

WPF

Praktik Opgave

30 May - 7 June,2022



Af:

Ramya Kailashnathan

TECHNICAL EDUCATION COPENHAGEN (TEC)

June 2022

Indhold

Opgave Logbog	3
Sprøgsmaal.....	4
1. Forklar hvad er WPF?	4
2. Hvad er forskellen på WPF og Windows Forms?	5
3. Hvad er fordelene med WPF?	6
4. Forklar i hvilken sammenhæng man bruger WPF?	8
5. Hvad er XAML?	9
6. Hvad er fordelene med XAML?	10
Øvelse 2. Intro øvelse til WPF.....	11
Example 1: Quiz game	11
1. Problemformulering og løsningsforslag.....	11
2. Mockup og UML af programmet.....	11
3. Beskrive hvilke objekter bliver brugt (Funktionsoversigt)	12
4. Kravspecifikation	13
5. Sourcecode med kommentar	13
6. Screenshots.....	14
7. Konklusion	18
Example 2: Data Binding	19
1. Problemformulering og løsningsforslag.....	19
Hvad er databinding?	19
1. Beskrive hvilke objekter bliver brugt (Funktionsoversigt)	20
2. Kravspecifikation	21
3. Sourcecode med kommentar (One way databinding)	21
4. Screenshots(one way databinding):.....	22
5. Konklusion(One way data binding).....	23
6. Sourcecode med kommentar (Twoway databinding)	23
9. Konklusion(Two way data binding)	25
Example 3 : Triggers(Property Triggers,Event Triggers,Data Triggers)	26
Hvad er Trigger?	26
Hvorfor bruge Trigger?.....	26
Hvor mange typer triggere er der i WPF?	26
1. Problemformulering og løsningsforslag	27
Property Trigger	27



Data Trigger	27
Event Trigger	27
2. Beskrive hvilke objekter bliver brugt (Funktionsoversigt)	27
3. Kravspecifikation	27
4. Sourcecode med kommentar	29
5. Konklusion	33



Indledning

Windows Presentation Foundation er en UI-ramme, der skaber desktop-klientapplikationer. WPF-udviklingsplatformen understøtter et bredt sæt af applikationsudviklingsfunktioner, herunder en applikationsmodel, ressourcer, kontroller, grafik, layout, databinding, dokumenter og sikkerhed.

WPF er en del af .NET, så hvis du tidligere har bygget applikationer med .NET ved hjælp af ASP.NET eller Windows Forms, bør programmeringsoplevelsen være bekendt. WPF bruger Extensible Application Markup Language (XAML) til at give en deklarativ model for applikationsprogrammering.

Her har jeg præsenteret tre eksempler. En quiz, hvor jeg har implementeret UI-designet og koden separat. Der er mange koncepter i WPF, men jeg har dækket databinding og triggere.

Opgave Logbog

Uge	22-23					
Dato	30	31	1	2	3	7
Dag	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Mandag
Læsning og forståelse af opgaven, video						
Programmering						
Report skrivning						

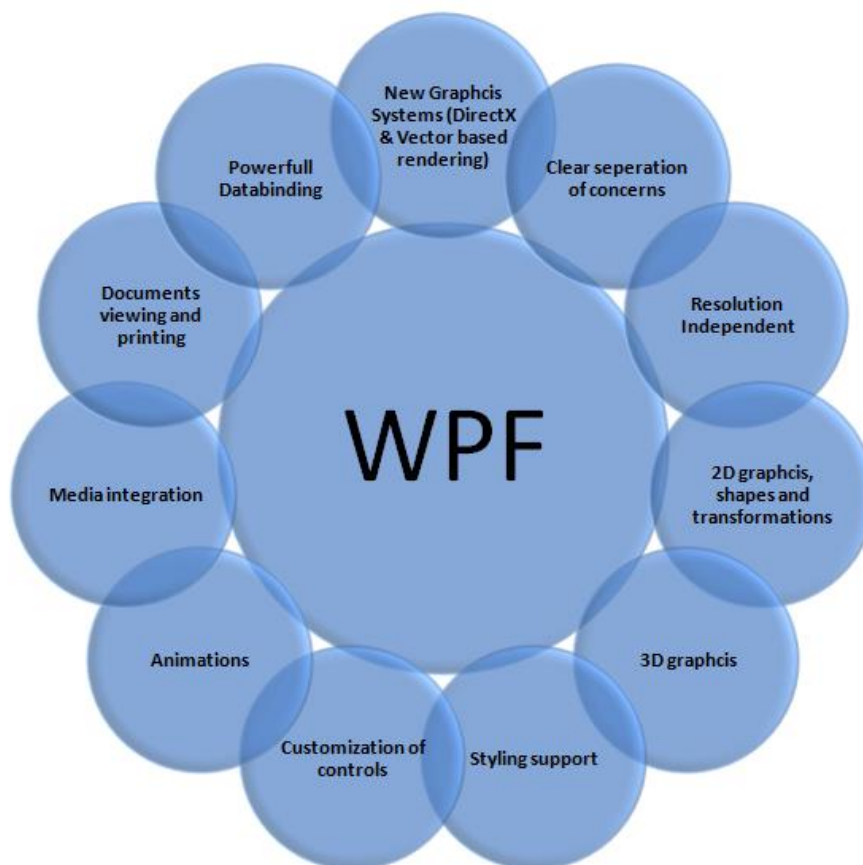


Sprøgsmål

1. Forklar hvad er WPF?

Windows Presentation Foundation er en UI-framework, der skaber desktop-klientapplikationer. WPF-udviklings platformen understøtter et bredt sæt af applikationsudviklingsfunktioner, herunder en applikationsmodel, ressourcer, kontroller, grafik, layout, databinding, dokumenter og sikkerhed.

WPF er en del af .NET, så hvis du tidligere har bygget applikationer med .NET ved hjælp af ASP.NET eller Windows Forms, bør programmeringsoplevelsen være bekendt. WPF bruger Extensible Application Markup Language (XAML) til at give en deklarativ model for applikationsprogrammering.



2. Hvad er forskellen på WPF og Windows Forms?

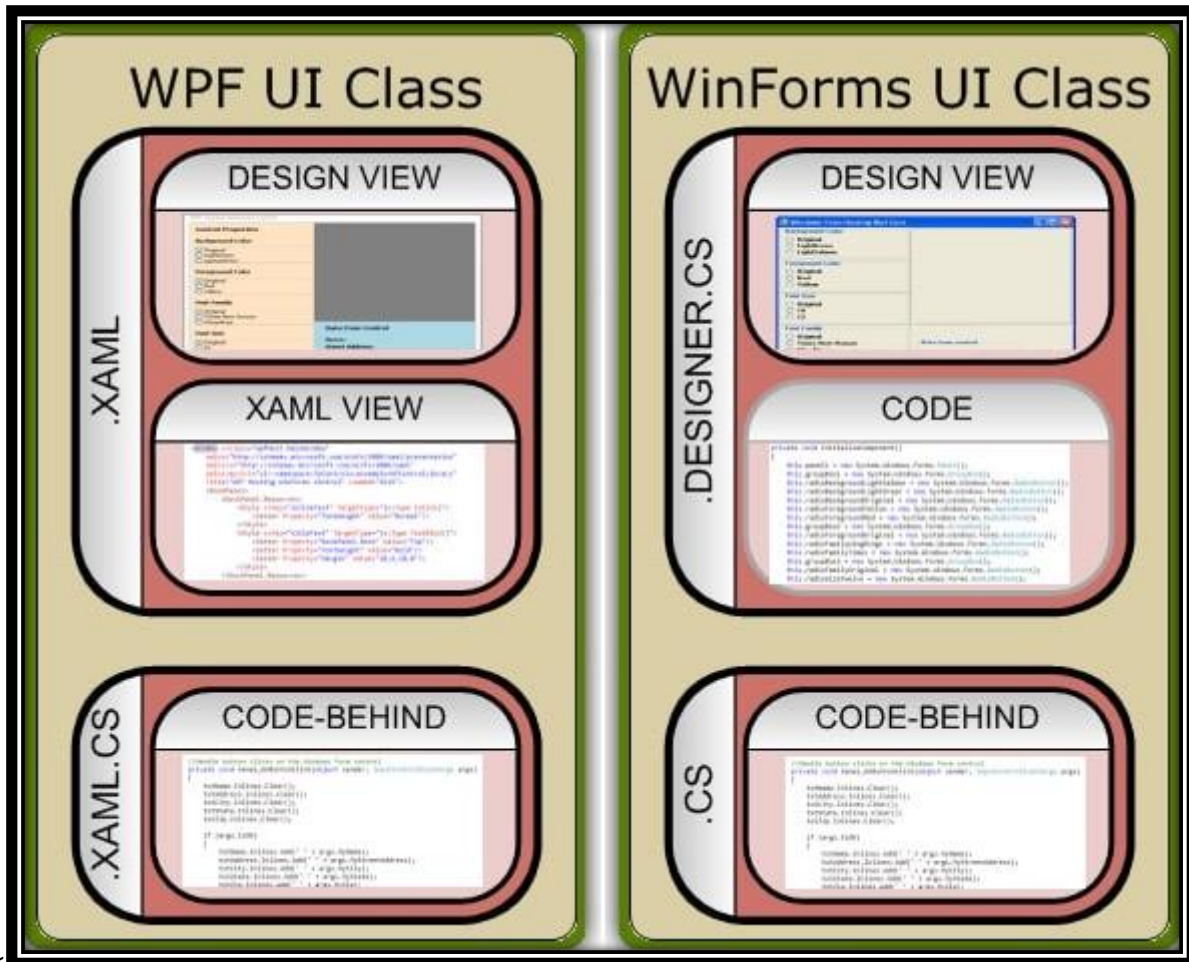
W.P.F refererer blot til Microsofts Windows Presentation Foundation, og WinForms er en simpel sammenkædning af Windows Forms Applications. Disse er begge Microsofts grafiske brugergrænseflader for Windows-applikationer, som udviklere kan bruge til at udvikle Windows-desktopapplikationer.

Windows-formularer WinForms blev introduceret i februar 2002 som en GUI baseret tilgang til .Net Framework. WinForms tillader stort set udvikleren at trække og slippe kontrolelementer på en Windows-formular og giver udvikleren mulighed for at manipulere disse kontrolelementer med en kode-bag-fil, der kan have C#, VB.NET eller et hvilket som helst andet .NET-sprog.

Hver WinForms kontrol er en forekomst af en klasse, fordi WinForms eksisterer som en wrapper, der har et sæt C++ klasser. Microsofts Visual Studio muliggør en lettere løsning med WinForms, da udviklere nemt kan trække og slippe kontrolelementer fra en værktøjskass.



3. Hvad er fordelene med WPF?



1. Windows-forms er ikke vektorbaserede brugergrænseflader. Hvorimod WPF er et vektorgrafikbaseret UI-præsentationslag. Ved hjælp af at være vektorbaseret tillader det præsentationslaget at skalere UI-komponenterne jævnt uden at have problemer med størrelsesforvrængning.
2. Windows-forms er nemmere at bruge under udvikling af applikationer, hvorimod WPF er lidt svær at bruge, da det kræver god viden at bruge kontrollerne.
3. I Windows-forms kan vi tilpasse kontrollerne i henhold til kravene. I WPF har vi også tredjepartskontroller for at berige applikationens muligheder.



