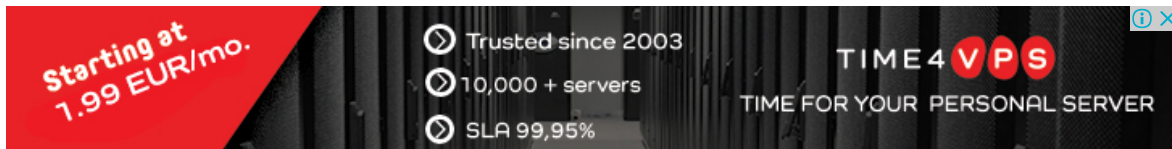




## Стек навигации

Последнее обновление: 25.06.2016



Когда происходит навигация на страницу с помощью вызова метода `PushAsync()` или `PushModalAsync()`, то возникает ряд действий:

- У страницы, с которой осуществляется переход, вызывается переопределенная версия метода `OnDisappearing()` (если данный метод в классе страницы переопределен)
- У страницы, на которую осуществляется переход, вызывается переопределенная версия метода `OnAppearing()` (если данный метод в классе страницы переопределен)
- После этого завершается выполнение методов `PushAsync/PushModalAsync`

Вызов методов `PopAsync()` и `PopModalAsync()` фактически приводит к тем же самым действиям, только в обратную сторону:

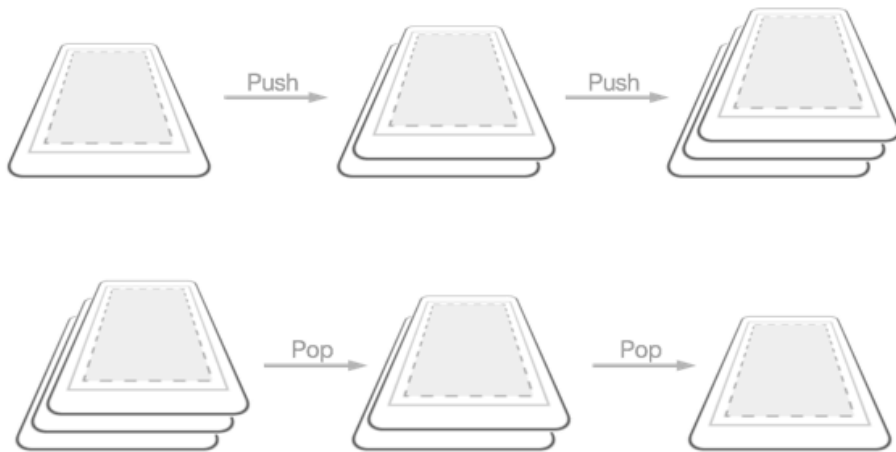
- У страницы, с которой осуществляется переход, вызывается переопределенная версия метода `OnDisappearing()` (если данный метод в классе страницы переопределен)
- У страницы, на которую осуществляется переход, вызывается переопределенная версия метода `OnAppearing()` (если данный метод в классе страницы переопределен)
- После этого завершается выполнение методов `PopAsync/PopModalAsync`

Из этих правил есть одно исключение: на платформе Android у страницы, которая вызывает метод `PushModalAsync`, не вызывается метод `OnDisappearing()`. И аналогично у страницы, которая вызвала метод `PopModalAsync()`, не вызывается метод `OnAppearing()`.

Для управления переходами интерфейс **INavigation** кроме вышеупомянутых методов также определяет два свойства:

- **NavigationStack**: стек страниц, который содержит все немодальные обычные страницы
- **ModalStack**: стек модальных страниц

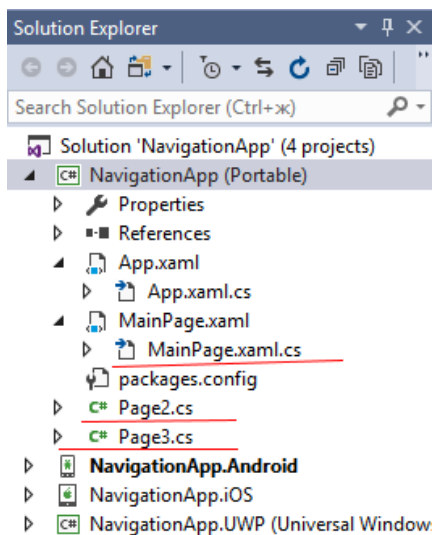
Эти свойства представляют коллекцию `IReadOnlyList<Page>`. Напрямую эти свойства доступны только для чтения, и мы можем влиять на них только с помощью вышеуказанных методов. Так, метод `PushAsync()` добавляет страницу в `NavigationStack`, а метод `PopAsync()`, наоборот, извлекает последнюю страницу из `NavigationStack`.



