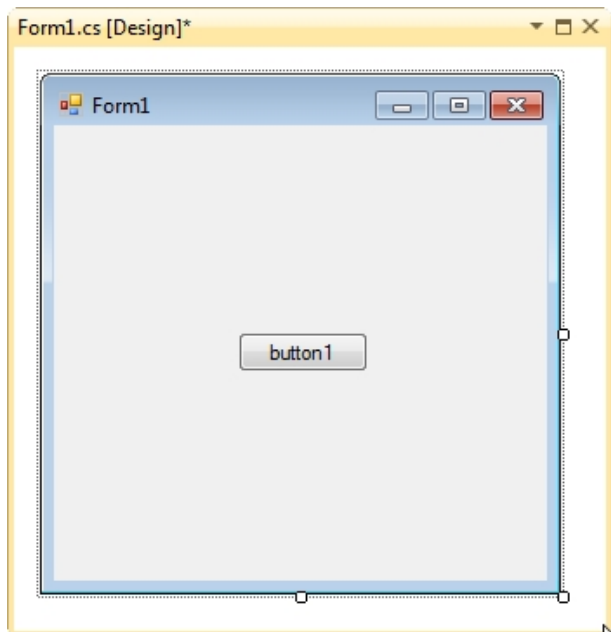
Tilføj kontrolelementer til en blank C# formular

Det første vi skal er at tilføje en knap til vores tomme formular. Vi skal derefter skrive en enkelt line kode, så vi kan se hvordan det arbejder sammen.

Hvis du skal tilføje en knap til din formular kan du bruge Toolbox i den venstre side. Flyt din mus hen til Toolbox og klik på plus tegnet ved siden af **Common Controls**. Du ser nu følgende liste med elementer du kan tilføje til din formular:



Klik på elementet **Button** under Common Controls overskriften. Dette vil vælge elementet. Klik nu en enkelt gang i din formular. En knap vil blive sat ind i din formular og ser ud på følgende måde:



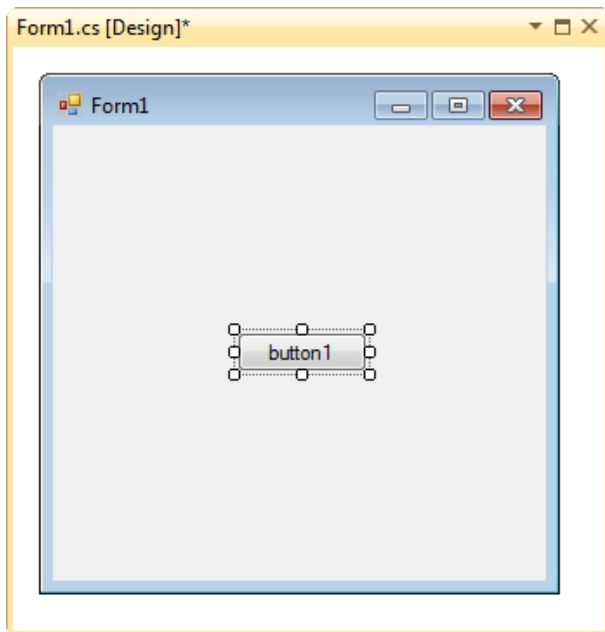
(Du kan også trække knappe ind i din formular med træk og slip metoden, og derefter ændre størrelsen på den.)

En knap er et objekt du gerne vil have brugeren til at klikke på. Når de gør det vil den kode du skrive blive afviklet. Teksten på knappen "button1" er en standard tekst, som kan ændres. Du kan skrive hvad du har lyst til, men det er en fordel at skrive noget fornuftigt i forhold til dine brugere, som for eksempel "Åbn en tekstfil" eller "Beregn nu".

Vi vil gerne have vist en simpel dialogboks, når der klikkes på knappen. Vi er derfor nød til at tilføje noget tekst til knappen. Du gør dette ved at ændre det man kalder properties (egenskaber). Det er ganske nemt og vi ser på det i næste del.

