



Avdeling for informatikk og e-l ring, H gskolen i S r-Tr ndelag

Introduksjon til programmering i VB.NET

Svend Andreas Horgen

L restoffet er utviklet for faget LN116D Programmering i Visual Basic

1. Introduksjon til programmering i VB.NET

Resym : Hva er egentlig .NET? Denne leksjonen starter med litt historikk om Visual Basic som programmeringsspr k og avklarer forskjellen p  begrepene spr k, utviklingsmilj  og rammeverk. Deretter tar vi for oss elementene i utviklingsmilj et, og lager et enkelt program. Dette utvides mot slutten av leksjonen, og vi kommer inn p  hvordan et program kan lages mer avansert ved relativt sm  endringer. M let med leksjonen er at du skal komme i gang med programmering i Visual Basic (VB) og at du skal f  lyst p  mer. Mye mer.

1.1.	VISUAL BASIC .NET	2
1.1.1.	Om programmering	2
1.1.2.	Lag ditt f�rste program.....	5
1.2.	UTVIDELSE AV PROGRAMMENE	9
1.2.1.	Mer informasjon, takk!	9
1.2.2.	Et litt mer avansert program	10
1.2.3.	Kj�r programmet utenfor utviklingsmilj�et ditt.....	12
1.3.	OPPSUMMERING	12

1.1. Visual Basic .NET

Programmering er gøy! Før vi begynner passer det å forklare kort hva Visual Basic .NET og programmering er.

Visual Basic er et programmeringsspråk som er laget av selskapet Microsoft. Det bygger på det gamle språket Basic, men har blitt endret flere ganger de siste årene. Overgangen fra 16-bit til 32-bit teknologi midt på 1990-tallet gjorde at Visual Basic 4 (VB 4) ble markant omskrevet fra den tidligere suksessen VB 3. Inntil nylig har VB 6.0 vært et populært programmeringsspråk for millioner av utviklere. Det fins derfor veldig mange lærebøker om VB 6.0, og utviklingsmiljøet VB 6.0 tilbød mange fordeler for hobbyutviklere så vel som profesjonelle programutviklere. Nå er ikke lenger VB 6.0 i salg, og det skyldes at Microsoft har skiftet strategi. Det er .NET som er gjeldende (uttales ”dått nett”) og har etablert seg som en seriøs konkurrent til Java de siste årene. Dette kurset vil ta for seg grunnleggende programmering i Visual Basic med Visual Studio/Basic .NET som utviklingsmiljø og .NET som rammeverk. Visual Studio fins i mange utgaver, 2003, 2005, 2008 og 2010, og et vell av varianter med alt fra fullversjon til den lette Visual Basic Express. I dette faget er vi mindre opptatt av versjoner og mer opptatt av generelle programmeringsferdigheter som fungerer uavhengig av utviklingsmiljø.

1.1.1. Om programmering

Du kan norsk, trolig engelsk og kanskje tysk, fransk eller spansk. Dette er språk du bruker når du snakker, og du gjør det for å uttrykke dine meninger og for å kommunisere med andre mennesker. Nå vil du lære deg å programmere. Vi som Høgskole tilbyr gjennom dette kurset en introduksjon til **grunnleggende programmering** og for å få til det, bruker vi språket Visual Basic (heter egentlig Visual Basic .NET, men vi bruker ofte navnet Visual Basic eller bare VB i dette kurset). Kurset vil i hovedsak fokusere på prinsipper i grunnleggende programmering, kun sekundært på mulighetene med .NET. Målet er at du skal lære deg programmering som kan brukes i andre sammenhenger. Visual Basic er ideelt til et slikt opplæringsformål, men kan også brukes til avanserte løsninger.

Det er forskjell på et *program*, et *programmeringsspråk*, et *utviklingsmiljø* og et *rammeverk*. Disse begrepene er viktige å skille fra hverandre og forstå, så vi tar kort for oss hvert enkelt i det følgende.

Et program er laget av noen

Et program er et sett av instruksjoner som kan utføres av en maskin til å gjøre noe. Du kjenner til mange programmer allerede – kanskje det mest nærliggende å nevne er Microsoft Word som mange bruker for å produsere dokumenter. Et enklere program som ofte brukes, er Kalkulatoren i Windows (tilsvarende finnes også på Linux, Mac og andre operativsystemer). Mange programmer kan lastes ned fra Internett, for eksempel mp3-spillerne Winamp og iTunes. Felles for alle programmer er at noen har skrevet dem slik at de kan brukes til å effektivisere eller berike den digitale hverdagen til brukere verden over.

Programmering er en måte å instruere datamaskinen, og er mer avansert enn vanlig digital kommunikasjon (så som tekstbehandling, epost-skriving, museklikking og tastetrykk). Når du bruker datamaskinen til daglig behandler du data og utfører oppgaver ved å *interagere* med programmer. Gjennom dette kurset skal du lære å lage dine egne programmer, og du vil snart erfare at uansett hvor skummelt dette måtte høres ut, så er det faktisk ganske enkelt. Med

Visual Basic kan du raskt lage enkle programmer, og med den teorien som dette kurset gjennomgår vil du også enkelt kunne bruke det du har lært til å raskere lære deg andre programmeringsspråk. Ved kursets slutt skal du dessuten kunne lage mer avanserte programmer i Visual Basic som kan ta vare på informasjon over tid, lage grafikk og bruke data fra for eksempel en database.

