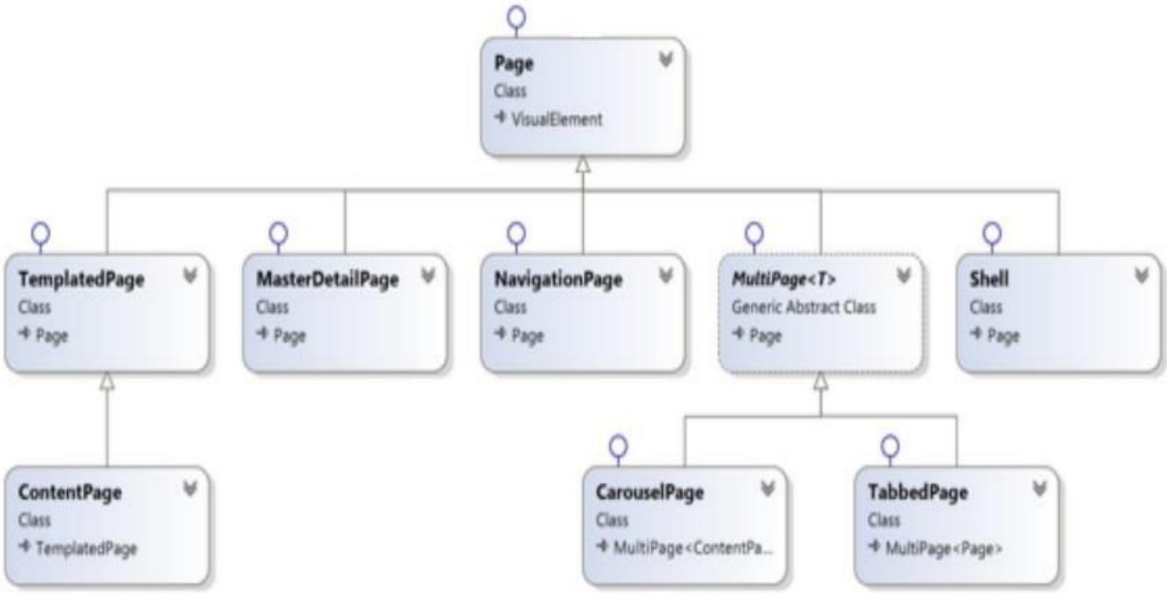
Giao diện ngư ời dùng: các trang như ContentPage, FlyoutPage, TabbedPage, Trang quay vòng và Trang mẫu

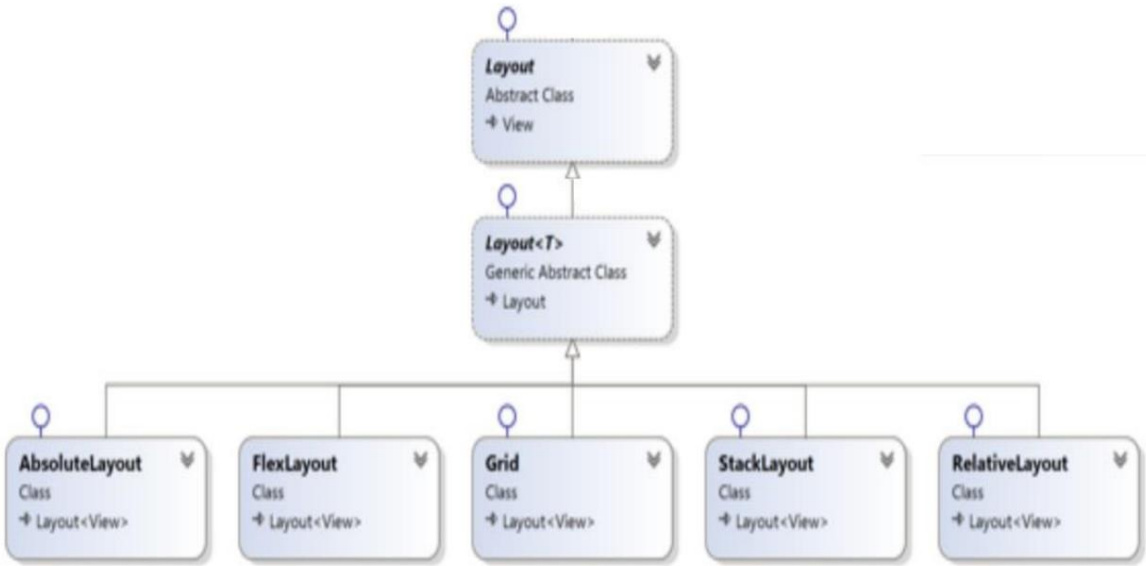
Dư ới đây là bốn nhóm yếu tố chính đư ợc sử dụng để Tạo giao diện ngư ời dùng của ứng dụng Xamarin.Forms.

- Trang
- Bố cục
- Điều khiển
- Tế bào

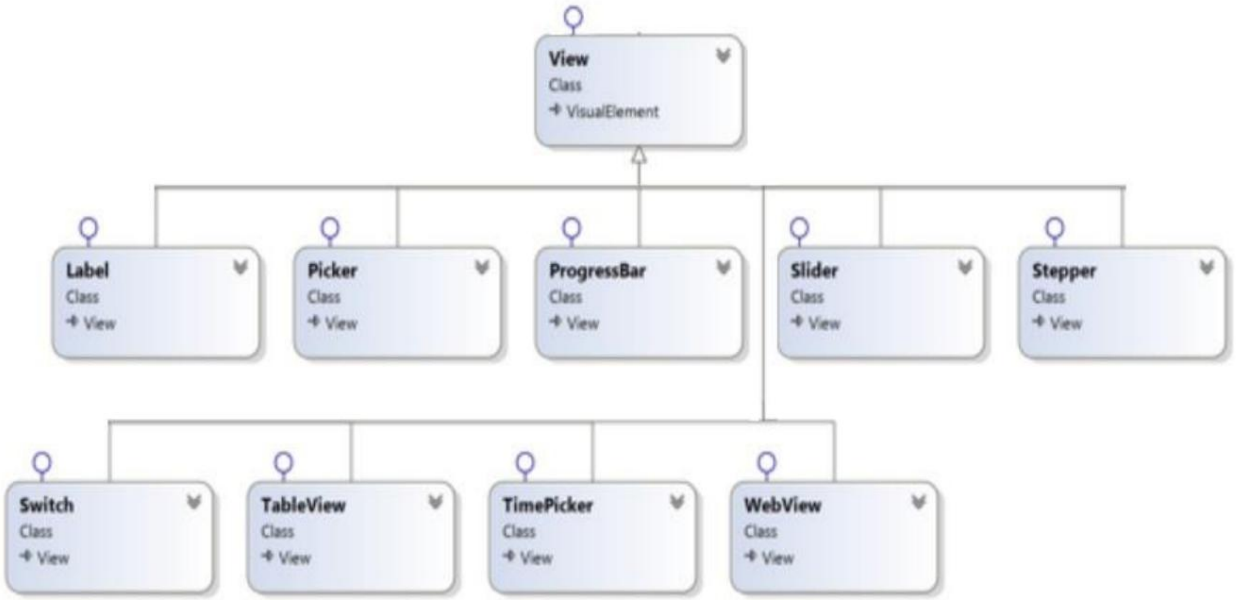
Trang ứng dụng Xamarin.Forms thư ờng chiếm toàn bộ màn hình. Tất cả các loại trang đều xuất phát từ lớp Trang.



Một trang thư ờng chứa bố cục chứa các điều khiển và có thể cả các bố cục khác. Tất cả các kiểu bố cục đều bắt nguồn từ lớp Bố cục.

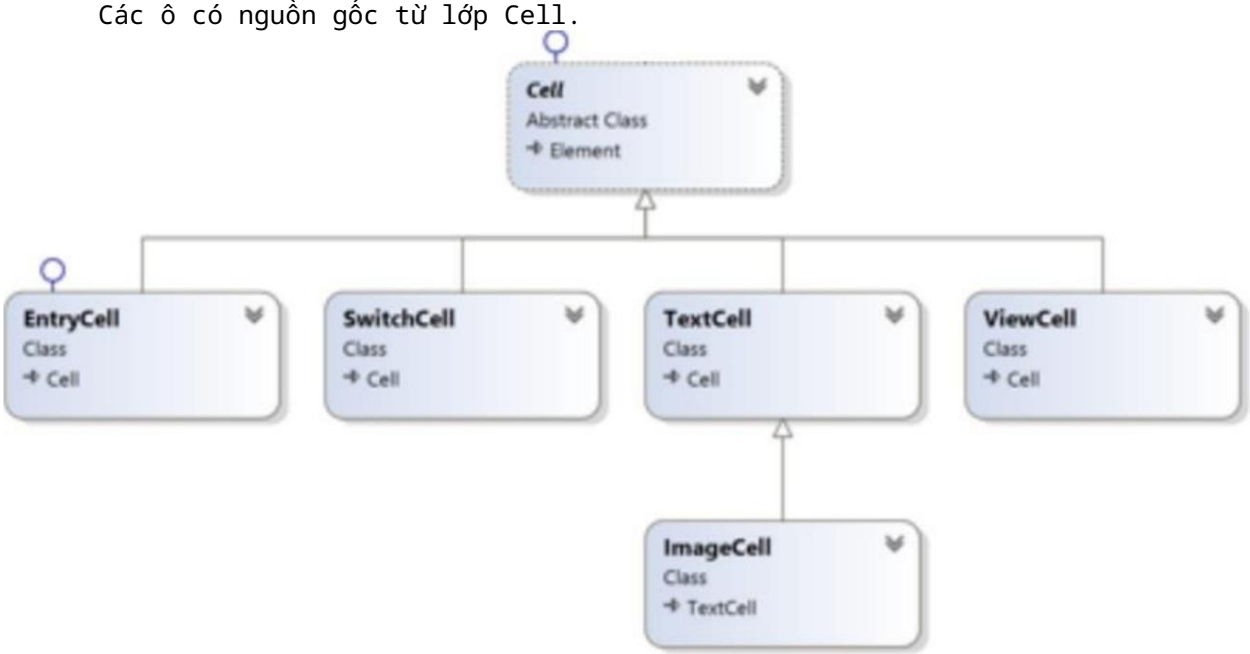


Bố cục chứa các điều khiển hoặc bố cục khác. Tất cả các loại các điều khiển có nguồn gốc từ lớp View.



