

Data binding overview

02/08/2017 9 minutes to read Contributors

[Updated for UWP apps on Windows 10. For Windows 8.x articles, see the [archive](#)]

This topic shows you how to bind a control (or other UI element) to a single item or bind an items control to a collection of items in a Universal Windows Platform (UWP) app. In addition, we show how to control the rendering of items, implement a details view based on a selection, and convert data for display. For more detailed info, see [Data binding in depth](#).

Prerequisites

This topic assumes that you know how to create a basic UWP app. For instructions on creating your first UWP app, see [Get started with Windows apps](#).

Create the project

Create a new **Blank Application (Windows Universal)** project. Name it "Quickstart".

Binding to a single item

Every binding consists of a binding target and a binding source. Typically, the target is a property of a control or other UI element, and the source is a property of a class instance (a data model, or a view model). This example shows how to bind a control to a single item. The target is the **Text** property of a **TextBlock**. The source is an instance of a simple class named **Recording** that represents an audio recording. Let's look at the class first.

Add a new class to your project, name it Recording.cs (if you're using C#, C++ snippets provided below as well), and add this code to it.

C#Copy

```
namespace Quickstart
{
    public class Recording
    {
```

```

public string ArtistName { get; set; }
public string CompositionName { get; set; }
public DateTime ReleaseDateTime { get; set; }
public Recording()
{
    this.ArtistName = "Wolfgang Amadeus Mozart";
    this.CompositionName = "Andante in C for Piano";
    this.ReleaseDateTime = new DateTime(1761, 1, 1);
}
public string OneLineSummary
{
    get
    {
        return $"{this.CompositionName} by {this.ArtistName}, released: "
            + this.ReleaseDateTime.ToString("d");
    }
}
}
public class RecordingViewModel
{
    private Recording defaultRecording = new Recording();
    public Recording DefaultRecording { get { return this.defaultRecording; } }
}
}

```

C++Copy

```

#include <sstream>
namespace Quickstart
{
    public ref class Recording sealed
    {
    private:
        Platform::String^ artistName;
        Platform::String^ compositionName;
        Windows::Globalization::Calendar^ releaseDateTime;
    public:
        Recording(Platform::String^ artistName, Platform::String^
compositionName,
        Windows::Globalization::Calendar^ releaseDateTime) :
            artistName{ artistName },
            compositionName{ compositionName },
            releaseDateTime{ releaseDateTime } {}
        property Platform::String^ ArtistName

```

```

    {
        Platform::String^ get() { return this->artistName; }
    }
    property Platform::String^ CompositionName
    {
        Platform::String^ get() { return this->compositionName; }
    }
    property Windows::Globalization::Calendar^ ReleaseDateTime
    {
        Windows::Globalization::Calendar^ get() { return
this->releaseDateTime; }
    }
    property Platform::String^ OneLineSummary
    {
        Platform::String^ get()
        {
            std::wstringstream wstringstream;
            wstringstream << this->CompositionName->Data();
            wstringstream << L" by " << this->ArtistName->Data();
            wstringstream << L", released: " <<
this->ReleaseDateTime->MonthAsNumericString()->Data();
            wstringstream << L"/" <<
this->ReleaseDateTime->DayAsString()->Data();
            wstringstream << L"/" <<
this->ReleaseDateTime->YearAsString()->Data();
            return ref new Platform::String(wstringstream.str().c-str());
        }
    }
};
public ref class RecordingViewModel sealed
{
private:
    Recording^ defaultRecording;
public:
    RecordingViewModel()
    {
        Windows::Globalization::Calendar^ releaseDateTime = ref new
Windows::Globalization::Calendar();
        releaseDateTime->Month = 1;
        releaseDateTime->Day = 1;
        releaseDateTime->Year = 1761;
        this->defaultRecording = ref new Recording{ L"Wolfgang Amadeus
Mozart", L"Andante in C for Piano", releaseDateTime };
    }
}

```

```

        property Recording^ DefaultRecording
        {
            Recording^ get() { return this->defaultRecording; };
        }
    };
}

```

Next, expose the binding source class from the class that represents your page of markup. We do that by adding a property of type **RecordingViewModel** to **MainPage**.

C#Copy

```

namespace Quickstart
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
            this.ViewModel = new RecordingViewModel();
        }
        public RecordingViewModel ViewModel { get; set; }
    }
}

```

C++Copy

```

namespace Quickstart
{
    public ref class MainPage sealed
    {
    private:
        RecordingViewModel^ viewModel;
    public:
        MainPage()
        {
            InitializeComponent();
            this->viewModel = ref new RecordingViewModel();
        }
        property RecordingViewModel^ ViewModel
        {
            RecordingViewModel^ get() { return this->viewModel; };
        }
    }
}

```

```
    };  
}
```

The last piece is to bind a **TextBlock** to the **ViewModel.DefaultRecording.OneLiner** property.

XMLCopy

```
<Page x:Class="Quickstart.MainPage" ... >  
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">  
        <TextBlock Text="{x:Bind ViewModel.DefaultRecording.OneLineSummary}"  
            HorizontalAlignment="Center"  
            VerticalAlignment="Center"/>  
    </Grid>  
</Page>
```