Forskjellige typer av programmeringsspråk

Et program må skrives, og for å skrive det trengs et *programmeringsspråk*. Noen eksempler på programmeringsspråk er:

- **Visual Basic** (fra Microsoft)
- **C++** (har noen år på baken, objektorientert og mye brukt)
- **C** (har noen år på baken, proseduralt, mye brukt)
- **Java** (objektorientert, blitt veldig populært siden lanseringen midt på 90-tallet)
- **C#** (uttales "Sii sharp", mange hevder dette er Microsofts Java-klone)
- **Pascal** (mye brukt til opplæring på 80-tallet)
- **Fortran** (brukt av matematikere)
- **Prolog** (passer til kunstig intelligens)
- **PHP** (veldig utbredt for webprogrammering)

Når du har valgt et språk består jobben mot å lage et ferdig program i å skrive kode i en *syntaks* som er i tråd med valgt språk. Syntaksen utgjør grunnsteinene i et språk og betyr de delene språket er bygget opp av. Visual Basic har en syntaks som er helt annerledes enn den som brukes i C++, som igjen er totalt forskjellig fra Prolog. Også selve idéen bak hvert språk er forskjellig, de har sine sterke og svake sider alle sammen og ulike bruksområder. Tilsvarende har norsk en annen syntaks og oppbygning enn kinesisk og arabisk.

! Det finnes flere hovedgrupper av programmeringsspråk. Tankemåtene, eller paradigmen, varierer. Objektorientert, funksjonell, prosedural og hendelsesorientert programmering er noen kjente grupperinger vi gjerne assosierer språk med. Det er også forskjell på hvilket nivå en programmerer. I dag bruker de fleste høynivå-språk, noe som gjør det relativt enkelt for Hvermannsen å lære programmering. De første pionerene måtte kode på lavt nivå, såkalt maskin-nivå. I dette kurset benyttes programmeringsspråket Visual Basic – i utgangspunktet et hendelsesorientert, høynivå språk som er blitt veldig populært grunnet blant annet dets enkle syntaks.

Hvis du har programmert tidligere vil du kanskje kjenne til at ulike språk har sine sterke og svake sider. Fordelen med Visual Basic er at det er enkelt å komme i gang med, og vi skal ganske snart lage vårt første program. Du trenger ikke kunne noe programmering fra før, vi skal ta det rolig i starten.

Utviklingsmiljø

Når et program skal lages, kan koden skrives inn i en enkel teksteditor, for eksempel Notisblokk eller TextPad for Windows, Pico eller VI for Unix/Linux. Å skrive programkode i en teksteditor bød på en revolusjon da det ble introdusert på 60-tallet, siden programmene før den tid måtte lages ved å hulle ut hullkort eller skifte om på en masse ledninger i en maskin. Det er nå klart for å se på utviklingsmiljøet som vi skal bruke i dette kurset.

I dag er det av og til hendig å bruke enkle teksteditorer når en programmerer, men stort sett er det mye raskere å bruke et *utviklingsmiljø*. Et utviklingsmiljø er et (stort) program som gjør arbeidet å programmere mye enklere ved å tilby programmereren å designe brukergrensesnittene med klikk-og-dra-operasjoner. En annen fordel er bruk av fargekoding for å skille faste kommandoer fra hverandre og annen kode. Det er også lettere å strukturere og dokumentere koden på en god måte, og mulig å jobbe med mange filer/prosjekter åpne samtidig. Du som utvikler skal få fokus på de kreative utfordringene i programmering slik at du kan løse problemer så effektivt og uhindret som mulig uten å måtte administrere rutinemessige ting og tenke på detaljer. Verktøy er til for at du skal slippe å finne hjulet opp på nytt hver gang, men også for å gi oversikt, tips, råd, hjelp, feilfinning og så videre.

Visual Studio Express-pakkene er en samling av utviklingsmiljø som tilbyr deg som programutvikler å programmere i enten Visual Basic, C#, C++ eller ASP.NET. Express-pakkene er lettversjonen av den noe tyngre storebroren (superpakken) **Visual Studio .NET**, og du kan for eksempel velge å installere bare Visual Basic Express. Da kan du programmere i Visual Basic. Express-utgavene har 90% av storebrorens funksjonalitet, og er myntet mot opplæring. Visual Studio .NET-miljøet er mye større enn de små Express-pakkene, koster mye, men tilbyr den profesjonelle programutvikler all mulig avansert funksjonalitet, samt å programmere i et hvilket som helst av språkene Visual Basic, C++, C# og J#. Alle ”Visual-miljøene” er laget av Microsoft. Mer informasjon om utviklingsmiljøene og hvilken versjon som passer deg best, finner du på informasjonssidene tilhørende dette faget.

Til sammenlikning kan du bruke miljø som for eksempel Metroworks Codewarrior for å programmere i C++. Dette betyr at C++ både kan brukes ved hjelp av verktøyene Visual Studio .NET, Visual C++ .NET og Codewarrior (og også enda flere). Det er altså slik at et utviklingsmiljø kan støtte mange språk, og et språk kan benyttes i mange utviklingsmiljø. *Et utviklingsmiljø er kun til for å lette arbeidet med programmeringen!*

Dette kurset bruker utviklingsmiljøet Visual Studio .NET i skjermbilder og eksempler, men Express-versjonen Visual Basic .NET kan også brukes uten problem. I teorien kan du bruke Notisblokk, men det anbefales ikke. Det kan være forvirrende at Microsoft opererer med så mange versjoner og varianter, og at de har gitt programmeringsspråket Visual Basic .NET og utviklingsmiljøet Visual Basic .NET samme navn. Du bør likevel nå ha en klar forståelse av skillet mellom et språk og et utviklingsmiljø.

Rammeverket .NET – forhistorie og motivasjon

Før du kan utvikle i Visual Basic, må du ha støtte for *rammeverket .NET*, på engelsk .NET Framework. Dette er flere hundre megabyte stort og kan lastes ned. Heldigvis følger det med Visual Studio-miljøet, og dersom du har Windows nyere enn XP med Service Pack 2, har du det trolig også fra før.

