eBook Gratuit

APPRENEZ mvvm-light

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#mvvm-light

Table des matières

À propos

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

Chapitre 1: Démarrer avec mvvm-light

Remarques

MVVM-light est une boîte à outils écrite en C # qui accélère la création et le développement d'applications MVVM dans WPF, Silverlight, Windows Store, Windows Phone et Xamarin.

Page Web: http://www.mvvmlight.net/

Il existe un exemple MVVM multi-plateforme de l'auteur de la bibliothèque, disponible sur https://github.com/lbugnion/sample-crossplatform-flowers

Examples

RelayCommand

`RelayCommand` implémente l'interface `ICommand` et peut donc être utilisé pour se lier à la `Command` s en XAML (en tant que propriété de `Command` de l'élément `Button`)

Le constructeur prend deux arguments; le premier est une `Action` qui sera exécutée si `ICommand.Execute` est appelé (par exemple, l'utilisateur clique sur le bouton), le second est un `Func<bool>` qui détermine si l'action peut être exécutée (true par défaut, appelé `canExecute` dans le paragraphe suivant).

la structure de base est la suivante:

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

Quelques effets notables:

- Si `canExecute` renvoie false, le `Button` sera désactivé pour l'utilisateur
- Avant que l'action ne soit vraiment exécutée, `canExecute` sera à nouveau vérifié
- Vous pouvez appeler `MyCommand.RaiseCanExecuteChanged();` forcer la réévaluation du `canExecute Func`

RelayCommand

`RelayCommand<T>` est similaire à `RelayCommand` , mais permet de passer directement un objet à la commande. Il implémente l'interface `ICommand` et peut donc être utilisé pour se lier à la `Command` s en XAML (par exemple, en tant que propriété de `Command` de l'élément `Button`). Vous pouvez ensuite utiliser la propriété `CommandParameter` pour transmettre l'objet à la commande.

Exemple XAML:

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

Le constructeur prend deux arguments; le premier est une Action qui sera exécutée si ICommand.Execute est appelée (par exemple, l'utilisateur clique sur le bouton), le second est un Func <string, bool> qui détermine si l'action peut être exécutée (true par défaut, appelé canExecute dans le paragraphe suivant). la structure de base est la suivante:

```
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";
    }
);
```

Quelques effets notables:

- Si `canExecute` renvoie `false` , le `Button` sera désactivé pour l'utilisateur
- Avant que l'action ne soit vraiment exécutée, `canExecute` sera à nouveau vérifié
- Vous pouvez appeler `MyCommand.RaiseCanExecuteChanged();` forcer la réévaluation du `canExecute` Func

ObservableObject

La classe `ObservableObject` contient des méthodes utiles pour vous aider avec le modèle MVVM.

`RaisePropertyChanged` fournit une méthode de compilation sûre pour générer des événements de modification de propriété.
On peut l'appeler avec

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

La méthode `Set` peut être utilisée dans le setter de propriétés pour définir la nouvelle valeur et déclencher l'événement de modification de propriété (uniquement en cas de modification). Il renvoie `true` si le changement s'est produit et `false` sinon.
exemple d'utilisation:

```
private string _myValue;
```

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

ViewModelBase

`ViewModelBase` étend `ObservableObject` et ajoute des méthodes utiles pour les modèles de vue.

La propriété `IsInDesignMode` ou `IsInDesignModeStatic` permet de déterminer si le code est exécuté en mode `IsInDesignModeStatic` (dans Visual Studio Design View) ou non. Les deux propriétés sont exactement les mêmes.

Lire Démarrer avec mvvm-light en ligne: https://riptutorial.com/fr/mvvm-light/topic/10610/demarrer-avec-mvvm-light

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec mvvm-light	Community, Florian Moser