4. Windows-forms har mindre indlæringskurve. Hvorimod WPF har mere indlæringskurve efter behov for at forstå det fulde flow af kontroller og designdelen.
5. Windows-forms er mindre tidskrævende eller mindre vanskelige. WPF er sværere og mere tid at forbruge for at få tingene på plads, mens applikationerne udvikles.
6. Windows-forms bliver ikke brugt til at udvikle nye applikationer. WPF bliver primært brugt til at udvikle nye applikationer.
7. Windows-forms er stor støtte i form af udviklere, online-fællesskab, biblioteker til at hjælpe i enhver form, mens de udvikler applikationen til begyndere. WPF har også tilstrækkelig support og biblioteker til rådighed til at udvikle applikationerne og få supporten hurtigt for begyndere.
8. I Windows-forms er kontrolelementer vanskelige at tilpasse, hvorimod kontrolelementer i WPF nemt kan tilpasses, da det er fuldstændigt skrevet fra bunden. Windows-formularer er dårlige til at give konsistens. WPF giver mere konsistens på tværs af applikationerne.
9. I Windows-forms er brugergrænsefladen designet ved hjælp af sproget i forretningslogikkoden. I WPF bruger den XAML som et markup-sprog til at designe UI-delen af applikationen.
10. Windows-forms er hovedsageligt baseret på en pixel, hvorimod WPF ikke er pixel-baseret, hvilket tillader skalerbarheden af UI-delen til applikationen. Windows-forms understøtter databinding på en begrænset måde, hvorimod WPF er fuldt understøttet databinding.
11. Windows-forms bruges ikke med forskellige temaer eller skins. WPF er hovedsageligt skinnable eller temaable, hvor forskellige skins eller temaer kan bruges til UI.
12. Windows-forms kræver mindre indsats for at designe brugergrænsefladen. WPF kræver mere indsats, da det meste af arbejdet skal udføres selv.



4. Forklar i hvilken sammenhæng man bruger WPF?

Nem adskillelse af Business Logic og UI

Microsoft designede WPF fra bunden til løst at koble præsentation og forretningslogik. Denne arkitektur giver dig mulighed for at bruge designmønstre som Model View Controller (MVC) eller Model View View Model (MVVM) i din udvikling. Disse mønstre adskiller ikke kun din forretningslogik og brugergrænseflade, de gør det også muligt for dig at enhedsteste hver linje kode i din applikation. Selvom du kan få en adskillelse af forretningslogik og brugergrænseflade med Windows Forms, gør krogene i WPF denne adskillelse meget nemmere.

Animation

Animation gør din applikation mere brugervenlig og selvdokumenterende. WPF gør det meget nemt at lave simple animationer. Faktisk kan du mange gange udtrykke animation i XAML, og du behøver ikke at skrive nogen traditionel VB eller C #-kode.

Skærmopløsning

WPF har indbyggede faciliteter til at håndtere forskellige skærmopløsninger svarende til den måde HTML fungerer med skærmopløsning. Men ligesom traditionelle Window Forms eller HTML kan du også layoute skærme forkert og ende med applikationer, der ikke skaleres på forskellige skærmopløsninger. Men mulighederne for at skalere er indbygget i WPF.

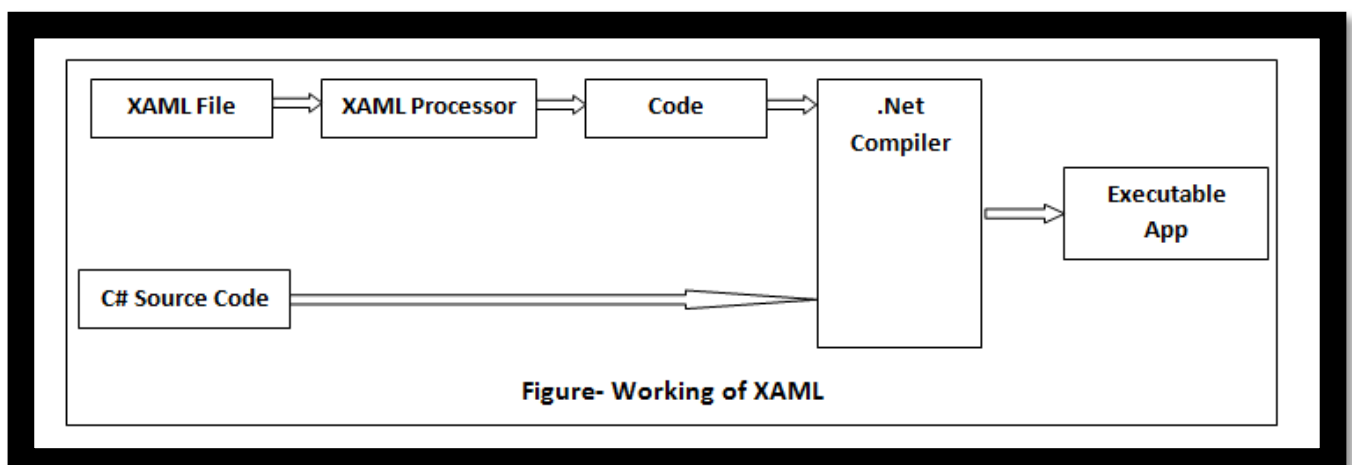
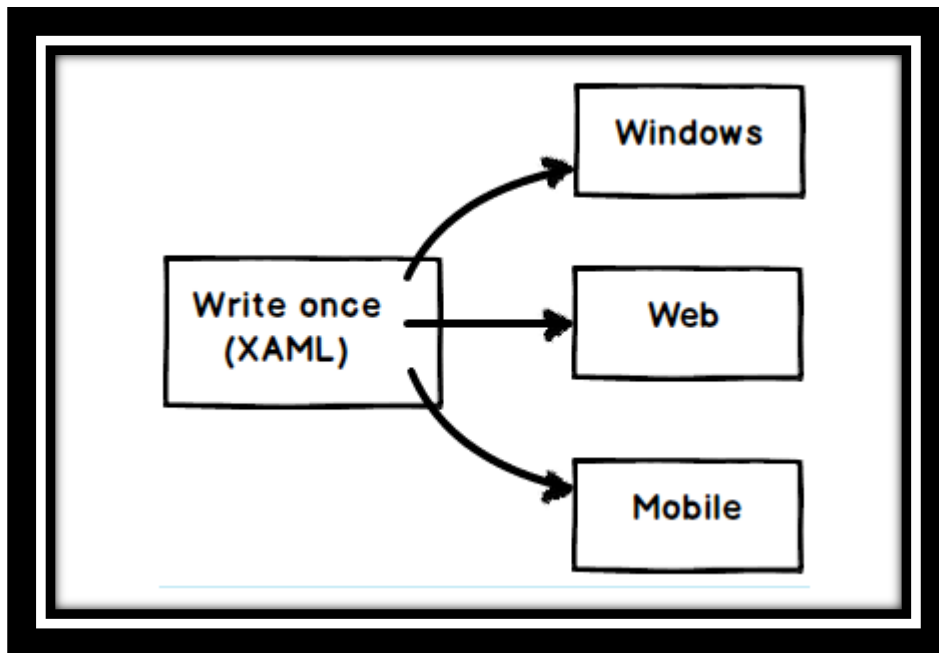
Databinding

Med WPF leverede Microsoft meget mere robust databinding, og der findes mange flere hooks, der tillader den nødvendige fleksibilitet til komplicerede forretningsapplikationer. Du vil finde dig selv at bruge XAML til at udtrykke den databinding, som du ville have skullet skrive kode til i Windows Forms-applikationer. Dette vil øge din produktivitet enormt, når du begynder at drage fordel af disse nye teknikker i WPF.



5. Hvad er XAML?

XAML er en XML-fil, som repræsenterer din WPF UI. Hele pointen med at skabe UI-repræsentationen i XML var at skrive én gang og køre den hvor som helst. Så den samme XAML UI kan gengives som Windows-applikation med WPF, og den samme UI kan vises på browseren ved hjælp af WPF-browser eller Silverlight-applikation.



Følgende figur beskriver virkemåden af XAML:

- En platformsspecifik XAML-processor fortolker .xaml-filen.
- XAML-processor konverterer XAML til intern kode, der beskriver UI-elementet.
- Den interne kode og C # kode kombineret gennem klassernes definition og derefter .Net compiler bygger appen.

6. Hvad er fordelene med XAML?

Fordele ved XAML

- XAML er baseret på XML
- XAML programmør og designer kan arbejde parallelt.
- XAML-koden er meget let at forstå.
- Giv en klar adskillelse mellem UI (XAML) og UI-logik (C #).
- Avanceret databinding



Øvelse 2. Intro øvelse til WPF

Example 1: Quiz game


1. Problemformulering og løsningsforslag

Brugeren præsenteres for seks spørgsmål, og hvert spørgsmål har fire mulige svar. Brugeren føres videre til næste spørgsmål, selvom hans svar er forkert. Det korrekte svarantal vises for brugeren på skærmen.

Efter det sjette spørgsmål får brugeren en pop-op-skærm for at afslutte quizen.

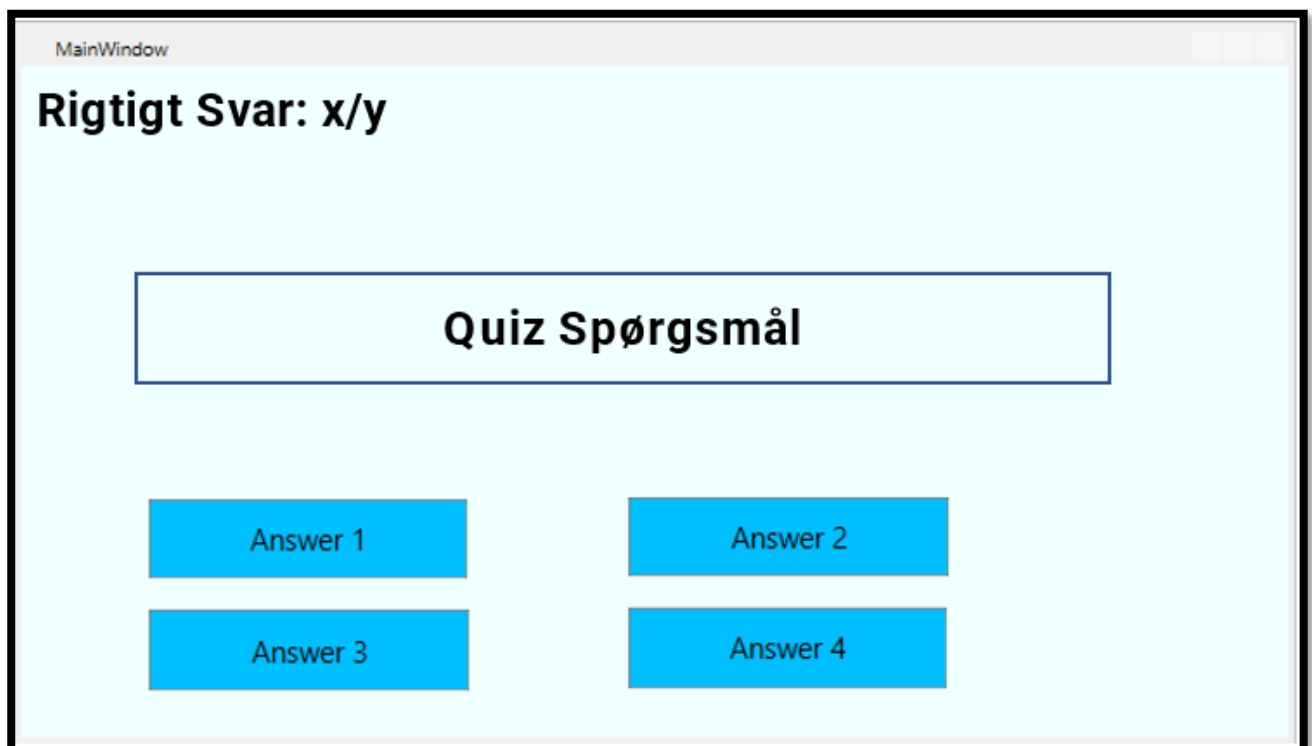
2. Mockup og UML af programmet

Use case Diagram:

Use case: Quiz Program 	
ID: 1	
Description:	En quiz med seks spørgsmål, hvor det rigtige antal svar vises på skærmen.
Primary Actors:	User
Secondary Actors:	None
Preconditions:	Alle spørgsmål skal være hårdkodet i C #-programmet i switch-case statement.
Trin:	<ol style="list-style-type: none">1. Brugeren vil være i stand til at se det første spørgsmål i quizen.2. Der vises fire knapper på knapperne.3. Uanset et klik på knappen (rigtigt eller forkert svar), vises næste spørgsmål.4. Efter det første spørgsmål vises det korrekte svarantal i venstre hjørne af skærmen.5. Den samme proces fortsættes for alle de seks spørgsmål.6. Efter at brugeren har besvaret det sjette spørgsmål, vises en pop-up-boks med en exit-knap, der beder brugeren om at afslutte quizen
Post Conditions:	<ul style="list-style-type: none">• None



Mockup screen:



3. Beskrive hvilke objekter bliver brugt (Funktionsoversigt)

Partial class- vi har brugte partial class hvor du kan opdele design code or Business logic for at læse og forstå koden.

Kan også vedligeholde din applikation på en effektiv måde ved at komprimere store klasser til små.

MainWindow() – Denne method har InitializeComponent(), en entry point i en WPF application.



checkAnswer() - tjek tag-værdi i XAML-filen

RestartGame() - genstart spil-funktionen vil indlæse alle standardværdierne for dette spil

NextQuestion() - denne funktion vil kontrollere, hvilket spørgsmål der skal vises næste gang, og det vil have alle spørgsmål og svar

4. Kravspecifikation

Brugeren præsenteres for seks spørgsmål, og hvert spørgsmål har fire mulige svar. Brugeren føres videre til næste spørgsmål, selvom hans svar er forkert. Det korrekte svarantal vises for brugeren på skærmen.

Efter det sjette spørgsmål får brugeren en pop-op-skærm for at afslutte quizen.

5. Sourcecode med kommentar



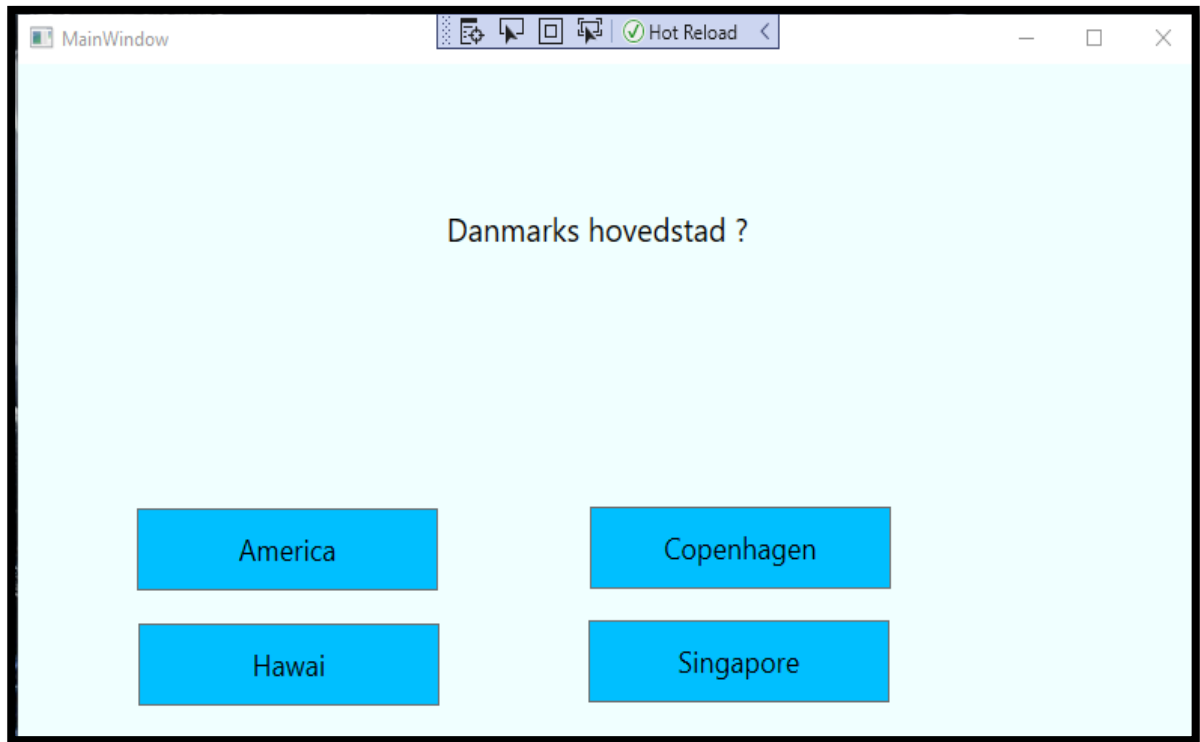
Quiz XAML Code

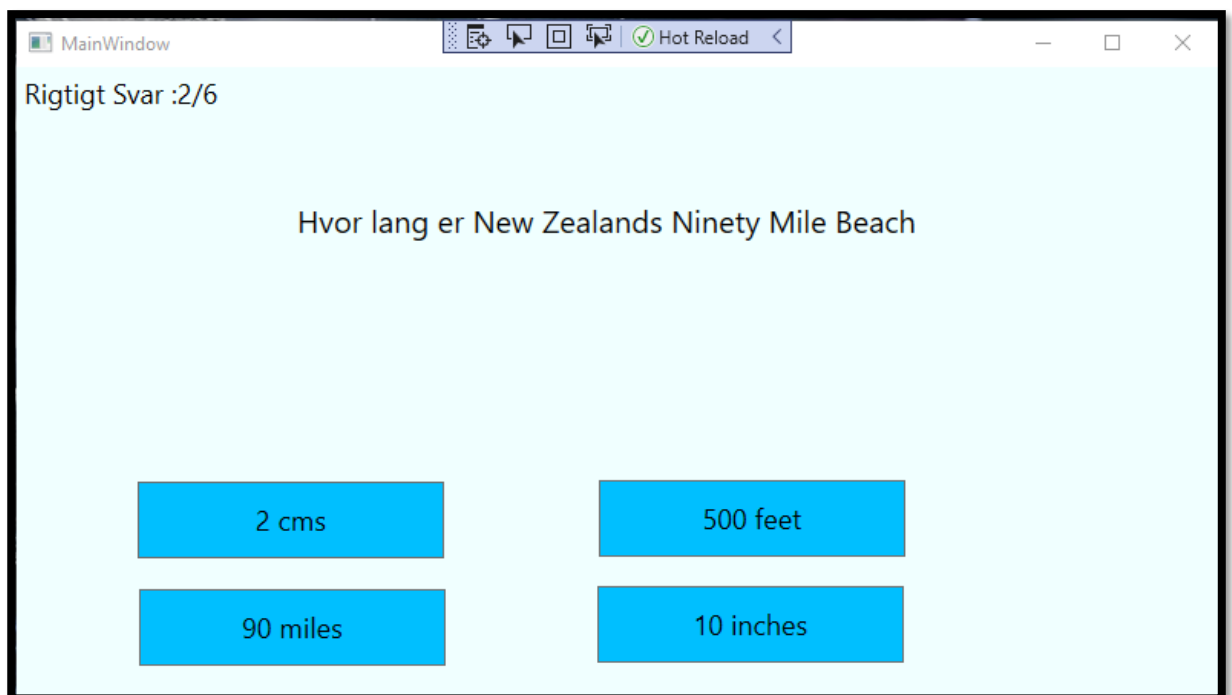
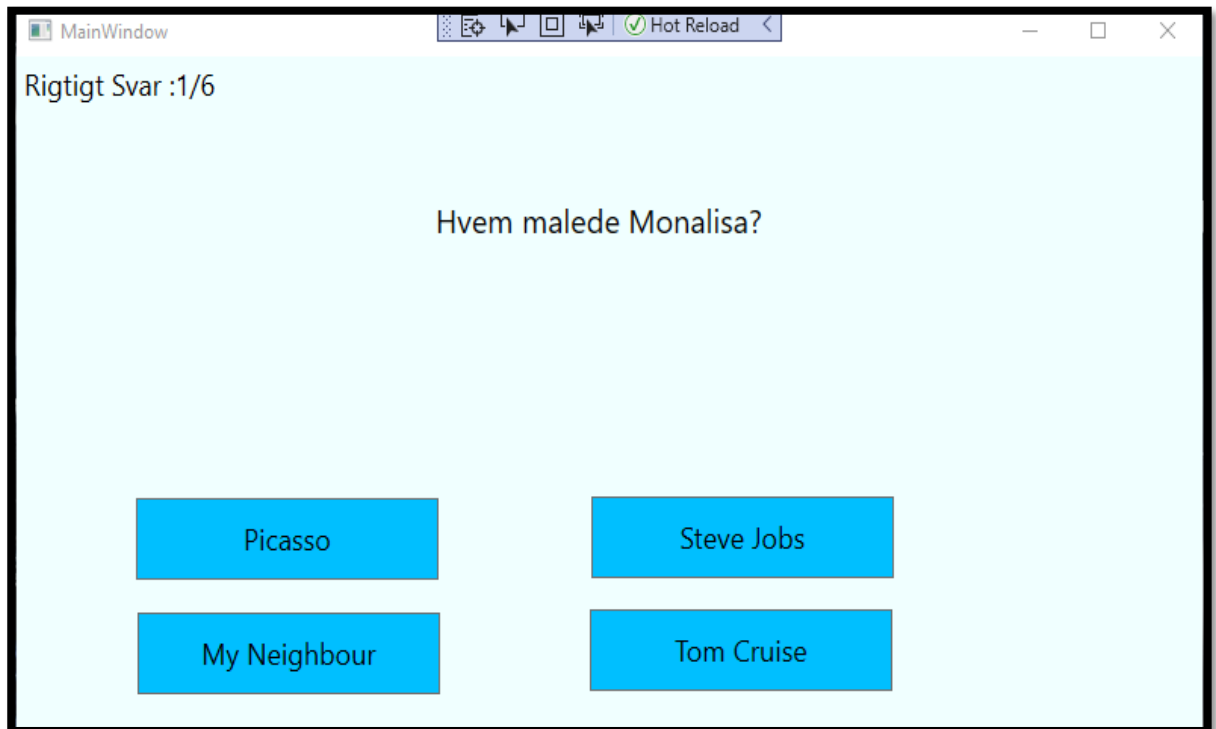