Аналогичным образом методы `PushModalAsync()` и `PopModalAsync()` изменяют содержимое в `ModalStack`.

Причем такое разделение на два стека имеет большое значение: мы можем перейти с обычной страницы на любую страницу, но с модальной мы можем перейти только на модальную страницу, или вернуться назад.

Для рассмотрения работы со стеками пусть в проекте будут три страницы - `MainPage` (главная) и дополнительные страницы `Page2` и `Page3`:



Код главной страницы `MainPage` будет выглядеть следующим образом:

```

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

        stackLabel = new Label();
        Content = new StackLayout { Children = { forwardButton, stackLabel } };
    }

    protected override void OnAppearing()
    {
        base.OnAppearing();
        if(loaded==false)
        {
            DisplayStack();
            loaded = true;
        }
    }
}
  
```

```
protected internal void DisplayStack()
{
    NavigationPage navPage = (NavigationPage)App.Current.MainPage;
    stackLabel.Text = "";
    foreach (Page p in navPage.Navigation.NavigationStack)
    {
        stackLabel.Text += p.Title + "\n";
    }
}
// Переход вперед на Page2
private async void GoToForward(object sender, EventArgs e)
{
    Page2 page = new Page2();
    await Navigation.PushAsync(page);
    page.DisplayStack();
}
}
```

На этой странице определяется кнопка для перехода вперед к странице Page2 и метка, в которую выводится текущее содержимое из NavigationStack.

Прежде всего здесь надо отметить метод DisplayStack(), который выводит все содержимое стека NavigationStack. Хотя в каждой странице мы нам напрямую доступно свойство Navigation.NavigationStack, однако это будет не общий стек, а стек, ассоциированный непосредственно с текущей страницей. Более того на момент использования по умолчанию он будет иметь 0 элементов. И чтобы получить общий стек, нам надо обратиться к NavigationPage:

```
NavigationPage navPage = (NavigationPage)App.Current.MainPage;
var stack = navPage.Navigation.NavigationStack;
```

Чтобы отобразить этот стек в MainPage мы переопределяем метод **OnAppearing()**, который срабатывает после загрузки страницы, в том числе после перехода на эту страницу. Причем здесь отображение стека срабатывает только один раз - при самой первой загрузке страницы. И чтобы этот момент отследить, применяется вспомогательная переменная loaded.

Третий момент - переход на страницу Page2 в обработчике кнопки. Здесь выполняется метод DisplayStack() у Page2 после того, как отработает метод await Navigation.PushAsync(page):

```
Page2 page = new Page2();
await Navigation.PushAsync(page);
page.DisplayStack();
```

Теперь определим страницу Page2:

```
public class Page2 : ContentPage
{
    Label stackLabel;
    public Page2()
    {
        Title = "Page 2";
        Button forwardBtn = new Button { Text = "Вперед" };
        forwardBtn.Clicked += GoToForward;

        Button backBtn = new Button { Text = "Назад" };
        backBtn.Clicked += GoToBack;

        stackLabel = new Label();
        Content = new StackLayout { Children = { forwardBtn, backBtn, stackLabel } };
    }
    protected internal void DisplayStack()
    {
        NavigationPage navPage = (NavigationPage)App.Current.MainPage;
        stackLabel.Text = "";
        foreach (Page p in navPage.Navigation.NavigationStack)
        {
            stackLabel.Text += p.Title + "\n";
        }
    }
    // Переход вперед на Page3
    private async void GoToForward(object sender, EventArgs e)
    {
        Page3 page = new Page3();
        await Navigation.PushAsync(page);
        page.DisplayStack();
    }
    // Переход обратно на MainPage
    private async void GoToBack(object sender, EventArgs e)
    {
        await Navigation.PopAsync();
    }
}
```

```

        NavigationPage navPage = (NavigationPage)App.Current.MainPage;
        ((MainPage)navPage.CurrentPage).DisplayStack();
    }
}

```

Здесь определены две кнопки для перехода вперед к Page3 и назад к MainPage. И также определен метод DisplayStack(), который аналогичен версии в MainPage.

Переход вперед к Page3 здесь аналогичен переходу к Page2 из MainPage.

А вот при переходе назад мы получаем страницу которая является последней в стеке (в данном случае MainPage) и вызываем у нее метод DisplayStack.

Класс NavigationPage определяет свойство **CurrentPage**. Это свойство указывает на страницу, которая находится последней коллекции NavigationStack.

