

Gegevensbeheer en Beveiliging – Les 6 – DataGrid

In deze les gaan we aan de slag met een nieuw WPF-element dat handig is om data te tonen: het DataGrid-element.

In XAML maken we een DataGrid waarin we de gegevens van enkele dieren (voor bijvoorbeeld de administratie van een dierenarts-praktijk) kunnen bewaren:

```
<StackPanel>
    <DataGrid x:Name="DierenGrid"></DataGrid>
</StackPanel>
```

Anders dan bij een ListBox is de Items property read only, dus je kan hier manueel niets in toevoegen met Add. In plaats daarvan werkt een DataGrid met ItemsSource die je naar een lijst (of lijst-achtig) object laat verwijzen. We maken hiervoor eerst een class.

```
public class Dier
{
    public string Naam { get; set; }
    public string Soort { get; set; }
    public float Gewicht { get; set; }
    public DateTime GeboorteJaar { get; set; }
}
```

In de code-behind (MainWindow.xaml.cs) voorzien we een **property** *Dieren* met enkele Dier objecten in. Na de InitializeComponent aanroep stellen we ItemsSource property in op onze lijst.

```
private List<Dier> Dieren {get; set;} = new List<Dier> {
    new Dier {Naam = "Fiffy", Soort = "Hond", Gewicht = 7F,
              GeboorteJaar = new DateTime(2012, 8, 30)},
    new Dier {Naam = "Billy", Soort = "Kat", Gewicht = 7F,
              GeboorteJaar = new DateTime(2020, 2, 7)},
    new Dier {Naam = "Flup", Soort = "Papegaai", Gewicht = 7F,
              GeboorteJaar = new DateTime(1980, 11, 3)}
};

public MainWindow()
{
    InitializeComponent();

    DierenGrid.ItemsSource = Dieren;
}
```

Het resultaat is een lijst met rijen en kolommen. De DataGrid creëert dus automatisch de kolommen. Je kan de kolommen sorteren en standaard kan je de lijst ook aanpassen en kan de gebruiker rijen verwijderen en nieuwe rijen aanmaken (probeer maar eens).



Naam	Soort	Gewicht	GeboorteJaar	
Fiffy	Hond	7	8/30/2012 12:00:00 AM	
Billy	Kat	13	2/7/2020 12:00:00 AM	
Flup	Papegaai	4.5	11/3/1980 12:00:00 AM	

Je kan de grid ook read only maken. Vind je hoe?

DataGrid koppelen aan SQLite

We bespreken een uitgewerkt voorbeeld van hoe je een DataGrid kan laten communiceren met een SQLite database. We maken een database voor de beheer van muzieknummers. We voorzien een class Nummer en een class Muzikant.

Nummer:

```
class Nummer
{
    public int ID { get; set; }
    public string Naam { get; set; }
    public int Lengte { get; set; }
    public int ArtiestID { get; set; }
}
```

Muzikant:

```
class Muzikant
{
    public int ID { get; set; }
    public string Naam { get; set; }
}
```

Om dit voorbeeld te kunnen draaien maak je bijhorende tabellen aan in SQLite en installeer je de SQLite Entity package. We voorzien 2 properties en 2 DataGrids:

```
private List<Muzikant> Muzikanten { get; set; } = new List<Muzikant> {};
private List<Nummer> Nummers { get; set; } = new List<Nummer> {};

public MainWindow()
{
```

```

InitializeComponent();

using (var db = new MusicDb())
{
    Muzikanten = db.Muzikant.ToList();
}

MuzikantenGrid.ItemsSource = Muzikanten;

using (var db = new MusicDb())
{
    Nummers = db.Nummer.ToList();
}

NummersGrid.ItemsSource = Nummers;
}

```

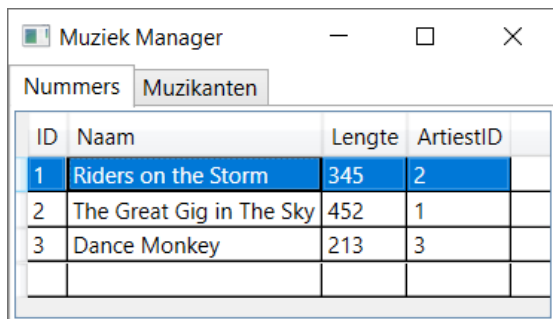
Je kan de 2 DataGrids in een eigen tabblad zetten met TabControl:

```

<TabControl>
    <TabItem Header="Nummers">
        <DataGrid x:Name="NummersGrid"></DataGrid>
    </TabItem>
    <TabItem Header="Muzikanten">
        <DataGrid x:Name="MuzikantenGrid"></DataGrid>
    </TabItem>
</TabControl>

```

Je zou zo iets moeten krijgen:



ID	Naam	Lengte	ArtiestID
1	Riders on the Storm	345	2
2	The Great Gig in The Sky	452	1
3	Dance Monkey	213	3

AutoGenerateColumns="False"

In plaats van de kolommen zelf te laten genereren kunnen we ook kiezen welke kolommen worden getoond. Daarvoor zet je AutoGenerateColumns op False op de DataGrid in XAML. Je kan dan kiezen welke properties er worden getoond via de Binding optie:

```

<DataGrid x:Name="NummersGrid" AutoGenerateColumns="False">
    <DataGrid.Columns>

```

```

        <DataGridTextColumn Header="Naam" Binding="{Binding Naam}" />
        <DataGridTextColumn Header="Lengte" Binding="{Binding Lengte}" />
    </DataGrid.Columns>
</DataGrid>

```

IsReadOnly="True"

Voor het wat eenvoudiger te houden gaan we voorlopig de DataGrid manipuleren via TextBoxen.