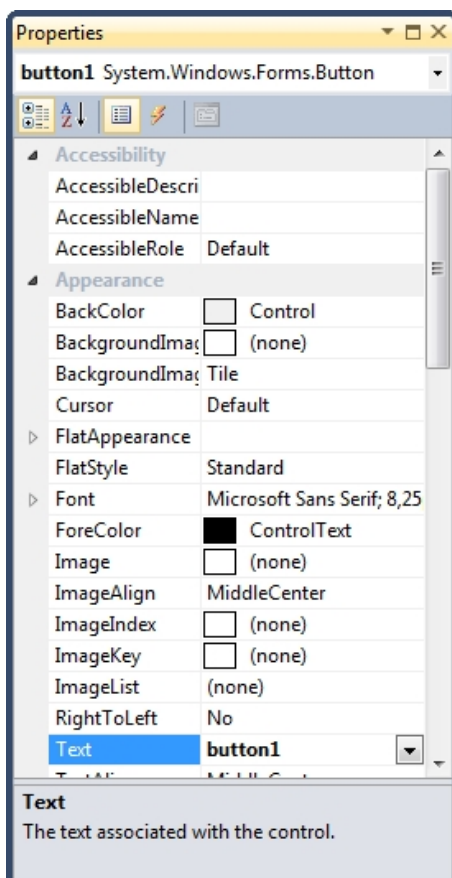
Kontrollementets egenskaber i C#

De kontrollementer du tilføjer til din formular har noget man kalder **Properties** (egenskaber). Et kontrollements egenskaber er ting som højde, bredde, navn, tekst og en lang række andre ting. For at se hvilke egenskaber der er til rådighed for en knap, skal du først markere knappen, som vist nedenfor:

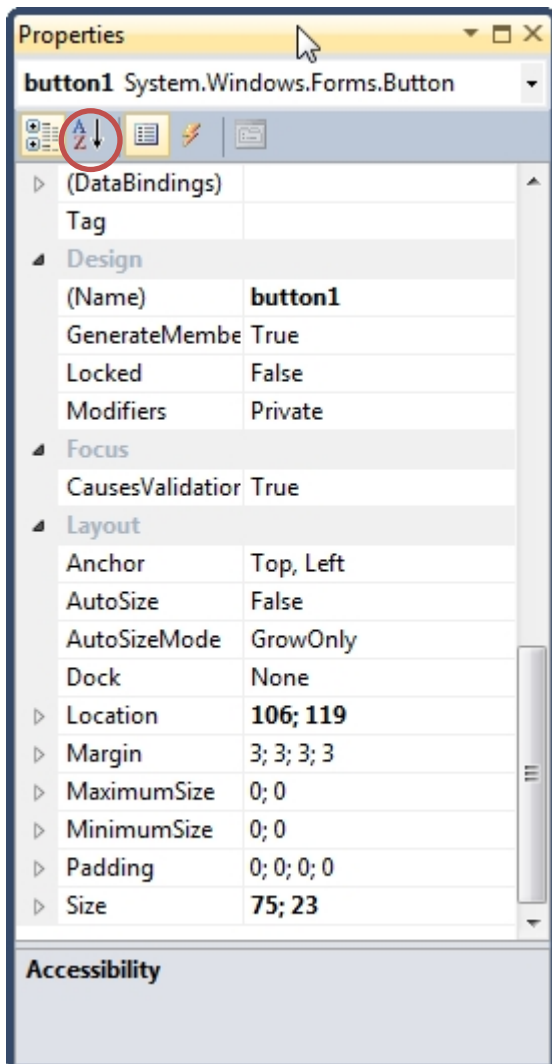


Hvis et kontrolelement er valgt vil det have hvide håndtag rundt omkring. Hvis knappen ikke er valgt klikker du bare på den en enkelt gang.

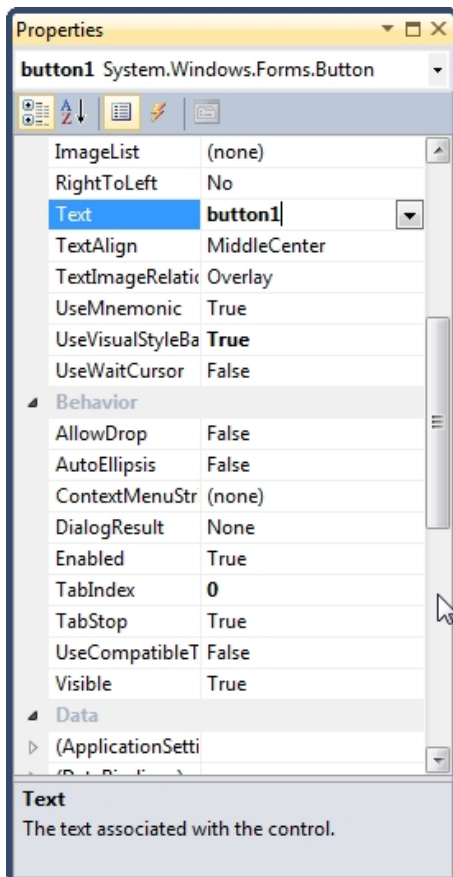
Se nu nederst i højre af Visual C#, lige nedenfor Solution Explorer. Her ser du vinduet Properties. (Hvis du ikke ser det vælger du **View**.):