Tế bào là những điều khiển chuyên biệt được sử dụng khi hiển thị dữ liệu trong TableView và ListView.

Các ô có nguồn gốc từ lớp Cell.



Các trang, bố cục, chế độ xem và ô cuối cùng đều bắt nguồn từ lớp Element.

ContentPage, FlyoutPage, NavigationPage, TabbedPage, TempatedPage, CarouselPage là các màn hình ứng dụng di động đa nền tảng. Tất cả các loại trang đều xuất phát từ lớp Trang.

Các thành phần hình ảnh này chiếm toàn bộ hoặc hầu hết màn hình màn hình.

Trong iOS, đối tượng lớp Trang được biểu thị dưới dạng ViewController và trong Android, mỗi trang là một Hoạt động nhưng không phải là đối tượng Hoạt động.

Hãy xem cách làm việc với các trang. ContentPage là loại trang đơn giản và phổ biến nhất, có thể chứa một điều khiển hoặc một bố cục. Để làm việc với trang ContentPage, bạn phải tạo một dự án có tên

ContentPageDemoPage cho Ứng dụng đa nền tảng Xamarin.Forms của mình bằng cách sử dụng mẫu Trống.

Xóa tự động được tạo và nằm trong tập tin
Mã giao diện người dùng MainPage.xaml.

Để thực hiện việc này, trong Solution Explorer, bấm chuột phải vào tệp MainPage.xaml và chọn Xóa. Hành động này cũng xóa tệp mã

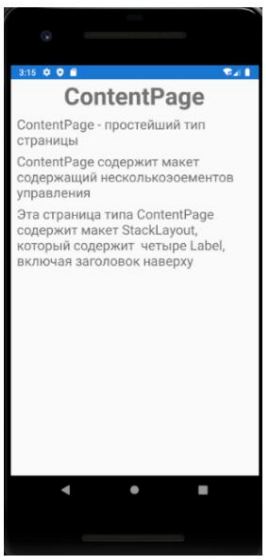
MainPage.xaml.cs.

Tạo một lớp trong một tệp mới có tên MainPage.cs. Làm cho lớp bắt nguồn từ lớp ContentPage. Thêm câu lệnh sử dụng Xamarin.Forms vì ContentPage nằm trong không gian tên
Xamarin.Forms.

```
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;

namespace ContentPageDemoPage
{
    Ссылок: 2
    public class MainPage: ContentPage
    {
        ссылка: 1
        public MainPage()
        {
            {
                Label header = new Label
                {
                    Text = "ContentPage",
                    FontSize = 40,
                    FontAttributes = FontAttributes.Bold,
                    HorizontalOptions = LayoutOptions.Center
                };

                Label label1 = new Label
                {
                    Text = "ContentPage - простейший тип страницы",
                    FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label)),
                };
```

```
            Label label2 = new Label
            {
                Text = "ContentPage содержит макет " +
                       "содержащий несколько" +
                       "эоементов управления",
                FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label)),
            };

            Label label3 = new Label
            {
                Text = "Эта страница типа ContentPage содержит " +
                   "макет StackLayout, который содержит  " +
                   "четыре Label, включая заголовок наверху",
                FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label)),
            };

            // Build the page.
            Title = "ContentPage Demo";
            Padding = new Thickness(10, 0);
            Content = new StackLayout
            {
                Children =
                {
                header,
                label1,
                label2,
                label3
                }
                };
            }
        }
    }
}
```

Tiếp theo, bạn cần xây dựng ứng dụng phần mềm.
Khi khởi chạy ứng dụng phần mềm, bạn sẽ thấy giao diện ngư ời dùng sau



FlyoutPage quản lý hai trang được xác định bởi thuộc tính Flyout và Chi tiết. Thuộc tính Flyout được sử dụng để chỉ định trang chính nơi hiển thị danh sách hoặc menu. Thuộc tính Chi tiết được sử dụng để chỉ định trang phụ hiển thị chi tiết hơn thông tin được chọn trên trang chính trong danh sách hoặc

thực đơn.

Thuộc tính IsPresented xác định trang nào đang được hiển thị - chính (thuộc tính IsPresented là đúng) hoặc phụ trợ (thuộc tính IsPresented là sai).

Hãy chuyển sang làm việc với FlyoutPage. Để thực hiện việc này, hãy tạo một dự án có tên FPageCS cho ứng dụng đa nền tảng Xamarin.Forms bằng cách sử dụng mẫu "Trống".

Hãy tạo một lớp phụ trợ NamedColor, lớp này cần thiết trong tương lai để làm việc với màu sắc và tên của chúng, lớp này được thực hiện trong tệp chương trình MainPage.cs, tệp này sẽ được tạo sau. Mở tập tin NamedColor.cs và nhập tệp mã chương trình bằng ngôn ngữ C Sharp được đưa ra trong bản trình bày. Trong mã chương trình đã cho, hàm tạo của lớp NamedColor có hai tham số đầu vào là kiểu chuỗi (để nhận tên màu) và Màu (để nhận và xuất chính màu đó).

```
sử dụng Hệ thống;
sử dụng System.Collections.Generic;
sử dụng System.Text;
sử dụng Xamarin.Forms;

không gian tên FPageCS
{
    lớp công khai NamedColor
  {
     công khai NamedColor()
     {
        ;
     }
     public NamedColor(tên chuỗi, Màu màu)
     {
        Tên = tên;
        Màu sắc = màu sắc;
     }
     Tên chuỗi công khai { bộ; lấy; }
     Màu công cộng Màu { bộ; lấy; }
     chuỗi ghi đè công khai ToString()
     {
        trả lại Tên;
     }
  }
}
```

Hãy tạo một lớp phụ trợ có tên NamedColorPage, lớp này cần thiết sau này để làm việc với trang phụ, tương ứng với thuộc tính Chi tiết của trang MainPage.

Mở file NamedColorPage.cs và gõ mã chương trình trong file
trong C Sharp, được đưa ra dưới đây.

Trong mã chương trình đã cho, một trang thuộc loại ContentPage được tạo, chứa hai bố cục thuộc loại Bố cục và điều khiển BoxView.

Mã chương trình trước tiên tạo điều khiển BoxView, điều khiển này được liên kết với màu được chọn trên trang chính trong điều khiển ListView bằng phương pháp SetBinding.

Tiếp theo, hàm CreateLabel được tạo, được thiết kế để tạo sáu điều khiển Nhãn hiển thị dữ liệu về màu được chọn trên trang chính trong điều khiển ListView.

Tiếp theo, sử dụng thuộc tính Nội dung, hình thức
trang có chứa:

bố cục kiểu StackLayout, bên trong có ba điều khiển Nhãn, hiển thị các thành phần màu được chọn trên trang chính (đỏ, lục, lam), một phần tử điều khiển;

điều khiển BoxView hiển thị màu được chọn trên trang chính;

Bố cục StackLayout chứa ba điều khiển Nhãn hiển thị màu sắc, độ bão hòa và độ sáng theo thang từ trắng đến đen (Độ sáng).

```
sử dụng Hệ thống;
sử dụng System.Collections.Generic;
sử dụng System.Text;
sử dụng Xamarin.Forms;

không gian tên FPageCS
{
    lớp NamedColorPage: ContentPage
  {
    công khai NamedColorPage()
    {
      // Điều khiển BoxView để hiển thị màu
      BoxView boxView = BoxView mới
      {
        ChiềurộngYêu cầu = 100,
        Chiều caoYêu cầu = 100,
        HorizontalOptions = LayoutOptions.Center
      };
      boxView.SetBinding(BoxView.ColorProperty, "Color");
```

```
        // Hàm tạo 6 điều khiển Label
        Func<string, string, Label> CreateLabel = (chuỗi nguồn, chuỗi fmt) => {

          Nhãn nhãn = Nhãn mới {

            FontSize = Device.GetNamedSize(NamedSize.Large, typeof(Label)),
            HorizontalTextAlignment = TextAlignment.End };

          label.SetBinding(Label.TextProperty,
            new Binding(source, BindingMode.OneWay, null, null, fmt));

          nhãn trả
        lại; };

        // Xây dựng trang (Stacklayout, BoxView, Stacklayout)
        Nội dung = StackLayout mới {

          Trẻ em = {

            StackLayout mới {

              HorizontalOptions = LayoutOptions.Center,
              VerticalOptions = LayoutOptions.CenterAndExpand,
              Children =
              {
                CreateLabel("Color.R", "R = {0:F2}"),
                CreateLabel("Color.G", "G = {0:F2}"),
                CreateLabel("Color.B", "B = { 0:F2}"),



            } },
            boxView,
            StackLayout mới {
              HorizontalOptions = LayoutOptions.Center,
              VerticalOptions = LayoutOptions.CenterAndExpand,
              Children =
              {
                CreateLabel("Color.Hue", "Hue = {0:F2}"),
                CreateLabel("Color.Saturation", "Saturation = {0:F2}"),
                CreateLabel("Color.Luminosity", "Luminosity = { 0:F2}")
              }
            }

          } };
      }

} }
```

Chúng ta sẽ tạo giao diện ngư ời dùng bằng cách sử dụng
C Mã chư ơng trình sắc nét.

