TSO_CSHARP_THEMA_5_COLLECTIES_BASIS

In deze opdracht gaan we de verkenningstocht binnen C# en WPF verderzetten.

- Je oefent jouw OOP-kennis.
- Je leert de List-collectie kennen.
- Je gebruikt een nieuwe vorm van iteratie.
- Je maakt kennis met het begrip 'generics'.
- Je ervaart de kracht van Grids bij WPF.
- Je maakt een bescheiden WPF-toepassing met meerdere pagina's.
- Je oefent het gebruik van een bescheiden beetje XAML

Je bouwt een toepassing voor éénvoudig gegevensbeheer. De gegevens bestaan uit naam, voornaam en email van je contactpersonen. Je gaat voor deze toepassing een eenvoudige GUI met WPF bouwen waarmee je contactpersonen kan toevoegen aan een lijst en waarmee je de lijst met contactpersonen kan weergeven.

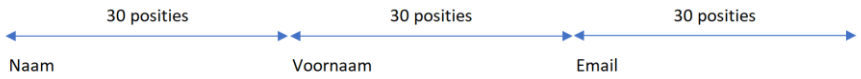
OPDRACHTEN:

1. Maak in Visual Studio een nieuw project aan gebaseerd op het WPF C# Application template. Geef de solution de naam `PersonsCollection`.
2. Klik in de Solution Explorer met de rechtermuistoets op de naam van de solution en kies in het popup menu voor `Add → Class`. Geef de klasse de naam `Persoon` en kies `Add` om de klasse toe te voegen aan je project.
3. Open vanuit de Solution Explorer het bestand `Persoon.cs` en schrijf de code voor de klasse `Persoon`. Deze klasse heeft *drie auto-implemented properties*:
    a. Naam
    b. Voornaam
    c. Email

    De klasse heeft een default constructor zonder parameters. De gegevens van die persoon worden dan gelijkgesteld aan "onbekend".

    Je voorziet ook *overloaded constructors* waarmee je naam en voornaam of naam, voornaam en email aan het object kan toekennen.

    Je voorziet een `ToString()` method die de Naam, Voornaam en Email retourneert. Deze drie gegevens worden uitgelijnd weergegeven:



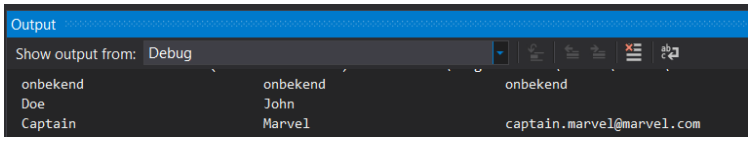
Pas hiervoor de String.Format method toe. Klik op de voorgaande link en ga na hoe je deze method kan gebruiken om het gestelde doel te bereiken.

4. Open vanuit de solution-explorer het bestand `mainWindow.xaml.cs`. Instantieer in de `MainWindow()` method drie objecten van de klasse persoon en voer voor elk object de `ToString()` method uit.

```
public MainWindow()
{
        InitializeComponent();

        /*
          instantieer hier de drie objecten p1, p2 en p3
          van de klasse persoon. Test vervolgens de
          ToString() method uit op elk object.
        */
}
```

Om het resultaat van de `ToString()`-method te testen maak je gebruik van de `Trace.WriteLine(string)` method in de Trace Class. De strings die je met de `Trace.WriteLine()` method schrijft worden tijdens het *debuggen* weergegeven in het `output`-venster van Visual Studio:



5. Het is in deze opdracht de bedoeling om de persoonsgegevens bij te houden in een `Collection`. Je hebt vorig schooljaar toen je C++ hebt geleerd al het array en de vector bestudeerd. Dat zijn collecties. C# kent heel wat geavanceerde collections. Voor deze opdracht zal je gebruik maken van een generieke List-collectie. We gebruiken een generieke lijst. Ga na wat dit wil zeggen.

    a. Open vanuit de solution-explorer het bestand `mainWindow.xaml.cs`. Declareer in de klasse `MainWindow` een List die objecten zal bevatten van het type Persoon. Geef deze List de naam *Personen*.