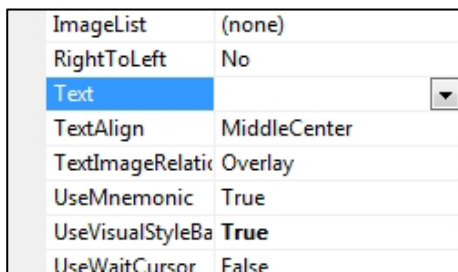
For at se Properties listen i alfabetisk orden, kan du klikke på AZ symbolet i toppen, her indtegnet med en rød cirkel i billedet nedenfor:



Som du kan se er der en lang række betingelse hørende til en knap. Rul ned på listen og find **Text** egenskaben:



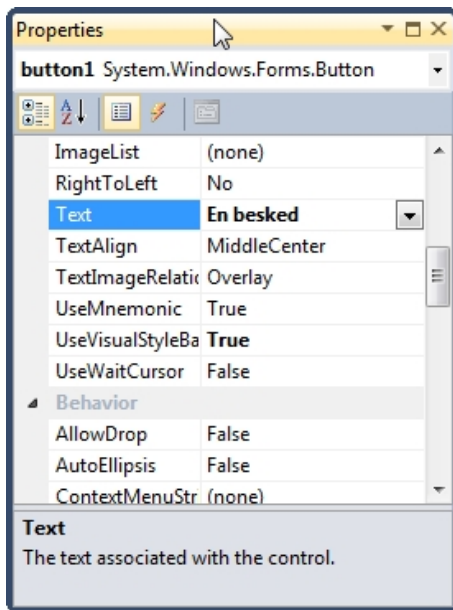
Text egenskaben, angiver som navnet også siger, den tekst du vil have stående på knappen. Lige nu står der button1. Klik indenfor i feltet og slet teksten.



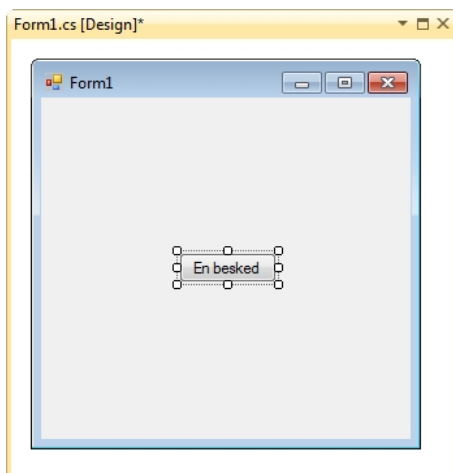
Indtast nu i stedet for:

En besked

Nu vil dit vindue se sådan ud:

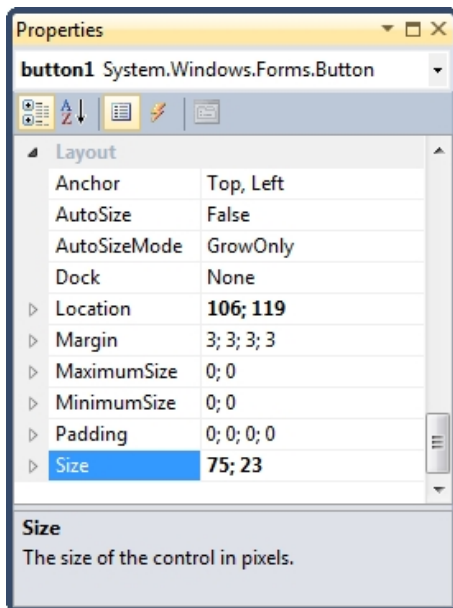


Tryk på din Enter tast. Se nu på din formular. Teksten på knappen er skiftet ud:

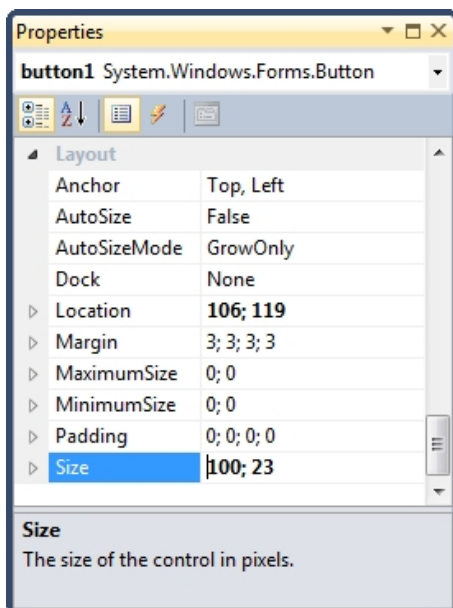


Der er lidt flere egenskaber vi kan ændre før vi begynder at kode.

