UWP-041 - Keeping Data Controls Updated with ObservableCollection

Up to this point, whenever we were binding to data, the only scenario we dealt with was binding to a static set of items, which would bind that data, once, whenever the app first loads. However, we need to discuss another scenario, in which the collection of data gets updated, and we then need to programmatically add/remove new items in the List.

In order for this to work, we would have to tell the GridView that there is some new information to bind to. For that to happen, you have to use a different type of collection than just an ordinary List, and that would be to use an ObservableCollection instead.

To illustrate this point, here is a very simple example application. Imagine an app that takes in contact information and adds it to a contact list, which can then be displayed back in the app once the data binding takes place in real time.

The contact information isn't appearing because, up to this point, the application has been using an ordinary List. Here we make a simple change in MainPage.xaml.cs to step in the right direction

```
public sealed partial class MainPage : Page
{
    private List<Icon> Icons;
    //private List<Contact> Contacts;
    private ObservableCollection<Contact> Contacts;
```

By making the collection an ObservableCollection, your DataBoundControl - whether it's a GridView, ListView, etc - will be watching the collection for changes. And whenever a change happens, the ObservableCollection<Contact>, will tell the the DataBoundControl to re-bind to it.

Now, the x:Bind references in MainPage.xaml have already been configured, in the background, to facilitate this binding process with a List (since, in this example, we started out with a simple List). It is configured to work fine for ordinary Lists which pre-compiles the binding procedure and establishes that right before runtime.

But when changing your binding process from one that is with a List, to one that is an ObservableCollection you are changing this process from pre-compiling, to happening at runtime. So you have to re-initialize this configuration. A simple way to do this is to highlight all of your code that now is supposed to bind to the ObservableCollection

```
<GridView Grid.Row="2" ItemsSource="{x:Bind Contacts}" Margin="20">
    <GridView.ItemTemplate>
        <DataTemplate x:DataType="data:Contact">
            <StackPanel HorizontalAlignment="Center" Margin="10">
                <Image Source="{x:Bind AvatarPath}" Width="100" Height
                <StackPanel Orientation="Horizontal"
                            Margin="0,10,0,0"
                            HorizontalAlignment="Center">
                    <TextBlock Text="{x:Bind FirstName}" Margin="0,0,5
                    <TextBlock Text="{x:Bind LastName}" />
                </StackPanel>
            </StackPanel>
        </DataTemplate>
    </GridView.ItemTemplate>

</GridView>
```

And simply cut it, then wait a few seconds and paste it back in. What that does it kicks off the background configuration to recognize it now needs to bind with an ObservableCollection.

Now, the application can run as intended, doing the binding at runtime.

When searching around for further explanations related to data binding, you may come across something called “MVVM.” This is related to binding to data with your Universal Windows Platform app, your Windows 8.1, or Windows Phone 8.1 app, or even the Windows Presentation Foundation app; all of these XAML-based platforms.

MVVM stands for Model-View-ViewModel. It is a design pattern for writing code on the user interface that binds to data. What we showed in this example, with the ObservableCollection, is really just a building block/foundational concept towards learning more about MVVM.

This topic is covered in depth in the previous version of this series, the Windows Phone 8.1 Development for Absolute Beginners series on Channel9. If you want to learn more about MVVM, you'll want to take a look at all lessons that have to do with ObservableCollection, MVVM, or the interface called INotifyPropertyChanged at the following URL

https://channel9.msdn.com/Series/Windows-Phone-8-1-Development-for-Absolute-Beginners