Code TextBoxen en Buttons

Hieronder staat de code voor de TextBoxen en Buttons te laten werken voor Nummers DataGrid te laten werken.

XAML:

```

<TabItem Header="Nummers">
    <StackPanel>
        <DataGrid x:Name="NummersGrid" AutoGenerateColumns="False" IsReadOnly="True" SelectionChanged="NummersGrid_SelectionChanged" SelectionMode="Single">
            <DataGrid.Columns>
                <DataGridTextColumn Header="Naam" Binding="{Binding Naam}"></DataGridTextColumn>
                <DataGridTextColumn Header="Lengte" Binding="{Binding Lengte}"></DataGridTextColumn>
            </DataGrid.Columns>
        </DataGrid>
        <StackPanel Width="200">
            <TextBlock>Naam</TextBlock>
            <TextBox x:Name="tbxNaam"></TextBox>
            <TextBlock Width="200">Lengte</TextBlock>
            <TextBox x:Name="tbxLengte"></TextBox>
            <TextBlock>Artiest</TextBlock>
            <ListBox x:Name="lb"></ListBox>
            <Button x:Name="btnAdd" Click="btnAdd_Click">Add</Button>
            <Button x:Name="btnUpdate" Click="btnUpdate_Click">Update</Button>
            <Button x:Name="btnDelete" Click="btnDelete_Click">Delete</Button>
        </StackPanel>
    </StackPanel>
</TabItem>

```

Code behind:

```

private void btnAdd_Click(object sender, RoutedEventArgs e)
{
    Muzikant muzikant = (Muzikant)lb.SelectedItem;

    if(muzikant == null)
    {
        MessageBox.Show("Selecteer een muzikant");
        return;
    }

    int lengte;

    try
    {
        lengte = int.Parse(tbxLengte.Text);
    }
    catch (FormatException)

```

```

    {
        MessageBox.Show("Voer een correcte lengte in (in seconden)");
        return;
    }

    using (var db = new MusicDb())
    {
        var nummer = new Nummer() { Naam = tbxNaam.Text, Lengte = lengte, ArtiestID = muzikant.ID };
        db.Add(nummer);
        Nummers.Add(nummer);
        NummersGrid.Items.Refresh();
        db.SaveChanges();
    }
}

private void btnUpdate_Click(object sender, RoutedEventArgs e)
{
    int lengte;

    try
    {
        lengte = int.Parse(tbxLengte.Text);
    }
    catch (FormatException)
    {
        MessageBox.Show("Voer een correcte lengte in (in seconden)");
        return;
    }

    using (var db = new MusicDb())
    {
        var nrInDB = db.Nummer.FirstOrDefault(n => n.ID == ((Nummer)(NummersGrid.SelectedItem)).ID);
        nrInDB.Naam = tbxNaam.Text;
        nrInDB.Lengte = lengte;

        var nrInLijst = Nummers.FirstOrDefault(n => n.ID == ((Nummer)(NummersGrid.SelectedItem)).ID);
        nrInLijst.Naam = tbxNaam.Text;
        nrInLijst.Lengte = lengte;

        db.SaveChanges();
        NummersGrid.Items.Refresh();
    }
}

private void btnDelete_Click(object sender, RoutedEventArgs e)
{
    if (NummersGrid.SelectedItem == null) return;

    using (var db = new MusicDb())
    {
        var nrInDB = db.Nummer.FirstOrDefault(n => n.ID == ((Nummer)(NummersGrid.SelectedItem)).ID);
        if (nrInDB == null) return;
        db.Nummer.Remove(nrInDB);

        var nrInLijst = Nummers.FirstOrDefault(n => n.ID == ((Nummer)(NummersGrid.SelectedItem)).ID);
        Nummers.Remove(nrInLijst);

        db.SaveChanges();
        NummersGrid.Items.Refresh();
    }
}

private void NummersGrid_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    var grid = (DataGrid)sender;

    Nummer geselecteerdeNummer = (Nummer)grid.SelectedItem;
}

```

```

    if (geselecteerdNummer == null) return;

    tbxNaam.Text = geselecteerdNummer.Naam;
    tbxLengte.Text = geselecteerdNummer.Lengte.ToString();

    int artiestID = geselecteerdNummer.ArtiestID;

    Muzikant m = Muzikanten.Find(m => m.ID == artiestID);

    lb.SelectedItem = m;
}

```

Voorzie zelf de code om de DataGridView van de Muzikanten aanpasbaar te maken.

Oefening 1

Maak een programma voor het beheer van de bestellingen in een café. Stel zelf een SQLite database op met tenminste 2 tabellen: dranken en bestellingen.

Voorzie functionaliteit om bestellingen en dranken toe te voegen met datagrids.

Oefening 2

Je kan de datagrids ook aanpasbaar maken, de gebruiker kan dan in de datagrid klikken en wijzigingen aanbrengen. Na een wijziging kan je deze automatisch laten bewaren in de SQLite tabel (of je kan eventueel werken met een “save” knop, misschien is dat beter?). Maak de datagrid van vorige oefening aanpasbaar (IsReadOnly weghalen op de datagrid) en zorg dat je kan bewaren na een wijziging in de datagrid.

Je zal hier waarschijnlijk wat voor moeten opzoeken...

<https://www.dotnetperls.com/datagrid-wpf>

<https://www.wpf-tutorial.com/datagrid-control/introduction/>