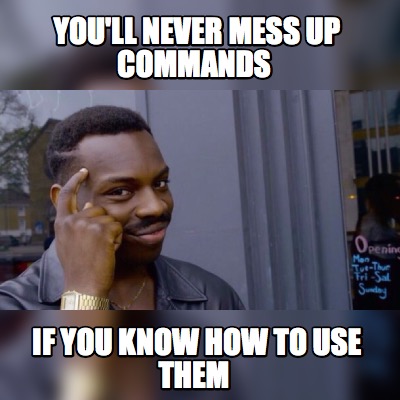
# WPF Command binding

|  |  |
| --- | --- |
| **Læringsmål** | Du kan:   * **1Pf1**: anvende centrale metoder til at specificere og konstruere algoritmer […] * **1Pf2**: anvende centrale faciliteter i programmeringssproget til realisering af […] brugergrænseflader |
| **Forventet læringsudbytte [SOLO]** | Programmeringssporet:   * [Unistrukturel] Du kan identificere enkelte kendetegn ved **GUI applikationstypen** – WPF, XAML, MVVM, UI-kontroller (ListBox, StackPanel, Grid, TextBox, Label, Button, mm.), databinding (inkl. interfacet INotifyPropertyChanged og ObservableCollection<T>-typen), command binding (ICommand) * [Unistrukturel] Du kan identificere enkelte kendetegn ved **arkitekturmønstre** – Model-View-ViewModel (MVVM) |
| **Din forberedelse** | Programmeringssporet:   * [WPF ICommand In MVVM](https://www.c-sharpcorner.com/UploadFile/e06010/wpf-icommand-in-mvvm/) * Commandbind 1 - Intro (video: 20:03) * Commandbind 2 – ListBox (video: 22:12) * Commandbind 3 – Button (video: 14:19)   + **Bemærk**: der vil være flere detaljer vedrørende et event i denne video. Events gennemgås om få uger; lige nu skal du bare anvende det viste. |

Denne opgave fokuserer på command binding, en indbygget funktionalitet i WPF til at skabe ikke alene en løs kobling mellem View-laget og ViewModel-laget ift. kommandoer, men også centralt styre aktiveringen af en kommando fra flere steder i ens GUI.

Formålet med dagens øvelser er fortsat at udvide dit repertoire i forbindelse med både command binding og databinding.

# Dagens ord



# Øvelse 1: Terminologi

Del teamet op i mindre grupper, og brug **Ordet rundt** til at reflektere over begreberne ”Command binding”, ”ICommand”.  
Sørg for at alle får mulighed for at tale.

*Tidsramme: 15 minutter*

**Benyt parprogrammering i alle følgende øvelser**.

# Øvelse 2: Command binding i XAML

WPF har indbygget et alsidigt kommando-koncept, som ikke alene skaber en løs kobling mellem View-laget og ViewModel-laget på samme måde som databinding gør, men også via XAML centraliserer bl.a. aktivering/deaktivering af kommandoer (fra knapper, menupunkter, toolbar-ikoner, keyboard shortcuts). Command binding minimerer desuden (eller fjerner helt) kode i code-behind. Du skal i denne øvelse implementere en kommando i en ViewModel og etablere en binding til den fra en knap i XAML i View-laget.

## Øvelse 2.1: Arkitekturovervejelse

Overvej følgende i en passende CL-struktur i dit team:

* I hvilket MVVM-lag hører en kommando-klasse til?
* Hvorfor mener du, kommando-klassen hører netop til i dette lag?
* Hvilke andre MVVM-lag må en kommando-klasse kende til?

## Øvelse 2.2: Vælg en tidligere løsning

Udfør følgende:

* Vælg en WPF-løsning fra tidligere opgaver, hvor du har implementeret en knap i brugergrænse­fladen, og nu vil anvende command binding i stedet for en eventhandler i code-behind for at udføre knappens funktionalitet.
* Opdatér din DCD for dit udvalgte GUI-eksempel, så din kommando-klasse indgår i den med tilhørende attributter, operationer og associationer.