Dette avsnittet skal prøve å forklare hvorfor .NET oppstod. Det er mildt sagt ikke trivielt å lage avansert programvare – men det er ofte nødvendig for aktører som skal lykkes i IT-markedet. For å oppnå suksess har utviklerne (det er ofte flere som programmerer ulike deler av et stort system) måttet ta en rekke hensyn:

- Velge blant og lære ulike fremgangsmåter eller modeller for å programmere grafikk.
- Ulike fremgangsmåter eller modeller for å aksessere databaser.
- Ulike versjoner av Windows med ulike krav (Windows 95, 98, Me, NT, 2000, XP)
- Versjonshåndteringen har ikke vært god.

- Vanskelig å gjenbruke kode.
- Vanskelig å kombinere kodebiter som er skrevet i forskjellig språk til en større applikasjon.

Det er vanskelig å utvikle med tanke på *robusthet og sikkerhet*. Som bruker av tegneprogrammer, tekstbehandlere, kalkulator, musikkprogrammer og liknende har du sikkert opplevd krøsj, bedrifter kan ha opplevd hacking eller andre sikkerhetshull. Det kan skyldes dårlig programmering og/eller de ytre forutsetningene nevnt over. Dårlig sikkerhet gir dårlig reklame – derom levnes ingen tvil. Samtidig har behovet for stadig tettere integrasjon mellom webapplikasjoner og tradisjonelle programmer økt. I sum har disse og andre faktorer ført til at Microsoft lanserte rammeverket .NET først på 2000-tallet som et steg i retningen mot mindre kompleksitet, større robusthet og nye muligheter for utviklere.

Idéen bak .NET er å overkomme de problemene en har hatt så langt. Som programutvikler kan du med .NET:

- Gjøre bruk av databaser mye lettere enn før.
- Slippe å tenke på underliggende sikkerhet knyttet til ulike versjoner av Windows.
- Bruke de vinduer, menyer, verktøy og annen funksjonalitet som rammeverket tilbyr.
- Lettere kunne utvikle større webløsninger med ASP.NET.
- Signere programmene du lager.
- Programmere i et hvilket som helst språk (som utviklingsmiljøet Visual Studio .NET støtter) mot det underliggende, felles rammeverket .NET.

Siden alle språk som støttes av .NET bygger på det samme rammeverket, skal kompatibilitet, sikkerhet og ikke minst robusthet bli resultatet. En følge av dette er at en programutvikler kan utnytte de sterke sidene ved ulike programmeringsspråk og lage en del av et program i språket Visual Basic, en annen del av samme program i språket C++ og en tredje del i språket C#.

1.1.2. Lag ditt første program

Vi skal nå lage vårt første program og samtidig bli kjent med utviklingsmiljøet. Hvorvidt du bruker en utgave av miljøet Visual Studio .NET eller en utgave av miljøet Visual Basic .NET spiller ingen rolle for dette kurset.

Ønsker å si "Velkommen, Kari Nordmann" til Kari Nordmann

Vi skal nå lage et program som ber en bruker skrive inn sitt navn og deretter ønsker denne velkommen når en knapp trykkes på.

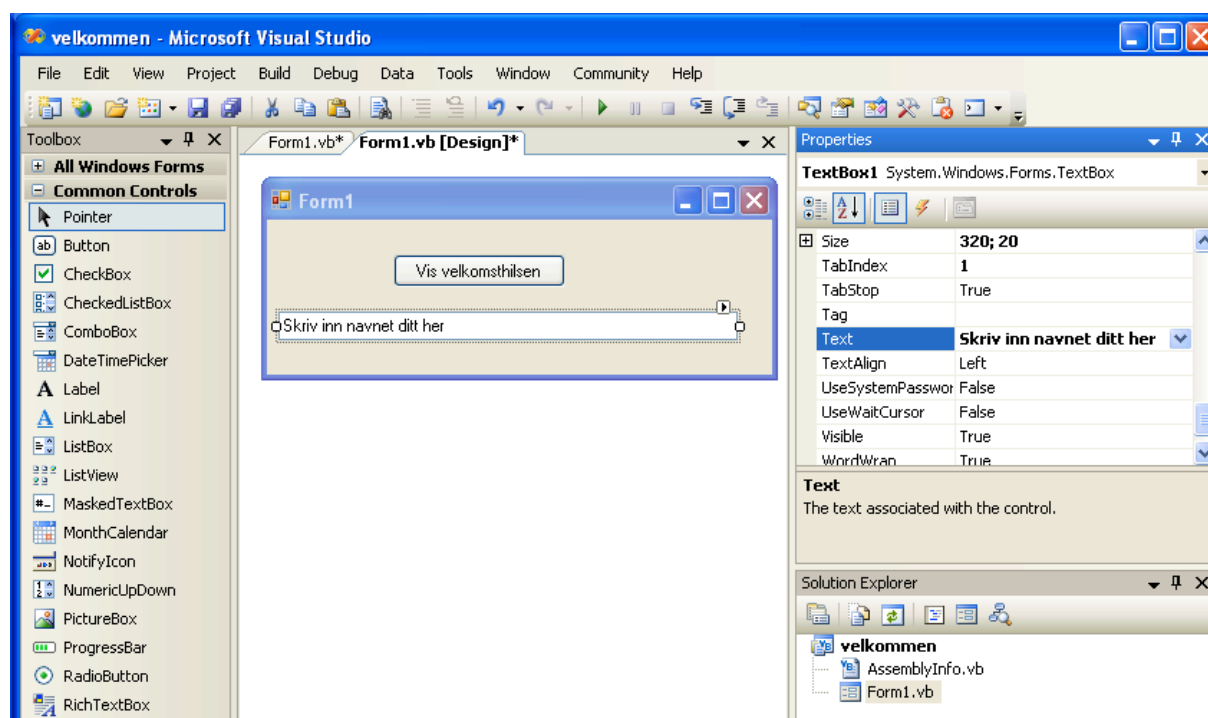
Et program må tilhøre et prosjekt. Start derfor med å lage et nytt prosjekt i ditt utviklingsmiljø og kall dette for "velkommen". Åpne gjerne mappen (fra Utforskeren i Windows) som du har opplyst at prosjektet skal lagres i, og observer at det har blitt laget en rekke filer og to undermapper kalt "bin" og "obj".

