Mishra, Siddharth

Beckman Coulter  [Company address]

WPF Concepts

Table of Contents

[WPF Command Model 2](#_Toc394861992)

[Commands 2](#_Toc394861993)

[Command Bindings 2](#_Toc394861994)

[Command Sources 2](#_Toc394861995)

[Command Targets 2](#_Toc394861996)

[ICommand Interface 2](#_Toc394861997)

[Execute() 2](#_Toc394861998)

[CanExecute() 2](#_Toc394861999)

[Event CanExecuteChanged 2](#_Toc394862000)

# WPF Command Model

The WPF command model consists of a surprising number of moving parts. Altogether, it has 4 key ingredients:

## Commands

A command represents an application task and keeps track of whether it can be executed. However, commands don’t actually contain the code that performs the application task.

## Command Bindings

Command binds to User Interface. You can use same command with different command bindings.

## Command Sources

A command source triggers a command. For example, Button is a command source.

## Command Targets

A command target is the element on which the command is being performed. For example, a Paste command might insert text into a TextBox, and an OpenFile command might pop a document into a DocumentViewer.

# ICommand Interface

The heart of the WPF command model is System.Windows.Input.ICommand interface that defoines how commands work. this interface includes two methods and an event:

|  |
| --- |
| public interface ICommand  {      void Execute(object parameter);        bool CanExecute(object parameter);      event EventHandler CanExecuteChanged;  } |

## Execute()

The Execute() method would contain the application task logic ( for example printing the document or opening another window). It accepts an additional parameter which user can use to pass along any extra information he needs.

## CanExecute()

The CanExecute() method will return state of the command. It will return True if it is enabled and false if it is disabled. It accepts an additional parameter which user can use to pass along any extra information he needs.

## Event CanExecuteChanged

This event is raised when the state changes. This is a signal to any control using the command that they should call the CanExecute() method to check the command’s state. This is part of the glue that allows command sources (Button or MenuItems) to automatically enable themselves when the command is available and to disable when it is not.

## Example:

<http://blogs.msdn.com/b/mikehillberg/archive/2009/03/20/icommand-is-like-a-chocolate-cake.aspx>

Given an instance of an ICommand, you just call Execute, and it does whatever it’s supposed to do. Except you shouldn’t call it if it’s CanExecute is false. If you want to know when CanExecute might be willing to give you a different answer, listen to the CanExecuteChanged event.

|  |
| --- |
| public class HelloWorldCommand : ICommand    {      #region ICommand Members        public void Execute(object parameter)      {        MessageBox.Show(parameter.ToString(), "Hello World Command", MessageBoxButton.OK, MessageBoxImage.Information);      }        public bool CanExecute(object parameter)      {        return true;      }      public event EventHandler CanExecuteChanged;        #endregion    } |

Once you have an ICommand instance handy, you can give it to a Button (on the Button.Command property), and Button knows what to do with it. As the simplest example, you can do this with the previous command:

|  |
| --- |
| <Window x:Class="WPFTutorial.XAMLTutorial"          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"          xmlns:cmd="clr-namespace:WPFTutorial.Command"          Title="XAMLTutorial"          Width="300"          Height="300">    <Window.Resources>      <cmd:HelloWorldCommand x:Key="HelloWorldCommand" />    </Window.Resources>    <Grid>  <Button x:Name="HelloWorldCommand"              Grid.Row="0"              Grid.Column="1"              Margin="5"              Command="{StaticResource HelloWorldCommand}"              CommandParameter="Hey, This is Hello world command."              Content="Hello World" />    </Grid>  </Window> |