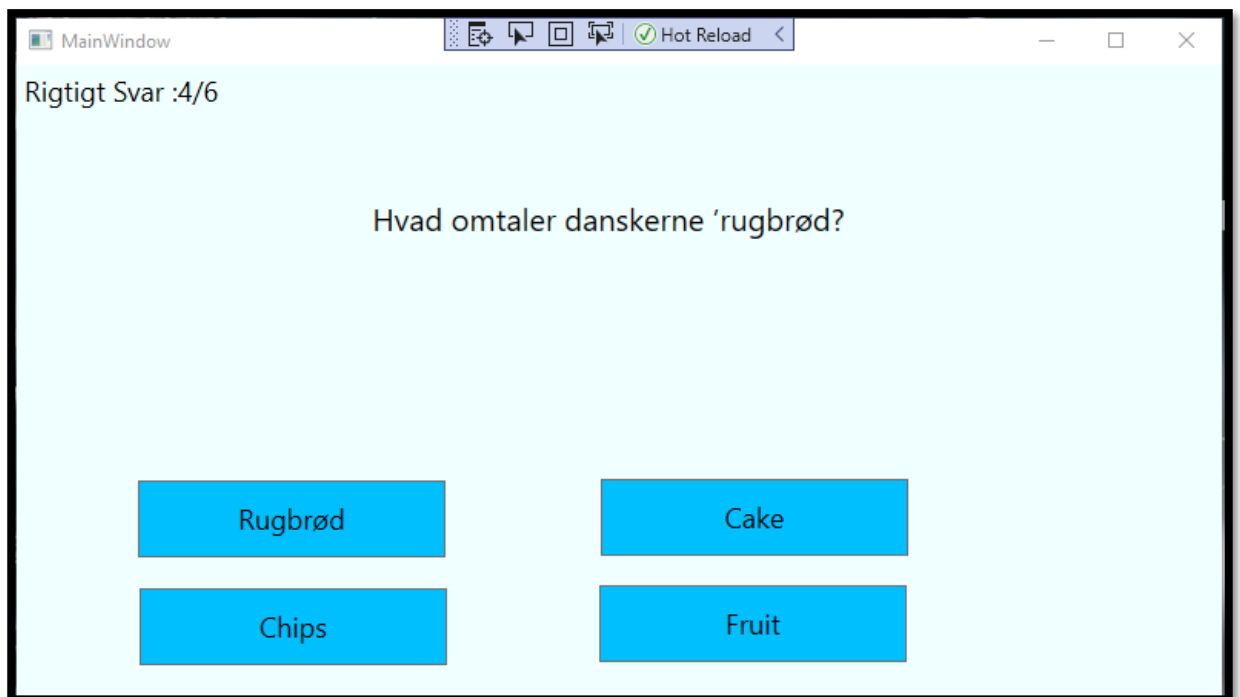
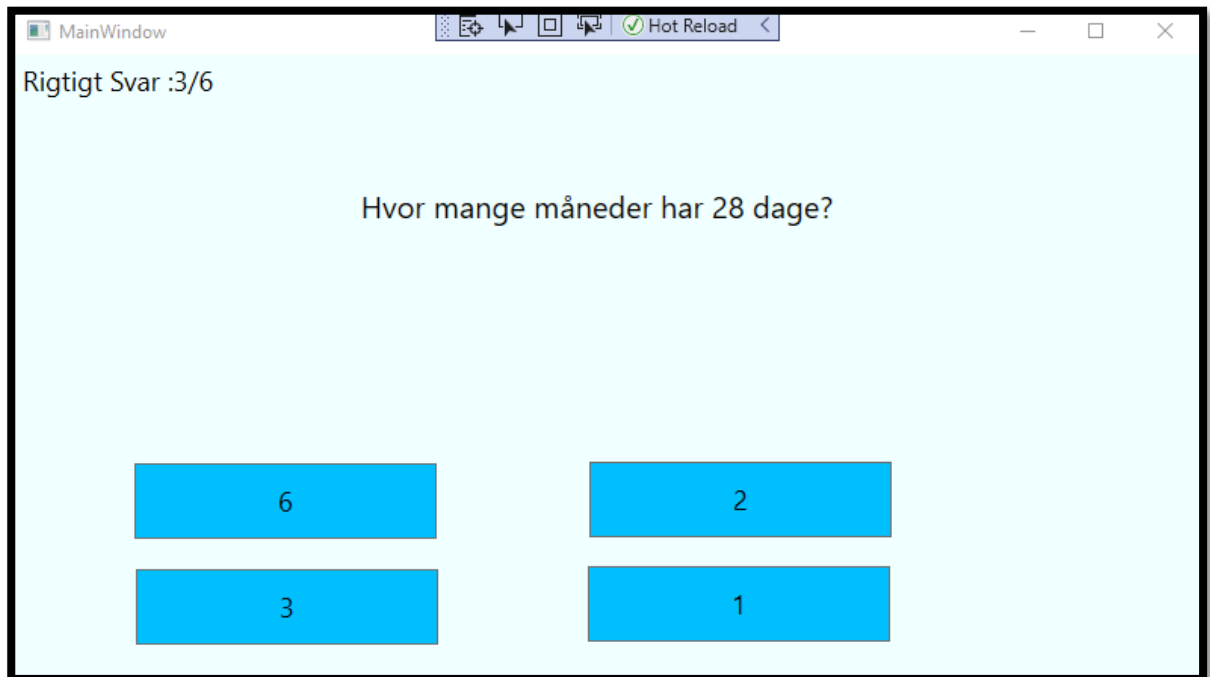
Quiz C# Code

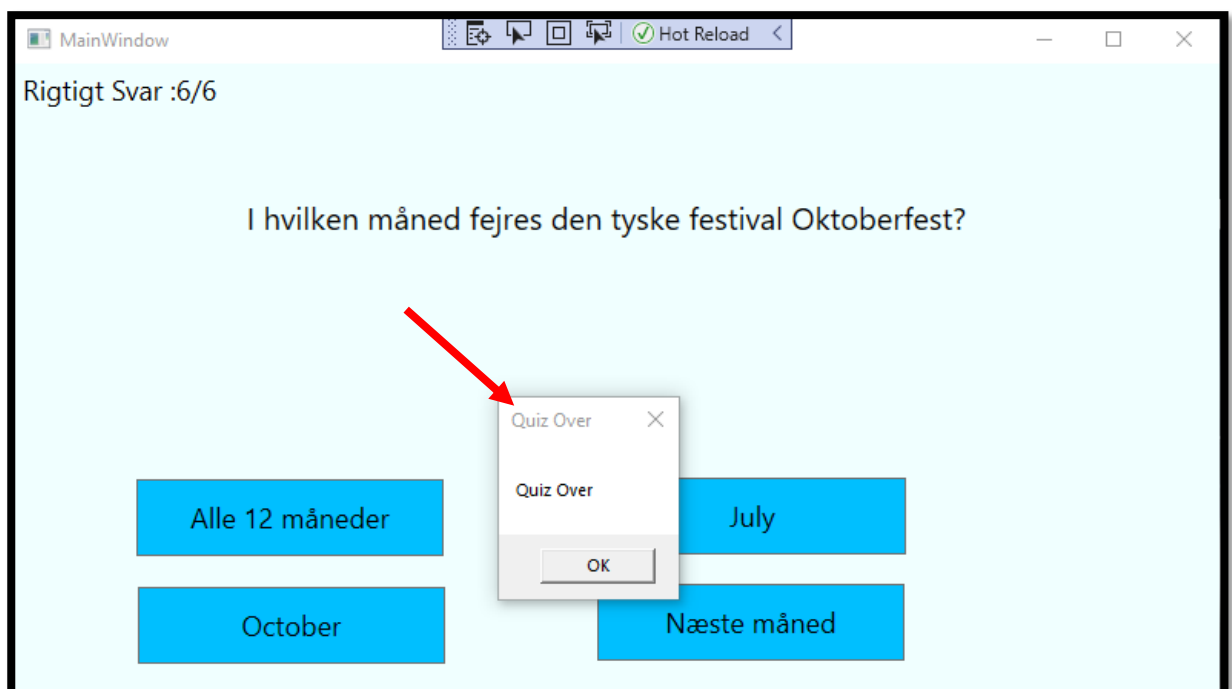


6. Screenshots









7. Konklusion

Brugeren præsenteres for seks spørgsmål, og hvert spørgsmål har fire mulige svar. Brugeren føres videre til næste spørgsmål, selvom hans svar er forkert. Det korrekte svarantal vises for brugeren på skærmen.

Efter det sjette spørgsmål får brugeren en pop-op-skærm for at afslutte quizzen.

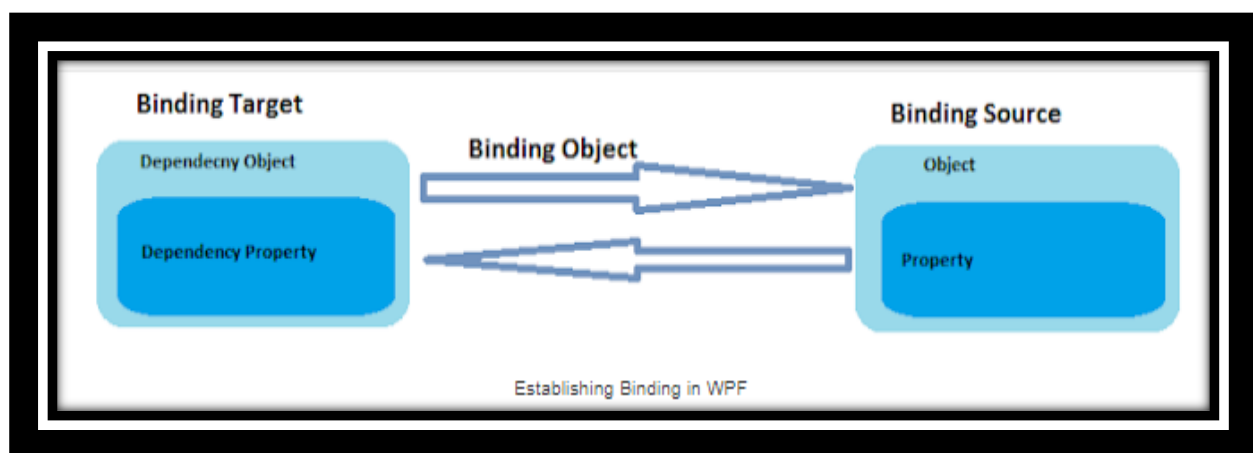


Example 2: Data Binding

1. Problemformulering og løsningsforslag

Vi er nødt til at vise databinding One way- og Two way databinding. Vi kan også se, hvordan data forbliver de samme på one way databinding, og hvordan data bliver opdateret i two way databinding.