## Øvelse 2.3: Opret en kommando-klasse, og etablér en command binding for den

Som du har set, implementerer en kommando-klasse ICommand-interfacet og er designet til at udføre en bestemt operation (forretningslogik), som aktiveres via en UI-kontrol (f.eks. en knap) i din GUI. I MVVM er der løs kobling mellem den UI-kontrol, der aktiverer kommandoen og selve implementeringen af kommandoen i kommando-klassens Execute()-metode.

Udfør følgende for dit udvalgte eksempel (som ikke nødvendigvis ligner nedenstående kode):

* Slet din eksisterende eventhandler for din knap i code-behind, f.eks.:

private void Button\_Click(object sender, RoutedEventArgs e)

{

mvm.MyLabelText = DateTime.Now.ToString();

}

* Slet dernæst referencen til eventhandler’en i XAML-elementet for knappen  
   <Button Content="Update Label" ... Click="Button\_Click"/>  
  Dvs. slet attributten: Click="Button\_Click"

Du er nu parat til at implementere en kommando samt oprette en command binding fra din ”Update Label”-knap (eller hvad nu din kommando hedder i dit eksempel) til denne kommando.   
**Bemærk: tilstræb at code-behind filen i din løsning ikke indeholder nogen som helst kode udover din MainWindow-constructor, som sætter DataContext til en instans af MainViewModel.**

Udfør:

* Implementér en kommando (via en C#-klasse, der implementerer ICommand-interfacet), der udfører den samme operation, som din knap gjorde før. Du vælger selv navnet på C#-klassen til kommandoen.
* Opret en property i din MainViewModel, som initialiseres til en instans af din nye kommando-klasse
* I XAML opret til sidst en command binding fra din knap til din kommando-property (husk du kan også overføre kommando-parametre fra XAML).
* Kør dit program, og ret i koden, indtil det virker.

## Øvelse 2.4: Aktivér/deaktivér knap via CanExcute()

En knap kan aktiveres/deaktiveres via CanExecute()-metode i din kommando-klasse. Du skal nu sørge for at styre denne aktivering på en passende måde for dit udvalgte eksempel.

Udfør følgende:

* Overvej, hvornår din knap skal være aktiveret og deaktiveret
* Ret Execute()-metoden i din kommando-klasse, så din knap (de)aktiveres korrekt
* Ret evt. CanExecuteChanged-eventet som angivet i forberedelsesvideoen ’Commandbind 3 - Button’, hvis din knap ikke responderer som forventet

# Øvelse 3: Databinding og command binding for ListBox

Du skal i det følgende øve både databinding og command binding for en ListBox-kontrol i GUI’en.

## Øvelse 3.1: Klargøring

Udfør følgende:

* Opret en ny WPF-løsning i Visual Studio med en mappe til hvert lag i MVVM-arkitektur. Du vælger selv navnet på løsningen.
* I WPF-løsningens mappe for Model-laget, opret en Product-klasse med to properties:
  + Name af typen string
  + Price af typen double

Opret dernæst en passende ProductRepo-klasse til at repræsentere dine produktdata:

* Overvej, hvilket lag dit repository skal placeres i
* Initialisér ProductRepo-listen med følgende 3 produkter:
  + ”Apple” med prisen 7.95
  + ”Orange” med prisen 5.50
  + ”Banana” med prisen 8.25

I din mappe for ViewModel-laget:

* Opret en passende ProductViewModel-klasse, som iscenesætter din Product-klasse også med de to properties nævnt foroven
* Opret tillige en MainViewModel-klasse med en ny property med navnet ”ProductsVM” af typen List<ProductViewModel>

## Øvelse 3.2: Databind fra ListBox til en Collection

Udfør følgende:

* Opret nu en ListBox-kontrol i GUI’en, dvs. i View-laget
* I XAML databind ListBox-kontrollen til ProductsVM-listen
* Kør programmet, og tjek om det virker, som det skal. Ret, hvis ikke.