Find feltet **Size** i Properties vinduet:

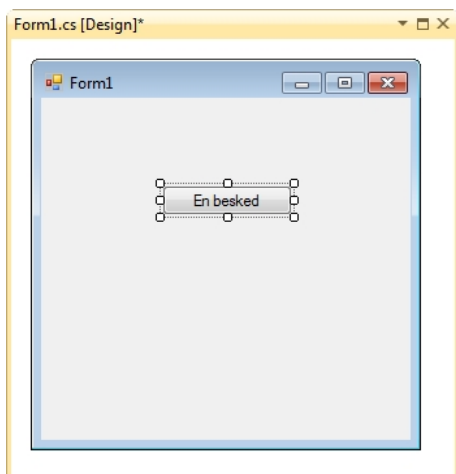


Det første tal, 75, er bredden på knappen. Det andet tal, 23, er højden på knappen. De to tal er adskilt med et semikolon¹. Skift tallene til 100;30:



Tryk igen på Enter tasten og din knapskifter størrelse:

¹ Afhængig af sprogopsætningen kan det også være et komma.



Du kan flytte din knap rundt i formularen med træk og slip metoden.

Prøv dette

Du kan også flytte din knap rundt ved at ændre på egenskaben **Location**. Brug Properties vinduet til at ændre placeringen af din knap i formularen til 100;50.

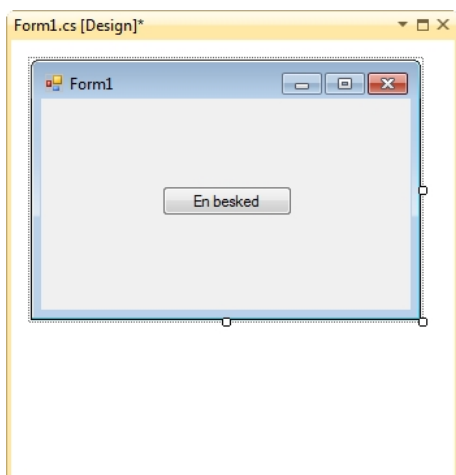
Prøv dette

Formularen har også en række egenskaber. Klik i selve formularen. Properties vinduet hørende til formularen vises. Skift egenskaben for feltet **Text** til **A First Message**.

Prøv dette

Du kan også ændre på størrelsen på formularen. Skift formularens størrelse **Size** til 300;200.

Efter du har været igennem disse øvelsen, ser din formular ud på følgende måde:

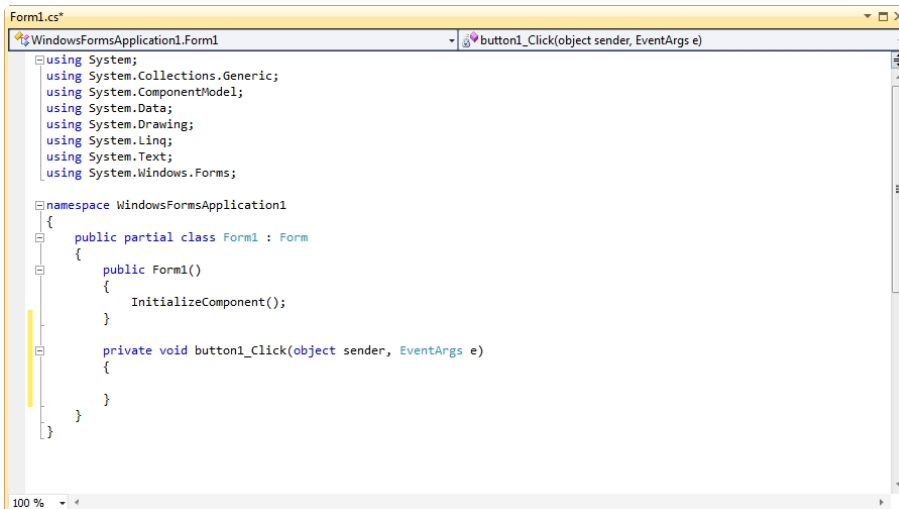


Da du ændrede egenskaben Text, skiftede du teksten der står i titellinjen i vinduet. Du kan skrive hvad du har lyst, men det er en god ide at skrive noget der beskriver formularens indhold.