Hvad er databinding?



Databinding er en mekanisme i WPF-applikationer, der giver en enkel og nem måde for Windows Runtime-apps at vise og interagere med data. I denne mekanisme er håndteringen af data fuldstændig adskilt fra den måde, data på.

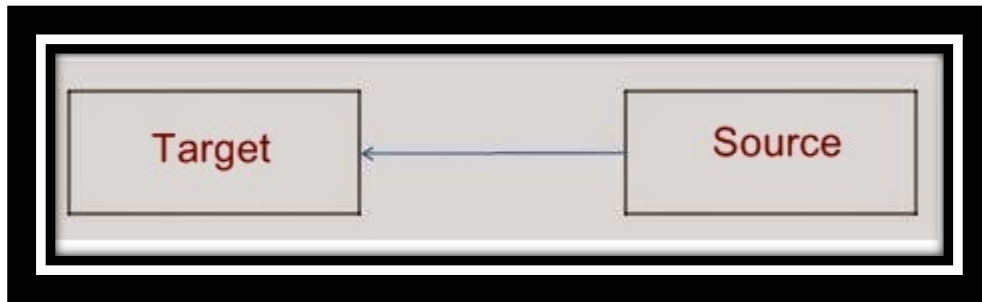
Databinding tillader strømmen af data mellem UI-elementer og dataobjekt på brugergrænsefladen. Når en binding er etableret, og dataene eller din forretningsmodel ændres, så afspejler det opdateringerne automatisk til UI-elementerne og omvendt. Det er også muligt at binde, ikke til en standarddatakilde, men til et andet element på siden.

Databinding er af to typer – One way databinding og Two way databinding.

Oneway databinding

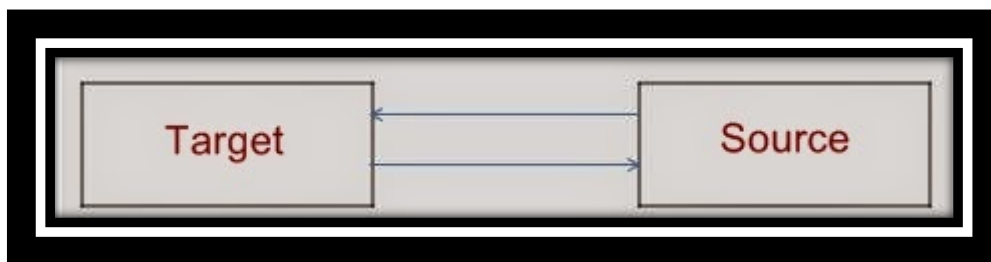
Ved Oneway binding er data bundet fra dets kilde (det er det objekt, der holder dataene) til dets mål (det er det objekt, der viser dataene).





Twoway databinding

Ved Twoway databinding kan brugeren ændre dataene gennem brugergrænsefladen og få disse data opdateret i kilden. Hvis kilden ændres, mens brugeren ser på visningen, ønsker du, at visningen skal opdateres.



1. Beskrive hvilke objekter bliver brugt (Funktionsoversigt)

One way databinding funktioner:

```
<TextBox x:Name = "countryText" Grid.Column = "1" Margin = "2"  
Text = "{Binding Country, Mode = OneWay}"/>  
  
<Label x:Name = "capitalLabel" Margin = "2" Grid.Row = "1" FontWeight="Bold">_CapitalCity:</Label>  
  
<TextBox x:Name = "capitalText" Grid.Column = "1" Grid.Row = "1" Margin = "2"  
Text = "{Binding City, Mode = OneWay}"/>
```



Two way databinding functioner:

```
<TextBox x:Name = "countryText" Grid.Column = "1" Margin = "2"  
Text = "{Binding Country, Mode = TwoWay}"/>  
<Label x:Name = "capitalLabel" Margin = "2" Grid.Row = "1" FontWeight="Bold">_CapitalCity:</Label>  
<TextBox x:Name = "capitalText" Grid.Column = "1" Grid.Row = "1" Margin = "2"  
Text = "{Binding City, Mode = TwoWay}"/>
```

2. Kravspecifikation

Vi tjekker, hvordan envejs databinding og tovejs databinding fungerer.

Oneway databinding - outputtet på messagebox opdateres ikke, når værdien i tekstbox ændres.

Twoway databinding - outputtet på messagebox opdateres, når værdien i tekstbox ændres.

3. Sourcecode med kommentar (One way databinding)



Oneway

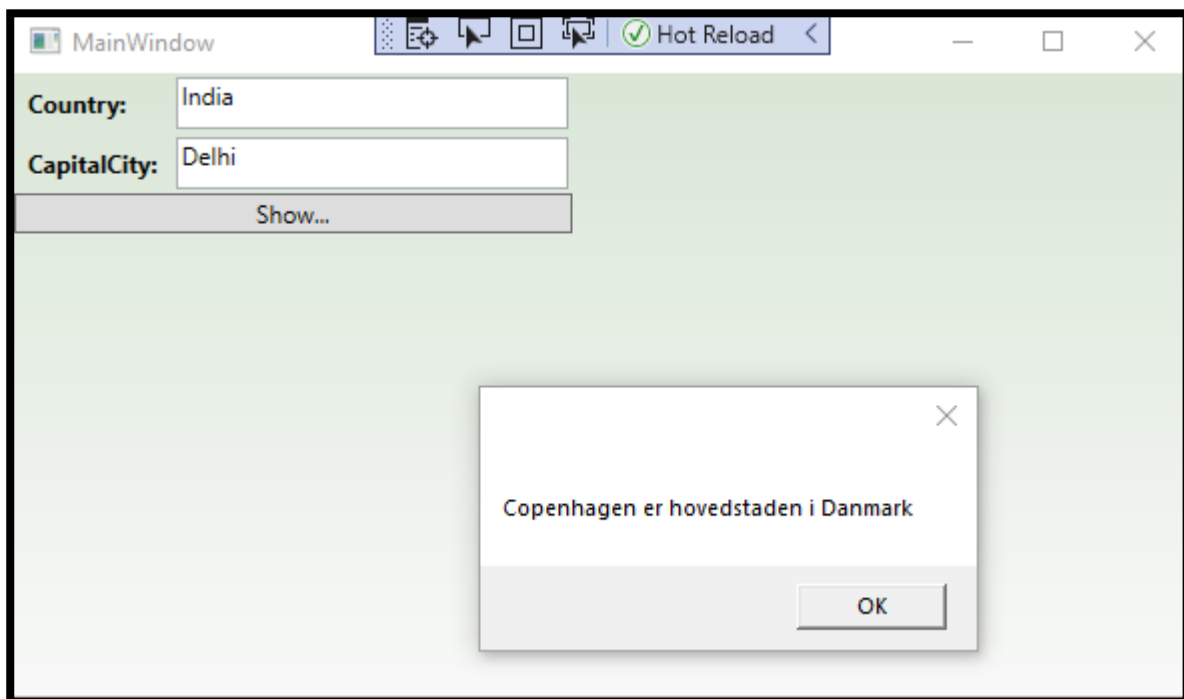
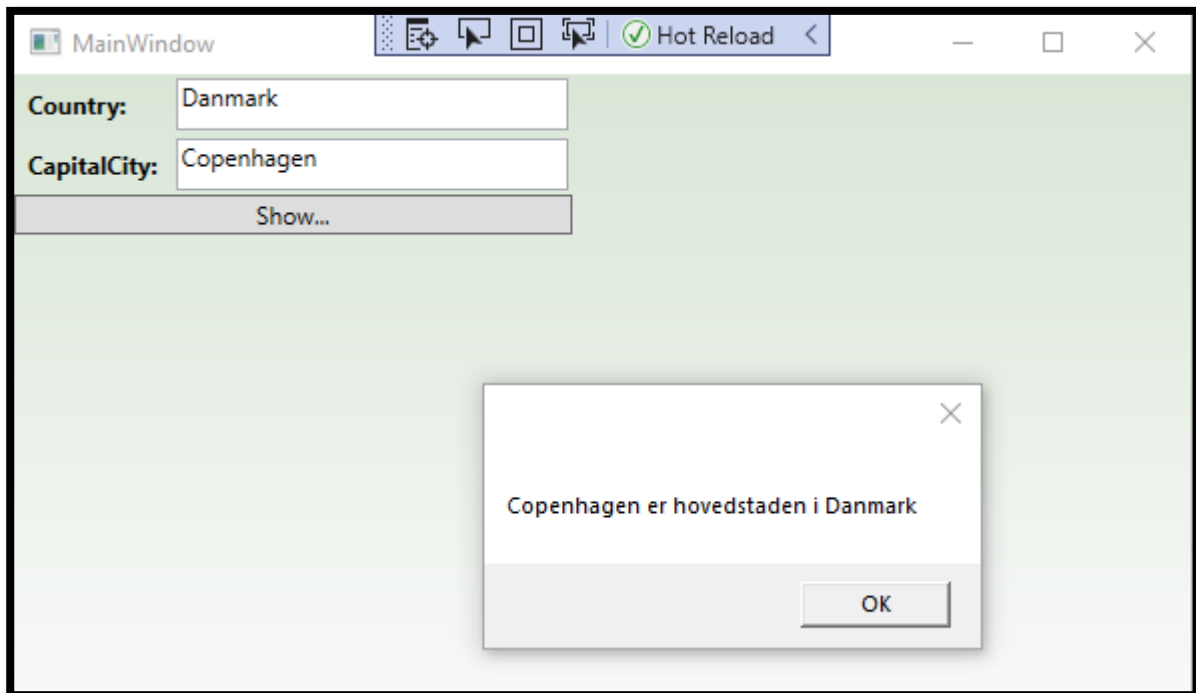


Oneway

Databinding XAML cDatabinding C# Coc



4. Screenshots(one way databinding):



5. Konklusion(One way data binding)

På one way databinding opdateres outputtet ikke. Men på two way databinding opdateres outputtet.