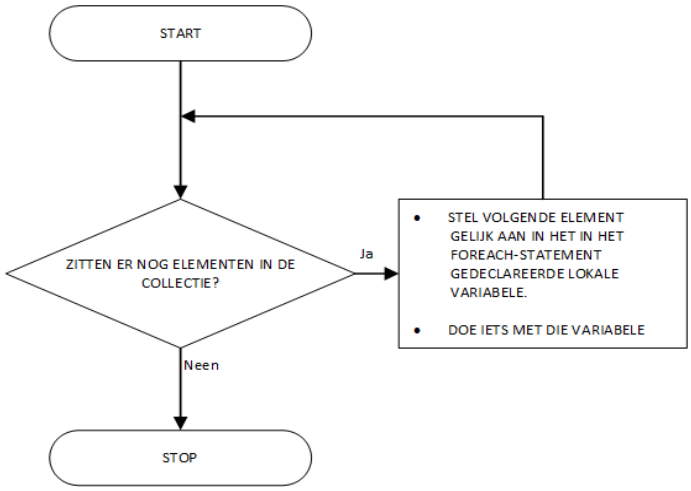
    b. Voeg de drie instanties p1, p2 en p3 die je in stap 4 hebt gemaakt toe aan de list *Personen*. Maak hiervoor gebruik van de `Add()` method van de klasse List<T>.

6. In opdracht 4 ben je nagegaan of de `ToString()` method van de klasse Persoon een correcte uitvoer gaf. We gaan dezelfde method gebruiken om na te gaan of de objecten effectief in de list Personen aanwezig zijn. We gaan hiervoor gebruik maken van een nieuw soort iteratie: `foreach`

De `foreach`-iteratie (lees dit als "for each", "voor elke") laat je toe om over een collectie van items te lopen:

```
//SYNTAX

foreach (type lokale variabele in collectienaam)
{
   //doe iets met lokale variabele
}
```



Pas het `foreach`-statement toe om voor elke `Persoon` in de list `Personen` de `ToString()` method aan te roepen en weer te geven in het outputvenster. Test je oplossing uit.

7. We willen de gebruiker van de software de mogelijkheid geven om personen toe te voegen aan de collectie. Om dit te realiseren gaan we aan de slag de GUI in WPF.

   Je zal een pagina aanmaken waarin je de naam, voornaam en mailadres van een persoon kan toevoegen aan de collectie.

   a. Klik op de naam van de solution en kies `Add → Page (WPF)`. Noem deze pagina '`Invoer.xaml`'.

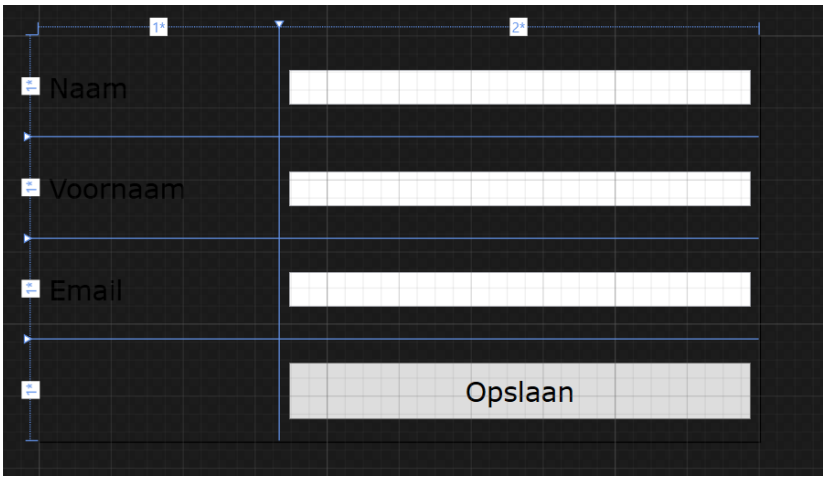
   b. Zet een Grid op deze pagina. Het Grid bestaat uit twee kolommen en vier rijen. De rijen zijn allemaal even groot. De verhouding tussen de eerste en de tweede kolom 1:2 is.