Để thực hiện việc này, trư ớc tiên bạn cần xóa mã giao diện ngư ời dùng đư ợc tạo tự động nằm trong tệp MainPage.xaml. Để thực hiện việc này, trong Solution Explorer, bạn cần nhấp chuột phải vào tệp MainPage.xaml và chọn Xóa. Hành động này cũng xóa tệp mã MainPage.xaml.cs.

Tiếp theo, bạn cần tạo lớp MainPage, bạn cần di chuyển con trỏ đến dự án không có hậu tố, nhấp chuột phải, chọn menu Thêm, chọn Lớp và nhập tên lớp MainPage.cs ở cuối hộp thoại hộp. Thao tác này sẽ tạo một tệp có tên MainPage.cs. Mở tệp MainPage.cs.

Thêm câu lệnh sử dụng Xamarin.Forms ở đầu mã của bạn vì FlyoutPage nằm trong không gian tên
Xamarin.Forms.

Trong mã của bạn, hãy làm cho lớp MainPage xuất phát từ lớp FlyoutPage. Để thực hiện việc này, sau tên lớp MainPage, hãy đặt dấu hai chấm và nhập FlyoutPage.

Đầu tiên, một điều khiển Nhãn có tên là tiêu đề đư ợc tạo, chứa tiêu đề của trang - văn bản “FlyoutPage Demo”. Sau đó, một mảng có tênColors
đư ợc hình thành chứa các màu và
danh hiệu.

Tên màu sẽ đư ợc hiển thị dư ới dạng danh sách trong điều khiển ListView sẽ nằm trên trang chính, tức là trang tư ơng ứng với thuộc tính Flyout.

Tiếp theo, chúng ta tạo điều khiển ListView. Đối
tư ợng ListView đư ợc điền dữ liệu bằng thuộc tính ItemSource,
trong đó chỉ định rằng tên của các màu từ mảng đư ợc đặt tênColors sẽ đư ợc sử dụng làm mục danh sách trong điều khiển ListView.

Thuộc tính Margin của điều khiển ListView chỉ định khoảng cách
giữa các cạnh của trang và các cạnh của ListView.

Khoảng cách đư ợc thiết lập bằng cấu trúc Độ dày, trong mã chư ơng trình có hai tham số, tư ơng ứng với việc thiết lập khoảng cách theo chiều ngang và chiều dọc.

Giá trị ngang sẽ đư ợc áp dụng đối xứng ở cạnh trái và phải của ListView, còn giá trị dọc sẽ đư ợc áp dụng đối xứng ở cạnh trên và dư ới của điều khiển ListView

Tiếp theo, sử dụng thuộc tính Flyout, trang chính của loại
ContentPage có tên là "Danh sách màu".

Bằng cách sử dụng thuộc tính Nội dung, bố cục thuộc loại StackLayout sẽ đư ợc đặt vào trang FlyoutPage, trong đó, bằng cách sử dụng thuộc tính Children, các điều khiển Nhãn đư ợc tạo trư ớc đó có tên là tiêu đề, cũng như điều khiển ListView, sẽ đư ợc đặt.

Tiếp theo, bằng cách sử dụng thuộc tính Chi tiết của trang MainPage, một trang phụ chi tiếtTrang được tạo dưới dạng một thể hiện của lớp NamedColorPage, mã chương trình đã được thảo luận trước đó. Trang
phụ hiển thị thông tin chi tiết hơn về màu được chọn trong danh sách màu hiển thị trong điều khiển ListView trên trang chính.

Sau đây là mã xử lý sự kiện ItemSelected khi một màu được chọn từ danh sách được hiển thị trong điều khiển ListView. Khi bạn chọn một màu, đối tượng SelectedItem sẽ được
truy xuất làm đối số cho sự kiện ItemSelected. Tham số người gửi của sự kiện ItemSelected tương ứng với phần tử

Điều khiển ListView, trong đó màu được chọn.
Trang phụ trợ chi tiếtTrang, là một phiên bản của lớp NamedColorPage, được truyền thông tin về màu đã chọn (BindingContext) bằng thuộc tính Chi tiết.

Thuộc tính IsPresented của trang MainPage thuộc loại FlyoutPage có nhiệm vụ hiển thị trang chính hoặc trang phụ. Nếu khi khởi chạy một
ứng dụng phần mềm trên màn hình của thiết bị di động, cần hiển thị trang chính trước thì thuộc tính IsPresented được gán giá trị true và nếu trang phụ được sử dụng thì thuộc tính IsPresented được gán giá trị giá trị sai.

Tùy thuộc vào loại thiết bị di động mà ứng dụng phần mềm đang chạy và hướng của màn hình thiết bị di động, một số phương pháp được sử dụng để kiểm soát quá trình chuyển đổi từ trang chính sang trang phụ.

Thuộc tính kiểm soát việc chuyển đổi giữa các trang FlyoutLayoutBehavior, có thể nhận nhiều giá trị:

- Mặc định - các trang được hiển thị theo mặc định theo nền tảng được sử dụng
- Popover - trang phụ chồng lên một phần trang chính trang.
- Tách - Trang chính được hiển thị ở bên trái và trang phụ được hiển thị ở bên phải.
- SplitOnLandscape - màn hình thiết bị di động được chia thành hai các bộ phận nếu hướng màn hình là ngang
- SplitOnPortrait - màn hình thiết bị di động được chia thành hai phần nếu hướng màn hình là dọc

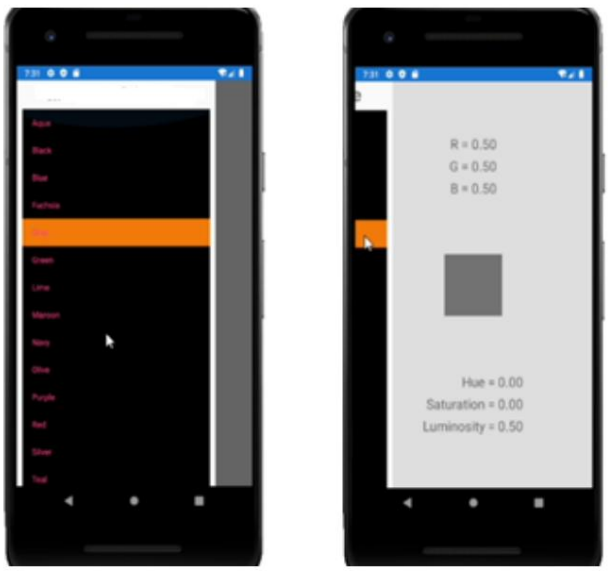
```
sử dụng Hệ thống;
sử dụng System.Collections.Generic;
sử dụng System.Text;
sử dụng Xamarin.Forms;

không gian tên FPageCS
```

```
{
   lớp MainPage: FlyoutPage {

      MainPage công khai()
      {
         Tiêu đề = "Bản demo FlyoutPage";

         Tiêu đề nhãn = Nhãn mới { Text

         = "FlyoutPage", FontSize =
            30, FontAttribut
            = FontAttribut.Bold, HorizontalOptions =
            LayoutOptions.Center };



         // Mảng các đối tư ợng NamedColor
         NamedColor[] tênColors = { new
            NamedColor("Aqua", Color.Aqua), new
            NamedColor("Black", Color.Black), new
            NamedColor("Blue", Color.Blue), new
            NamedColor ("Fuchsia", Color.Fuchsia), NamedColor
            mới ("Gray", Color.Gray), NamedColor mới
            ("Green", Color.Green), NamedColor mới ("Lime",
            Color.Lime), NamedColor mới (" Maroon ",
            Color.Maroon), NamedColor mới ("Navy", Color.Navy),
            NamedColor mới ("Olive", Color.Olive),
            NamedColor mới ("Purple", Color.Purple),
            NamedColor mới ("Đỏ" , Color.Red), NamedColor
            mới("Silver", Color.Silver), NamedColor
            mới("Teal", Color.Teal), NamedColor mới("Trắng",
            Color.White), NamedColor mới ("Vàng", Màu .
            Màu vàng) };



         // Tạo một điều khiển ListView cho trang thả xuống.
         ListView listView = ListView mới {

            ItemSource = tênColors, Margin =
            new Độ dày (10, 0),
            BackgroundColor=Color.Black,

         };

         // Tạo một trang thả xuống bằng cách sử dụng điều khiển ListView.
         Flyout = Trang nội dung mới {

            Tiêu đề = "Danh sách màu",
            Nội dung = StackLayout mới
```

```
            {
               Trẻ em = {
                  tiêu đề,
                  danh sáchXem
               }
            }
         };

         // Tạo một trang chi tiết sử dụng lớp NamedColorPage
         NamedColorPage chi tiếtPage = new NamedColorPage();
         Chi tiết = trang chi tiết;

         // Xác định một trình xử lý đã chọn cho ListView.
         listView.ItemSelected += (người gửi, args) => {
            // Đặt BindingContext của trang chi tiết.
            Chi tiết.BindingContext = args.SelectedItem;
            // Hiển thị trang chi tiết.
            IsPresented = đúng;
         };
         // Khởi tạo lựa chọn ListView.
         listView.SelectedItem = tênColors[5];
         FlyoutLayoutBehavior = FlyoutLayoutBehavior.Popover;
      }
   }
}
```

Sau khi khởi chạy ứng dụng phần mềm, trang chính sẽ hiển thị, trong đó hiển thị danh sách các màu được trang bị khả năng tua lại. Đầu tiên, màu xám được chọn từ danh sách màu.

Để xem trang phụ cung cấp thông tin chi tiết hơn về màu đã chọn, bạn cần trượt trang chính sang trái. Tiếp theo chọn màu đỏ và cũng dịch chuyển sang bên trái trang chính để xem trang phụ.