Vi kan nu gå i gang med noget kodning, og køre formularen så vi kan se hvordan det hele tager sig ud.

Tilføj C# kode til en knap

Det vi skal lave nu er, at vi skal have vist en meddelelsesboks hver gang der klippes på knappen. Vi skal derfor over i kodevinduet. For at se koden bag knappen, skal du dobbeltklikke på knappen i din formular. Når du gør det åbner du kodevinduet, og din markør står og blinker inde i koden for knappen. Det ser ud på følgende måde:



Den eneste forskel fra sidst du så vinduet er at du har fået tilføjet koden bag knappen. Det er denne kode:

```
private void button1_Click(object sender, EventArgs e)
{
}
```

Dette er bare endnu en Method, et stykke kode som udfører en opgave. Navnet på denne Method er **button1_click**. Den hedder button1 fordi det er navnet til knappen. Når du ændre Text, Location og Size egenskaberne for knappen, kunne du også ændre navnet på knappen fra button1 (som er standard navnet) til et hvilket som helst andet navn.

Udtrykket **_Click** efter button1 kaldes en Event (hændelse). Andre typer af hændelser kan for eksempel være MouseDown, LocationChange, TextChange og mange andre. Du lærer mere om hændelser senere.

Efter **_Click**, og mellem et par bløde parenteser har vi:

object sender, EventArgs e

Disse to udtryk kaldes **arguments** (argumenter). Et argument kaldes **sender**, og det andet kaldes **e**. Du vil også lære mere om argumenter senere, så du skal ikke bekymre dig mere om dem lige nu.

Bemærk der er et par krøllede parenteser, hvor vi skal tilføje koden:

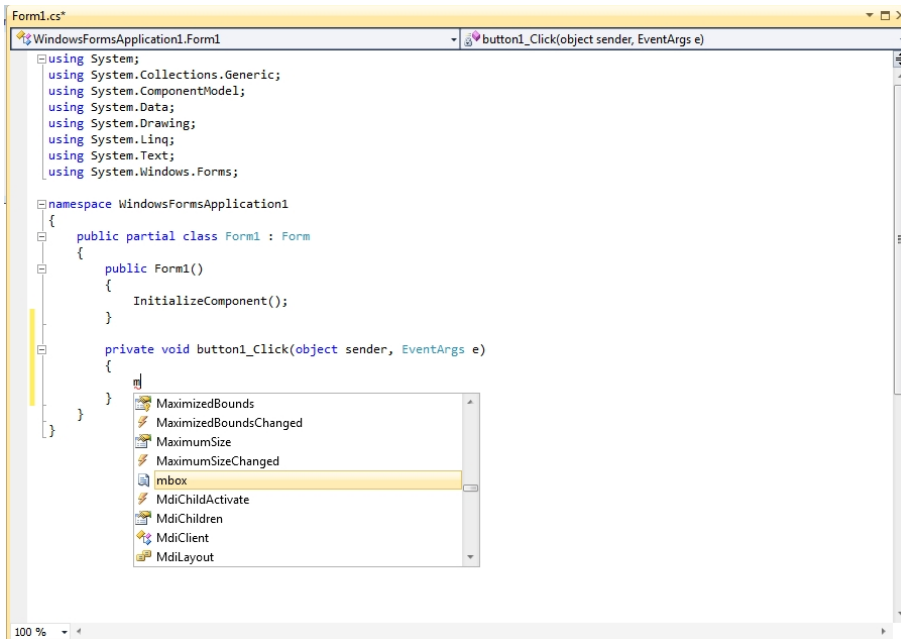
```
private void button1_Click(object sender, EventArgs e)
{
}
```

Når du vil tilføje kode til en knap, skal du skrive koden mellem de krøllede parenteser. Vi vil tilføje en enkelt linje kode.

