



Команды и взаимодействие с пользователем в MVVM

Последнее обновление: 21.06.2016



Удобный конструктор сайтов



В прошлой теме описывался паттерн MVVM, позволяющий отображать связанные данные. Однако, как правило, необходимо не только отображать данные, но и использовать какую-то логику взаимодействия с пользователем, обрабатывать пользовательский ввод. Рассмотрим, как это сделать в рамках паттерна MVVM.

Ключевой идеей паттерна является взаимодействие с моделью через ViewModel, то есть в данном случае использование событий визуальных компонентов и их обработчиков нежелательно. Чтобы решить эту задачу, в платформе Xamarin Forms имеется механизм команд.

Механизм команд раскрывается через интерфейс **ICommand**:

```
public interface ICommand
{
    void Execute(object arg);
    bool CanExecute(object arg);
    event EventHandler CanExecuteChanged;
}
```

Интерфейс ICommand определен в пространстве имен System.Windows.Input и расположен в сборке System.ObjectModel.

Данный интерфейс затем уже реализуется классом **Command**, который может использоваться следующими элементами управления:

- Button
- MenuItem
- ToolbarItem
- SearchBar
- TextCell
- ImageCell
- ListView
- TapGestureRecognizer

Итак, для применения команд возьмем проект приложения из прошлой темы. Вначале изменим в нем класс PhoneViewModel следующим образом:

```
using System.ComponentModel;
using System.Windows.Input;
using Xamarin.Forms;

public class PhoneViewModel : INotifyPropertyChanged
{
    // реализации ICommand
    public ICommand SavePhoneCommand { protected set; get; }
    public ICommand DeletePhoneCommand { protected set; get; }

    public event PropertyChangedEventHandler PropertyChanged;
    public Phone Phone { get; set; }

    public PhoneViewModel()
    {
        Phone = new Phone();
        this.SavePhoneCommand = new Command(SavePhone);
        this.DeletePhoneCommand = new Command(DeletePhone);
    }

    private void SavePhone()
    {
        // код по сохранению объекта Phone в бд, внешнем файле и т.д.
    }

    private void DeletePhone()
    {
        // код по удалению объекта Phone из бд и т.д.
        // в данном примере просто очищаем поля
        this.Title = "";
        this.Company = "";
        this.Price = 0;
    }

    public string Title
    {
        get { return Phone.Title; }
        set
        {
            if (Phone.Title != value)
            {
                Phone.Title = value;
                OnPropertyChanged("Title");
            }
        }
    }

    public string Company
    {
        get { return Phone.Company; }
        set
        {
            if (Phone.Company != value)
            {
                Phone.Company = value;
                OnPropertyChanged("Company");
            }
        }
    }

    public int Price
    {
        get { return Phone.Price; }
        set
```

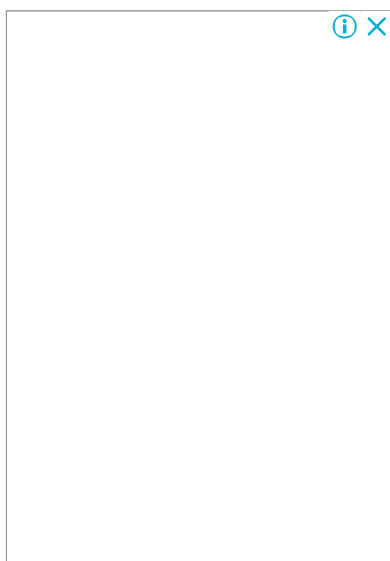
```
{
    if (Phone.Price != value)
    {
        Phone.Price = value;
        OnPropertyChanged("Price");
    }
}
protected void OnPropertyChanged(string propName)
{
    if (PropertyChanged != null)
        PropertyChanged(this, new PropertyChangedEventArgs(propName));
}
}
```

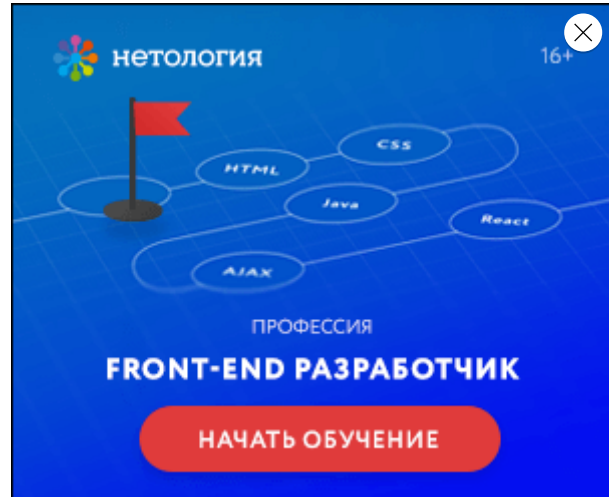
Здесь добавлены две команды `SavePhoneCommand` и `DeletePhoneCommand`. В конструкторе класса происходит создание этих команд. В реальном приложении эти команды могли бы выполнять функции по взаимодействию с бд или другим внешним хранилищем. В данном случае для краткости примера опустим данный функционал.