Tiếp theo, chúng ta sẽ xem xét làm việc với một trang thuộc loại NavigationPage, loại trang này kiểm soát việc điều hướng giữa các trang (hộp thoại) bằng cách sử dụng kiến trúc dựa trên ngăn xếp. Khi điều hướng các trang trong một ứng dụng phần mềm, một phiên bản của trang phải được chuyển tới hàm tạo của đối tượng NavigationPage. Lớp học

NavigationPage cung cấp điều hướng phân cấp nơi người dùng có thể di chuyển tiến và lùi qua các trang. Lớp học
NavigationPage triển khai điều hướng dựa trên một chồng đối tượng Trang bằng phương thức LIFO (vào sau, ra trước). Khi điều hướng từ trang này sang trang khác, ứng dụng phần mềm sẽ đẩy trang mới (đang được điều hướng đến) vào ngăn điều hướng, tại đây nó được đẩy lên đầu và trở thành trang hoạt động.

Để quay lại từ trang hiện tại về trang trước, ứng dụng sẽ xóa trang hiện tại khỏi đầu ngăn xếp điều hướng, sau đó trang ở đầu ngăn xếp sẽ hoạt động.

Các phương thức điều hướng được thuộc tính Điều hướng cung cấp cho các trang thuộc bất kỳ loại nào xuất phát từ lớp Trang. Trong điều hướng phân cấp, lớp

NavigationPage được sử dụng để điều hướng qua một chồng đối tượng ContentPage. Trang đầu tiên được thêm vào ngăn điều hướng được gọi là trang gốc của ứng dụng phần mềm.

Để làm việc với một trang thuộc loại NavigationPage, một dự án được tạo với tên NavPage dành cho ứng dụng đa nền tảng Xamarin.Forms sử dụng mẫu Trống. Giao diện người dùng sẽ được tạo bằng mã chương trình C Sharp. Để thực hiện việc này, trước tiên bạn cần xóa mã giao diện người dùng được tạo tự động nằm trong tệp MainPage.xaml. Để thực hiện việc này, trong Solution Explorer, bạn cần nhấp chuột phải vào

MainPage.xaml và chọn menu “Xóa”. Sau đó, tệp MainPage.xaml.cs được liên kết với tệp MainPage.xaml cũng bị xóa.

Tiếp theo, bạn cần mở tệp App.xaml.cs trong dự án của mình và thêm mã để thêm trang gốc, Trang1, vào ngăn điều hướng bằng cách sử dụng lớp NavigationPage. Kết quả là trang Trang 1

được đặt ở đầu ngăn xếp, bắt đầu hoạt động và hiển thị đầu tiên khi ứng dụng khởi động.

```
sử dụng Hệ thống;
sử dụng Xamarin.Forms;
sử dụng Xamarin.Forms.Xaml;

không gian tên NavPage
{
  lớp học công khai Ứng dụng: Ứng dụng
  {
    publicApp()
    {
      MainPage = new NavigationPage(new Page1());
    }
  }
}
```

Tiếp theo, một trang LabelPage được tạo, quá trình chuyển đổi sẽ được thực hiện từ Trang 1. Cũng từ Trang1 sẽ có sự chuyển đổi sang BoxViewPage.

Lớp LabelPage được tạo bằng các bước tạo lớp đã thảo luận trước đó. Tiếp theo bạn cần mở file đã tạo
LabelPage.cs và đặt lớp LabelPage thành lớp con của lớp ContentPage. Vì ContentPage và NavigationPage được Xamarin.Forms hỗ trợ, nên bạn phải thêm câu lệnh sử dụng Xamarin.Forms ở đầu mã để truy cập vào không gian tên thích hợp.

Tiếp theo, bạn cần nhập mã chương trình bằng ngôn ngữ C Sharp bên dưới vào tệp. Trang chứa
StackLayout với hai nhãn và hai nút để đi tới BoxViewPage và quay lại Trang1. Việc chuyển đổi sang BoxViewPage được thực hiện bằng cách nhấp vào nút NextPage.
Trình xử lý sự kiện nhấp vào nút NextPage gọi phương thức PushAsync của thuộc tính Điều hướng của LabelPage. Việc quay lại Trang 1 được thực hiện bằng cách nhấp vào nút PrevPage. Trình xử lý sự kiện xảy ra khi nút PrevPage được nhấp vào sẽ gọi phương thức PopAsync.

Kết quả là LabelPage bị xóa khỏi ngăn điều hướng và
Trang trên cùng trong ngăn xếp, tức là Trang 1, sẽ hoạt động.

```
không gian tên NavPage
{
  lớp NhãnTrang: Nội dungTrang
  {
    trang nhãn công khai()
    {
      Tiêu đề nhãn = Nhãn mới
      {
```

```
  Văn bản = "Nhãn",
  Cỡ chữ = 50,
  FontAttribut = FontAttribut.Bold,
  HorizontalOptions = LayoutOptions.Center
};

Nhãn nhãn = Nhãn mới
{
  Text = "Xamarin.Forms cho phép bạn tạo"giao diện        " +
 ngư ời dùng "phổ biến cho nền tảng Android và iOS"        " +

  FontSize = Device.GetNamedSize(NamedSize.Large,
                                    typeof(Nhãn)),
  Tùy chọn dọc = LayoutOptions.CenterAndExpand,
  Lề = Độ dày mới (10, 0)
};

Nút nextPageButton = Nút mới { Text =
"Trang tiếp theo",
 VerticalOptions = LayoutOptions.CenterAndExpand };

nextPageButton.Clicked += NextPage;

Nút trư ớcPageButton = Nút mới { Text =
"Trang trư ớc",
 VerticalOptions = LayoutOptions.CenterAndExpand };

trư ớcPageButton.Clicked += PrevPage;

//Xây dựng trang
Tiêu đề = "Bản trình diễn nhãn";
Nội dung = StackLayout mới
{
  Trẻ em =
  {
    tiêu đề,
    nhãn
    nút trang tiếp theo
    trư ớcTrangNút
  }
};

async void NextPage(ngư ời gửi đối tư ợng, EventArgs e)
{
  đang chờ Navigation.PushAsync(new BoxViewPage());
```

```
            }
            async void PrevPage(ngư ời gửi đối tư ợng, EventArgs e)
            {
               đang chờ Navigation.PopAsync();
            }
         }
      }
   }
```

Một trang BoxViewPage đư ợc tạo, quá trình chuyển đổi sẽ đư ợc thực hiện từ các trang Page1 và LabelPage. Để làm điều này bạn cần tạo một lớp BoxViewPage, mở tệp BoxViewPage.cs đã tạo và tạo một lớp BoxViewPage là con của lớp ContentPage.

Tiếp theo, bạn cần thêm câu lệnh sử dụng Xamarin.Forms ở đầu mã của mình. Sau đó, bạn cần nhập mã chư ơng trình bằng ngôn ngữ C Sharp bên dư ới vào tệp.

Trang chứa StackLayout với các điều khiển Label và BoxView cũng như một nút để quay lại LabelPage và Page1.

Việc quay lại các trang trư ớc đư ợc thực hiện bằng cách nhấp vào nút NextPage. Điều này gọi phư ơng thức PopAsync. Kết quả là BoxViewPage bị xóa khỏi ngăn xếp điều hư ớng và trang trên cùng trong ngăn xếp, tức là Trang1 hoặc LabelPage, sẽ hoạt động.

```
không gian tên NavPage
{
   lớp BoxViewPage: ContentPage
   {
      BoxViewPage công khai()
      {
         Tiêu đề nhãn = Nhãn mới
         {
            Văn bản = "BoxView",
            Cỡ chữ = 50,
            FontAttribut = FontAttribut.Bold,
            HorizontalOptions = LayoutOptions.Center
         };
         BoxView boxView = BoxView mới
         {
            Màu = Màu.Orange,
            Chiều rộngYêu cầu = 200,
            Chiều caoYêu cầu = 200,
            Tùy chọn ngang = LayoutOptions.Center,
            VerticalOptions = LayoutOptions.CenterAndExpand
         };
         Nút trư ớcPageButton = Nút mới
         {
            Text = "Trang trư ớc",
```

```
            VerticalOptions = LayoutOptions.CenterAndExpand
        };
        trước PageButton.Clicked += PrevPage;
        //Xây dựng trang
        Tiêu đề = "Bản demo BoxView";
        Nội dung = StackLayout mới
        {
            Trẻ em =
            {
                tiêu đề,
                hộpXem
                trước TrangNút
            }
        };
        async void PrevPage(người gửi đối tượng, EventArgs e)
        {
            đang chờ Navigation.PopAsync();
        }
    }
  }
}
```

Hãy chuyển sang tạo Trang1. Tạo một lớp Page1, mở tệp Page1.cs đã tạo và đặt lớp Page1 thành lớp con của lớp ContentPage. Thêm câu lệnh sử dụng Xamarin.Forms ở đầu chương

trình

mã vì ContentPage nằm trong không gian tên Xamarin.Forms.

Nhập vào file mã chương trình bằng ngôn ngữ C Sharp được đưa ra trong bài thuyết trình. Bên trong trang có bố cục có hai nút để điều hướng đến LabelPage và BoxViewPage. Ngoài ra, trình xử lý sự kiện nhấn nút

cũng được hình thành trong mã chương trình. Trình xử lý sử dụng phương thức PushAsync của thuộc tính Điều hướng của Trang1 để điều hướng đến LabelPage và BoxViewPage.

