

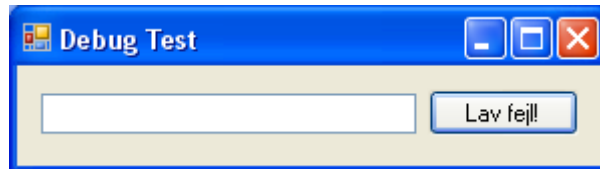
Opgave 4.

Den brette app. debugges

Vi har lavet den fede applikation, men der er noget galt... Det virker nogle gange men ikke altid. Så nu skal der debugges.

Visual Studio har mange muligheder for at debugge dit program, i denne opgave skal vi kort gennemgå de mest almindelige metoder.

Start med at oprette et nyt WFC projekt:



Indsæt en tekstboks og en knap på brugerfladen og indtast koden nedenfor i knappens "Click" event handler:

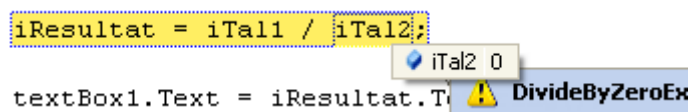
```
private void button1_Click(object sender, RoutedEventArgs e)
{
    int iTal1 = 4;
    int iTal2 = 0;
    int iResultat = 0;

    iResultat = iTal1 / iTal2;

    textBox1.Text = iResultat.ToString();
}
```

Prøv at køre programmet, hvad sker der når du trykker på knappen?

Eftersom du kørte programmet i debuggeren, returnerer du til Visual Studio når fejlen opstår. Her har du nu mulighed for at teste forskellige ting. Bl.a. hvad de enkelte variabler indeholder:

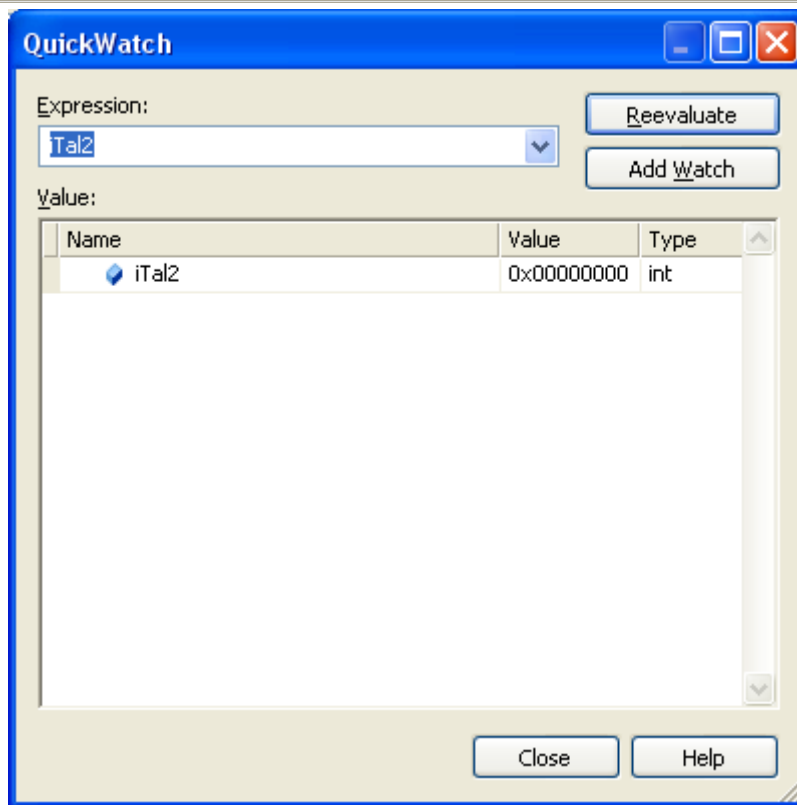


Ved at holde musen hen over en given variabel vil du se variabelens indhold. (Hvis den ellers er tilgængelig fra den del af koden vi befinder os i!)

Hvis du ønsker at ændre på variabelen kan du højreklikke på den og vælge Quick Watch

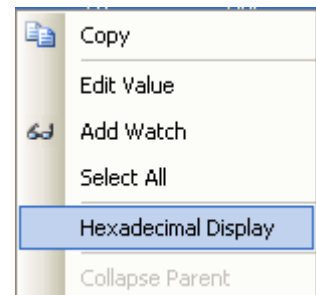
Quick Watch vinduet giver dig mulighed for at undersøge de enkelte variabler i dybden. Hvis du for eksemplets skyld markerer textBox1 og vælger Quick Watch vil du få en helt masse forskellige properties frem.

