Бесплатная электронная книга

УЧУСЬ

mvvm-light

Free unaffiliated eBook created from
Stack Overflow contributors.

#mvvm-light

Около

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

глава 1: Начало работы с mvvm-light

замечания

MVVM-light - это инструментарий, написанный на C #, который помогает ускорить создание и разработку приложений MVVM в WPF, Silverlight, Windows Store, Windows Phone и Xamarin.

Веб-страница: http://www.mvvmlight.net/

Существует кросс-платформенный образец MVVM от автора библиотеки, который можно найти на странице https://github.com/lbugnion/sample-crossplatform-flowers

Examples

RelayCommand

`RelayCommand` реализует интерфейс `ICommand` и поэтому может использоваться для привязки к `Command` s в XAML (как свойство `Command` элемента `Button`)

Конструктор принимает два аргумента; первый - это `Action` которое будет выполняться, если вызывается `ICommand.Execute` (например, пользователь нажимает кнопку), второй - это `Func<bool>` который определяет, можно ли выполнить действие (по умолчанию - true, называется `canExecute` в следующий пункт).

базовая структура выглядит следующим образом:

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

Некоторые заметные эффекты:

- Если `canExecute` возвращает false, `Button` будет отключена для пользователя
- Прежде чем действие действительно будет выполнено, `canExecute` будет снова проверено
- Вы можете вызвать `MyCommand.RaiseCanExecuteChanged();` для принудительной

переоценки `canExecute Func`

RelayCommand

`RelayCommand<T>` похож на `RelayCommand` , но позволяет напрямую передать объект команде. Он реализует интерфейс `ICommand` и поэтому может использоваться для привязки к `Command` s в XAML (например, как свойство `Command` элемента `Button`). Затем вы можете использовать свойство `CommandParameter` для передачи объекта команде.

Пример XAML:

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

Конструктор принимает два аргумента; первый - это действие, которое будет выполнено, если вызывается ICommand.Execute (например, пользователь нажимает кнопку), второй - Func <string, bool>, который определяет, может ли действие быть выполнено (по умолчанию используется true, canExecute в следующем абзаце). базовая структура выглядит следующим образом:

```
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";
    }
);
```

Некоторые заметные эффекты:

- Если `canExecute` возвращает `false` , `Button` будет отключена для пользователя
- Прежде чем действие действительно будет выполнено, `canExecute` будет снова проверено
- Вы можете вызвать `MyCommand.RaiseCanExecuteChanged();` для принудительной переоценки `canExecute Func`

ObservableObject

Класс `ObservableObject` содержит некоторые полезные методы, помогающие с шаблоном MVVM.

`RaisePropertyChanged` обеспечивает безопасный способ компиляции для изменения событий, связанных с изменением свойств.
Его можно вызвать с помощью

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

Метод `Set` может использоваться в настройщике свойств для установки нового значения и повышения события изменения свойства (только в случае изменения). Он возвращает `true` если изменения произошли, а `false` противном случае.
пример использования:

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

ViewModelBase

`ViewModelBase` расширяет `ObservableObject` и добавляет некоторые методы, полезные для viewmodels.

Свойство `IsInDesignMode` или `IsInDesignModeStatic` позволяет определить, выполняется ли код в режиме разработки (в Visual Studio Design View) или нет. Оба свойства абсолютно одинаковы.

Прочитайте Начало работы с mvvm-light онлайн: https://riptutorial.com/ru/mvvm-light/topic/10610/начало-работы-с-mvvm-light

кредиты

S. No	Главы	Contributors
1	Начало работы с mvvm-light	Community, Florian Moser