Get SDK and tools

Quick access

My samples

Upload a sample

Browse sample requests

126,099

Points

Top 0.1%

XAML guy

Partner

Joined Sep 2008

View samples

Show activity

More from XAML guy

Best ComboBox Tutorial Ever, Ever! (57)

How to Build, Manage and Navigate the User Interface of a WPF Application (23)

WPF Printing Overview (9)

Easy WPF Login & Navigate Tutorial. Simple WPF Examples, in code-behind or MVVM (12)

See all

Easy MVVM Example

This project will give you crash course on WPF MVVM that you can do in your lunch break! Everything you need to know about INotifyPropertyChanged, Dependency Objects & Properites, POCO objects, Business Objects, Attached Properties and much m

Download	C# (380.0 KB)		
Ratings	(72)	Updated	9/20/2012
Downloaded	40,761 times	License	Apache License, Version 2.0
Favorites	Add to favorites	Share	
Requires	Visual Studio 2010		
Technologies	WPF, XAML		
Topics	Data Binding, events, MVVM, POCO, INotifyPropertyChanged, Attached Properties, WPF Binding, DataTem Dependency Properties, DataContext, DependencyObject, Business Objects		

Report abuse to Microsoft

Description | Browse code | Q and A (1)

Introduction

This project will give you crash course on WPF MVVM that you can do in your lunch break! Everything you need to know about binding, INotifyProp Dependency Objects & Properites, POCO objects, Business Objects, Attached Properties and much more!

For a full discussion and detailed breakdown of this project, please read below

http://social.technet.microsoft.com/wiki/contents/articles/13536.easy-mvvm-examples.aspx

Building the Sample

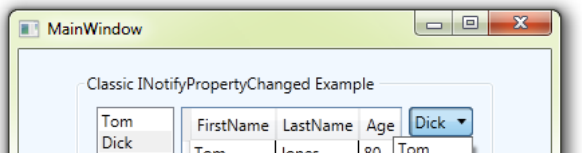
Just download, unzip, open and run!

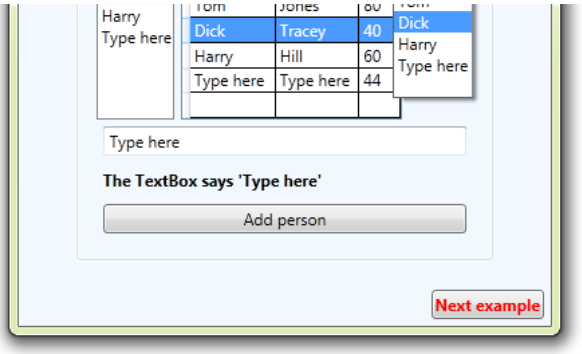
Description

This project consists of five windows, with practicly no code behind.

All application functionality and navigation is done by the ViewModels

MainWindow - Classic INotifyPropertyChanged





This is the classic MVVM configuration, implementing INotifyPropertyChanged in a base class (ViewModelBase)

The ViewModel is attached by the View itself, in XAML. This is fine if the ViewModel constructor has no parameters.

It has a ListBox, DataGrid and ComboBox all with ItemsSource to the same collection, and the same SeletedItem.

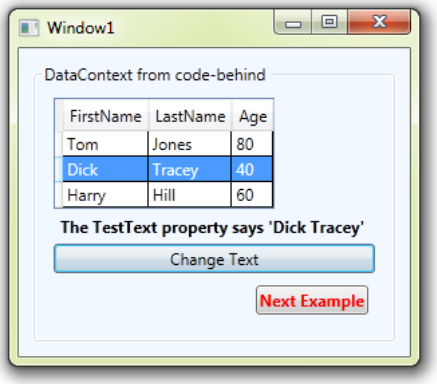
As you change selected Person, you will see all three controls change together.

A TextBox and TextBlock share the same property, and changes in the TextBox reflect in the TextBlock.

Click the button to add a user, it shows in all three controls.

Closing the Window is just a nasty code behind hack, the simplest and worst of the examples.

Window1



This window simply shows how you can attach the ViewModel to the DataContext in code, done by MainWindow.

C#

```
var win = new Window1 { DataContext = new ViewModelWindow1(tb1.Text) };
```

This ViewModel is derived from ViewModelMain, with an extra public property and command to pull from the base class and update the new prope

XAML

```
<Button Content="Change Text" Command="{Binding ChangeTextCommand}" CommandParameter="{Binding SelectedItem, Element
```

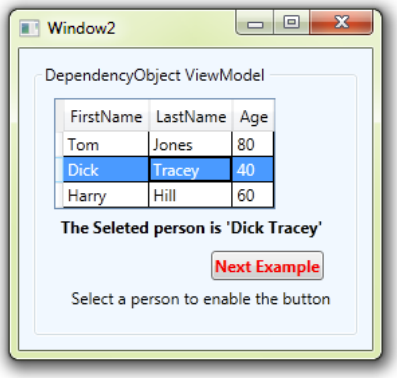
C#

```
void ChangeText(object selectedItem)
{
    //Setting the PUBLIC property 'TestText', so PropertyChanged event is fired
    if (selectedItem == null)
        TestText = "Please select a person";
    else
    {
        var person = selectedItem as Person;
        TestText = person.FirstName + " " + person.LastName;
    }
}
```

You can see I'm having to check for null here, "boiler plating" we could do without, as shown in CanExecute below.

Closing this Window uses the nicest way to do it, using an Attached Behaviour (Property) with a binding to a flag in the ViewModelBase. In our View call CloseWindow()

Window 2



This example shows the alternative to INotifyPropertyChanged - DependencyObject and Dependency Properties.