Men i denne omgang koncentrerer vi os omkring variabelen "iTal2". Marker den og vælg Quick Watch.

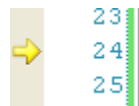


Som du kan se er informationerne rimelig begrænset omkring denne type variabel. Og godt det samme, for alt vi kan i dette tilfælde er at ændre værdien som iTal2 indeholder.

I vinduet ovenfor er værdien af iTal2 angivet som et hexadecimal tal. Du kan ændre dette ved at højreklikke på variablen og fjerne flue benet fra "hexadecimal Display" som vist her til højre:




Som du også kan se på menuen til højre kan du ligeledes ændre værdien, prøv at sætte den til 2 i stedet for 0 som den står til lige nu. og luk så quick watch vinduet igen. Når du nu holder musen hen over iTal2 vil du se at den nu indeholder værdien 2.



```
iResultat = iTal1 / iTal2;
```

iTal2 2

For at teste denne stump kode skal du nu tage fat i den lille gule pil vist på billedet ovenfor og trække det en kode linje op, det vil resultere i at debuggeren "spoler tilbage" i programmet og prøver den defekte linje kode endnu en gang.

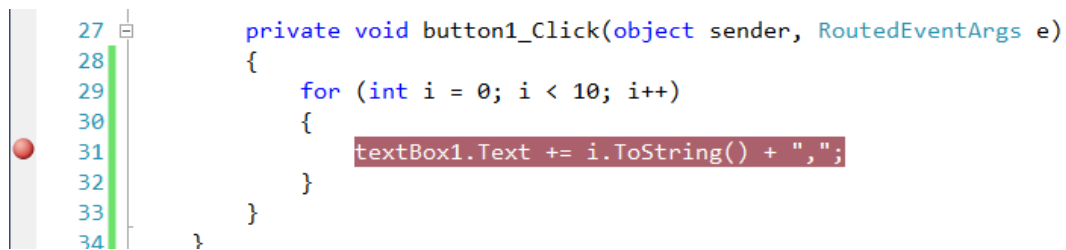
Hvis du trykker på "play" knappen  oppe i debug menuen, skulle der gerne komme noget i tekst boksen?

Men vi behøver ikke generere fejl for at stoppe programmet et bestemt sted. Vi kan også indsætte break points. Break points er en hjælp til os programmører hvis vi ønsker at se programmet i en bestemt tilstand eller bare ønsker at standse programmet på et bestemt sted for at "kigge ind".

Prøv at ændre koden i button1_Click funktionen så det ligner det nedefor:

```
private void button1_Click(object sender, EventArgs e)
{
    for (int i = 0; i < 10; i++)
    {
        textBox1.Text += i.ToString() + ",";
    }
}
```

Problemet med koden ovenfor er nu at den ikke tæller 10 med, også selv om der jo står 10 i koden. Så det vil vi lige teste med et break point... Find den linje du ønsker at standse programmet på og tryk helt ude i den grå margin:

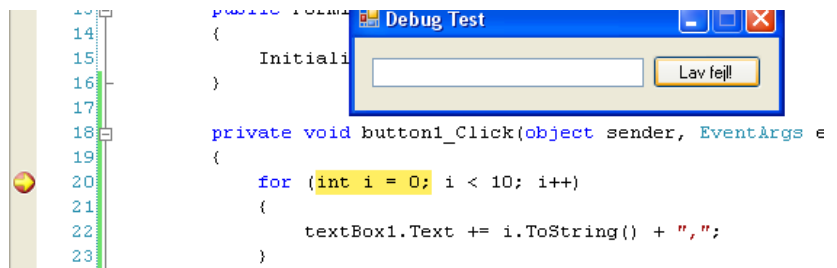



Der vil nu komme en rød cirkel og noget af koden vil blive farvet rød. Den røde cirkel indikerer break pointet og den røde kode indikerer den del af koden hvor debuggeren vil standse.



Prøv at starte dit program i debuggeren (F5) og se hvad der sker:

Når du trykker på knappen returnerer du til Visual Studio. Her er den kode, **som eksekveres næste gang** markeret med gult.

Hvis du vil gennemgå programgangen i enkelte trin (single



step) så skal du nu trykke på step over knappen , eller trykke F10. Så vil du se at den gule markering flytter sig rundt imellem de stumper kode som eksekveres.

Du kan også steppe out  hvis du eksempelvis ønsker at komme ud af for loopet (eller en funktion eller en while løkke eller....) Og step into  som virker lige modsat step out.