И также изменим код страницы в `xaml`, добавив в него кнопки с привязкой к вышеопределенным командам:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
              x:Class="MvvmApplication.MainPage">
    <StackLayout>
        <Entry Text="{Binding Title}" FontSize="Medium" />
        <Entry Text="{Binding Company}" FontSize="Medium" />
        <Entry Text="{Binding Price}" FontSize="Medium" />
        <StackLayout Orientation="Horizontal">
            <Button Text="Сохранить" Command="{Binding SavePhoneCommand}" />
            <Button Text="Удалить" Command="{Binding DeletePhoneCommand}" />
        </StackLayout>
    </StackLayout>
</ContentPage>
```

И при нажатии на кнопку происходит действие, заложенное в связанной команде. В итоге мы можем отказаться от событийной модели обработки пользовательского ввода и заменить ее на механизм команд.





4 Комментариев

metanit.com

1 Войти ▾

♥ Рекомендовать

🔗 Поделиться

Лучшее в начале ▾



Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS ?

Имя

**Александр** • 4 дня назад

А мне кажется люди придумали себе геморрой. Нет что бы использовать событие OnClick, они плодят куча кода: Реализация, назначение, описание, всё это ещё сбиндить. Это что бы выполнить в итоге 1 команду, в одну строчку... мда.

^ | ▾ • Ответить • Поделиться ›

**Ярослав Орлов** • год назад

"Ключевой идеей паттерна является взаимодействие с моделью через ViewModel, то есть в данном случае использование событий визуальных компонентов и их обработчиков нежелательно."

Подскажите, а чем нежелательность использования событий компонентов?

^ | ▾ • Ответить • Поделиться ›

**Unknown** ➔ Ярослав Орлов • 8 месяцев назад

Шаблон Command используют для взаимодействия с бизнес логикой.

1. Если вам нужно загрузить\изменить\создать\удалить модель бизнес логики(предметной области) - используем команды, которые, как правило, инкапсулируются в ViewModel.

2. Если вам нужно реагировать на события UI компонентов, и впоследствии менять способ отображения(например: зум) - это логика UI, для нее нужно использовать обычные события и обработчики инкапсулируются в классах контролов или страниц, но не в ViewModel

страниц, но не в ViewModel.

3. Не будет лишним познакомиться с Behaviors: <https://blog.xamarin.com/tu...>

^ | v • Ответить • Поделиться ›



Metanit Модератор ➔ Ярослав Орлов • год назад


потому что в идеале ViewModel взаимодействует с компонентами только через команды. Но это не значит, что мы вообще не можем использовать события, просто там, где возможно, вместо событий лучше использовать команды.

^ | v • Ответить • Поделиться ›

ТАКЖЕ НА METANIT.COM


C# и .NET | Раннее и позднее связывание

4 комментариев • 3 месяца назад

 **dev loop** — спасибо, ответили и на мой вопрос тоже)


Go | Синхронизация

1 комментарий • месяц назад

 **Roman** — Получается, что каналы и карты передаются в функцию по ссылке, верно?


Kotlin | Переменные



2 комментариев • 2 месяца назад

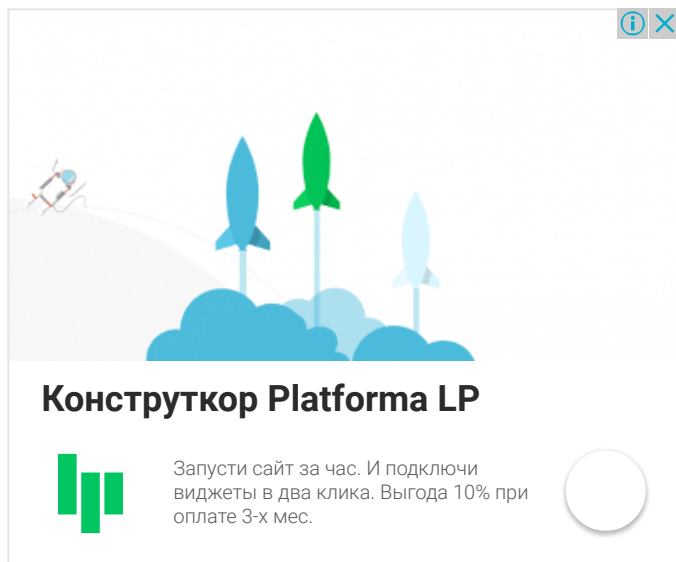
 **Иван Китаев** — Подскажите, как мы можем ввести переменные с клавиатуры?String Вводится ...

Angular в ASP.NET Core | Введение

16 комментариев • 3 месяца назад

 **Igor Teterkin** — Если у меня есть в контроллере (UserController) метод Send (отправка почты ...

 Подписаться  Добавить Disqus на свой сайтДобавить DisqusДобавить  Конфиденциальность



[Вконтакте](#) | [Twitter](#) | [Google+](#) | [Канал сайта на youtube](#) | [Помощь сайту](#)

Copyright © metanit.com, 2012-2017. Все права защищены.