   c. Plaats in de eerste kolom in de bovenste drie rijen een TextBlock. De weer te geven tekst is respectievelijk "Naam", "Voornaam" en "Email". Verander het font naar Verdana, 22 Px.

   d. Plaats in de tweede kolom in de bovenste drie rijen een TextBox. Deze TextBoxes zijn leeg. Geef zowel de TextBlocks als de TextBoxes een linker en rechtermarge van 10.

e. Plaats een Button in de tweede kolom van de onderste rij. Pas voor deze Button ook de tekstgrootte aan naar Verdana, 22 Px. Geef de button boven, onder links en rechts een marge van 10.

   Het scherm zou er moeten uitzien zoals in onderstaande afbeelding:

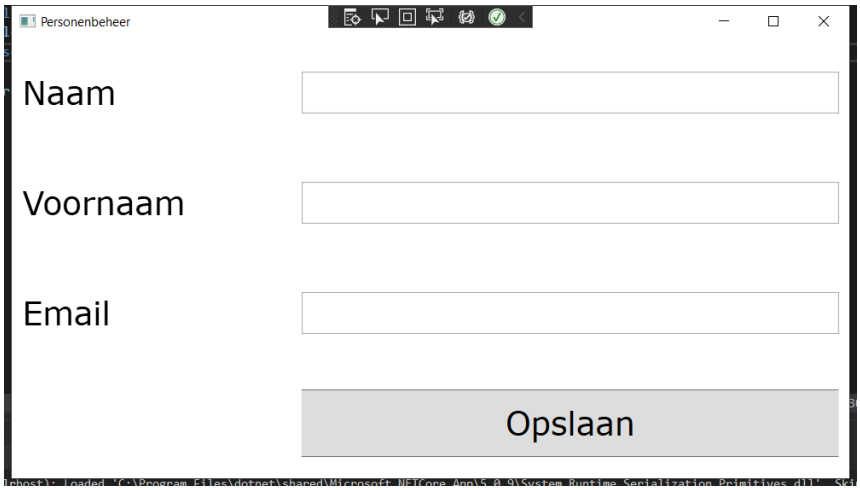


f. Open via de solution-explorer het `MainWindow.xaml` bestand. Voeg een Frame-tag toe zodat de XAML-code tussen de grid-tags eruitziet zoals volgt:

```
<Window …>
    <Grid>
      <Frame Source=”/Invoer.xaml”/>
    </Grid>
</Window>
```

g. Verander in MainWindow.xaml de Title=”MainWindow” naar “Personenbeheer”

h. Run de code. Op het scherm zou het volgende beeld moeten verschijnen:



8. Je hebt nu een pagina aangemaakt waarmee je gegevens kan invoeren. In de toekomst zal er ook een pagina nodig zijn om de personen weer te geven, om personen te verwijderen, om een e-mail te sturen naar een specifieke persoon,… Het is dus handig dat we een menupagina hebben waarmee we dit kunnen realiseren. In deze opdracht maak je deze menupagina aan:

    a. Klik op de naam van de solution en kies `Add → Page (WPF)`. Noem deze pagina 'Menu.xaml'.

    b. Plaats op deze pagina een grid met vier kolommen en drie rijden. De kolommen en rijen zijn gelijkmatig verdeeld over de pagina.

    c. Zet op de eerste rij een TextBlock met als tekst "PersonenBeheer". Stel het font in op "Verdana, 48 px"

    d. Plaats op rij drie, in de eerste kolom, een knop met als opschrift "Invoer". Stel het font in op "Verdana, 24 px".

    e. Voeg op rij drie, in de tweede kolom, een knop met als opschrift "Weergave". Stel het font in op "Verdana, 24 px".