6. Sourcecode med kommentar (Twoway databinding)



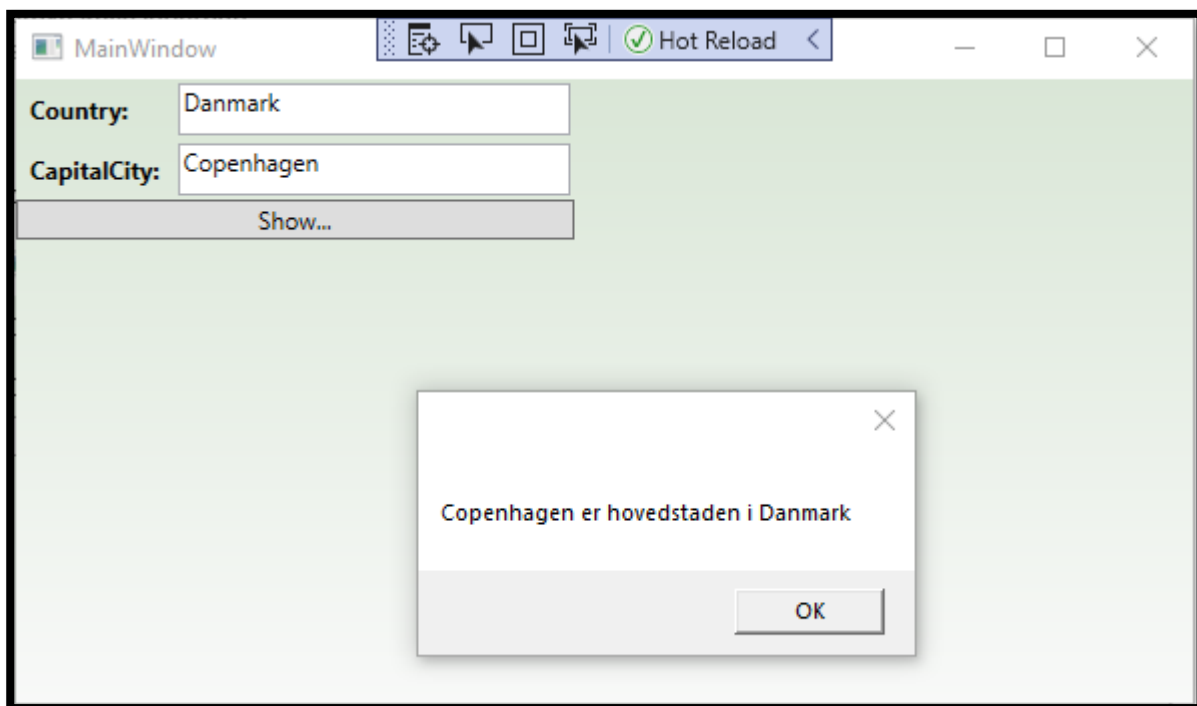
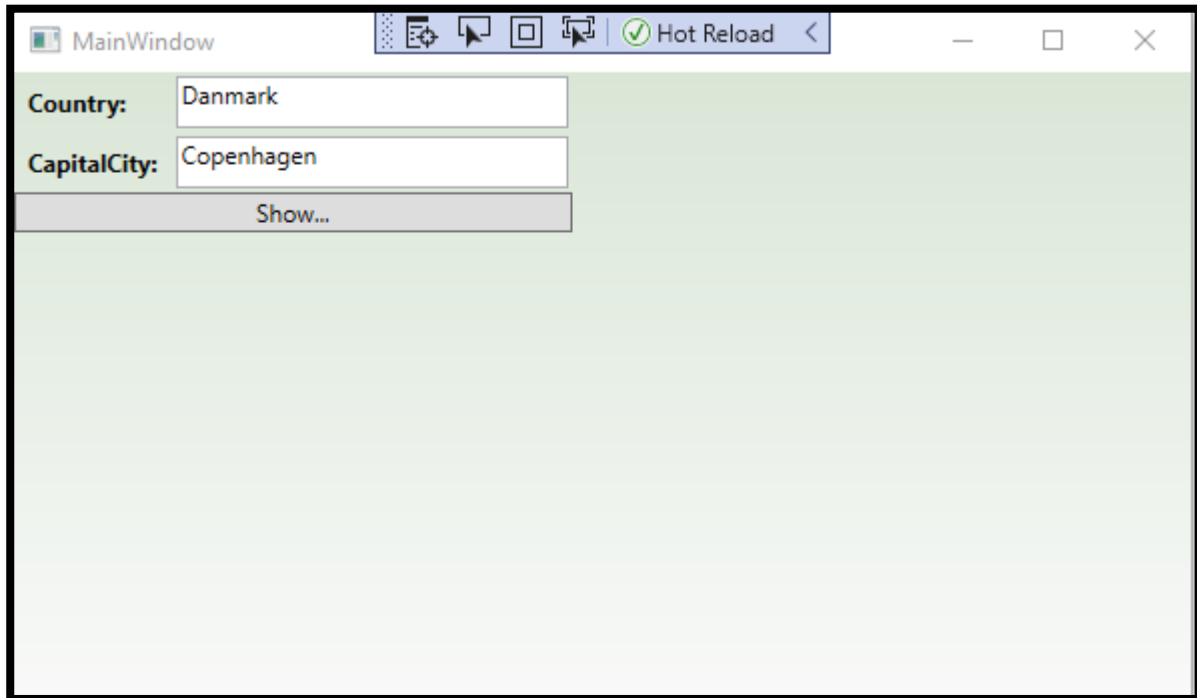
Twoway
Databinding Xaml code

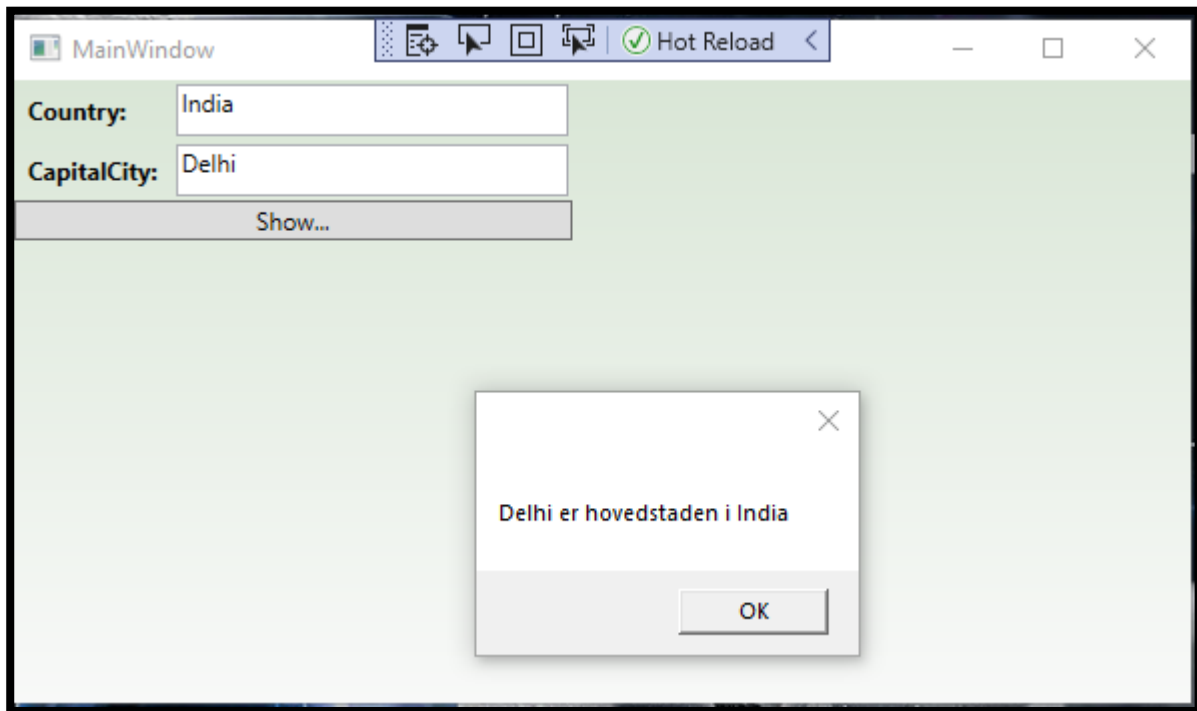


Twoway Binding C#
Code

Screenshots(two way databinding):







9. Konklusion(Two way data binding)

På one way databinding opdateres outputtet ikke. Men på two way databinding opdateres outputtet.



Example 3 : Triggers(Property Triggers,Event Triggers,Data Triggers)

Hvad er Trigger?

En trigger bruges typisk i en stilart eller kontrolskabelon. Den udløser egenskaber for det, der skabes, og indstiller andre egenskaber for kontrolelementet (eller specifikke skabelonelementer). For eksempel vil du bruge en Trigger på IsMouseOver til at reagere på, at musen er over kontrollen, og indstillingerne kan opdatere en pensel for at vise en "hot" effekt.

Hvorfor bruge Trigger?

Triggere bruges i stile til at udføre handlinger på en ændring af en ejendomsværdi eller hændelsesbrand. Triggere skaber visuelle effekter på kontroller. Ved at bruge Triggers kan vi ændre udseendet af rammeelementer.

Hvor mange typer triggere er der i WPF?

Der er fem typer triggere, der understøttes af WPF; de er:

- 1. Property trigger**
- 2. Data Trigger**
- 3. MultiTrigger**
- 4. MultiDataTrigger**
- 5. Event trigger**



Jeg har brugt kun tre triggers(**Property trigger ,Data Trigger,Event Trigger**).

1. Problemformulering og løsningsforslag

Property Trigger

Den enkleste form for en udløser er en egenskabstrigger, der holder øje med, om en afhængighedsejendom har en vis værdi. For eksempel, hvis vi ønskede at lyse en knap i gult, når brugeren bevæger musen hen over den, kan vi gøre det ved at se efter, om `IsMouseOver`-egenskaben har en værdi på "True".

Data Trigger

`DataTriggers` er triggere, der overvåger en bundet værdi i stedet for en afhængighedsegenskab. De giver dig mulighed for at se et bundet udtryk og vil reagere, når denne binding vurderes lig med din værdi.

Event Trigger

`EventTrigger` bruges til at udløse handlinger som reaktion på hændelser.

For eksempel: antag, at der er behov for at affyre en kommando ved at klikke på knappen eller på valghændelsen i en `ComboBox` og vælge en række af en gittervisning i WPF. Du kan derefter bruge en `EventTrigger`.

2. Beskrive hvilke objekter bliver brugt (Funktionsoversigt)

Vi har brugt: `IsMouseOver`, `IsChecked`, og `FontSize`

3. Kravspecifikation

En **DataTrigger** giver dig mulighed for at indstille egenskabsværdier, når egenskabsværdien for dataobjektet matcher en specificeret værdi.

Her, når afkrydsningsfeltet ikke er markeret, er tekstfarven rød, og når afkrydsningsfeltet er markeret, er tekstfarven grøn.



```
<Style TargetType="TextBlock">
  <Setter Property="Text" Value="Not Checked" />
  <Setter Property="Foreground" Value="■"Red" />
  <Style.Triggers>
    <DataTrigger Binding="{Binding ElementName=cbSample, Path=IsChecked}" Value="True">
      <Setter Property="Text" Value="Checked" />
      <Setter Property="Foreground" Value="■"Green" />
    </DataTrigger>
  </Style.Triggers>
</Style>
```

En **EventTrigger** udfører en handling, når en specifik hændelse udløses. Det bruges normalt til at udføre nogle animationer såsom DoubleAnimation, ColorAnimation osv.

Her udløses **MouseEnter-** og **MouseLeave**-udløserne, når musen er over teksten på skærmen.

```
<EventTrigger RoutedEvent="MouseEnter">
  <EventTrigger.Actions>
    <BeginStoryboard>
      <Storyboard>
        <DoubleAnimation Duration="0:0:0.300" Storyboard.TargetProperty="FontSize" To="28" />
      </Storyboard>
    </BeginStoryboard>
  </EventTrigger.Actions>
</EventTrigger>
```

I **Property Trigger**, når en ændring sker i en ejendom, vil det resultere i enten en øjeblikkelig eller en animeret ændring i en anden ejendom.