## Øvelse 3.3: Brug af DataTemplate i en ListBox

Måske viser ListBox-kontrollen ikke lige den mest brugervenlige tekst, så overvej hvordan du kan vise en tekst for hvert produkt, hvor både produktets navn og pris indgår (vink: det er vist noget med en DataTemplate i XAML)

* Ret koden, så ListBox-kontrollen viser en mere forståelig tekst for hvert produkt.

## Øvelse 3.4: Visning af det valgte element i ListBox-kontrollen

Du vil nu gerne kunne vælge et produkt i ListBox-kontrollen og gøre det muligt at tilgå hver af de to oplysninger for et produkt.

Udfør følgende:

* Opret to TextBox-kontroller i GUI’en (den ene over den anden) til højre for din ListBox. Den øverste TextBox skal vise navnet på et produkt og den nederste prisen på et produkt. Indsæt evt. nogle labels, så du kan se, hvad der er hvad. Det kunne se ud som nedenstående:



* Opret nu en property f.eks. med navnet SelectedProduct i din MainViewModel af type ProductViewModel.
* I XAML opret en databinding fra det udvalgte element i din ListBox (vink: SelectedItem-attributten i XAML) til ovenstående property SelectedProduct.
* I XAML etablér dernæst databindings, så navn og pris for det udvalgte produkt i ListBox-kontrollen bliver vist i de to TextBox-kontroller
* Kør dit program, og ret koden, indtil det virker.

## Øvelse 3.5: Redigering af produkt

Nu hvor du har to TextBox-kontroller for produktnavn og -pris, så vil du gerne kunne redigere navn og/eller pris via disse for et udvalgt produkt i listen.

Undersøg følgende:

* Kør dit program, og vælg et produkt i listen.
* Ret produktets navn eller pris i den tilsvarende TextBox.
* Observér om produktet ændrer navn eller pris i ListBox-kontrollen efter din ændring
* Ret koden, hvis det ikke virker, som ønsket.

## Øvelse 3.6: Indsæt nyt produkt

Du vil nu gerne kunne indsætte et nyt produkt og automatisk få ListBox-kontrollen i GUI’en opdateret.

Udfør følgende:

* Implementér en ny C#-klasse ”NewProduct”, der implementerer ICommand-interfacet og sørger for at indsætte et nyt produkt i ProductRepo-listen samt ProductsVM-listen, når kommandoens Execute()-metode kaldes. Lad det nye produkt få navnet ”New Product” med default-prisen 0.0.
* Udvid din MainViewModel med en kommando-property (med et passende navn), der instantierer den nye kommando.
* I XAML indsæt en knap under ListBox-kontrollen med teksten ”New Product”.
* I XAML etablér command binding for denne knap til din nye kommando-property i MainViewModel.
* Kør dit program, og afprøv din nye kommando.
* Kører det som det skal? Kommer der er nyt produkt ind i ListBox-kontrollen, når du klikker på knappen?
* Overvej, hvorfor det fungerer, som det gør.

Din ListBox indeholder en databinding (via ItemsSource) til en ProductsVM-liste (Collection) i MainViewModel. Hvis der sker en ændring i Product-listen, f.eks. der indsættes et nyt produkt, eller slettes et eksisterende produkt, via C#-kode (f.eks. via NewProduct.Execute()-metoden), så skal ListBox-kontrollen notificeres om denne ændring.

* Overvej hvorledes ListBox-kontrollen bliver notificeret, når den liste, den er databindet til, ændres.
* Overvej hvad du skal rette i koden for at sikre, at ListBox-kontrollen bliver notificeret om en ændring
* Implementér din rettelse, kør programmet og se, om det virker (vink: ObservableCollection)

## Øvelse 3.7: Slet valgt produkt

Du vil også kunne slette produkter fra din ProductRepo og ProductsVM-liste i MainViewModel og automatisk få opdateret din ListBox.

* Implementér denne funktionalitet via en ”Delete Product”-knap i GUI’en med inspiration fra forrige øvelse. Vær opmærksom på knappens funktionalitet, hvis der ikke er valgt et produkt fra ListBox-kontrollen, eller listen er tom.