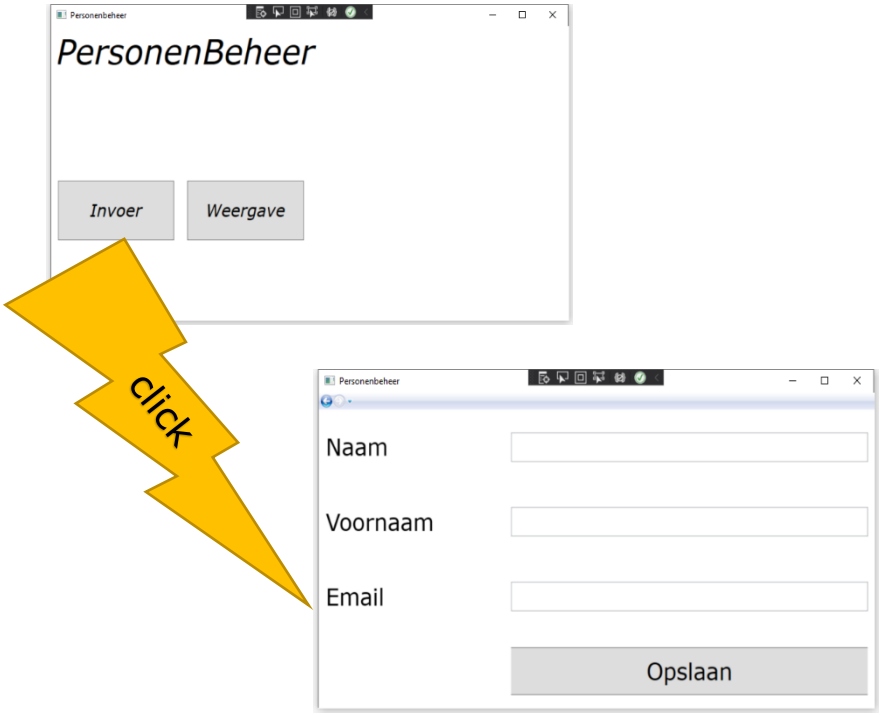
    f. Dubbelklik op de knop met het opschrift "Invoer". Aan de achterliggende code wordt er dan een click-event toegevoegd. Vul in dit clickevent de onderstaande code in.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
        NavigationService.Navigate(new Uri("/Invoer.xaml", UriKind.Relative));
}
```

g. Open via de solution-explorer het `MainWindow.xaml` bestand. Pas het Frame-tag toe aan de XAML-code tussen de grid-tags er uitziet zoals volgt:

```
<Window …>
    <Grid>
      <Frame Source=”/Menu.xaml”/>
    </Grid>
</Window>
```

i. Test de code uit en klik op de button met het opschrift “Invoer”. Je zou de onderstaande schermen moeten zien. Je kan nu vanuit één Window door gebruik te maken van meerdere pagina’s en goed gekozen events navigeren tussen de pagina’s.



TSO_CSHARP_THEMA_5_COLLECTIES_UITBREIDING

In deze opdracht bouwen we verder op het project rond .

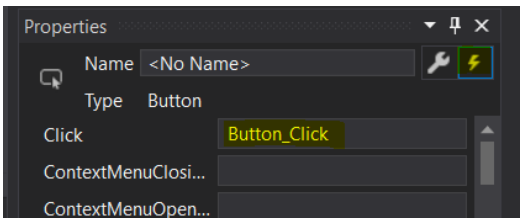
- Je leert wat *data binding* is.
- Je verneemt wat *MVVM* is.
- Je leert hoe je met data binding éénvoudig controls kan koppelen met databronnen in de code.
- Je past achterliggende collecties aan vanuit de GUI.
- Je geeft achterliggende collecties weer in de GUI

Je bouwt verder aan de toepassing voor éénvoudig gegevensbeheer. De gegevens bestaan uit naam, voornaam en email van je contactpersonen. Je voegt de code toe om vanuit de GUI gegevens toe te voegen aan de achterliggende List.

OPDRACHTEN

1. Het is de bedoeling dat de op de invoerpagina ingegeven naam, voornaam en email worden toegevoegd aan een object van de klasse Persoon. Dit object moet geïnstantieerd worden wanneer je op de knop "Opslaan" klikt. Daarna kan dit object opgeslagen worden in de List `Personen`.
   Er moet dus een koppeling gemaakt worden tussen de elementen in de GUI en de achterliggende code. In deze opdracht ga je een '*innige*' koppeling tussen GUI en achterliggende code maken. Je gaat eerst aan de `Opslaan`-knop een event koppelen. Vervolgens ga je de elementen op de GUI-pagina `invoer` een naam geven. Daarna koppel je de C#-code met de gegevens die in de GUI worden ingevoerd.

   a. Open het bestand `Invoer.xaml`
   b. Selecteer de knop 'Opslaan'.
   c. Klik in het *Properties* venster op de bliksemschicht.
   d. Dubbelklik naast 'Click'. Er zal een event *Button_Click* worden toegevoegd.



   e. Als alternatief kan je ook de event via de xaml-code toevoegen:

```
<Button Content="Opslaan" … Click="Button_Click"/>
```

f. Ga naar de xaml-code van de TextBoxes en voeg bij elke TextBox een naam toe:

```
<TextBox x:Name="NaamInput"… />
<TextBox x:Name="VoornaamInput"… />
<TextBox x:Name="EmailInput"… />
```

g. Open het bestand invoer.xaml.cs en zoek de `Button_Click` method.

```
private void Button_Click(object sender, RoutedEventArgs e)
{

}
```

h. Het is de bedoeling dat bij het klikken op de knop 'Opslaan' er een object wordt geïnstantieerd waarvan de Naam, Voornaam en Email gelijk wordt aan de in de TextBoxes ingevoerde gegevens. Voeg daarvoor de volgende code toe aan de Button_Click event:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
        Persoon p = new Persoon();
        p.Naam = NaamInput.Text;
        p.Voornaam = VoornaamInput.Text;
        p.Email = EmailInput.Text;
        Trace.WriteLine(p.ToString());
}
```

Het Trace-statement is toegevoegd om te controleren of de data die via de GUI is ingetikt correct in de C#-code toekomt.

i. Als je alles correct hebt uitgevoerd dan heb je in opdracht g. een object geïnstantieerd dat de in de GUI ingevoerde gegevens bevat. Dit object moet opgeslagen worden in de List `Personen`. Dat lijkt éénvoudig, maar er zit een addertje onder het gras:

Voeg het vet gemarkeerde statement toe aan de event en test de code uit. Welke foutmelding wordt er gegeven?

```
private void Button_Click(object sender, RoutedEventArgs e)
{
        Persoon p = new Persoon();
        p.Naam = NaamInput.Text;
        p.Voornaam = VoornaamInput.Text;
        p.Email = EmailInput.Text;
        Trace.WriteLine(p.ToString());

        Personen.Add(p);
}
```

j. Het object personen is gedefinieerd in de MainWindow-klasse. Het is zelfs 'public' gedefinieerd. Niettemin is het blijkbaar onbereikbaar vanuit de klasse `Invoer`.

De List Personen kan je echter niet bereiken via de klasse MainWindow. Je kan de List enkel bereiken via een instantie van de klasse MainWindow. Als de code opstart wordt er uiteraard een object van de klasse MainWindow geïnstantieerd. Als je de naam van dat object kent dan kan je ook de List Personen gebruiken die bij dat object hoort . Gelukkig is de naam van de MainWindow makkelijk te achterhalen. Pas de code van de Button_Click event aan en test de code uit.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
        MainWindow mw = (MainWindow)Application.Current.MainWindow;

        Persoon p = new Persoon();
        p.Naam = NaamInput.Text;
        p.Voornaam = VoornaamInput.Text;
        p.Email = EmailInput.Text;
        Trace.WriteLine(p.ToString());

        mw.Personen.Add(p);
}
```

Opmerking: Het alternatief is om de List Personen als static te declareren. De List kan dan via de klasse MainWindow i.p.v. via het geïnstantieerd object van de klasse worden benaderd. Pas daartoe de code als volgt aan.

In MainWindow.xaml.cs:

```
public static readonly List<Persoon> Personen = new List<Persoon>();
```

In de Button_Click event:

```
private void Button_Click(object sender, RoutedEventArgs e)
{
        Persoon p = new Persoon();
        p.Naam = NaamInput.Text;
        p.Voornaam = VoornaamInput.Text;
        p.Email = EmailInput.Text;
        Trace.WriteLine(p.ToString());

        MainWindow.Personen.Add(p);
}
```

Vaak kan je problemen op meer dan één wijze oplossen. Elke oplossing heeft zijn voor- en nadelen.

2. In opdracht 1 is er een hecht verband gelegd tussen de gegevens die via de GUI zijn ingegeven en de code. Er is echter een andere oplossing mogelijk. Je kan 'data binding', kortweg 'binding' gebruiken. Hier ga je aan een element op de GUI als het ware vertellen met welk achterliggend object of ander GUI-element het is verbonden. Wanneer dit principe doorgedreven wordt toegepast dan volgt men het MVVM-ontwerpprincipe: Model View ViewModel. Bij dit model wordt de GUI (=View) maximaal gescheiden van de code (=Model). Via een link (ViewModel) wordt er een verbinding (binding) gelegd tussen de GUI en de Code. Dit reduceert sterk te hoeveelheid code die moet worden geschreven en onderhouden.

    a. Open het bestand Invoer.xaml
    b. Selecteer de TextBox die gebruikt wordt om de Naam in te voeren
    c. Identificeer in de xaml-code het Tekst-veld. Wijzig dit veld als volgt:

    ```
    Text="{Binding Path=Naam}"
    ```

    Je leest deze xaml-code als volgt : koppel ("bind") het tekst-property van de TextBox met een element dat men "Naam" noemt.

    d. Doe hetzelfde met de twee andere TextBoxes:

    ```
    Text="{Binding Path=Voornaam}"
    Text="{Binding Path=Email}"
    ```

    e. De elementen op de invoer-pagina weten nu dat ze gekoppeld worden met iets wat Naam, Voornaam en Email noemt. We moeten nu in onze code duidelijk maken waar deze elementen de data kunnen vinden die Naam, Voornaam en Email voorstellen. Dat noemt men de Datacontext.

        i. Open het Invoer.xaml.cs bestand.
        ii. Voeg de vetgedrukte lijnen toe aan de onderstaande code:

```
public partial class Invoer : Page
    {
        Persoon p = new Persoon();

        public Invoer()
        {
            InitializeComponent();
            DataContext = p;
        }
```

Het onderliggend mechanisme dat de xaml-code met de C#-code verbindt weet nu dat naam, Voornaam en Email slaan op het object p.

iii. Wijzig de code van de `Button_Click` event als volgt (zet de vorige code in een commentaarblok):

```
private void Button_Click(object sender, RoutedEventArgs e)
{

        MainWindow mw = (MainWindow)Application.Current.MainWindow;
        mw.Personen.Add(p);
        Trace.WriteLine(p.ToString());

}
```

f. Je kan meerdere keren dezelfde persoonsgegevens toevoegen aan de lijst. Een object van de klasse List kan gebruik maken van de Contains-method. Wanneer een item al in de list aanwezig is dan zal deze method `true` retourneren. Gebruik in de Button_Click-event de Contains-method om te vermijden dat er duplicaten in de List worden opgeslagen. Test de code uit.

3. Je kan nu persoonsgegevens toevoegen aan de List. Via een nieuwe pagina `weergave.xaml` ga je de in de list is opgeslagen gegevens weergeven. De knop om deze pagina te activeren heb je al op de `Menu.xaml` pagina gezet. Maak alvast pagina Weergave.xaml aan en zorg dat deze pagina verschijnt wanneer je op de menu-knop "*Weergave*" drukt.