И также определим страницу Page3:

```

public class Page3 : ContentPage
{
    Label stackLabel;
    public Page3()
    {
        Title = "Page 3";
        Button backBtn = new Button{ Text = "Назад"};
        backBtn.Clicked += GoToBack;

        stackLabel = new Label();
        Content = new StackLayout { Children = { backBtn, stackLabel } };
    }
    protected internal void DisplayStack()
    {
        NavigationPage navPage = (NavigationPage)App.Current.MainPage;
        stackLabel.Text = "";
        foreach (Page p in navPage.Navigation.NavigationStack)
        {
            stackLabel.Text += p.Title + "\n";
        }
    }
    // Переход обратно на Page2
    private async void GoToBack(object sender, EventArgs e)
    {
        await Navigation.PopAsync();

        NavigationPage navPage = (NavigationPage)App.Current.MainPage;
        ((Page2)navPage.CurrentPage).DisplayStack();
    }
}

```

Здесь также определены кнопка назад для перехода к Page2 и метка для вывода стека.

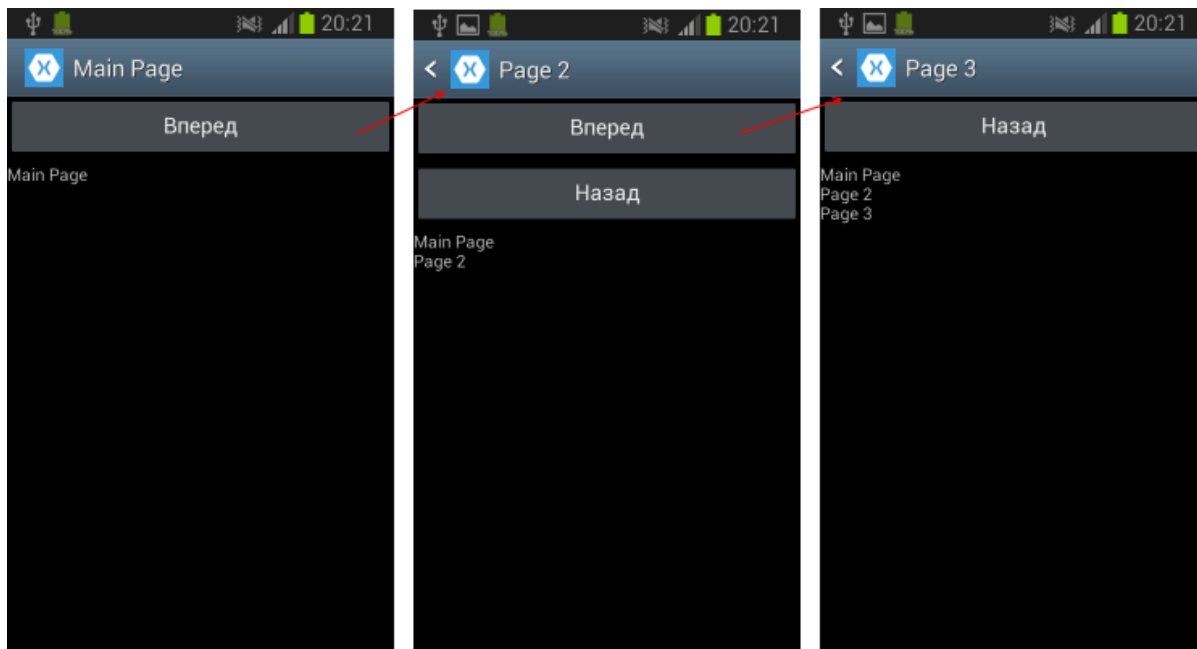
Может возникнуть вопрос: а зачем нам вызывать метод DisplayStack() у каждой страницы при переходе, если мы, допустим, можем это сделать в переопределенном методе OnAppearing(), который в любом случае вызывается системой при переходе на страницу?

Дело в том, что момент вызова OnAppearing() неопределен в том смысле, что он может происходить в то время, пока вызовы методов перехода PushAsync() и PopAsync() еще полностью не завершились. Соответственно пока не завершатся эти методы, в стеке может сохраняться страница, с которой был осуществлен переход. Поэтому, чтобы убедиться, что стек изменен, вывод стека делается в обработчиках кнопок именно после завершения методов навигации:

```

private async void GoToBack(object sender, EventArgs e)
{
    await Navigation.PopAsync();
    // переход завершен, стек изменился, можно выводить содержимое стека
    NavigationPage navPage = (NavigationPage)App.Current.MainPage;
    ((Page2)navPage.CurrentPage).DisplayStack();
}

```



Кроме того, мы можем получать различные страницы по индексу в стеке и тем самым манипулировать ими. Например, переход назад из Page3 на Page2 можно было бы осуществить так:

```
private async void GoToBack(object sender, EventArgs e)
{
    await Navigation.PopAsync();

    NavigationPage navPage = (NavigationPage)App.Current.MainPage;
    // получаем последнюю страницу в стеке
    Page page2 = navPage.Navigation.NavigationStack[navPage.Navigation.NavigationStack.Count - 1];
    ((Page2)page2).DisplayStack();
}
```