Skjema, designmodus og kodemodus

Det kommer automatisk opp et såkalt skjema (eller form som det heter på engelsk, leksjonene bruker begge begrepene). Det er dette skjemaet som utgjør basisen i programmet når det kjører. Du står fritt til å legge til knapper, tekstbokser, menyer, informasjonsbokser, listebokser, radioknapper, avkrysningsbokser og mye mer i skjemaet. Disse ser du i

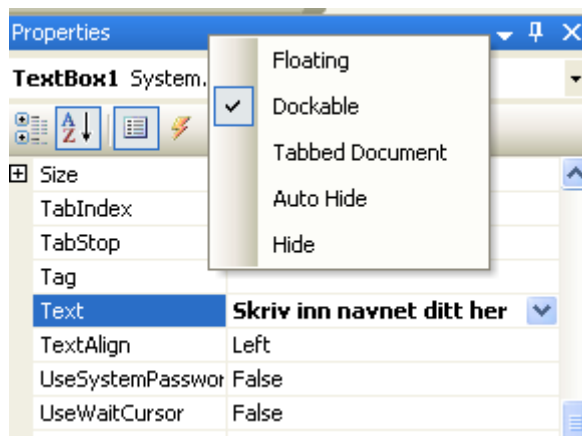
verktøykassen til venstre i utviklingsmiljøet (se Figur 1). Brukere trenger noe for å kommunisere med programmet på, og programmet trenger noe for å vise resultater og de muligheter brukeren har. Det brukeren ser når programmet starter, og får resultater på, kalles *grensesnittet* i et program. Det fine med Visual Studio er at du enkelt kan forme *grensesnittet* i programmet slik du vil ha det.

Du kan se av Figur 1 at vi har dratt skjemaet til en passende størrelse, og plassert en knapp og en tekstboks i skjemaet. I det du lager en ny knapp vil teksten på den bli "Button1", mens et nytt tekstfelt får teksten "TextBox1". Du ser at teksten i knappen og i tekstboksen er mye mer beskrivende enn utgangspunktet. For å få til det må du markere for eksempel tekstboksen ved å klikke en gang på den, gå til egenskapsvinduet (engelsk: Properties, befinner seg vanligvis nederst til høyre i utviklingsmiljøet). Deretter må du finne frem til egenskapen "Text" og skrive inn teksten "Skriv inn navnet ditt her". Dermed vil det som vises i tekstboksen i skjemaet endres. Tilsvarende har vi endret tekstegenskapen til knappen fra Button1 til "Vis velkomsthilsen". Du kan endre en rekke andre egenskaper i Properties-vinduet, prøv deg gjerne frem med farger, skrifttyper og liknende.



Figur 1: Vi lager et enkelt program i VB, med en knapp og en tekstboks i et skjema (form). Merk: Bildet kan variere på tvers av de mange versjonene av Visual Studio. Hovedprinsippene er derimot like.

- ! Dersom du ikke ser seksjonen med "Common Controls" så må du velge menyen View... Toolbox. Det er mulig å stikke om på rekkefølgen av vinduer i Visual Studio.
- Hvis du skulle komme til å miste vinduselementer som *Properties*, *Solution Explorer* og andre, finner du også disse igjen i View-menyen. Høyreklikk på den blå tittellinjen (for eksempel i properties-vinduet) og du får mulighet til å velge om vinduet skal være "dockable", "floating" og så videre. Prøv ut dette hvis du vil, men ikke gjør det i starten før du er fortrolig med miljøet. Det kan være litt vanskelig å få tilbake et fint oppsett igjen.



Figur 2: Du kan styre utseendet i Visual Studio på mange måter, men det kan være vanskelig å rekonstruere igjen, så vær forsiktig i starten.

Det du ser i Figur 1 er designmodus (også kalt utformingsmodus) til Visual Studio. Over skjema-vinduet er det to arkfaner, en som heter "Form1.vb (Design)" og en som heter "Form1.vb". Med disse kan du veksle mellom *designmodus* og *kodemodus*. Designmodus brukes for å lage grensesnittet bestående av knapper, tekstfelt og liknende. Kodemodus brukes for å skrive inn kode som skal utføres. Visual Basic er et hendelsesbasert språk, og det betyr at det er hendelser i brukergrensesnittet som bestemmer hvilken kode som skal utføres. Et klikk på en knapp er en hendelse. Dersom du dobbeltklikker på knappen, vil Visual Studio bytte til kodemodus og opprette følgende kode:

```
Private Sub Button1_Click(ByVal sender As System.Object,
                          ByVal e As System.EventArgs)
    Handles Button1.Click
End Sub
```

Legg merket til at selv om teksten etter Private Sub... er brutt i denne leksjonen for oversiktens skyld, vil det hele mest trolig vises på en og samme linje hos deg. Det som skrives innenfor Private Sub... og End Sub vil nå bli kode som tilhører denne knappen.

MERK: Det er veldig viktig å dobbeltklikke på knappen og ikke i bakgrunnen i skjemaet eller på tekstfeltet. Bommer du på knappen vil koden nemlig knyttes til andre hendelser, og det er ikke det vi ønsker i dette programmet. Det er en vanlig nybegynnerfeil å dobbeltklikke på tekstfelt eller selve skjemaet utilsiktet.

