Сообщения и MessagingCenter

Последнее обновление: 15.08.2016

Xamarin Forms поддерживает такую функциональность, как отправку сообщений. Для этого в Xamarin определен класс **MessagingCenter**, который поддерживает ряд методов:

- **Send<TSender>(TSender sender, string message)**: посылает сообщение message, в качестве отправителя выступает объект TSender
- **Subscribe<TSender>(object subscriber, string message, Action<TSender> callback)**: устанавливает подписку для объекта subscriber на получение сообщения message. При получении события будет вызываться действие callback
- **Unsubscribe<TSender>(object subscriber, string message)**: прекращает подписку для объекта subscriber на сообщение message

Рассмотрим применение сообщений на примере. Добавим в проект класс страницы, который назовем MessagePage:

```
using System;
using Xamarin.Forms;

namespace NavigationApp
{
    public class MessagePage : ContentPage
    {
        public MessagePage()
        {
            Title = "MessagePage";
            Button backBtn = new Button
            {
                Text = "Назад"
            };
            backBtn.Clicked += GoToBack;

            Content = new StackLayout { Children = { backBtn} };
        }
        // Переход обратно на MainPage
        private async void GoToBack(object sender, EventArgs e)
        {
            // отправляем сообщение
            MessagingCenter.Send<Page>(this, "LabelChange");
            await Navigation.PopAsync();
        }
    }
}
```

Здесь определена одна кнопка для возврата на предыдущую страницу. При возврате с помощью метода `MessagingCenter.Send` отправляется сообщение. Название сообщения - "LabelChange". В качестве отправителя установлен текущий объект, поэтому метод типизирован типом `Page`. Хотя можно было бы и конкретизировать тип отправителя:

```
MessagingCenter.Send<MessagePage>(this, "LabelChange");
```

Таким образом, сообщение отправляется. Теперь подпишемся на это событие. Пусть у нас есть главная страница MainPage:

```
using System;
using Xamarin.Forms;

namespace NavigationApp
{
```

```
    public partial class MainPage : ContentPage
    {
        Label stackLabel;
        public MainPage()
        {
            Title = "Main Page";
            Button forwardButton = new Button
            {
                Text = "Вперед"
            };
            forwardButton.Clicked += GoToForward;

            stackLabel = new Label
            {
                FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label))
            };
            Content = new StackLayout { Children = { forwardButton, stackLabel } };
            // устанавливаем подписку на сообщения
            Subscribe();
        }
        // установка подписки на сообщения
        private void Subscribe()
        {
            MessagingCenter.Subscribe<Page>(
                this, // кто подписывается на сообщения
                "LabelChange",   // название сообщения
                (sender)=> { stackLabel.Text = "Получено сообщение"; });    // вызываемое действие

        }
        // Переход вперед на MessagePage
        private async void GoToForward(object sender, EventArgs e)
        {
            MessagePage page = new MessagePage();
            await Navigation.PushAsync(page);
        }
    }
}
```

Здесь определена кнопка для перехода к странице MessagePage, и также определен метод подписки на сообщение, в котором вызывается `MessagingCenter.Subscribe`. Причем метод MessagingCenter.Subscribe типизирован именно тем типом, который является отправителем, то есть тип Page.

Получателем является текущий объект, поэтому в метод передается в качестве первого параметра this.