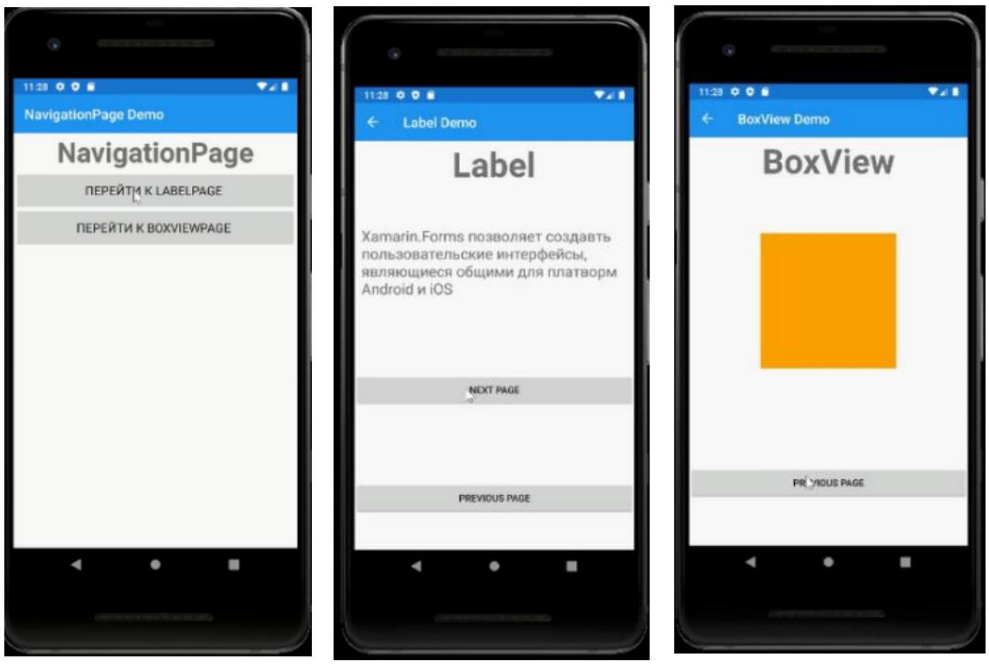
```
không gian tên NavPage
{
  lớp Trang1: Nội dungTrang
  {
    Trang công khai1()
    {
      Tiêu đề nhãn = Nhãn mới
      {
        Text = "Trang điều hướng",
        Cỡ chữ = 40,
        FontAttribut = FontAttribut.Bold,
        HorizontalOptions = LayoutOptions.Center
      };
```

```
            Nút nút1 = Nút mới {

               Text = "Vào LabelPage ", Font =
               Font.SystemFontOfSize(NamedSize.Large), BorderWidth =
               1,
               widthRequest=400 };

            nút1.Clicked += async (ngư ời gửi, args) =>
               đang chờ Navigation.PushAsync(new LabelPage());

            Nút nút2 = Nút mới {

               Text = "Vào BoxViewPage", Font =
               Font.SystemFontOfSize(NamedSize.Large), BorderWidth =
               1, widthRequest =
               400 }; nút2.Clicked

            += async (ngư ời gửi, args) =>
               đang chờ Navigation.PushAsync(new BoxViewPage());
            //Xây dựng trang
            Title = "Bản trình diễn trang điều hư ớng";
            Nội dung = StackLayout mới {

               Trẻ em = {

                  tiêu
                  đề, StackLayout
                  mới {
                     HorizontalOptions = LayoutOptions.Center,
                     Children = {

                        nút1,
                        nút2
                     }
                  }

               } };
         }
      }
```

} Khi bạn khởi chạy ứng dụng phần mềm, trang Page1 với hai nút “Go to LabelPage” và “Go to BoxViewPage” sẽ hiển thị trên màn hình của thiết bị di động.

Nhấp vào nút “Chuyển đến LabelPage” sẽ đư a bạn đến trang LabelPage, từ đó bạn có thể truy cập BoxViewPage hoặc

quay lại trang gốc Trang1. Nếu bạn điều hư ớng đến BoxViewPage, bạn có thể quay lại LabelPage.



Tiếp theo, hãy xem xét cách làm việc với trang TabbedPage, một trang con của lớp MultiPage trừu tư ợng và cho phép bạn điều hư ớng giữa các trang bằng cách sử dụng danh sách các tab.

Trên iOS, danh sách các tab xuất hiện ở cuối màn hình cũng như trong vùng dữ liệu ở trên cùng.

Mỗi tab có tiêu đề và cũng có thể có biểu tư ợng, biểu tư ợng này phải là tệp PNG.

Ở hư ớng dọc, các biểu tư ợng thanh tab xuất hiện phía trên
tiêu đề tab.

Ở hư ớng ngang, các biểu tư ợng và tiêu đề xuất hiện cạnh nhau.

Ngoài ra, tùy thuộc vào thiết bị và hư ớng của bạn, bạn có thể thấy thanh tab thông thư ờng hoặc thanh tab nhỏ gọn.

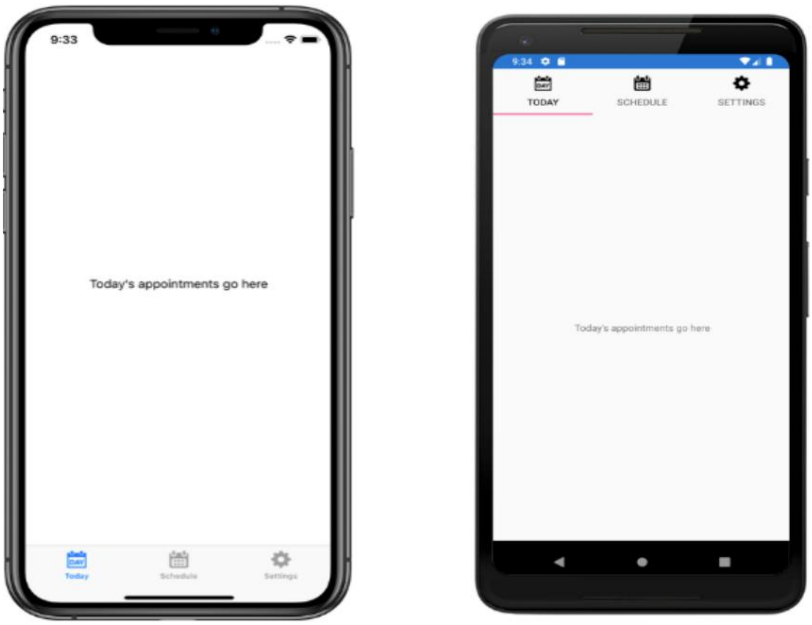
Nếu bạn có nhiều hơn năm tab, tab Nâng cao sẽ xuất hiện và đư ợc sử dụng để truy cập các tab bổ sung.

Trên Android, danh sách các tab xuất hiện ở đầu màn hình cũng như trong vùng dữ liệu bên dư ới.

Mỗi tab có tiêu đề và biểu tư ợng, phải là tệp PNG có kênh alpha.

Như ng những tab này có thể đư ợc di chuyển xuống cuối màn hình trong
tùy thuộc vào nền tảng cụ thể.

Nếu bạn có nhiều hơn năm tab đư ợc liệt kê ở cuối màn hình, tab Nâng cao sẽ xuất hiện mà bạn có thể sử dụng để truy cập các tab bổ sung.



Có hai cách để tạo TabbedPage:

1. TabbedPage chứa một tập hợp các đối tượng con
gõ Trang, ví dụ: các trang thuộc loại ContentPage.

2. Gán tập hợp các trang Trang cho thuộc tính ItemsSource của trang TabbedPage, sau đó gán mẫu DataTemplate cho thuộc tính ItemTemplate để hiển thị thống nhất các trang có trong
thành phần của bộ sưu tập.

Với cả hai cách tiếp cận, TabbedPage hiển thị một trang
tương ứng với tab đã chọn.

TabbedPage có các thuộc tính sau:

BarBackgroundColor (loại Màu) - màu nền của thanh tab;

BarTextColor (Loại màu) - màu văn bản trên thanh tab;

SelectedTabColor (Loại màu) - màu của tab đã chọn;

UnselectedTabColor (loại Màu) - màu của tab nếu nó không được chọn.

Hãy tạo dự án TabPage cho dự án đa nền tảng Xamarin.Forms bằng mẫu Trống. Hãy mở tập tin
MainPage.xaml.cs và điều chỉnh mã chương trình cho phù hợp với cách trình bày. Từ tệp này, bạn có thể thấy rằng trang MainPage là trang TabbedPage.

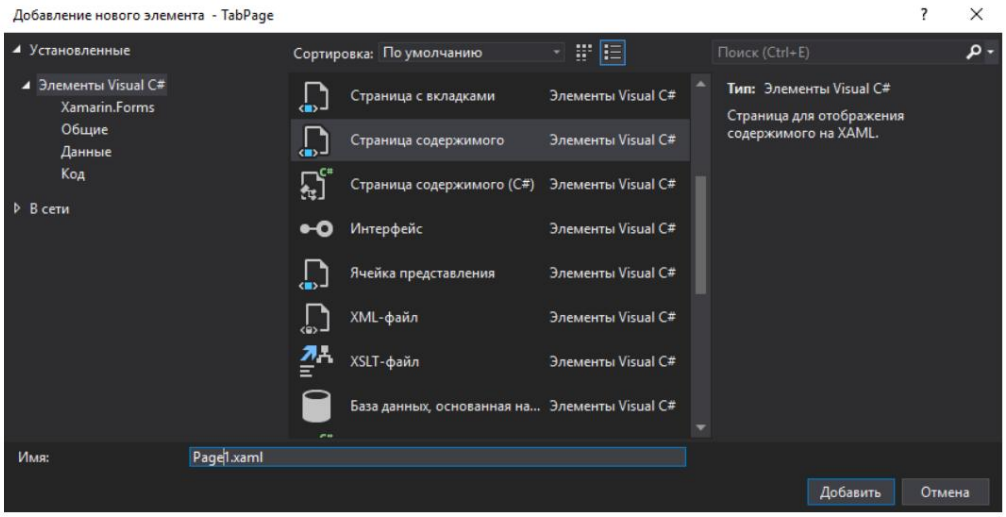
```
sử dụng Xamarin.Forms;
không gian tên TabPage
{
	lớp công khai MainPage : TabbedPage
	{
		MainPage công khai()
		{
			Khởi tạoComponent();
		}
	}
}
```

Hãy mở tệp MainPage.xaml và nhập mã chương trình được cung cấp trong bài thuyết trình.