Skriv nå inn koden som for lesbarhetens skyld er markert i grått her:

```
Private Sub Button1_Click(ByVal sender As System.Object,
                          ByVal e As System.EventArgs)
    Handles Button1.Click
    MsgBox("Velkommen, " & TextBox1.Text)
End Sub
```

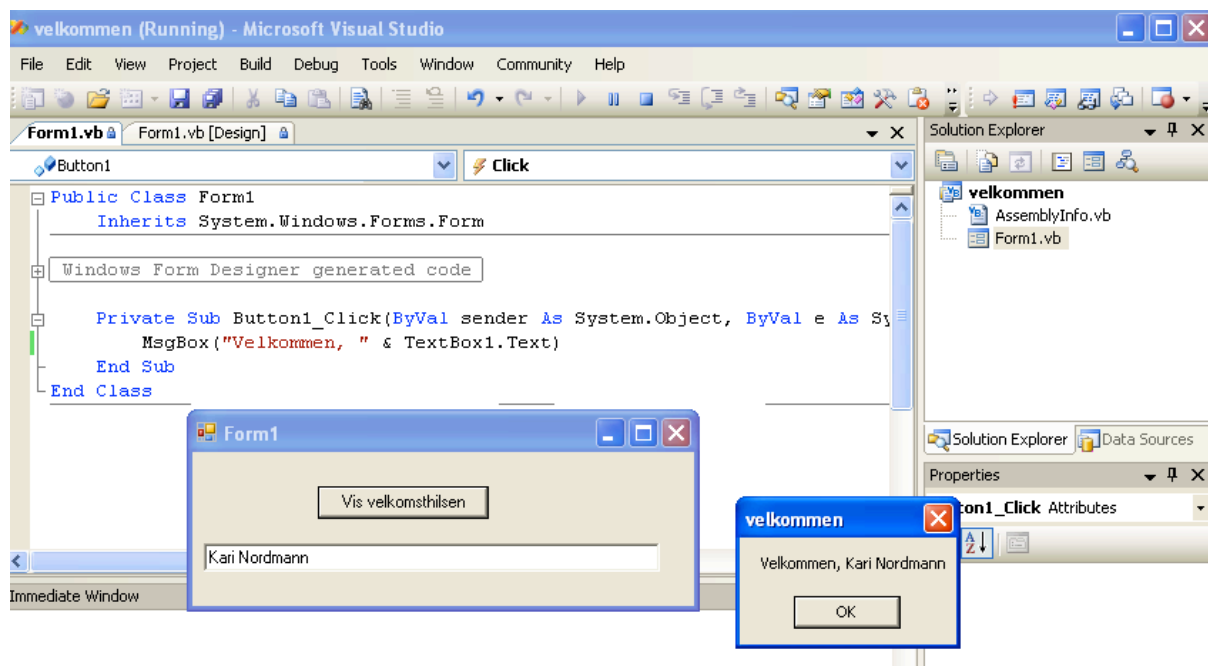
Du må skrive setningen nøyaktig riktig og på riktig sted. Vær spesielt oppmerksom på parenteser, anførselstegn (dobbel, ikke enkel), &-tegnet og punktet foran Text:

```
MsgBox("Velkommen, " & Textbox1.Text)
```

Hva betyr det å kjøre et program?

Du har nå laget ditt første program! Start programmet ved å enten trykke på den liggende blå trekanten ved siden av "Debug" øverst mot midten av skjermbildet, eller ved å trykke

funksjonstasten F5 på tastaturet ditt. Visk så ut det navnet som står i tekstboksen og erstatt det med ditt eget. Trykk deretter på knappen, og du skal da få fram følgende resultat:



Figur 3: Når et program kjører, blir det moduset du står i lagt i bakgrunnen, og det programmet du har laget får fokus og kan testes.

Vi sier at vi nå er i *kjøremodus* eller at ”programmet kjører”. Det kommer fram et vindu som tillater oss å skrive inn et navn, og du ser at det er skjemaet fra designmoduset som nå vises som et eget, lite vindu. Tekstboksen har innholdet ”Skriv inn navnet ditt her”. Brukeren kan viske ut denne teksten og skrive inn sitt eget navn i stedet. Når knappen med teksten ”Vis velkomsthilsen” trykkes, spretter det opp en ny boks som viser en beskjed og har en OK-knapp. Figur 3 er tatt når dette er gjort. Du ser at det er navnet Kari Nordmann som er skrevet inn. Når OK-knappen i meldingsboksen trykkes, forsvinner meldingsboksen. En meldingsboks er bare en boks som viser en melding og tillater brukeren å akseptere denne beskjeden. Deretter kan et nytt navn skrives inn og knappen trykkes på nytt for å vise en ny meldingsboks (med en ny velkomsthilsen).

For å avslutte kjøringen av programmet kan du trykke på det røde krysset i det programmet du har laget (ikke den i Visual Studio :-)) eller du kan klikke på den blå stopp-knappen i utviklingsmiljøets verktøylinje. Når du avslutter vil du gå tilbake til det moduset du stod i når programmet startet.

Noen få ord om det som skjer bak kulissene i det du starter programmet er på sin plass. En datamaskin kan bare forstå maskinkode i form av 0-ere og 1-ere. Prosessen med å oversette den koden du har skrevet inn til maskinkode, kalles ”kompilering”. Dersom det er noen syntaktiske feil i programmet (skrivefeil eller ulovlige kommandoer), vil kompileringen mislykkes og du vil få beskjed om at programmet ikke kan kjøre enda. Mer om dette siden.

