Der zugehörige XAML-Code sieht so aus:

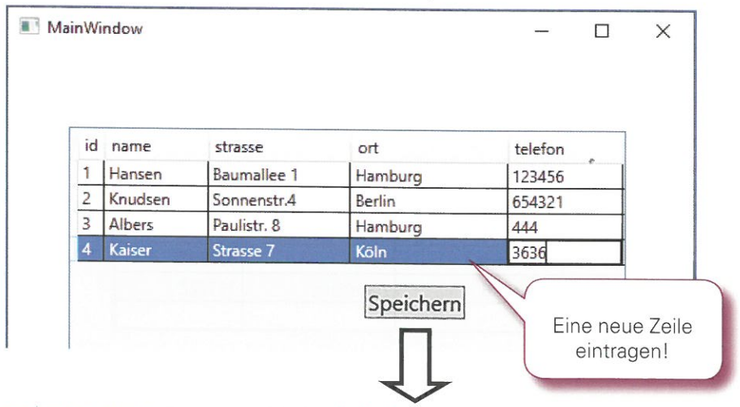
```
<DataGrid x:Name="kundenDataGrid" AutoGenerateColumns="False" Ena-
bleRowVirtualization="True" ItemsSource="{Binding}"
Margin="45,59,51,64" RowDetailsVisibilityMode="VisibleWhenSelected">

    <DataGrid.Columns>
        <DataGridTextColumn x:Name="idColumn"
        Binding="{Binding id}" Header="id" Width=
        "SizeToHeader"/>
        <DataGridTextColumn x:Name="nameColumn"
        Binding="{Binding name}" Header="name"
        Width="SizeToHeader"/>
        <DataGridTextColumn x:Name="strasseColumn"
            Binding="{Binding strasse}" Header="strasse"
            Width="SizeToHeader"/>
        <DataGridTextColumn x:Name="ortColumn" Binding=
        "{Binding ort}"Header="ort" Width="SizeToHeader"/>
        <DataGridTextColumn x:Name="telefonColumn"
        Binding="{Binding telefon}" Header="telefon"
        Width="SizeToHeader"/>
    </DataGrid.Columns>
</DataGrid>
```

Die Spalten passen sich der Überschrift an.

Die WPF-Datenbindung!

In die Tabelle können auch neue Werte eingetragen werden, allerdings findet keine automatische Synchronisierung statt. Dafür muss in der *Code-behind*-Datei eine bestimmte Methode der `DatenAdapterklasse` aufgerufen werden. Die folgende Ereignismethode zeigt die Vorgehensweise der Synchronisierung.




```
private void Button_Click(object sender, RoutedEventArgs e)
{
    try
    {
        OleDbCommandBuilder cmbKunden = new
            OleDbCommandBuilder(kundenDataSetKundenTableAdapter.
                                                    Adapter);
```

Durch die Instanzierung eines `CommandBuilder`-Objektes erhält die Adapter-Instanz alle nötigen Informationen für das ***Update***!


```
        kundenDataSetKundenTableAdapter.Update(kundenDataSet);
        MessageBox.Show("Update erfolgreich");
    }
    catch (System.Exception ex)
    {
        MessageBox.Show("Update-Fehler:" + ex);
    }
}
```

Update veranlassen!

Allerdings muss dazu in der *Code-behind*-Datei der Quellcode so geändert werden, dass die Verweise für die Datenbankanbindung als Attribute angelegt werden. Ansonsten wäre ein Zugriff auf das Adapterobjekt wie in der obigen Methode nicht möglich.

Der angepasste Quellcode sieht so aus:

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
    }

    private KundenDataSet kundenDataSet;

    private KundenDataSetTableAdapters.KundenTableAdapter
                                    kundenDataSetKundenTableAdapter;
    private System.Windows.Data.CollectionViewSource
                                    kundenViewSource;
    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        kundenDataSet =
        ((KundenDataSet)(this.FindResource("kundenDataSet")));
        kundenDataSetKundenTableAdapter = new
        KundenDataSetTableAdapters.KundenTableAdapter();
        kundenDataSetKundenTableAdapter.Fill(kundenDataSet.
        Kunden);
        kundenViewSource =
        ((Sysem.Windows.Data.CollectionViewSource)
            (this.FindResource("kundenViewSource")));

        kundenViewSource.View.MoveCurrentToFirst();
    }
    :
    :
}
```

Die Verweise als private Attribute anlegen.