Mã MainPage XAML đã thêm Page1, Page2 và Page3 vào bộ sưu tập TabbedPage bằng thuộc tính Children.

```
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:local="clr-namespace:TabPage ;assembly=TabPage"
       x:Class="TabPage.MainPage">
  <TabbedPage.Children>
    <local:Trang1 />
    <local:Trang2 />
    <local:Trang3 />
  </TabbedPage.Children>
</TabbedPage>
```

Hãy thêm trang Page1 vào dự án. Để thực hiện việc này, hãy di chuyển con trỏ chuột đến dự án TabPage được chia sẻ, nhấp chuột phải và chọn “Thêm” - “Mục mới”. Trong hộp thoại Thêm mục mới, chọn Trang nội dung để tạo một trang thuộc loại ContentPage. Đặt tên cho trang Trang1.



Bạn cần mở file Page1.xaml và gõ mã xaml trang Trang 1, được tham chiếu từ mã chương trình Trang chính. Trang này chứa bố cục kiểu StackLayout, bên trong có điều khiển Nhãn với văn bản giải thích và điều khiển BoxView màu đỏ cam.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:d="http://xamarin.com/schemas/2014/forms/design"
       xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
       mc:Có thể bỏ qua="d"
```

```
        x:Class="TabPage.Page1"
        BackgroundColor="San hô"
        Tiêu đề="BoxView">
  <ContentPage.Content>
    <StackLayout>
      <Label Text="BoxView dành cho
          hiển thị một hình chữ nhật có chiều rộng, chiều
          cao và màu sắc nhất định"
        VerticalTextAlignment="Trung tâm"
        HorizontalTextAlignment="Trung tâm"
        widthRequest="300" BackgroundColor="Nâu"
        HeightRequest="300" TextColor="Vàng"
        FontAttribut="Đậm"
        FontSize="30" /
        >
      <BoxView
        widthRequest="250"
        Chiều caoYêu cầu="250"
        VerticalOptions="CenterAndExpand"
        HorizontalOptions="Trung tâm"
        BackgroundColor="OrangeRed"/> </
    StackLayout> </
  ContentPage.Content> </
ContentPage>
```

Hãy thêm trang Page2 vào dự án. Cần mở tập tin Page2.xaml và nhập mã xaml của trang Page2 đư ợc liên kết từ mã trang MainPage. Trang này chứa bố cục kiểu StackLayout, bên trong có tùy chọn kiểm soát Nhãn có văn bản giải thích và tùy chọn kiểm soát Trình soạn thảo có màu “Bisque”.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://
       schemas.microsoft.com/winfx/2009/xaml" xmlns:d="http://
       xamarin .com/schemas/2014/forms/design" xmlns:mc="http://
       schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="d"
       x:Class="TabPage.Page2"

       BackgroundColor="DarkGray"
       Tiêu đề="Biên tập viên">
  <ContentPage.Content>
    <StackLayout>

      <Label Text="Trình soạn thảo cho phép ngư ời dùng
          nhập và chỉnh sửa nhiều dòng văn bản"

        VerticalTextAlignment="Trung tâm"
        HorizontalTextAlignment="Trung tâm"
        widthRequest="300" BackgroundColor="Nâu"
```

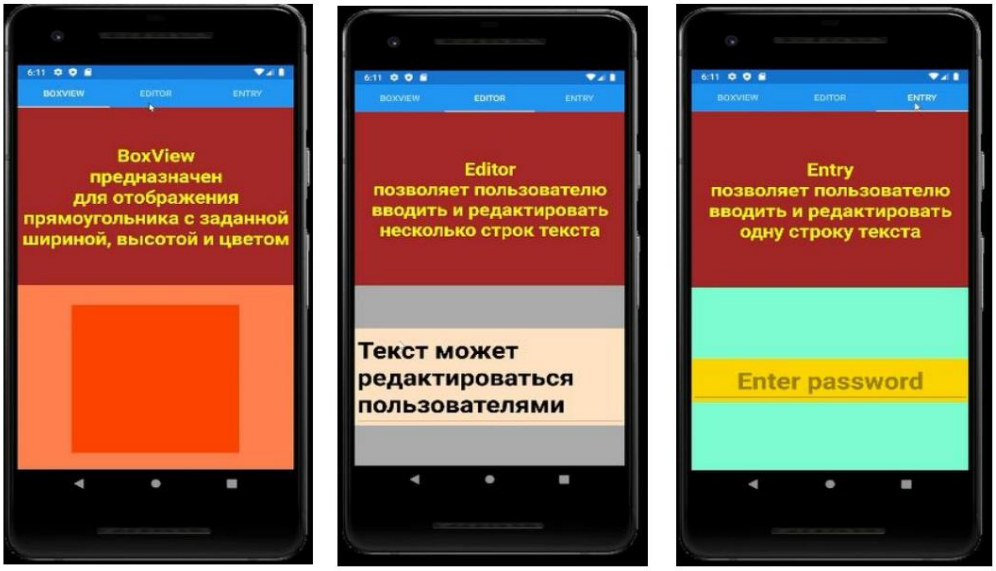
```
          HeightRequest="300" TextColor="Vàng"
          FontAttribut="Đậm"
          Kích thư ớc phông chữ="30"/>

        <Biên tập viên
          BackgroundColor="Bisque"
          VerticalOptions="CenterAndExpand"
                  TextColor="Đen"
          Text="Ngư ời dùng có thể chỉnh sửa văn bản"
          FontSize="40" FontAttribut="Bold"/> </
    StackLayout> </
  ContentPage.Content> </
ContentPage>
```

Hãy thêm trang Page3 vào dự án. Cần mở tập tin Page3.xaml và gõ mã chư ơng trình xaml của trang Page3, đư ợc liên kết từ mã chư ơng trình của trang MainPage. Trang này chứa bố cục kiểu StackLayout, bên trong có điều khiển Nhãn với văn bản giải thích và điều khiển Mục nhập có màu “Vàng”.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:d="http://xamarin .com/schemas/2014/forms/design"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        x:Class="TabPage.Page3"
        BackgroundColor="Aquamarine"
        Title="Entry">
  <ContentPage.Content>
    <StackLayout>

      <Label Text="Mục nhập cho phép ngư ời dùng
          nhập và chỉnh sửa một dòng văn bản."

        VerticalTextAlignment="Trung tâm"
        HorizontalTextAlignment="Trung tâm"
        widthRequest="300" BackgroundColor="Nâu"
        HeightRequest="300" TextColor="Vàng"
        FontAttribut="Đậm"
        Kích thư ớc phông chữ="30"/>

      <Bàn phím nhập="Văn bản"
        BackgroundColor="Vàng"
        HorizontalTextAlignment="Trung tâm"
        Giữ chỗ="Nhập mật khẩu"
        VerticalOptions="CenterAndExpand"
        FontSize="40" FontAttribut="Đậm"/>
```

```
    </StackLayout>
  </ContentPage.Content>
</Trang nội dung>
```

Hãy khởi chạy ứng dụng phần mềm. Một trang có ba tab được hiển thị. Bạn có thể chuyển đổi giữa các tab hiển thị tên của các điều khiển được sử dụng để tạo giao diện người dùng và giải thích mục đích của các điều khiển.



CarouselPage trong Xamarin.Forms cho phép bạn điều hướng giữa các trang bằng cử chỉ cuộn, nghĩa là khi người dùng vuốt màn hình cảm ứng sang phải để chuyển tiếp qua một tập hợp các trang hoặc sang trái để di chuyển lùi để lật trang.

Nếu một CarouselPage được nhúng vào trang Chi tiết của FlyoutPage thì thuộc tính FlyoutPage.IsGestureEnabled phải được đặt thành false để tránh xung đột giữa các trang

CarouselPage và FlyoutDetailPage.

Có hai cách để tạo CarouselPage:

1. Bạn cần chỉ định một tập hợp các trang cho thuộc tính Trẻ em của trang CarouselPage. Trong trường hợp này, bộ sưu tập chỉ có thể bao gồm các trang thuộc loại ContentPage hoặc các đối tượng thuộc loại ContentPage.

2. Gán thuộc tính ItemsSource của trang CarouselPage cho một tập hợp các trang. Sau đó gán thuộc tính ItemTemplate DataTemplate (mô tả về mẫu chỉ định cách hiển thị giống hệt nhau của từng trang từ bộ sưu tập được chỉ định trong thuộc tính ItemSource).

Cả hai phương pháp đều khiến các trang thuộc bộ sưu tập CarouselPage xuất hiện theo trình tự khi bạn vuốt màn hình.

Hãy tạo dự án CarPage cho dự án đa nền tảng Xamarin.Forms bằng mẫu Trống. Hãy mở tập tin MainPage.xaml.cs và điều chỉnh mã chương trình cho phù hợp

bài thuyết trình. Từ tệp này, bạn có thể thấy rằng trang MainPage là một trang thuộc loại CarouselPage.

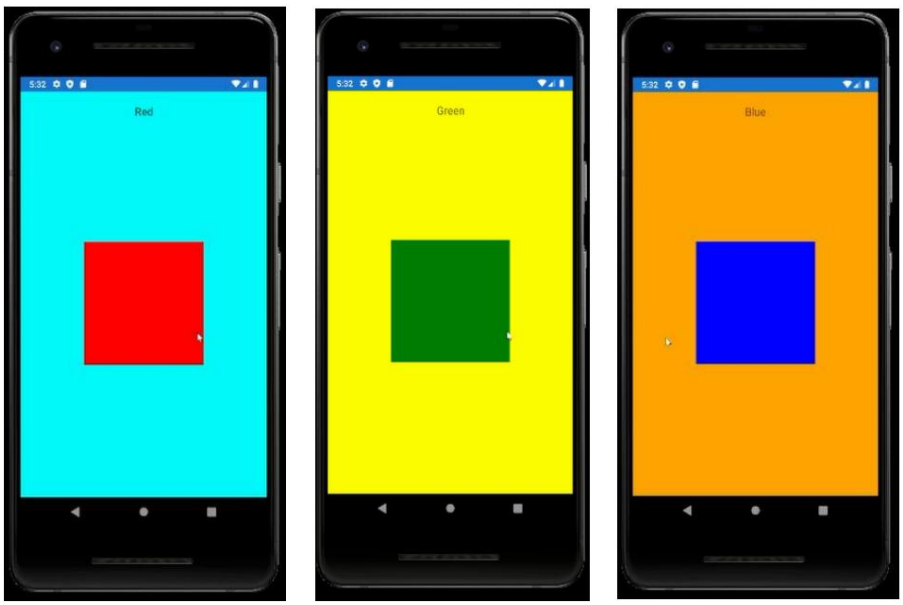
```
sử dụng Xamarin.Forms;
không gian tên CarPage
{
        lớp một phần công khai MainPage : CarouselPage
        {
                MainPage công khai()
                {
                        Khởi tạoComponent();
                }
        }
}
```

Hãy mở tệp MainPage.xaml. Tập tin này chứa mã chương trình XAML tương ứng với CarouselPage có chứa một bộ sưu tập gồm ba trang thuộc loại ContentPage. Trong ứng dụng phần mềm này, CarouselPage được tạo bằng cách sử dụng phương pháp đầu tiên trong số các phương pháp trên (nghĩa là tạo một tập hợp các trang thuộc loại ContentPage). Bộ sưu tập bao gồm ba trang thuộc loại ContentPage. Mỗi trang chứa một StackLayout chứa các điều khiển Label và BoxView. Trên trang đầu tiên, màu nền là "Cyan" và điều khiển BoxView có màu đỏ. Trang thứ hai có màu nền vàng và điều khiển BoxView màu xanh lá cây. Trang thứ ba có màu nền màu cam và điều khiển BoxView màu xanh lam.