Forklaring til programmet

Det er flere ting å merke seg allerede nå. Koden er bare halve programmet. I tillegg til setningen

```
MsgBox("Velkommen, " & TextBox1.Text)
```

som vi skrev inn etter å ha dobbeltklikket på knappen i designmodus, er selve brukergrensesnittet helt nødvendig for at programmet skal gjøre det vi ønsker. I gamle dager måtte brukergrensesnittet med alle dets knapper, tekstfelt, vindusstørrelse og så videre programmeres inn, men det slipper du å gjøre i moderne utviklingsmiljø. Vårt grensesnitt består av et skjema som igjen består av en knapp med navnet `Button1` og en tekstboks med navnet `TextBox1`. Selv om vi endret det som vises i tekstboksen og i knappen til henholdsvis ”Skriv inn navnet ditt her” og ”Vis velkomsthilsen”, endret vi ikke *navnene* på disse knappene. Det vi oppnådde ved å endre `Text`-egenskapen var bare å lage brukergrensesnittet mer brukervennlig å se på.

Tekstbokser og knapper har fellesbenevnelse *kontroll*. Hver kontroll har en rekke egenskaper som vises i properties-vinduet når kontrollen er markert. Fra dette vinduet kan du endre egenskapene til ønsket kontroll.

Alle tekstbokser har altså en egenskap som heter `Text` og som reflekterer innholdet som vises i tekstboksen når programmet starter opp. I egenskapsvinduet finner vi også et felt som heter `(name)` som angir navnet til tekstboksen. Kommandoen `msgbox()` fører til at det som står innenfor parentesene vises på skjermen i en boks med en OK-knapp i. Teksten ”Velkommen, ” er en såkalt *tekststreng*. Det som står i tekstboksen er også en tekststreng. Du ser at programmet vårt tar de to tekststrengene og slår dem sammen til en beskjed i den meldingsboksen som spretter fram når knappen trykkes. Det er tegnet `&` som slår sammen to tekststrenger, og sørger for at velkomsthilsenen blir fullstendig. Tekststrenger kan du utforske nærmere i leksjonen om variabler, men godta bare syntaksen foreløpig.

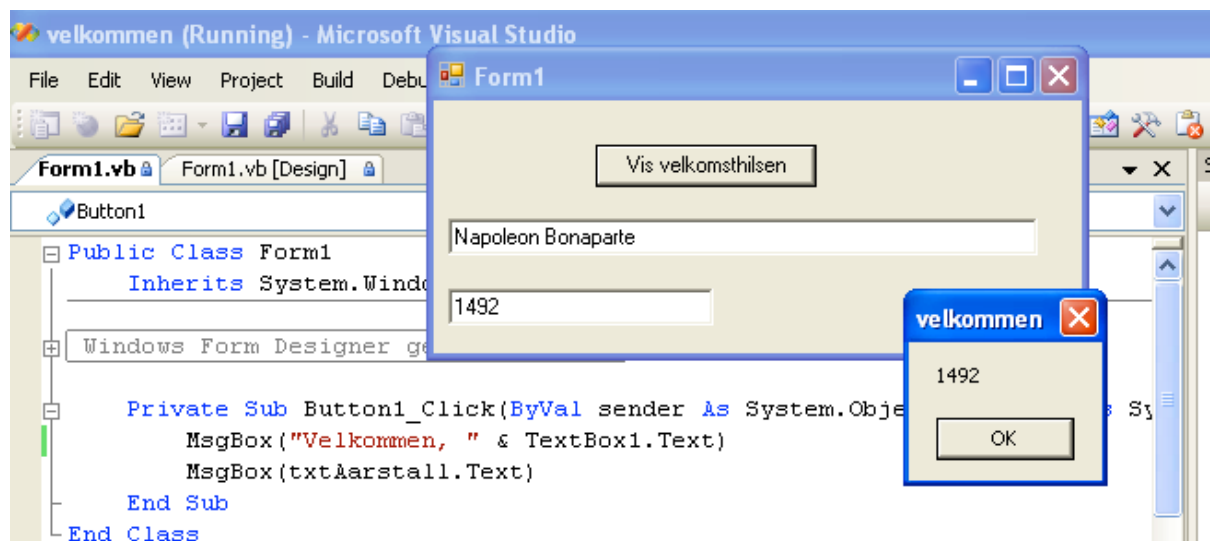
1.2. Utvidelse av programmene

Det som er bra med programmering er at det er enkelt å utvide programmene vi lager gradvis. Nå som vi har fått vårt første program til å fungere er det en smal sak å lage et program som ikke bare ønsker velkommen, men også skriver ut årstallet som er oppgitt.

1.2.1. Mer informasjon, takk!

Videreutviklingen av programmet er enkel. Vi har lagt til en ny tekstboks som vi har gitt navnet `txtAarstall`, i tråd med de prefiks-kodene som læreboka introduserer i kapittel 2. Med mange ulike kontroller i grensesnittet er det lurt å navngi alle tekstbokser med prefikset `txt`, alle labeler med `lbl` og så videre. Du er vel enig i at navnet `txtAarstall` er mer beskrivende enn `TextBox2`?

Når vi nå legger til koden som skimtes i bakgrunnen for deretter å kjøre programmet, skriver inn et nytt navn og et årstall, og til slutt trykker på OK-knappen i den første meldingsboksen som spretter opp, kommer en ny meldingsboks fram. Denne boksen viser årstallet vi skrev inn.