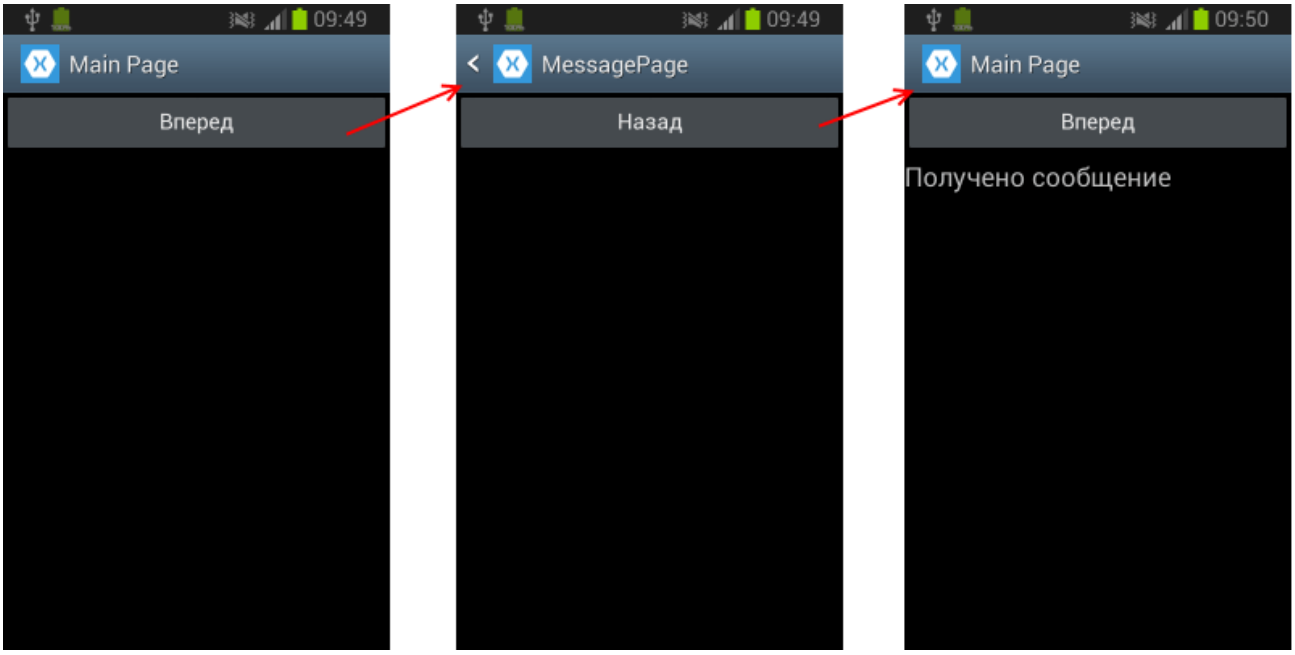
Название сообщения должно совпадать с тем, которое используется при отправке, то есть в нашем случае "LabelChange".

Третий параметр - действие просто выполняет установку текста метки, хотя тут могла бы быть и более сложная логика.

Ну и в главном классе приложения App страница MainPage должна передаваться в NavigationPage:

```
using Xamarin.Forms;

namespace NavigationApp
{
    public partial class App : Application
    {
        public App()
        {
            MainPage = new NavigationPage(new MainPage());
        }

        protected override void OnStart()
        {}

        protected override void OnSleep()
        {}

        protected override void OnResume()
        {}
    }
}
```

Теперь запустим приложение и перейдем с MainPage на MessagePage и обратно:



Назад Содержание Вперед

Andrey Lysenko • месяц назад

Смешно, набодобие такого "вселенского центра распределения и подписки событий" я пытался "слепить" еще когда учился программированию в классических win forms)))) приятно осозновать, что я был тогда на правильном пути)))

Ответить • Поделиться ›

XEN • 6 месяцев назад

Ну MessagingCenter это великая штука, можно перекидываться месседжами между нативной и ненативной частью

Ответить • Поделиться ›

Phenman • год назад

Что-то не могу понять в чем соль этого центра сообщений? Для чего или вместо чего он применяется на практике и чем он лучше обычных классических событий?

Ответить • Поделиться ›

Kreator ➚ Phenman • 7 месяцев назад

Насколько я понял, это очередь сообщений. Наподобие RabbitMQ и подобных. Но реализована в рамках отдельного приложения.
Это удобно: можно не пробрасывать данные в страницы через конструкторы и тд.
А записывать в очередь сообщений. И , соответственно, если в приложении есть дополнительные потоки, они могу подписываться на сообщения и выполнять действия.
Пусть меня поправят, если я не прав.

Ответить • Поделиться ›