Derudover har du følgende menu:



Start debugger(eller fortsæt programmet),

Pause programmet,

Stop ,

Reset programmet

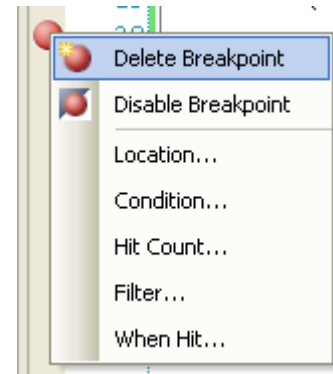
Opgave 4.

Den borte app. debugges

Men break points kan en del mere end bare stoppe programmet.

Vi kan også sætte parametre op for hvornår et break point skal "trigger".

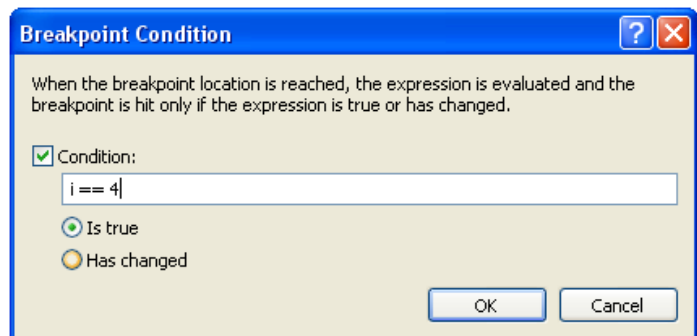
Hvis du højreklikker på breakpointet får du en lille menu frem som vist her til højre. Menuen giver dig mulighed for at opsætte flere forskellige parametre for hvornår break pointet skal standse koden:



Hit counters, angiver hvor mange gange programmet skal passere dette break point inden det skal stoppes.

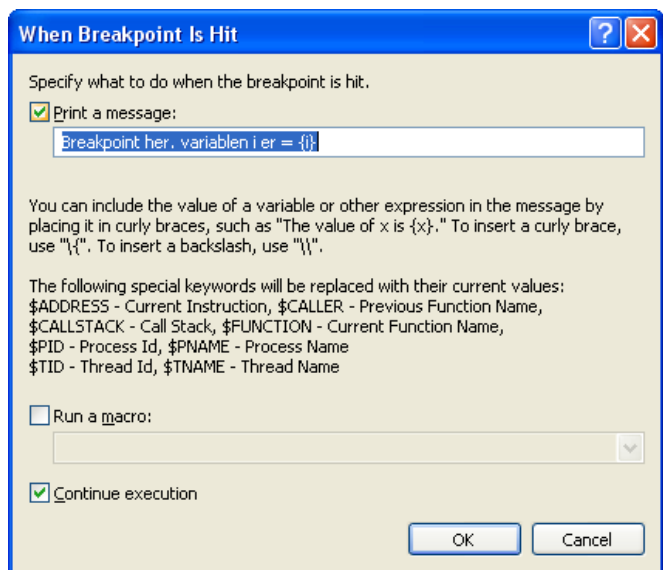
Location giver mulighed for at flytte break pointet ud fra linjenummer og filnavn

Condition. Her kan vi opsætte en logisk tilstand som skal evalueres til sand (som en if statement) Og bedst af alt, kan vi benytte kodens variabler direkte.

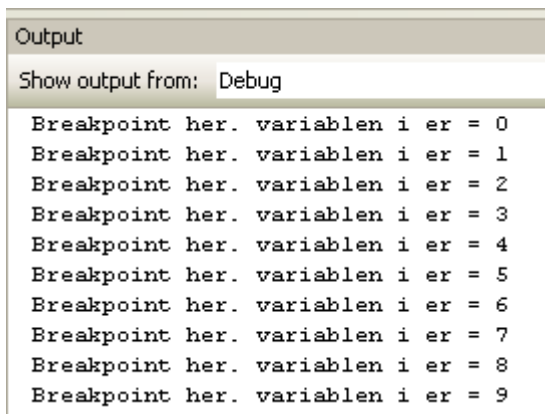


Filter giver nogle af de mest avancerede muligheder. Her kan vi vælge *kun* at standse ud fra process, maskine eller tråd ID.

When Hit er den sidste mulighed og også en af de mere praktiske muligheder vi har. Her kan vi nemlig vælge at skrive en kort besked til "output" vinduet.



Eksemplet her til højre skriver følgende til output vinduet:



Du bør som minimum afprøve Hit counter og When hit parametrene for du vil i fremtiden blive *rigtig* glad for disse to muligheder for at debugge dit program.

Der er selvfølgelig en hel håndfuld yderligere muligheder for at debugge dit program, men med break points og de muligheder du har set her, kan du komme langt, inden du får behov for andet.