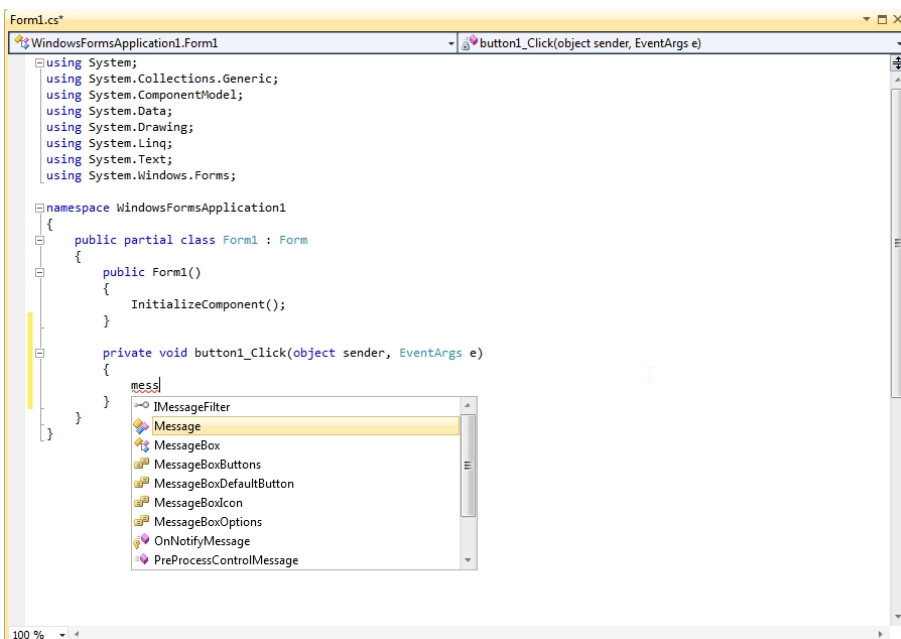
En C# dialogboks

Vi vil gerne have vist en dialogboks med lidt tekst. Det er ganske nemt i C#.

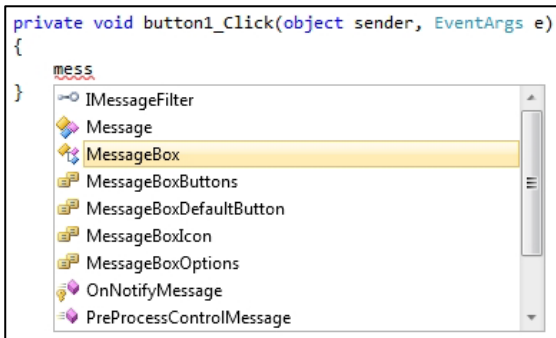
Placer din cursor mellem de krøllede parenteser. Tast nu et stort "M". Du vil nu se IntelliSense dukke op:



Tast nu "ess" efter "M". IntelliSense springer nu ned til:



De eneste muligheder der starter med Mess er alle de muligheder der hedder Messages. Den vi skal bruge er MessageBox. Du kan nu enten skrive det sidst, eller det nemmere flytte ned til MessageBox:



Når du har valgt MessageBox trykker du på Enter (eller dobbeltklikker med musen). Koden vil nu blive tilføjet:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox
}
```

Tast nu et punktum (.) efter "x" i MessageBox. Igen dukker IntelliSense op:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.
}
```

Der er kun tre linjer på listen, og alle Methods (du kan se det er en Methods på grund af det lille ikon foran). Dobbeltklik på **Show**, og det vil blive tilføjet din C# kode:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show
}
```

Fordi Show er en Method skal vi bruge et par bløde parenteser. Teksten du vil have vist dialogboksen skal stå mellem parenteserne. Tast en venstre parentes lige efter "w" i "Show":

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show(
}
```

Straks efter du har tastet en venstre parentes efter "w" ser du de forskellige måder Show metoden kan bruges. Der er i alt 21 forskellige måder. Heldigvis skal du ikke gå igennem dem alle sammen. Tast følgende efter den venstre parentes (Glem ikke anførselstegnet.):

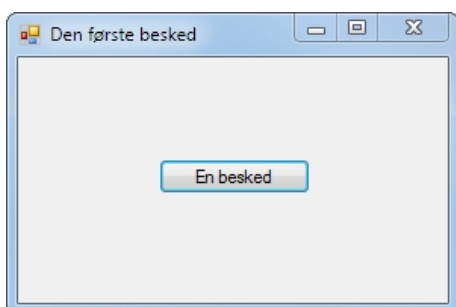
"Min første besked"

Efter det sidste anførselstegn taster du en højre parentes. Afslut linjen med et semikolon (;). Din kode vil nu se ud på denne måde:

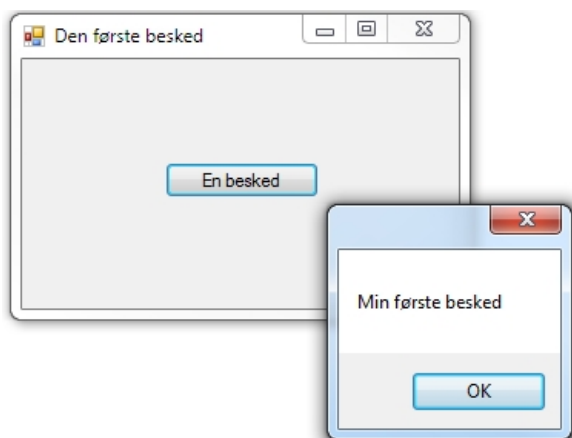
```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Min første besked");
}
```

Teksten med den røde farve er det der vil blive vist i din dialogboks. Prøv den. Gem dit arbejde ved at klikke på **File** og **Save All**. Du vil se den samme dialogboks **Save** som da du arbejdede med din Console Application. Gem dit projekt.

Kør programme ved at vælge **Debug** og **Start Debugging**. Eller tryk F5 på tastaturet. Dit program vil se ud som dette:



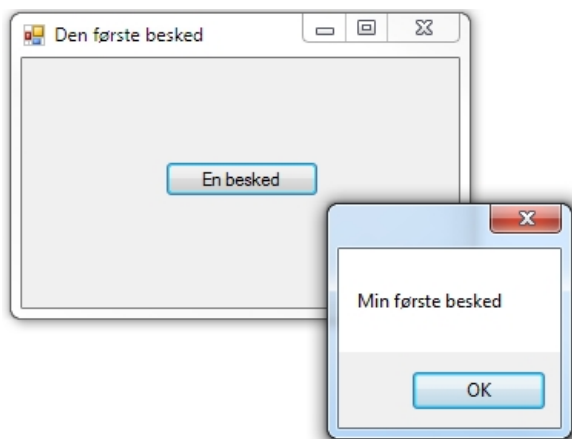
Klik på knappen for at se din dialogboks:



Tillykke! Det er din første besked! I den næste del vil vi undersøge andre ting vi kan foretage med dialogboksen.

Mere om C# dialogboksen

Hvis du ser på dialogboksen vi har lavet i den foregående sektion, vil du bemærke at der ikke står noget i titellinjen – det blå område ved siden af det røde X:



Du kan hurtigt tilføje en titellinje.

Klik OK i dialogboksen. Klik på det røde X i dit program for at afslutte. Du kommer nu tilbage til Visual C#. Vend tilbage til kodevinduet (tryk på F7 på tastaturet, hvis du ikke kan se det).

Placer din cursor efter det sidste anførselstegn i "Min første besked", afmærket med en cirkel i billedet vist nedenfor:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Min første besked");
}
```

Tast nu et komma. Straks du har tastet et komma vil du se en liste af **Show** muligheder:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Min første besked",);
}
```

▲ 2 of 21 ▼ System.Windows.Forms.DialogResult MessageBox.Show(TWin32Window owner, **string text**)
 Displays a message box in front of the specified object and with the specified text.
text: The text to display in the message box.

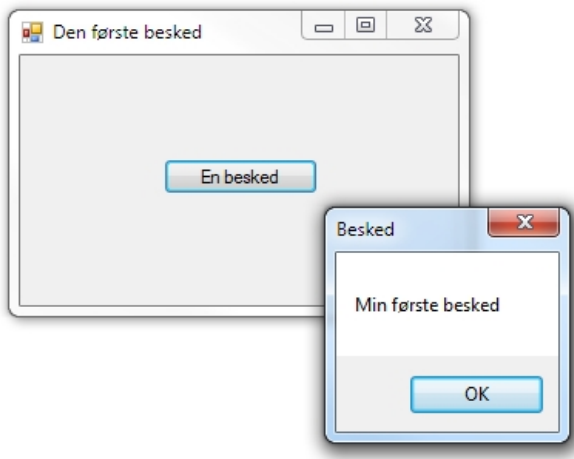
Tast følgende:

"Besked"

Igen skal du bruge et anførselstegn. Din linje med kode skal se ud på følgende måde:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Min første besked", "Besked");
}
```

Når din linje med kode se ud som vist ovenfor kører du igen dit program. Klik på din knap og du ser nu en titel i din dialogboks:



Flere Knap muligheder

I stedet for kun at have en OK knap, kan du tilføje knapper som Ja, Nej og Annuller til din C# dialogboks. Vi vil tilføje en Ja og en Nej knap.

Returner til dit kodevindue. Efter det andet anførselstegn i titlen du lige har tilføjet stater du et komma. Indsæt et mellemrum så dukker IntelliSense's liste op. (Hvis den ikke dukker op taster du et "M").

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Min første besked", "Besked", me);
}

```

▲ 4 of 21 ▼ System.Windows.Forms.DialogResult MessageBox.Show(IWin32Window owner, string text, string caption)
 Displays a message box in front of the specified object and with the specified text and caption.
caption: The text to display in the title bar of the message box.