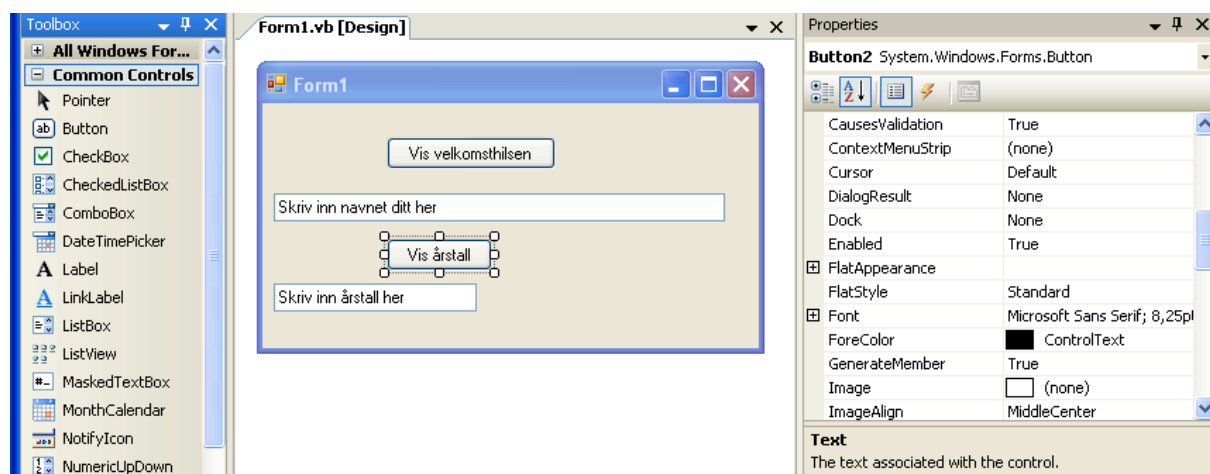


Figur 4: Utvidelse av det første programmet, her med to tekstfelt i skjemaet.

Programmet er nå utvidet til å ta i mot en ny mengde data fra brukeren og til å vise mer informasjon på skjermen. En ekstra setning og en ekstra tekstboks var alt som skulle til fra programmerers side. Merk at vi kan kjøre programmet uavhengig av om vi står i kode-modus eller i design-modus.

1.2.2. Et litt mer avansert program

Nå skal vi lage et helt nytt, interessant program, nemlig et som forandrer seg utseendemessig mens det kjører. Det første steget er å lukke forrige prosjekt (husk å lagre alle filene). Lag deretter en kopi av mappen som programmet ditt ligger lagret i og kall denne kopien for "velkommen_utvidet". Åpne dette nye prosjektet ved å dobbeltklikke på prosjektfilen (filnavn.proj) direkte fra Utforskeren, eller velg File...Open fra utviklingsmiljøet og bla deg fram til riktig mappe. Skjermbildet skal se ut som i Figur 5:



Figur 5: Egenskaper kan for eksempel settes i Properties-vinduet. Det fine med dette vinduet, er at du ser de muligheter som fins for hver kontroll. Legg merke til muligheten for å sortere egenskapene alfabetisk og kategorisk.

Det spesielle å merke seg er at vi har lagt til en knapp og et tekstfelt. Egenskapen *visible* til både den nye knappen og det nye tekstfeltet er satt til "false". Når denne egenskapen til en kontroll settes til false i egenskapsvinduet, vil kontrollen bli usynlig (ikke vises) når programmet startes. Det morsomme er at vi kan endre egenskapene til en kontroll både ved å manipulere egenskaps-vinduet, og *ved å skrive kode*. La oss derfor legge til et par setninger og få følgende kode bak første knapp:

Kodesnutt som viser usynlige kontroller etterhvert

```
1. Private Sub Button1_Click(ByVal sender As System.Object,  
                             ByVal e As System.EventArgs)  
                             Handles Button1.Click  
2.     MsgBox("Velkommen, " & TextBox1.Text)  
3.     Button2.Visible = True  
4.     txtAarstall.Visible = True  
5. End Sub
```

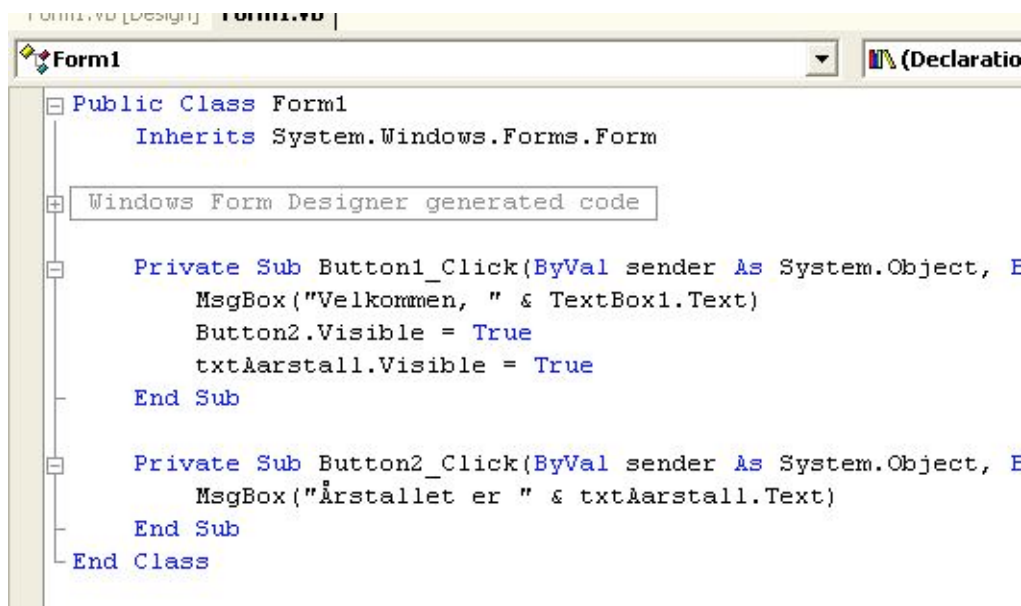
Vi har lagt til linjenummer for å lette lesbarheten. Linje 1 består av den lange setningen `Private Sub osv...` og linje 5 har de to kodeordene `End Sub`