```
<?xml version="1.0" mã hóa="UTF-8"?>
<CarouselPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://
        schemas.microsoft.com/winfx/2009/xaml"
              x:Class="CarPage.MainPage">
  <Trang nội dung
    BackgroundColor="Cyan" Padding="20">
    <StackLayout>
      <Văn bản nhãn="Đỏ"
          FontSize="Trung bình" FontAttribution="Đậm"
          HorizontalOptions="Trung tâm" />
      <BoxView Color="Đỏ"
           Chiều rộngRequest="200" Chiều caoRequest="200"
           HorizontalOptions="Trung tâm"
           VerticalOptions="CenterAndExpand" />
    </StackLayout>
  </Trang nội dung>
  <Trang nội dung
    BackgroundColor="Vàng" Padding="20">
    <StackLayout>
      <Văn bản nhãn="Xanh"
          FontSize="Trung bình"
          HorizontalOptions="Trung tâm" />
      <BoxView Color="Xanh"
```

```
            Chiều rộngRequest="200" Chiều caoRequest="200"
            HorizontalOptions="Trung tâm"
            VerticalOptions="CenterAndExpand" />
    </StackLayout>
  </Trang nội dung>
  <Trang nội dung
    BackgroundColor="Orange" Padding="20">
    <StackLayout>
      <Văn bản nhãn="Màu xanh"
          FontSize="Trung bình"
          HorizontalOptions="Trung tâm" />
      <BoxView Color="Xanh"
          Chiều rộngRequest="200" Chiều caoRequest="200"
          HorizontalOptions="Trung tâm"
          VerticalOptions="CenterAndExpand" />
    </StackLayout>
  </Trang nội dung>
</CarouselPage>
```

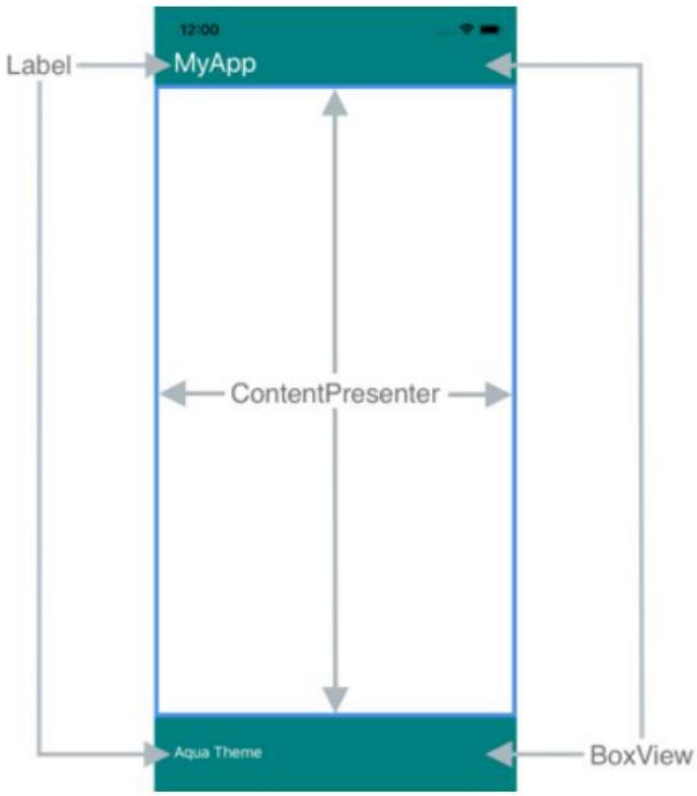
Khi bạn khởi chạy ứng dụng phần mềm, trang đầu tiên sẽ xuất hiện với điều khiển BoxView có màu đỏ. Khi bạn cuộn, các trang có điều khiển BoxView màu xanh lá cây và xanh lam sẽ xuất hiện.



Trang Bản mẫu cho phép bạn xác định một mẫu để xác định cấu trúc trực quan tiêu chuẩn cho các trang thuộc loại ContentPage. Bằng cách này, một mẫu trang có thể được sử dụng để hiển thị nhiều trang theo cùng một cách. Trong trường hợp này, các điều khiển trên các trang sẽ được hiển thị thống nhất, có tính đến các cài đặt có trong mẫu.

Đối với trang mẫu có chứa một số điều khiển, ControlTemplate được sử dụng để chỉ định

điều khiển có trong mẫu. Trong mẫu, bạn có thể chỉ định điều khiển ContentPresenter lưu trữ các phần tử
các điều khiển được chuyển đến mẫu từ trang mà mẫu được áp dụng. Hình này hiển thị ControlTemplate cho một trang chứa các điều khiển Nhãn, điều khiển BoxView và phần tử ContentPresenter ở dạng hình chữ nhật.



Hãy tạo dự án TemPage cho dự án đa nền tảng Xamarin.Forms bằng mẫu Trống. Hãy tạo một tập tin
HeaderFooterPage.xaml, chứa các mẫu "TealTemplate" cho
hình thành trang chính MainPage, mã chương trình trong đó
đề cập đến một mẫu và sẽ được tạo sau. Hãy gõ mã chương trình theo đúng bài trình bày.

Phần đầu của trang đặt cài đặt cho các điều khiển Mục nhập nằm trong Trang Chính. Cài đặt được đặt bằng cách sử dụng thuộc tính Style và TargetType. Kiểu hiển thị trực quan của các điều khiển được định cấu hình khi làm việc với chúng (nghĩa là khi tiêu điểm được chuyển sang chúng).

Thông thường, ControlTemplate được khai báo là tài nguyên. Nếu ControlTemplate được khai báo là tài nguyên thì nó phải có khóa được chỉ định bằng thuộc tính x:Key.

ControlTemplate chỉ có một điều khiển làm phần tử gốc. Trong trường hợp này, điều khiển gốc chứa các điều khiển khác.

Mã hiển thị OrangeTemplate và GreenTemplate. Phần tử gốc của mỗi mẫu là phần tử

Kiểm soát lưới. Sự kết hợp của các điều khiển trong điều khiển gốc tạo nên cấu trúc trực quan của trang chính. Bên trong phần tử gốc có ba ô nằm

ở dưới nhau. Ô đầu tiên chứa tiêu đề trang.

Ô đầu tiên chiếm 10% chiều cao của trang và chứa điều khiển BoxView lấp đầy toàn bộ ô cũng như điều khiển Nhãn hiển thị tên của mẫu đang được sử dụng.

Đối với các điều khiển này, vị trí, màu tô, màu phông chữ và kích thước cũng như mức thụt lề từ cạnh của điều khiển đã được định cấu hình.

Ô thứ hai chứa điều khiển ContentPresenter sẽ lưu trữ các điều khiển xuất hiện trên trang truy cập vào mẫu. Ô thứ hai có chiều cao bằng 80% chiều cao của trang. Ô thứ ba là tiêu đề trang, có chiều cao bằng 10% chiều cao của trang mẫu và chứa điều khiển BoxView lấp đầy toàn bộ ô cũng như điều khiển Nhãn mà bạn phải bấm vào

nếu bạn muốn chuyển đổi mẫu đang được sử dụng. Đối với các điều khiển này, vị trí, màu tô cũng như màu và kích thước phông chữ, kiểu viết văn bản và mức thụt lề từ cạnh của điều khiển cũng đã được định cấu hình.

Tiện ích mở rộng đánh dấu templateBinding với thuộc tính HeaderText cho phép một giá trị được chuyển đến thuộc tính Text của mẫu Kiểm soát nhãn từ mã trong tệp MainPage.xaml.

Điều quan trọng cần lưu ý là điều khiển Nhãn nằm ở cuối trang có trình xử lý sự kiện được đính kèm.
OnChangeTheme, được thiết kế để xử lý cử chỉ chạm và được cung cấp trong tệp HeaderFooterPage.xaml.cs, tệp này sẽ được thảo luận sau. Điều khiển này cũng được liên kết với thủ tục OnApplyTemplate, thủ tục này chịu trách nhiệm thay đổi mẫu hiển thị trang và được cung cấp trong tệp MainPage.xaml.cs.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:controls="clr-namespace: TemPage"
       x:Class="TemPage.HeaderFooterPage">
  <ContentPage.Resources>
    <Style TargetType="Entry">
      <Setter Property="VisualStateManager.VisualStateGroups">
        <VisualStateGroupList>
          <VisualStateGroup x:Name="CommonStates">
            <VisualState x:Name="Bình thường">
              <VisualState.Setters>
                <Thuộc tính Setter="Kích thước phông chữ"
                     Giá trị="18" />
              </VisualState.Setters>
```

```
            </VisualState>
            <VisualState x:Name="Tập trung">
              <VisualState.Setters>
                <Setter Property="FontSize"
                     Value="36" />
              </VisualState.Setters>
            </VisualState>
          </VisualStateGroup> </
        VisualStateGroupList> </
      Setter> </
    Style>

    <ControlTemplate x:Key="OrangeTemplate">
      <Lư ới>
        <Grid.RowDefinitions>
          <RowDefinition Height="0.1*" />
          <RowDefinition Height="0.8*" />
          <RowDefinition Height="0.1*" /> </
        Grid.RowDefinitions>
        <BoxView Color="Orange" />
        <Label Margin=" 20.0.0.0"
            Text="{TemplateBinding HeaderText}"
            TextColor="Trắng"
            Kích thư ớc phông chữ="20"
            HorizontalOptions="Trung tâm"
            VerticalOptions="Center" />
        <ContentPresenter Grid.Row="1" />
        <BoxView Grid.Row="2"
             Màu="Cam" />
        <Nhãn x:Name="ChangeThemeLabel"
            Lư ới.Row="2"
            Ký quỹ="20,0,0,0"
            Text="Thay đổi chủ đề"
            TextColor="Trắng"
            Kích thư ớc phông chữ="20"
            HorizontalOptions="Trung tâm"
            VerticalOptions="Center">
          <Label.GestureRecognizers>
            <TapGestureRecognizer
              Tapped="OnChangeTheme" />
          </Label.GestureRecognizers> </
        Label> </
      Grid> </
    ControlTemplate>

    <ControlTemplate x:Key="GreenTemplate">
      <Grid>
        <Grid.RowDefinitions>
          <RowDefinition Chiều cao="0,1*" />
```

```
            <RowDefinition Chiều cao="0,8*" />
            <RowDefinition Chiều cao="0,1*" />
          </Grid.RowDefinitions>
          <BoxView Color="Green" />
          <Lề nhãn="20,0,0,0"
              Text="{TemplateBinding HeaderText}"
              TextColor="Vàng"
              Kích thước phông chữ="20"
              HorizontalOptions="Trung tâm"
              VerticalOptions="Trung tâm" />
          <ContentPresenter Grid.Row="1" />
          <BoxView Grid.Row="2"
               Màu="Xanh" />
          <Nhãn x:Name="ChangeThemeLabel"
              Lư ới.Row="2"
              Ký quỹ="20,0,0,0"
              Text="Thay đổi chủ đề"
              FontSize="20" FontAttribut="Đậm"
              TextColor="Vàng"
              HorizontalOptions="Trung tâm"
              VerticalOptions="Trung tâm">
            <Label.GestureRecognizers>
              <TapGestureRecognizer
                Tapped="OnChangeTheme" />
            </Label.GestureRecognizers>
          </Nhãn>
        </Grid>
      </ControlTemplate>
    </ContentPage.Resources>
</Trang nội dung>
```