## Управление навигацией

Для управления стеком страниц интерфейс `INavigation` определяет три дополнительных метода:

- **RemovePage(page):** удаляет страницу `page` из стека
- **InsertPageBefore(pageA, pageB):** вставляет страницу `pageA` в стек перед страницей `pageB`
- **PopToRootAsync():** переходит на главную страницу

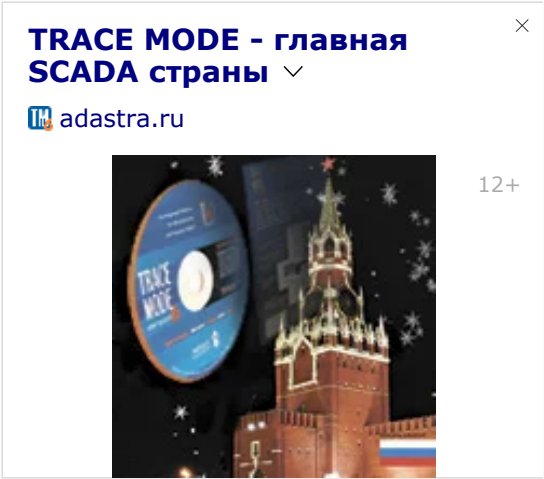
Например, добавим в стек страницу Page3 перед Page2:

```
NavigationPage navPage = (NavigationPage)App.Current.MainPage;
Page page2 = navPage.Navigation.NavigationStack[navPage.Navigation.NavigationStack.Count - 1];
navPage.Navigation.InsertPageBefore(new Page3 { Title = "Новая Page 3" }, page2);
```

Или добавим на Page3 новую кнопку, а в качестве обработчика кнопки назначим следующий метод:

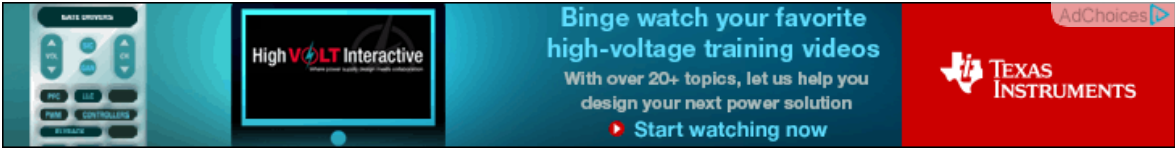
```
private async void GoToRoot(object sender, EventArgs e)
{
    await Navigation.PopToRootAsync();
    NavigationPage navPage = (NavigationPage)App.Current.MainPage;
    ((MainPage)navPage.CurrentPage).DisplayStack();
}
```

И в данном случае будет идти перенаправление на главную страницу MainPage.



Яндекс.Директ

[Назад](#) [Содержание](#) [Вперед](#)



2 Комментариев metanit.com

1 Войти ▾

 Рекомендовать
  Поделиться

Лучшее в начале ▾



Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS 

Имя

**Phenman** • год назад

Хотелось бы уточнить:

Согласно этой главе переходы осуществляются следующим образом:

С 1 на 2 страницу:

`page1.OnDisappearing();``page2.OnAppearing();`

Обратно со 2 на 1 страницу:

`page1.OnDisappearing();``page2.OnAppearing();`

Мне кажется это очень нелогичным. Может в тексте ошибка? по логике и если учитывать, что вы говорите об обратном порядке, должно быть так:

Обратно со 2 на 1 страницу:

`page2.OnDisappearing();``page1.OnAppearing();`

 • Ответить • Поделиться ›
**Metanit** Модератор → Phenman • год назад

да, поправил


 • Ответить • Поделиться ›

ТАКЖЕ НА METANIT.COM

**Vue.js | Навигация и ссылки**

1 комментарий • 3 месяца назад


**Delirium4Dude** — Огромное спасибо тебе за твои труды!  
Одни из самых толковых мануалов в Рунете
**Vue.js | Локальные и глобальные фильтры**

1 комментарий • 3 месяца назад



**eugene81** — Можно ли создать фильтр для фильтрации массивов?
**Go | Сервер. Обработка подключений**

2 комментария • 17 дней назад

**Metanit** — Да**Angular в ASP.NET Core | CRUD и маршрутизация. Часть 1**

1 комментарий • 2 месяца назад


**Vadim Prokopchuk** — Добрый вечер. Попытался в текущий пример добавить стилизацию, но столкнулся с проблемой. Как подгружать стили через webpack? Для

 Подписаться
  Добавить Disqus на свой сайт
 [Добавить Disqus](#)
[Добавить](#)
 Конфиденциальность