Når programmet kjøres og knapp nummer én trykkes, vil boksen med velkomsthilsenen komme fram. Dette bestemmes av koden i linje 2 og medfører at programmet stopper midlertidig. Når OK-knappen trykkes er linje 2 ferdigkjørt og det er linje 3 sin tur. Den gjør at knapp nummer to vises på skjermen. Tilsvarende vil linje 4 sørge for at tekstfeltet vises. Deretter er det mulig å skrive inn et årstall i det andre tekstfeltet og trykke på knappen "Vis årstall". Programsetningen

```
Msgbox(txtAarstall.Text)
```

som opprinnelig stod under knapp nummer en, har vi nå fjernet og flyttet til koden for knapp nummer to. Et utklipp fra Visual Studio bør fjerne enhver tvil om hvordan den totale koden nå ser ut. Gjør det samme, kjør programmet selv og observer hva som skjer. På en-to-tre har vi laget en enkel form for dynamikk i programmet vårt.

Dersom det kommer fram en rekke kodelinjer over "Private Sub Button1_Click", går det an å trykke på minus-tegnet i margin ved siden av "Windows Form Designer generated code" for å skjule all koden som utviklingsmiljøet selv legger til automatisk. I Visual Studio 2005 er ikke denne kodeblokken til stede, men i tidligere utgaver av Visual Studio er det automatisk generert kode som kan vises/skjules. Det går også an å skjule/vise egen kode ved å trykke på riktig pluss og minus, noe som er hendig for å få bedre oversikt i større program.



```
Public Class Form1
    Inherits System.Windows.Forms.Form

    Windows Form Designer generated code

    Private Sub Button1_Click(ByVal sender As System.Object, E
        MsgBox("Velkommen, " & TextBox1.Text)
        Button2.Visible = True
        txtAarstall.Visible = True
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, E
        MsgBox("Årstallet er " & txtAarstall.Text)
    End Sub
End Class
```

Figur 6: Her er kode knyttet til to ulike knapper. Merk at det er knappens navn som følger etter Private Sub, og du kan også lese deg til at det er hendelsen Click (et trykk på knappen) som setter i gang denne prosedyren.

1.2.3. Kjør programmet utenfor utviklingsmiljøet ditt

En bra ting med utviklingsmiljøet er at det lages en programfil som du kan kjøre uavhengig av om Visual Studio/Visual Basic er installert på maskinen. Du har allerede installert miljøet, og bra er det. Det er derimot tungvindt å skulle starte Visual Studio/Visual Basic hver eneste gang du skal kjøre et program.

Åpne mappen hvor et av prosjektene dine ligger lagret, og bla deg fram til mappen "bin" Her finner du programmet som en kjørbare exe-fil. Start den og se hva som skjer. Du vil kanskje merke at filen starter opp mye raskere enn den gjorde i utviklingsmiljøet ditt? Grunnen til dette er at programfilen er gjort kjørbare og er gjort om til maskinkode som kan kjøres direkte. Sender du denne filen til en venn per epost, kan han/hun kjøre programmet du har laget. Dersom vennen din ikke får til å kjøre programmet, skyldes det at .NET-rammeverket ikke er installert på dennes maskin. Det kan lastes ned via Windows Update fra Microsofts hjemmesider. Når du har Visual Studio har du også .NET Framework, men det er ikke sikkert at andre har det. Etter hvert blir det nok mer og mer vanlig. Merk at .NET foreløpig bare er for Windows.

Gitt muligheten for ikke bare å lage egne programmer, men også distribuere disse til de du kjenner: Protesterer du nå på utsagnet om at programmering er gøy? ☺

1.3. Oppsummering

Vi har i denne leksjonen sett litt på historikken til Visual Basic, introdusert rammeverket .NET og prøvd å gi en forståelse av hvordan et program kan skrives i ett av mange språk. Prosessen med å utvikle kan gjøres enklere ved å bruke et godt utviklingsmiljø.

Det er enkelt å lage programmer med språket Visual Basic, og enda enklere med et utviklingsmiljø som Visual Studio .NET. Vi lagde to programmer, og du aner kanskje at det er lurt å lagre hvert program du lager i forskjellig mappe. En oppdatert kjørbare programfil

legges i mappen "bin" hver gang du kjører programmet fra utviklingsmiljøet, og denne programfilen kan kopieres over til en hvilket som helst annen mappe eller maskin.

Visual Basic er et hendelsesbasert språk, og det betyr at det er hendelser (knappeklikk, mus over knapp, endring i tekstfelt og liknende) som bestemmer hvilken kodebit som skal utføres til enhver tid. For å skrive kode må du kjenne til de grunnleggende byggestenene i Visual Basic. Disse skal du bli mer kjent med etter hvert. I denne leksjonen har du fått se hvordan en meldingsboks kan kombinere statisk innhold med verdier fra ulike tekstfelt.

Egenskaper er sentralt i Visual Basic-sammenheng. Det viktige å merke seg etter denne leksjonen, er at hver knapp, tekstfelt og til og med skjemaet disse ligger i, har en rekke egenskaper. Navn, innhold og utseende, men også synbarhet er eksempler på egenskaper som de fleste kontroller må ha. Ved å bruke punktum-notasjonen "kontrollnavn.egenskapsnavn" henviser du til egenskapen til en bestemt kontroll. Du kan legge inn informasjon i en egenskap, slik:

```
TextBox1.Text = "hei på deg"
```

Du kan også bruke informasjonen som akkurat nå ligger i en egenskap til en kontroll, for eksempel skrive ut innholdet som er i en tekstboks i en meldingsboks:

```
MsgBox (TextBox1.Text)
```

Du vil lære mye mer i dette kurset. Lek deg nå til å begynne med, og ikke vær redd for å utforske ulike egenskaper i properties-vinduet!