Her kan du bruge en Property Trigger som **IsMouseOver** til at ændre tekstfarven, når musen er over teksten.

```
<Style TargetType="{x:Type TextBlock}">
  <Setter Property="Foreground" Value="■"Blue"/>
  <Style.Triggers>
    <Trigger Property="IsMouseOver" Value="True">
      <Setter Property="Foreground" Value="■"Red" />
      <Setter Property="TextDecorations" Value="Underline" />
    </Trigger>
  </Style.Triggers>
</Style>
```



4. Sourcecode med kommentar



Data Trigger XAML
code

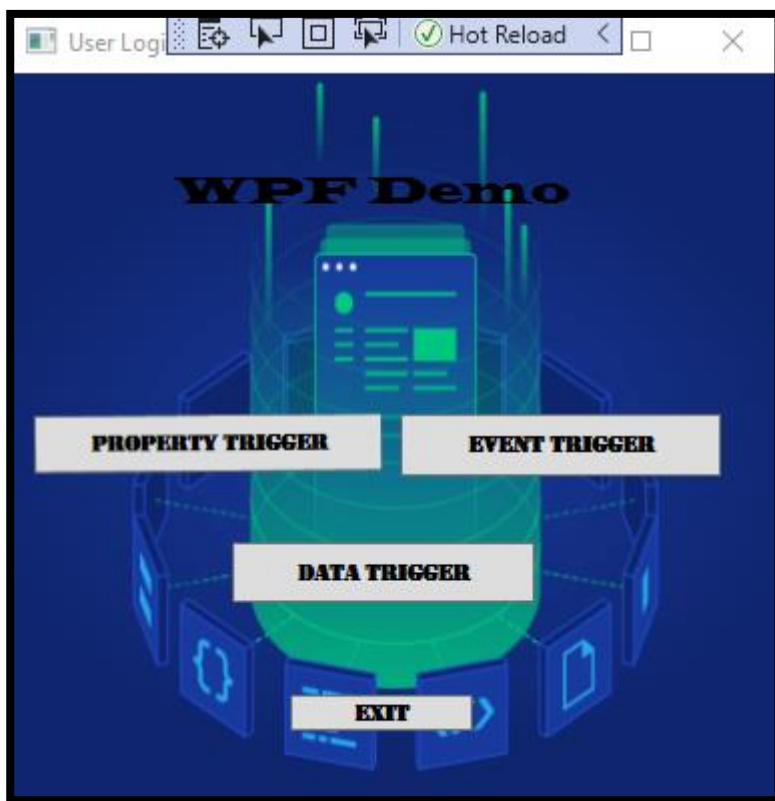


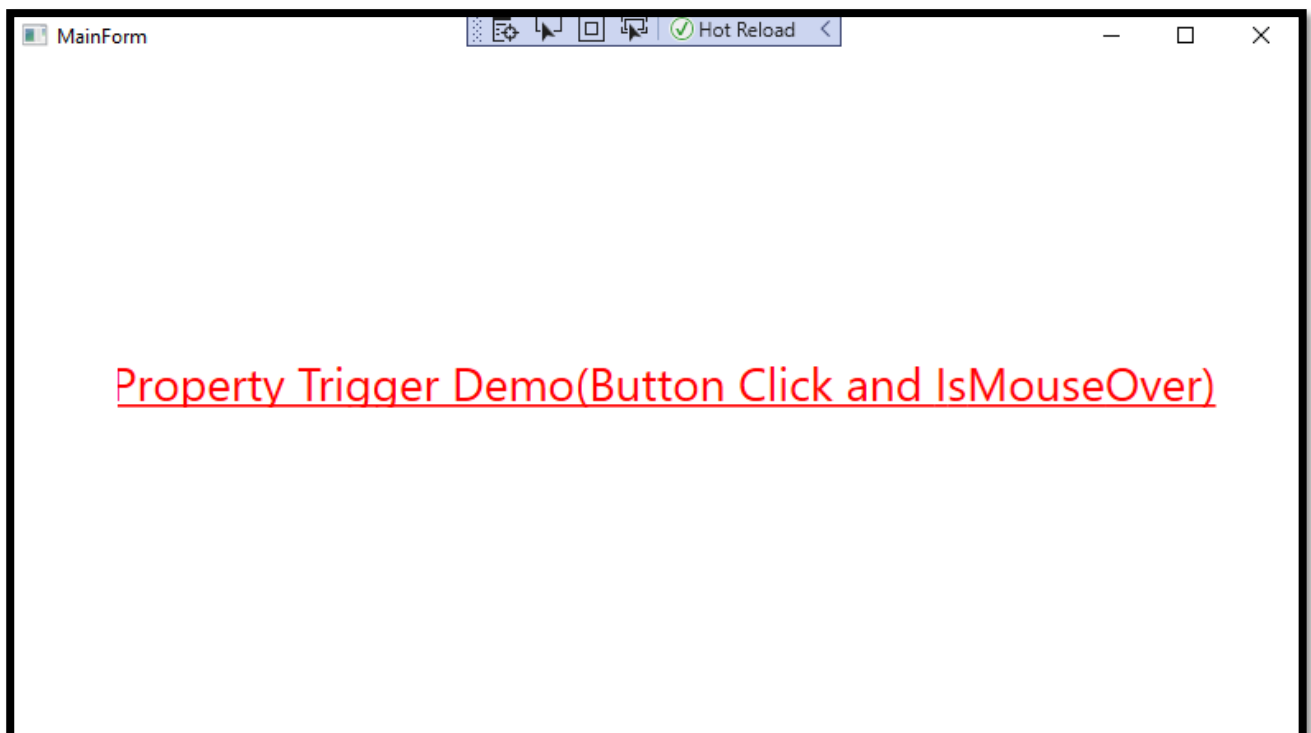
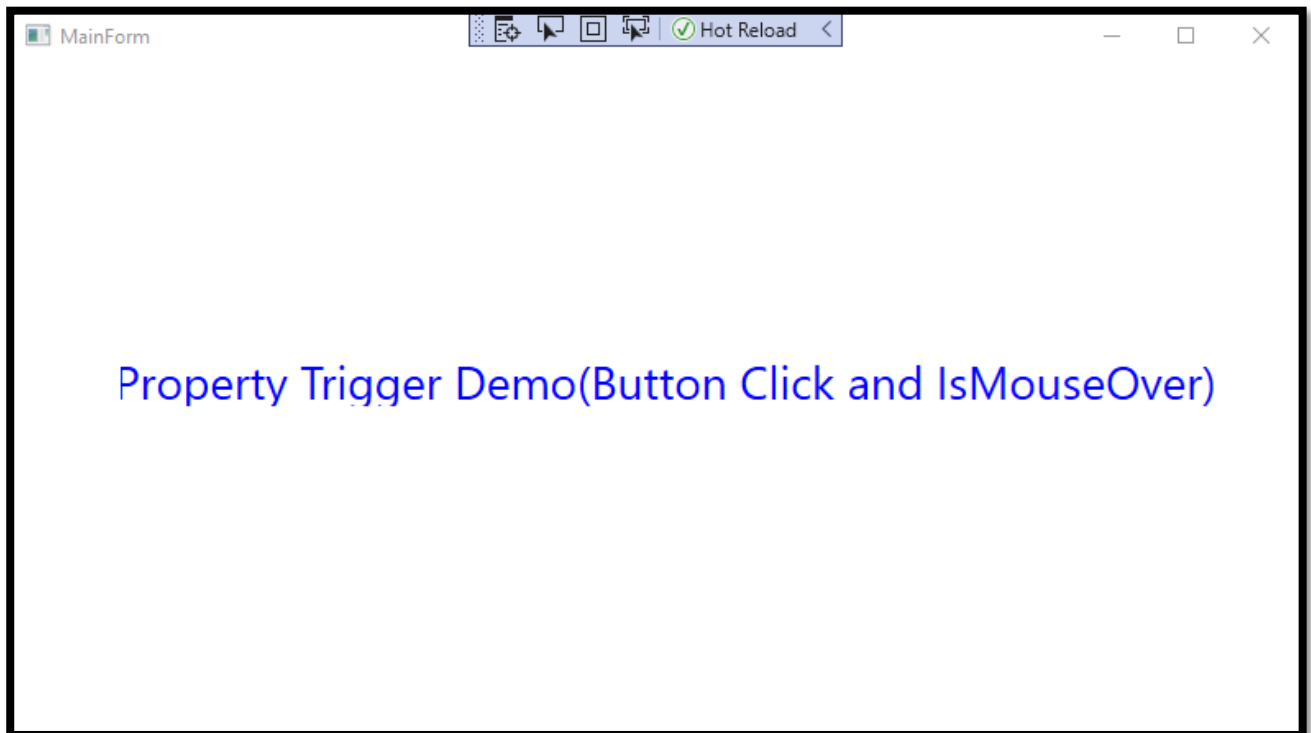
Property Trigger
XAML Code