Hãy mở tệp HeaderFooterPage.xaml.cs. Hãy gõ vào tập tin chư ơng trình mã theo bản trình bày.

Trong mã chư ơng trình, những cái đư ợc thảo luận trư ớc đó đư ợc khai báo đầu tiên Mẫu OrangeTemplate và GreenTemplate. Sau

đó, HeaderTextProperty đư ợc khai báo để ghi lại giá trị của thuộc tính mẫu HeaderText. Trang HeaderFooterPage thuộc loại ContentPage đư ợc khởi tạo.

Để theo dõi việc chuyển đổi giữa các mẫu OrangeTemplate và GreenTemplate, hãy sử dụng thuộc tính OriginalTemplate và biến phụ ban đầu. Để xử lý cử chỉ nhấp vào điều khiển

Nhãn nhằm thay đổi mẫu trang, một trình xử lý đư ợc thiết kế

OnChangeTheme, đư ợc tham chiếu bởi mã xaml của trang

Trang đầu trang chân trang.

```
sử dụng Xamarin.Forms;
sử dụng Xamarin.Forms.Xaml;
```

```
không gian tên TempPage
{
  [XamlCompilation(XamlCompilationOptions.Compile)]
  lớp một phần công khai HeaderFooterPage : ContentPage
  {
    ControlTemplate OrangeTemplate;
    ControlTemplate GreenTemplate;

    công khai tĩnh chỉ đọc BindableProperty
      HeaderTextProperty =
      BindableProperty.Create(nameof(HeaderText),
                  typeof(string),
                  typeof(HeaderFooterPage),
                  default(string));
    Tiêu đề chuỗi công khai
    {
      get => (string)GetValue(HeaderTextProperty);
      set => SetValue(HeaderTextProperty, value);
    }
    bool gốc = đúng;
    bool công khai OriginalTemplate
    {
      lấy { trả lại bản gốc; }
    }

    công khai HeaderFooterPage()
    {
      Khởi tạoComponent();
      Mẫu màu cam =
        (ControlTemplate)Tài nguyên["OrangeTemplate"];
      Mẫu xanh =
        (ControlTemplate)Tài nguyên["GreenTemplate"];
    }
    void OnChangeTheme(ngư ời gửi đối tư ợng, EventArgs e)
    {
      gốc = !nguyên gốc;
      ControlTemplate =
        (bản gốc)? Bản mẫu màu cam: Bản mẫu màu xanh lá cây;
    }
  }
}
```

Chúng ta hãy mở file MainPage.xaml và gõ mã chư ơng trình theo đúng bài trình bày. Cần lư u ý rằng mã

chư ơng trình tham chiếu đến HeaderFooterPage, nơi lư u trữ các mẫu cũng như cài đặt cho các điều khiển Mục nhập.

Thuộc tính HeaderText được thiết kế để chuyển chuỗi văn bản "TemplatePage" cho giá trị của thuộc tính Text của điều khiển Nhãn nằm trong tiêu đề của trang mẫu.

MainPage mà mẫu được áp dụng sẽ chuyển điều khiển ContentPresenter của trang mẫu sang bố cục StackLayout với hai điều khiển Mục nhập và một điều khiển Nút. Các điều khiển mục nhập áp dụng cài đặt hiển thị tiêu chuẩn có trong mã HeaderFooterPage.

```
<controls:HeaderFooterPage xmlns="http://xamarin.com/schemas/2014/forms"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:controls="clr-namespace:TemPage"
       x:Class="TemPage.MainPage"
       Title="Truy cập mẫu"
       HeaderText="Trang mẫu"
       ControlTemplate="{StaticResource OrangeTemplate}">
  <Lề StackLayout="10">
    <Entry Placeholder="Nhập tên người dùng" />
    <Entry Placeholder="Nhập mật khẩu"
         IsPassword="True" />
    <Nút văn bản="Đăng nhập" />
  </StackLayout>
</controls:HeaderFooterPage>
```

Chúng ta mở file MainPage.xaml.cs và gõ mã chương trình vào theo phần trình bày. Biến

TemLabel được sử dụng để liên kết với điều khiển Nhãn nằm ở đầu trang và chân trang của trang mẫu và được đặt tên là ChangeThemeLabel theo thuộc tính x:Name.

Quy trình OnApplyTemplate được thiết kế để thay đổi văn bản trong điều khiển Nhãn trên trang mẫu khi điều khiển được nhấp vào khi mẫu hiển thị của trang được thay đổi.

```
sử dụng Xamarin.Forms;
không gian tên TempPage
{
  lớp công khai MainPage : HeaderFooterPage
  {
    Nhãn TemLabel;
    MainPage công khai()
    {
      Khởi tạoComponent();
    }
    ghi đè được bảo vệ void OnApplyTemplate()
    {
      base.OnApplyTemplate();
```

```
            TemLabel = (Nhãn)GetTemplateChild("ChangeThemeLabel");
            TemLabel.Text =
                Mẫu gốc? “Chủ đề màu cam” : “Chủ đề màu xanh lá cây”;
        }
    }
}
```

Chúng ta sẽ lắp ráp ứng dụng phần mềm, sau đó sẽ không có có lỗi và khởi chạy ứng dụng phần mềm.

Đầu tiên trang đư ợc hiển thị theo mẫu Mẫu màu cam.

Bên trong trang có hai điều khiển Mục nhập cũng như điều khiển Nút, đư ợc chuyển như một phần của điều khiển StackLayout sang điều khiển mẫu ContentPresenter.

Khi bạn bấm vào điều khiển Nhãn trong tiêu đề, hiển thị tên của mẫu đang đư ợc sử dụng, mẫu hiển thị của trang MainPage sẽ đư ợc thay đổi.

Khi bạn bắt đầu làm việc với các điều khiển Mục nhập, chúng sẽ thay đổi trạng thái trực quan (thay đổi kích thư ớc phông chữ).

Sau khi bạn làm việc xong với các điều khiển Mục nhập, chúng trở về trạng thái ban đầu của chúng.