Kostenloses eBook

LERNEN

mvvm-light

Free unaffiliated eBook created from
Stack Overflow contributors.

#mvvm-light

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: mvvm-light

It is an unofficial and free mvvm-light ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official mvvm-light.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit mvvm-light

Bemerkungen

MVVM-light ist ein in C # geschriebenes Toolkit, das die Erstellung und Entwicklung von MVVM-Anwendungen in WPF, Silverlight, Windows Store, Windows Phone und Xamarin beschleunigt.

Webseite: http://www.mvvmlight.net/

Es gibt ein plattformübergreifendes MVVM-Beispiel des Autors der Bibliothek, das Sie unter https://github.com/lbugnion/sample-crossplatform-flowers finden

Examples

RelayCommand

Der `RelayCommand` implementiert die `ICommand` Schnittstelle und kann daher zum Binden an `Command` s in XAML (als `Command` Eigenschaft des `Button` Elements) verwendet werden.

Der Konstruktor akzeptiert zwei Argumente. Die erste ist eine `Action` die ausgeführt wird, wenn `ICommand.Execute` aufgerufen wird (z. B. der Benutzer klickt auf die Schaltfläche), die zweite ist eine `Func<bool>` die festlegt, ob die Aktion ausgeführt werden kann (standardmäßig true, `canExecute` in genannt folgenden Absatz).

Die Grundstruktur ist wie folgt:

```
public ICommand MyCommand => new RelayCommand(
    () =>
    {
        //execute action
        Message = "clicked Button";
    },
    () =>
    {
        //return true if button should be enabled or not
        return true;
    }
);
```

Einige bemerkenswerte Effekte:

- Wenn `canExecute` false zurückgibt, die `Button` wird für den Benutzer deaktiviert
- Bevor die Aktion wirklich ausgeführt wird, wird `canExecute` erneut geprüft
- Sie können `MyCommand.RaiseCanExecuteChanged();` aufrufen `MyCommand.RaiseCanExecuteChanged();` um eine Neubewertung der `canExecute Func`

RelayCommand

Der `RelayCommand<T>` ähnelt dem `RelayCommand` , ermöglicht jedoch die direkte Übergabe eines Objekts an den Befehl. Es implementiert die `ICommand` Schnittstelle und kann daher zum Binden an `Command` s in XAML verwendet werden (z. B. als `Command` Eigenschaft des `Button` Elements). Sie können dann die `CommandParameter` Eigenschaft verwenden, um das Objekt an den Befehl zu übergeben.

XAML-Beispiel:

```
<Button Command="{Binding MyCommand}" CommandParameter="{Binding MyModel}" />
```

Der Konstruktor akzeptiert zwei Argumente. Die erste ist eine Aktion, die ausgeführt wird, wenn ICommand.Execute aufgerufen wird (z. B. der Benutzer klickt auf die Schaltfläche), die zweite ist eine Func <string, bool>, die bestimmt, ob die Aktion ausgeführt werden kann (standardmäßig auf true gesetzt, aufgerufen canExecute im folgenden Absatz). Die Grundstruktur ist wie folgt:

```
public RelayCommand<string> MyCommand => new RelayCommand<string>(
    obj =>
    {
        //execute action
        Message = obj;
    },
    obj =>
    {
        //return true if button should be enabled or not
        return obj != "allowed";
    }
);
```

Einige bemerkenswerte Effekte:

- Wenn `canExecute` kehrt `false` , die `Button` wird für den Benutzer deaktiviert
- Bevor die Aktion wirklich ausgeführt wird, wird `canExecute` erneut geprüft
- Sie können `MyCommand.RaiseCanExecuteChanged();` aufrufen `MyCommand.RaiseCanExecuteChanged();` um eine Neubewertung der `canExecute` `Func`

ObservableObject

Die `ObservableObject` Klasse enthält einige hilfreiche Methoden für das MVVM-Muster.

`RaisePropertyChanged` bietet eine sichere Methode zum Kompilieren, um Ereignisse mit Eigenschaftsänderungen `RaisePropertyChanged` .
Es kann mit aufgerufen werden

```
RaisePropertyChanged(() => MyProperty);
```

Die `Set` Methode kann im Eigenschaftssetter verwendet werden, um den neuen Wert festzulegen und das geänderte Eigenschaftsereignis auszulösen (nur bei Änderung). Bei einer Änderung wird `true` andernfalls `false` .
Verwendungsbeispiel:

```
private string _myValue;
public string MyValue
{
    get { return _myValue; }
    set { Set(ref _myValue, value); }
}
```

ViewModelBase

`ViewModelBase` erweitert `ObservableObject` und fügt einige für Viewmodels nützliche Methoden hinzu.

Mit der Eigenschaft `IsInDesignMode` oder `IsInDesignModeStatic` können Sie feststellen, ob der Code im Entwurfsmodus (in Visual Studio-Entwurfsansicht) ausgeführt wird oder nicht. Die beiden Eigenschaften sind genau gleich.

Erste Schritte mit mvvm-light online lesen: https://riptutorial.com/de/mvvm-light/topic/10610/erste-schritte-mit-mvvm-light

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit mvvm-light	Community, Florian Moser