EBook Gratis

APRENDIZAJE

mvvm-light

Free unaffiliated eBook created from
Stack Overflow contributors.

#mvvm-light

Tabla de contenido

Acerca de

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Capítulo 1: Empezando con mvvm-light

Observaciones

MVVM-light es un kit de herramientas escrito en C # que ayuda a acelerar la creación y desarrollo de aplicaciones MVVM en WPF, Silverlight, Windows Store, Windows Phone y Xamarin.

Página web: http://www.mvvmlight.net/

Hay una muestra de MVVM multiplataforma del autor de la biblioteca que se puede encontrar en https://github.com/lbugnion/sample-crossplatform-flowers

Examples

RelayCommand

`RelayCommand` implementa la interfaz `ICommand` y, por lo tanto, se puede usar para enlazar con los `Command` en XAML (como la propiedad `Command` del elemento `Button`)

El constructor toma dos argumentos; la primera es una `Action` que se ejecutará si se llama a `ICommand.Execute` (por ejemplo, el usuario hace clic en el botón), la segunda es una función `Func<bool>` que determina si la acción se puede ejecutar (el valor predeterminado es verdadero, llamado `canExecute` en el siguiente párrafo).

La estructura básica es la siguiente:

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

Algunos efectos notables:

- Si `canExecute` devuelve false, el `Button` se desactivará para el usuario
- Antes de que la acción se ejecute realmente, `canExecute` se comprobará de nuevo
- Puede llamar a `MyCommand.RaiseCanExecuteChanged();` para obligar a la reevaluación de la `canExecute Func`

RelayCommand

El `RelayCommand<T>` es similar al `RelayCommand` , pero permite pasar directamente un objeto al comando. Implementa la interfaz `ICommand` y, por lo tanto, se puede usar para enlazar con los `Command` en XAML (por ejemplo, como la propiedad `Command` del elemento `Button`). Luego puede usar la propiedad `CommandParameter` para pasar el objeto al comando.

Ejemplo de XAML:

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

El constructor toma dos argumentos; la primera es una Acción que se ejecutará si se llama a ICommand.Execute (por ejemplo, el usuario hace clic en el botón), la segunda es una Func <string, bool> que determina si la acción se puede ejecutar (el valor predeterminado es true, llamado Puede ejecutar en el siguiente párrafo). La estructura básica es la siguiente:

```
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";
    }
);
```

Algunos efectos notables:

- Si `canExecute` devuelve `false` , el `Button` se desactivará para el usuario
- Antes de que la acción se ejecute realmente, `canExecute` se comprobará de nuevo
- Puede llamar a `MyCommand.RaiseCanExecuteChanged();` para obligar a la reevaluación de la `canExecute` Func

Objeto observable

La clase `ObservableObject` contiene algunos métodos útiles para ayudar con el patrón MVVM.

`RaisePropertyChanged` proporciona un método seguro de compilación para generar eventos de propiedades modificadas.
Se puede llamar con

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

El método `Set` se puede usar en el establecedor de propiedades para establecer el nuevo valor y elevar el evento de cambio de propiedad (solo si ocurrió el cambio). Devuelve `true` si ocurrió el cambio y `false` contrario.
ejemplo de uso:

```
private string _myValue;
```

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

ViewModelBase

`ViewModelBase` amplía `ObservableObject` y agrega algunos métodos útiles para los modelos de vista.

La propiedad `IsInDesignMode` o `IsInDesignModeStatic` permite determinar si el código se ejecuta en el modo de diseño (en la vista de diseño de Visual Studio) o no. Las dos propiedades son exactamente las mismas.

Lea Empezando con mvvm-light en línea: https://riptutorial.com/es/mvvm-light/topic/10610/empezando-con-mvvm-light

Creditos

S. No	Capítulos	Contributors
1	Empezando con mvvm-light	Community, Florian Moser