- MergedMenu
- MergeFailedEventArgs
- MergeFailedEventHandler
- Message
- MessageBox**
- MessageBoxButtons
- MessageBoxDefaultButton
- MessageBoxIcon
- MessageBoxOptions

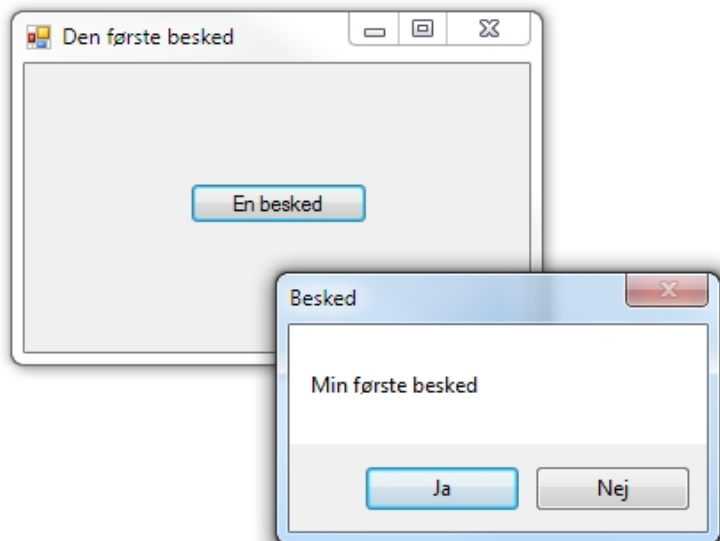
Den der tilføjer knapper til din dialogboks er ikke overraskende **MessageBoxButtons**. Tast Enter når denne er markeret. Den vil da blive tilføjet din kode. Tast nu et punktum efter det sidste "s" i MessageBoxButtons. Du ser nu de forskellige knap muligheder:

```
"Besked", MessageBoxButtons.);
```

- AbortRetryIgnore
- OK
- OKCancel
- RetryCancel
- YesNo**
- YesNoCancel

Dobbeltklik på **YesNo** og den vil blive tilføjet din kode.

Kør dit program igen og klik på din knap. Din dialogboks ser nu ud på følgende måde:

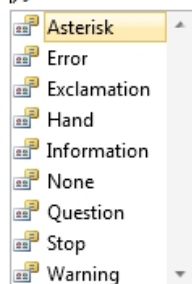


Tilføj ikoner til din C# dialogboks

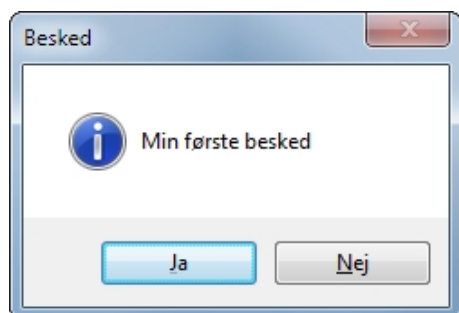
En anden ting du kan peppe din dialogboks op med er et ikon. Det nemmere at se hvad det er end at forklare det!

Tast endnu et komma efter **MessageBoxButtons.YesNo**. Efter kommaet taster du et "M". Fra IntelliSense kan du nu se en liste med muligheder. Dobbeltklik på **MessageBoxIcon**. Efter **MessageBoxIcon** taster du et punktum for at se de mulige ikoner:

```
"Besked", MessageBoxButtons.YesNo, MessageBoxIcon.);
```



Vi skal bruge et Asterisk. Dobbeltklik på denne og tilføj den til din kode. Kør dit program igen og se hvordan ikonet ser ud i din dialogboks:



Det ser meget imponerende ud! Og alt dette med bare en linjes kode!

Prøv de andre ikoner på IntelliSense listen for at se hvordan de ser ud når du kører dit program. Er der forskel på Information ikonet og Asterisk ikonet? (For hurtigt at se de muligheder der er på IntelliSense listen, sletter du ordet Asterisk og derefter punktummet. Tast igen et punktum og IntelliSense listen vises igen.)