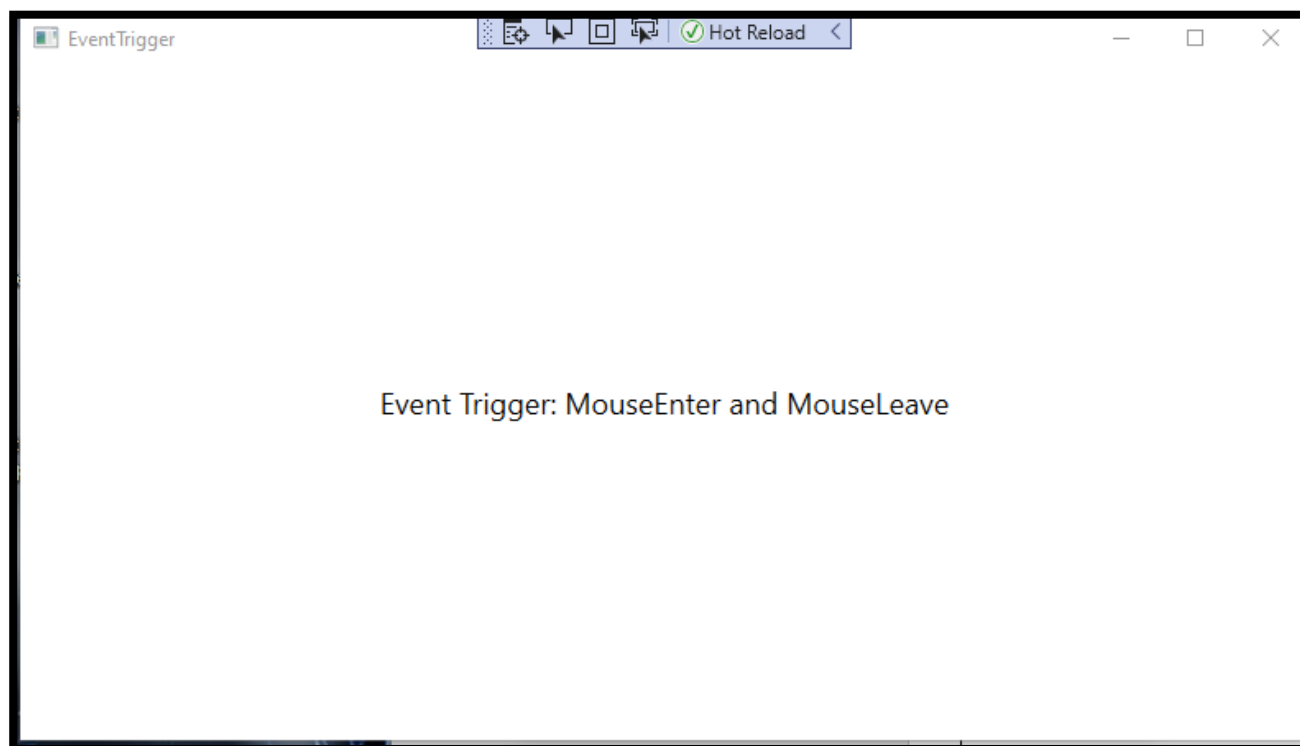


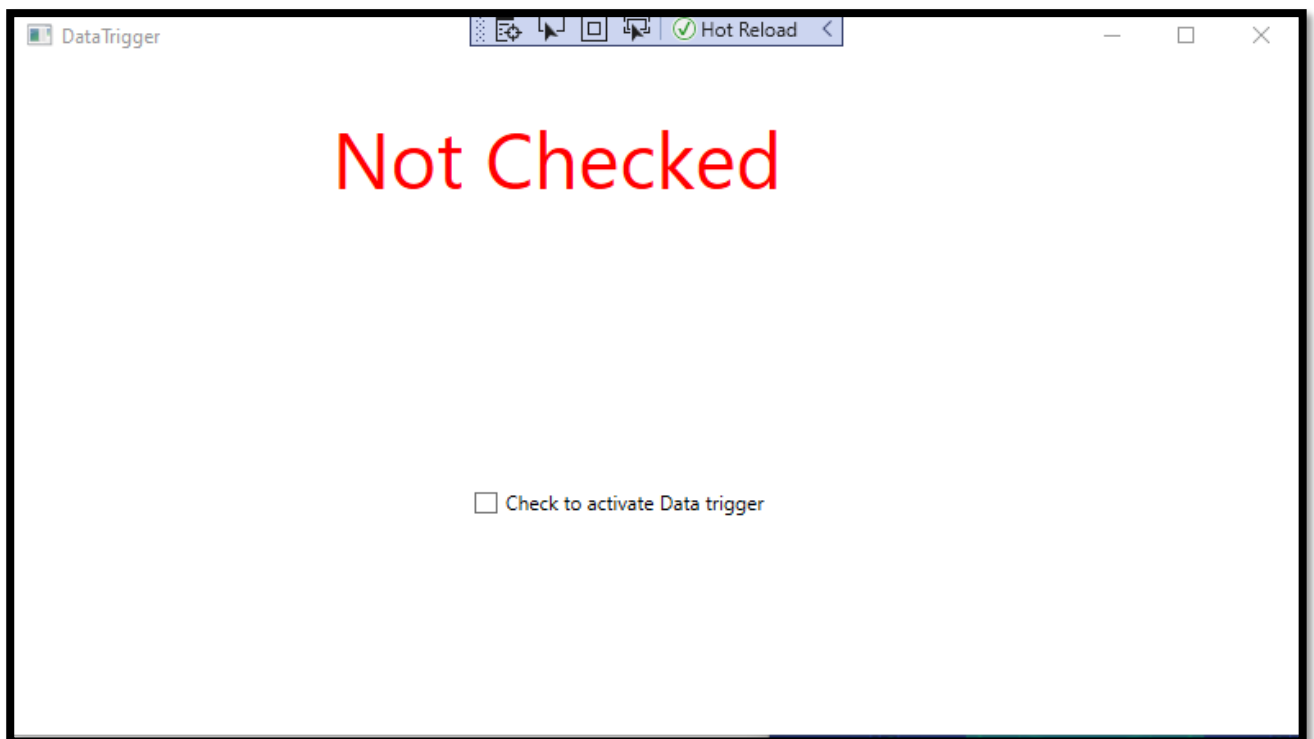
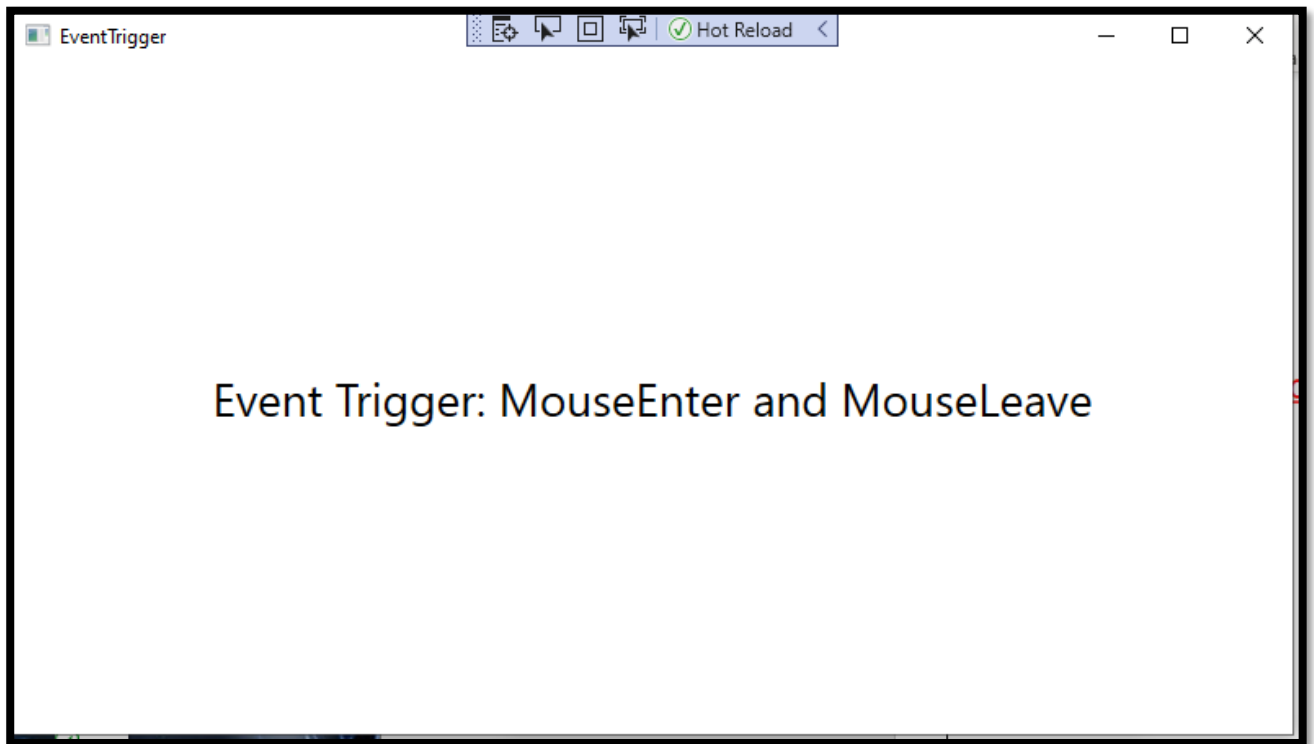
Event Trigger XAML
Code

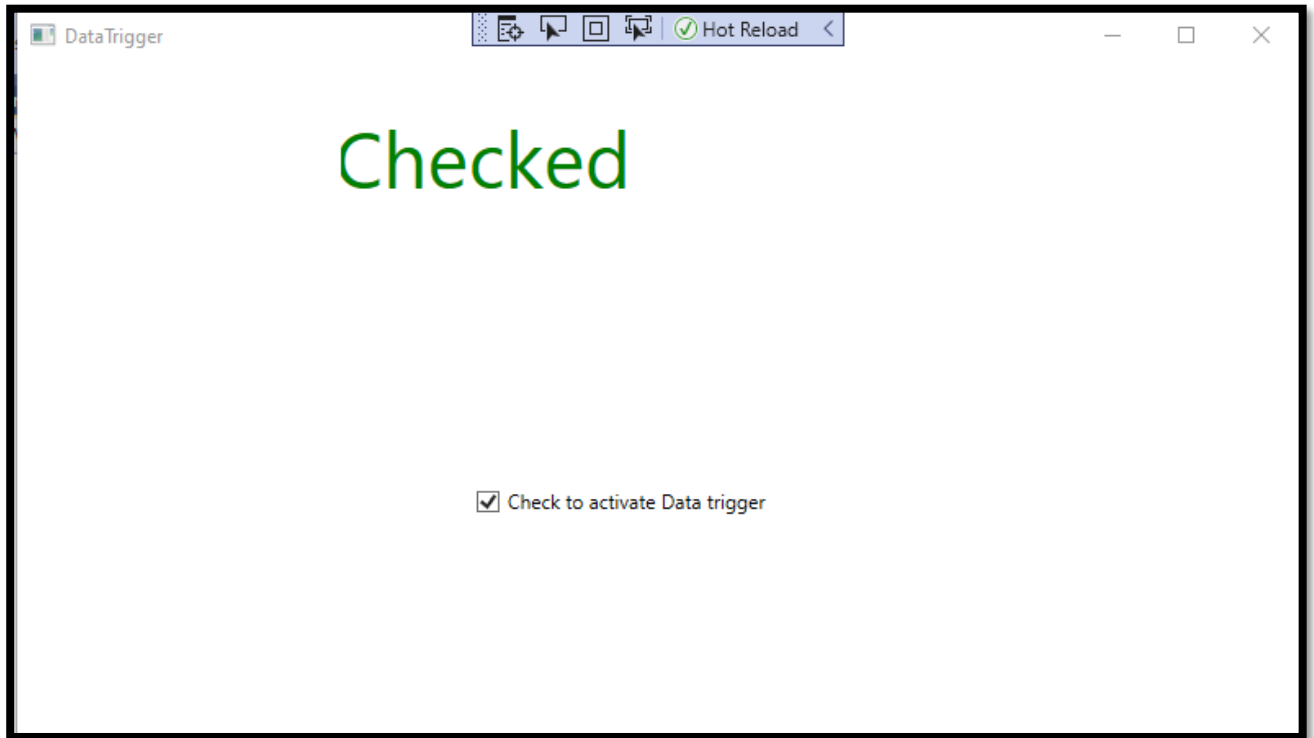
Screenshots:











5. Konklusion

Vi arbejdede med styles ved at sætte en statisk værdi for en specifik egenskab. Men ved at bruge triggere kan du ændre værdien af en given property, når en bestemt tilstand ændres. Triggere kommer i flere varianter: **Property Trigger**, **Event Trigger** og **Data Trigger**. De giver dig mulighed for at lave ting, der normalt ville blive gjort i kode-bag fuldstændigt i markup i stedet, som alt sammen er en del af den igangværende proces med at adskille stil og kode.