C#

```
public Person SelectedPerson
{
    get { return (Person)GetValue(SelectedPersonProperty); }
    set { SetValue(SelectedPersonProperty, value); }
}

// Using a DependencyProperty as the backing store for SelectedPerson.  This enables animation, styling, binding, et
public static readonly DependencyProperty SelectedPersonProperty =
    DependencyProperty.Register("SelectedPerson", typeof(Person), typeof(ViewModelWindow2), new UIPropertyMetadata(
```

Dependency Properties also fire PropertyChanged events, and also have callback and coerce delegates.

The only drawback to Dependency Properties for general MVVM use is they need to be handled on the UI layer.

This example also shows how a command can also control if a button is enabled, through it's CanExecute delegate.

As we are not using the parameter, but relyng on the ViewModel selected item, if there is none, the CanExecute method returns false, which disables done by behaviour, no messy code or boiler plating.

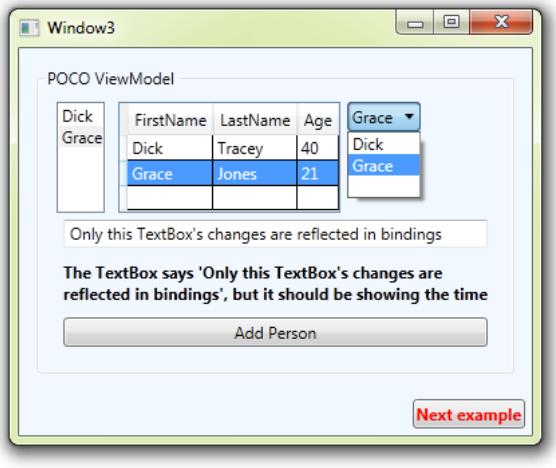
C#

```
public ViewModelWindow2()
{
    People = FakeDatabaseLayer.GetPeopleFromDatabase();
    NextExampleCommand = new RelayCommand(NextExample, NextExample_CanExecute);
}

bool NextExample_CanExecute(object parameter)
{
    return SelectedPerson != null;
}
```

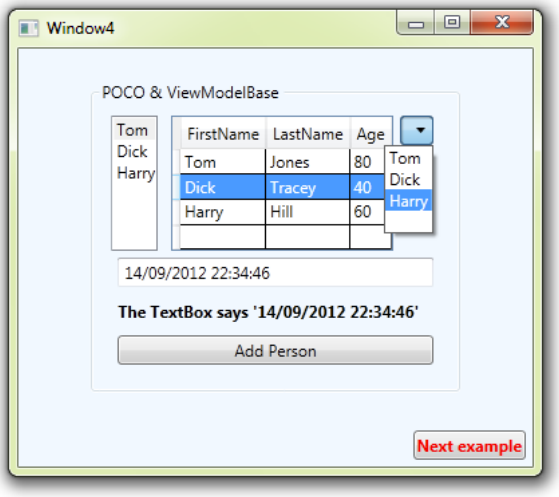
In this example, we still use the Attached Property in the Window XAML, to close the Window, but the property is another Dependency Property in t

Window 3



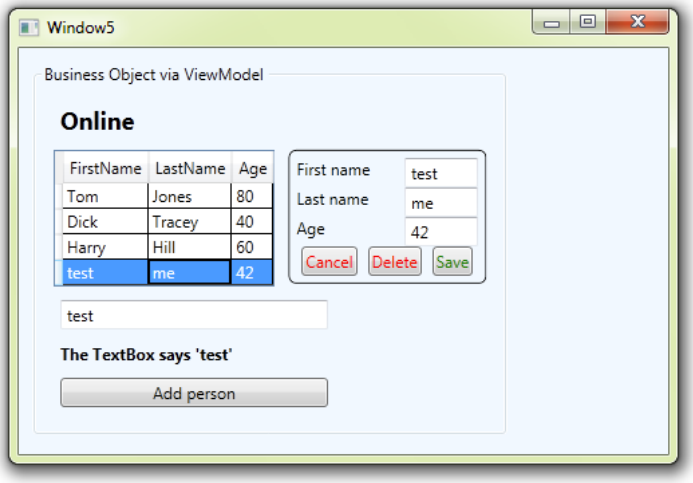
A POCO class in WPF/MVVM terms is one that does not provide any PropertyChanged events. This would usually be legacy code modules, or conver
WinForms.
If a POCO class is used in the classic INPC setup, things start to go wrong.
At first, everything seems fine. Selected item is updated in all, you can change properties of existing people, and add new people through the DataG
However, the textbox should actually be showing a timestamp, as set by the code behind Dispatcher Timer.
Also, clicking the button to add a new person does not seem to work, until you try to add a user in the DataGrid.

Window 4



This example is simply an extension to the previous example, where I have added the ViewModelBase and PropertyChanged event on the timer proj
see the time updating.

Window 5



What if you have a business object that handles all the work, like a database layer or web service?
This may therefore be a closed object that you cannot enrich with INPC on it's properties.
In this case you have to fall back on wrappers and polling.
This example shows a complete and virtually codeless master/detail, edit & add control.

For a full discussion and detailed breakdown of this project, please read below:

http://social.technet.microsoft.com/wiki/contents/articles/13536.easy-mvvm-examples.aspx

Downloads and tools
Windows 10 dev tools
Visual Studio
Windows SDK
Windows Store badges

Essentials
API reference (Windows apps)
API reference (desktop apps)
Code samples
How-to guides (Windows apps)

Learning resources
Microsoft Virtual Academy
Channel 9
Video gallery
Windows 10 by 10 blog series

Programs
Get a dev account
App Developer Agreement
Windows Insider Program
Microsoft Affliate Program

English

Privacy and cookies Terms of use Trademarks